Lecture 17:
3D Vision

Reminder: A4

A4 due **Today, Wednesday, November 13, 11:59pm**

A4 covers:
- PyTorch autograd
- Residual networks
- Recurrent neural networks
- Attention
- Feature visualization
- Style transfer
- Adversarial examples

Recall: Course Structure

We are here!

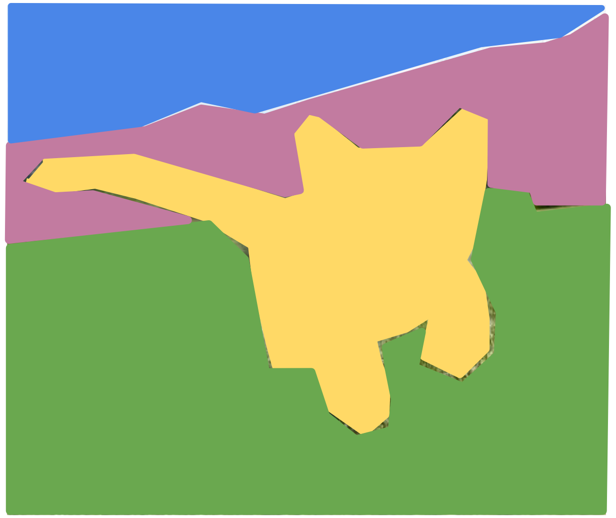
- First half: Fundamentals
  - Details of how to implement and train different types of networks
  - Fully-connected networks, convolutional networks, recurrent networks
  - How to train and debug, very detailed
- Second half: Applications and “Researchy” topics
  - Object detection, image segmentation, 3D vision, videos
  - Attention, Transformers
  - Vision and Language
  - Generative models: GANs, VAEs, etc
  - Less detailed: provide overview and references, but skip some details

Last Time: Predicting 2D Shapes of Objects

Classification

CAT

Semantic Segmentation

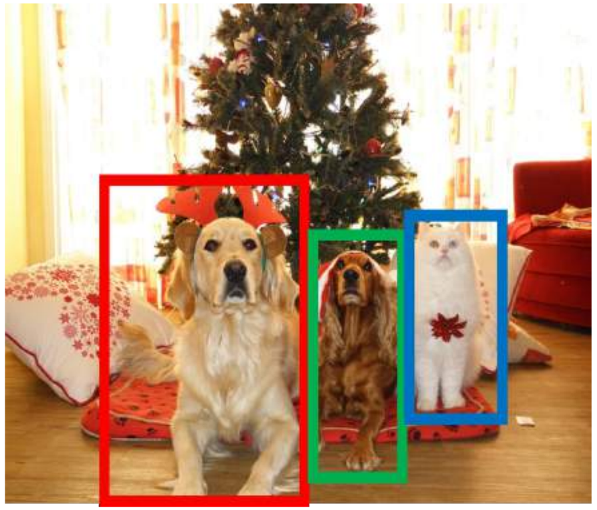


GRASS, CAT, TREE, SKY

No spatial extent

No objects, just pixels

Object Detection



DOG, DOG, CAT

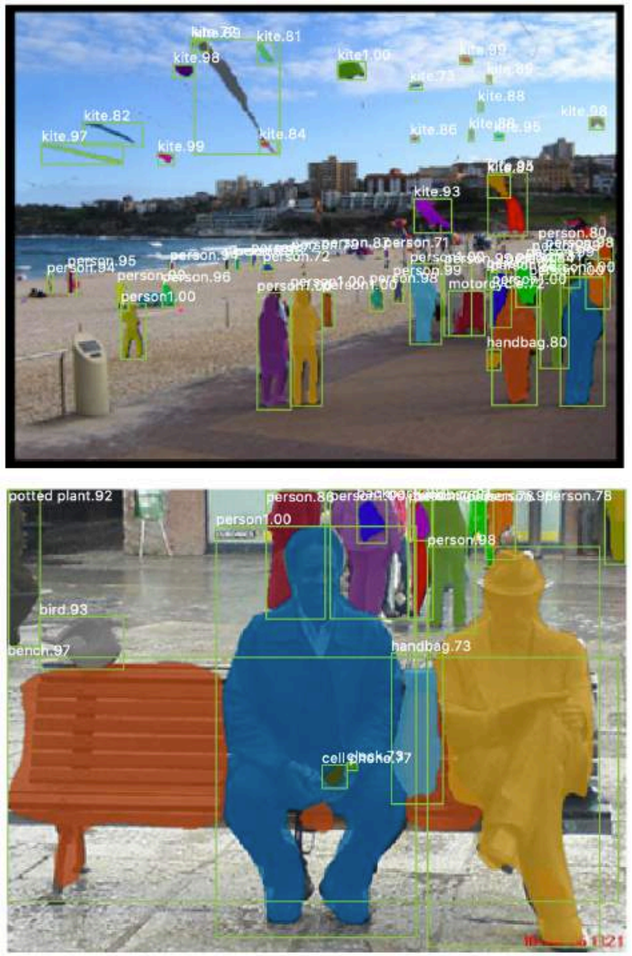
Instance Segmentation

DOG, DOG, CAT

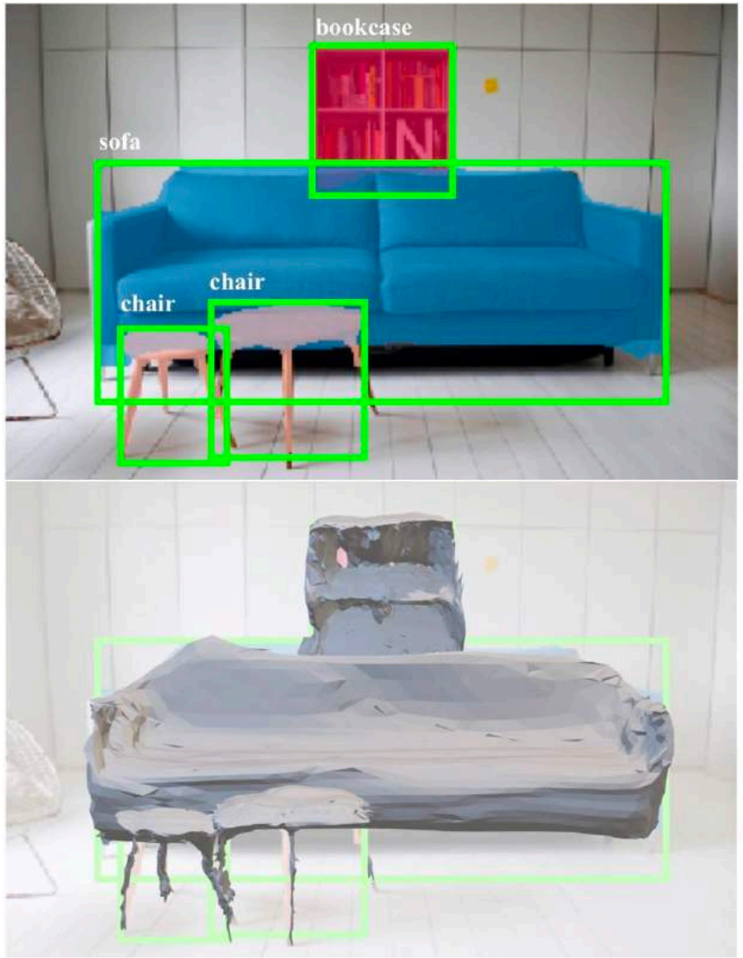
Multiple Objects

Today: Predicting **3D Shapes of Objects**

Mask R-CNN:
2D Image -> 2D shapes



Mesh R-CNN:
2D Image -> **3D** shapes



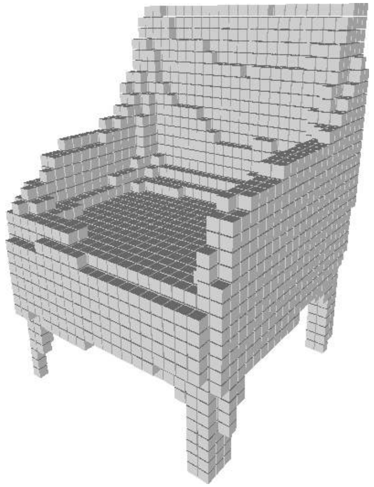
He, Gkioxari, Dollár, and Girshick, "Mask R-CNN", ICCV 2017

Gkioxari, Malik, and Johnson, "Mesh R-CNN", ICCV 2019

Focus on Two Problems today

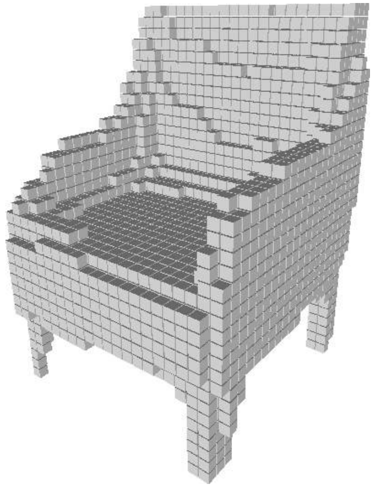
Predicting 3D Shapes
from single image

Input Image



3D Shape

Processing 3D
input data



3D Shape

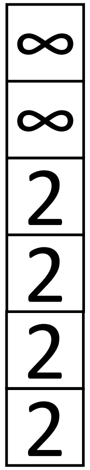
Chair

Many more topics in 3D Vision!

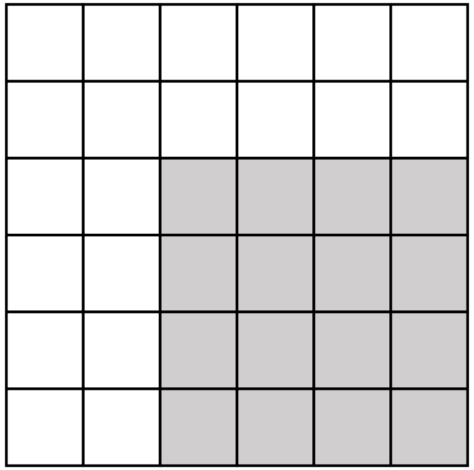
Computing correspondences
Multi-view stereo
Structure from Motion
Simultaneous Localization and Mapping (SLAM)
Self-supervised learning
View Synthesis
Differentiable graphics
3D Sensors

Many non-Deep Learning methods alive and well in 3D!

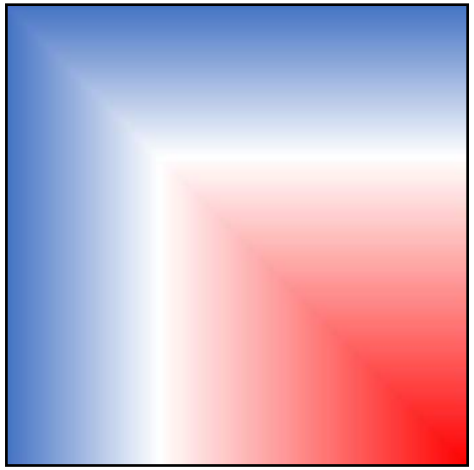
3D Shape Representations



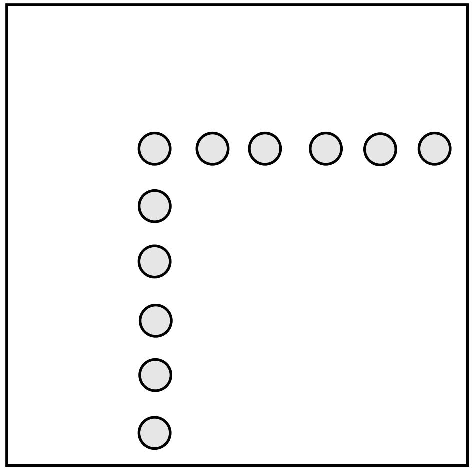
Depth Map



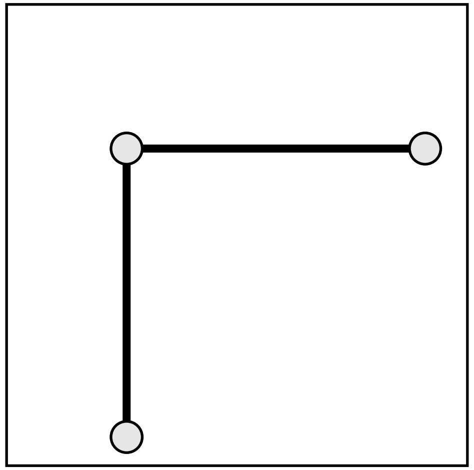
Voxel Grid



Implicit Surface

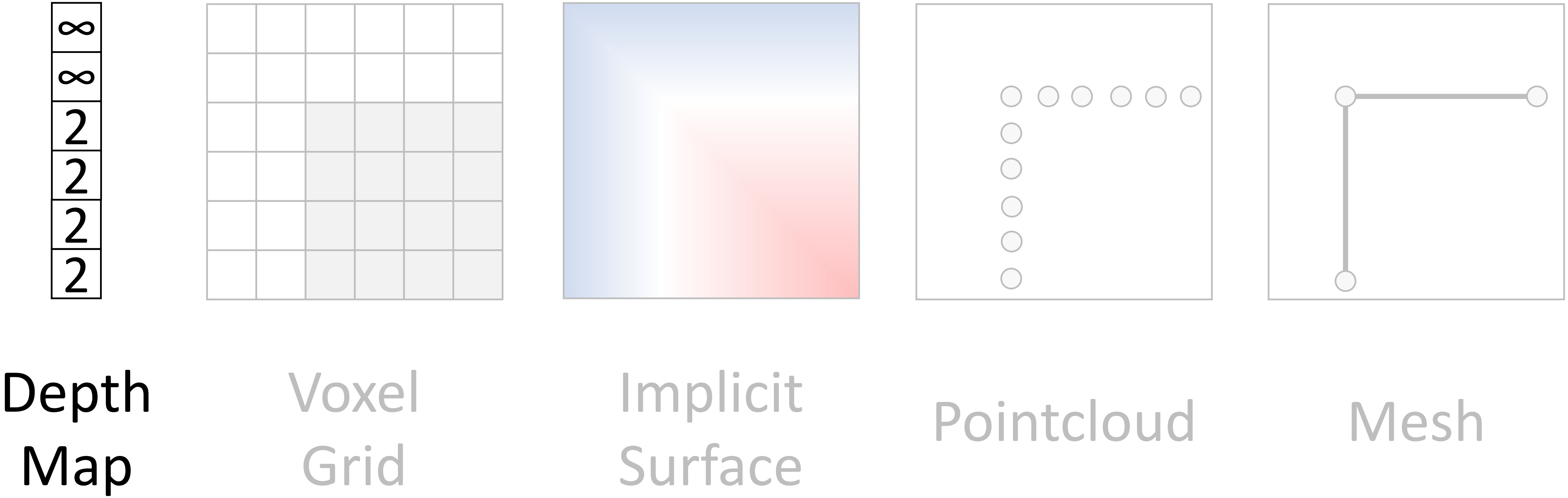


Pointcloud



Mesh

3D Shape Representations

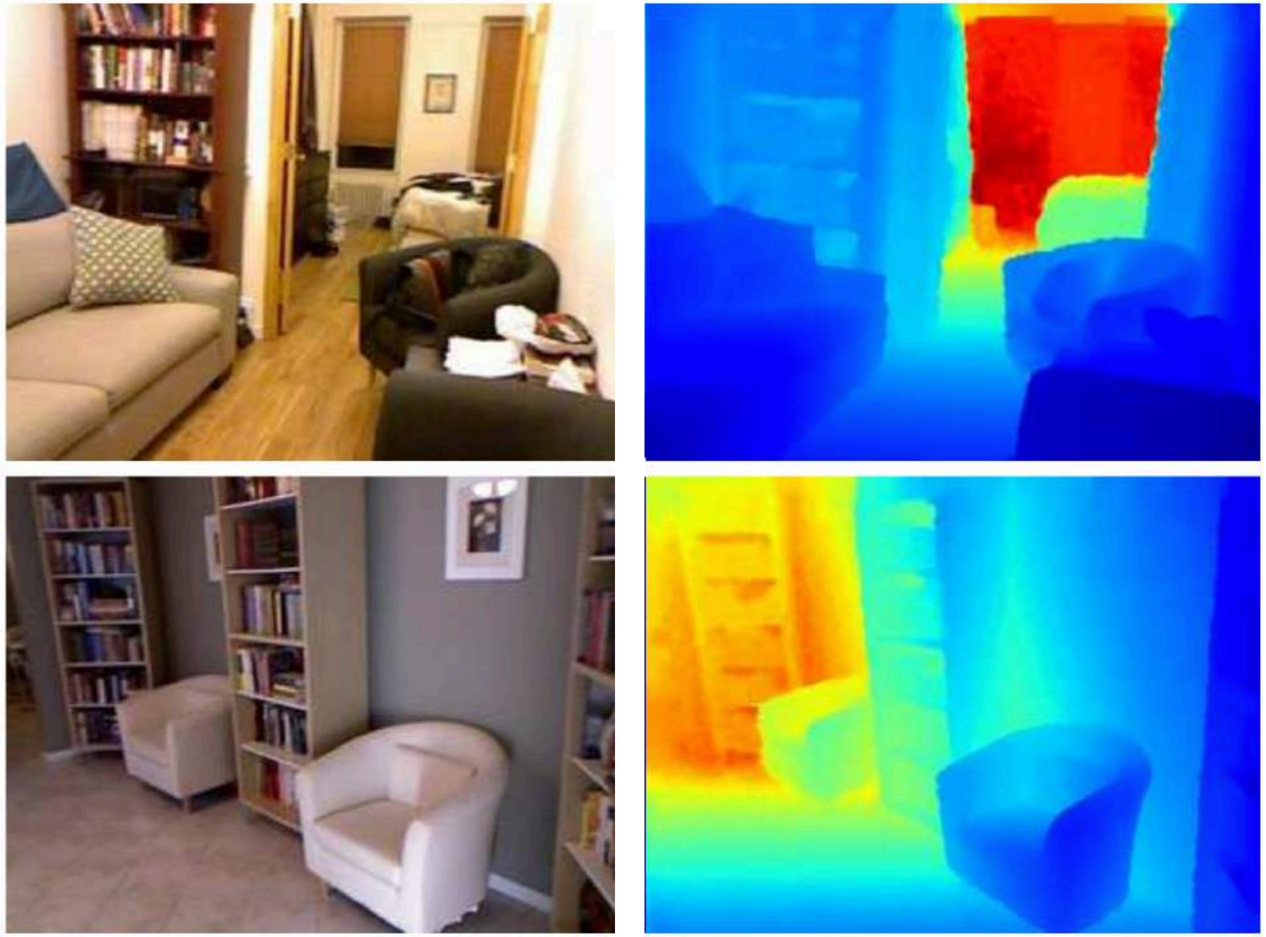


3D Shape Representations: Depth Map

For each pixel, **depth map** gives distance from the camera to the object in the world at that pixel

RGB image + Depth image
= RGB-D Image (2.5D)

This type of data can be recorded directly for some types of 3D sensors (e.g. Microsoft Kinect)

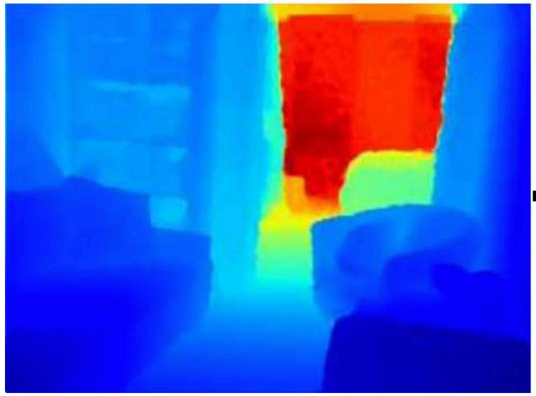


RGB Image: 3 x H x W   Depth Map: H x W

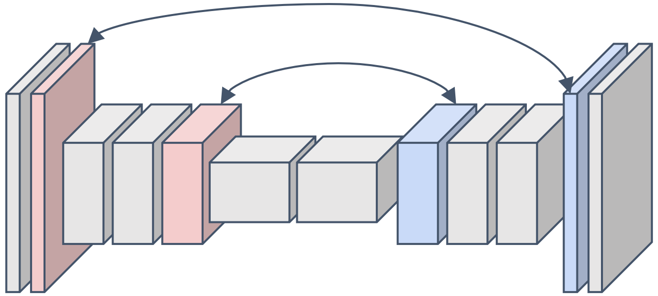
Eigen and Fergus, "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture", ICCV 2015

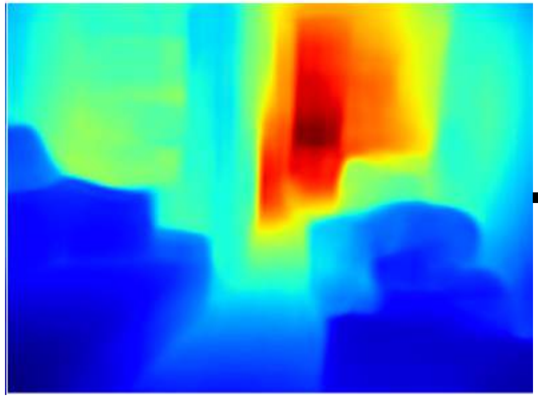
Predicting Depth Maps

Predicted Depth Image:
1 x H x W



Per-Pixel Loss
(L2 Distance)





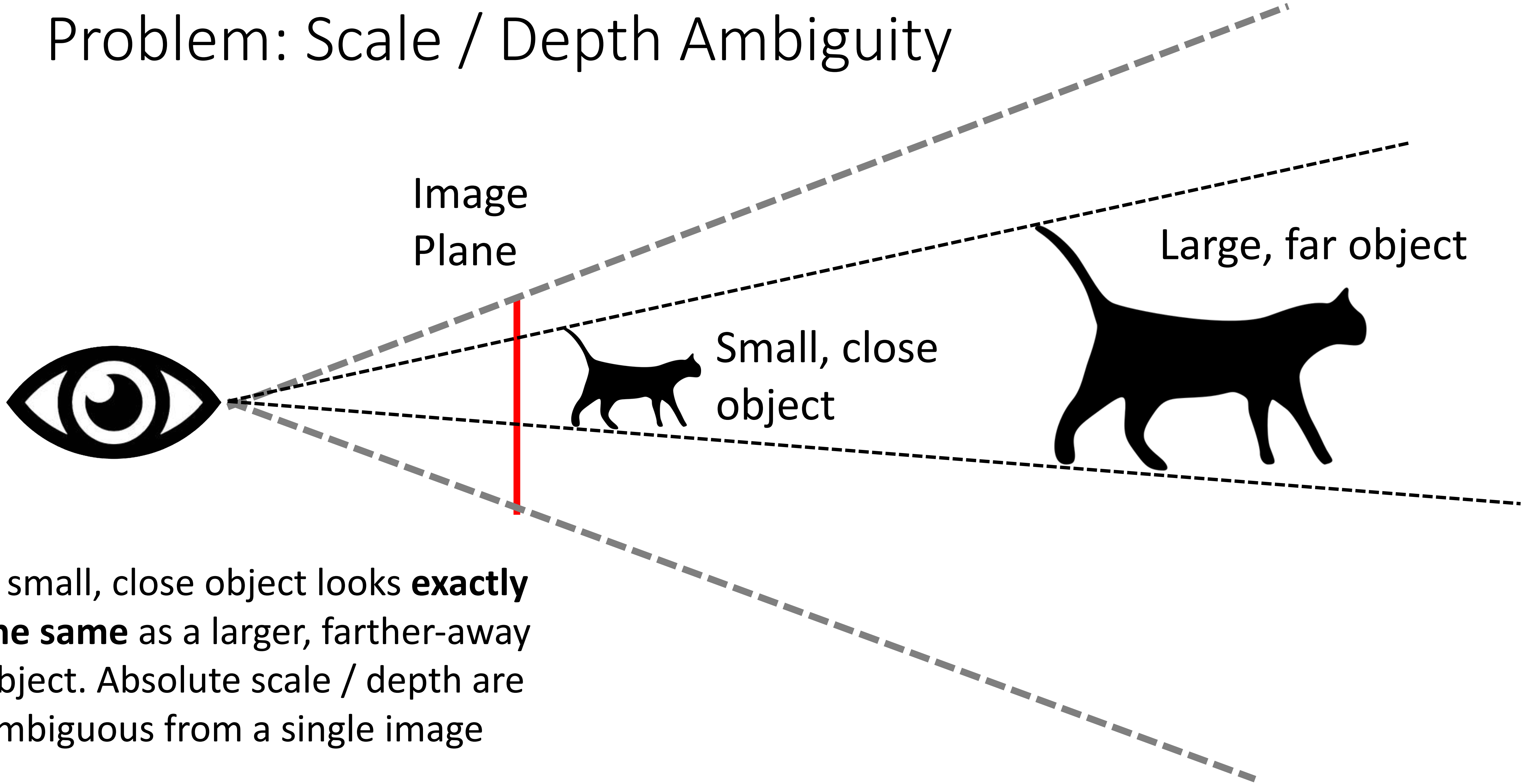
RGB Input Image:
3 x H x W

Fully Convolutional network

Predicted Depth Image:
1 x H x W

Eigen, Puhrsh, and Fergus, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network", NeurIPS 2014
Eigen and Fergus, "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture", ICCV 2015

Problem: Scale / Depth Ambiguity



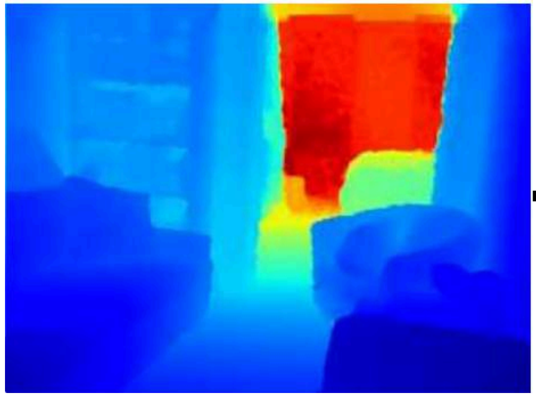
A small, close object looks **exactly the same** as a larger, farther-away object. Absolute scale / depth are ambiguous from a single image

Predicting Depth Maps

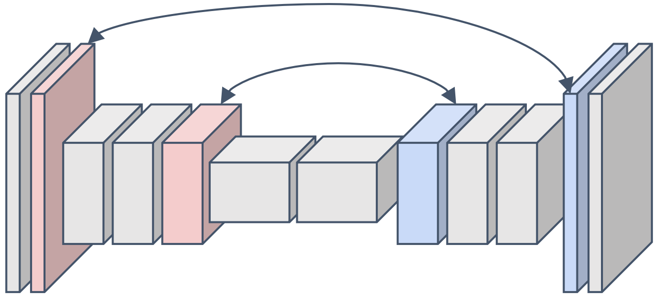
Scale invariant loss

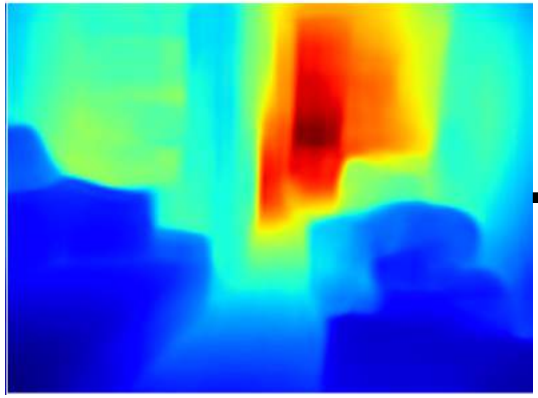
$$
\begin{aligned}
D(y, y^*) &= \frac{1}{2n^2} \sum_{i,j} \left((\log y_i - \log y_j) - (\log y_i^* - \log y_j^*) \right)^2 \\
&= \frac{1}{n} \sum_i d_i^2 - \frac{1}{n^2} \sum_{i,j} d_i d_j \;=\; \frac{1}{n} \sum_i d_i^2 - \frac{1}{n^2} \left(\sum_i d_i \right)^2
\end{aligned}
$$

Predicted Depth Image:
1 x H x W



Per-Pixel Loss
(Scale invariant)





RGB Input Image:
3 x H x W

Fully Convolutional network

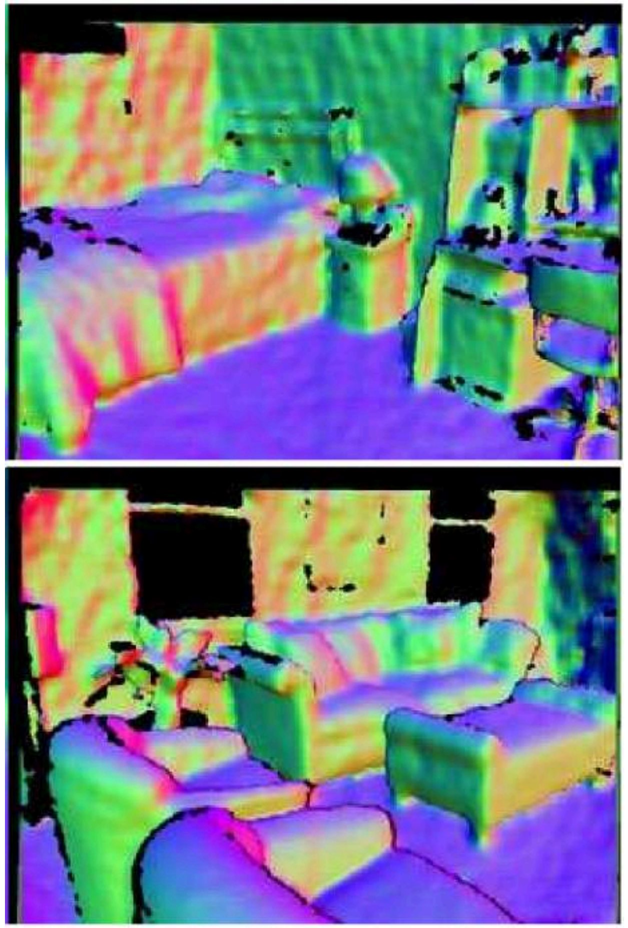
Predicted Depth Image:
1 x H x W

Eigen, Puhrsh, and Fergus, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network", NeurIPS 2014
Eigen and Fergus, "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture", ICCV 2015

3D Shape Representations: Surface Normals

For each pixel, **surface normals** give a vector giving the normal vector to the object in the world for that pixel

RGB Image: 3 x H x W

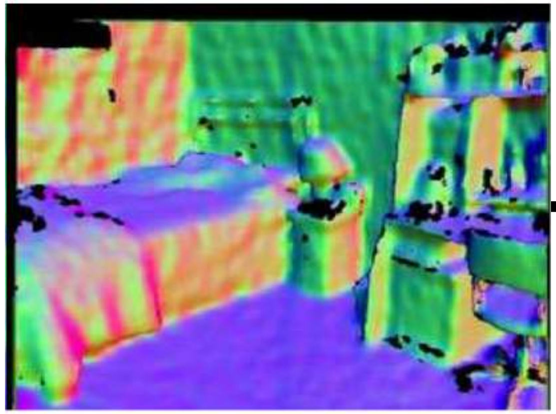


Normals: 3 x H x W

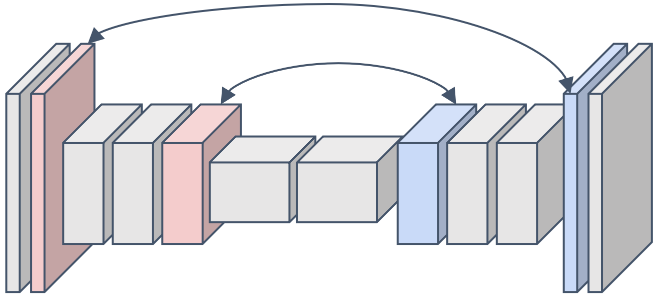
Eigen and Fergus, “Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture”, ICCV 2015

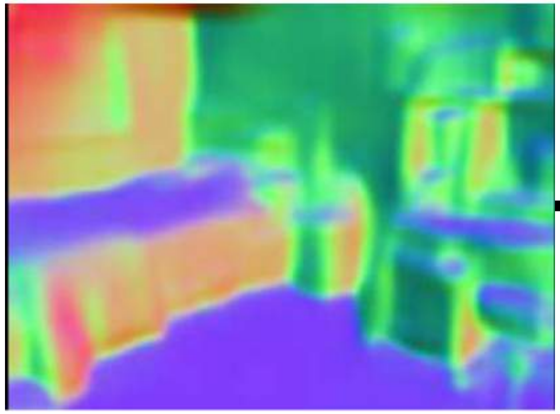
Predicting Normals

Ground-truth Normals:
3 x H x W



Per-Pixel Loss:
$(x \cdot y) / (|x| \, |y|)$





RGB Input Image:
3 x H x W

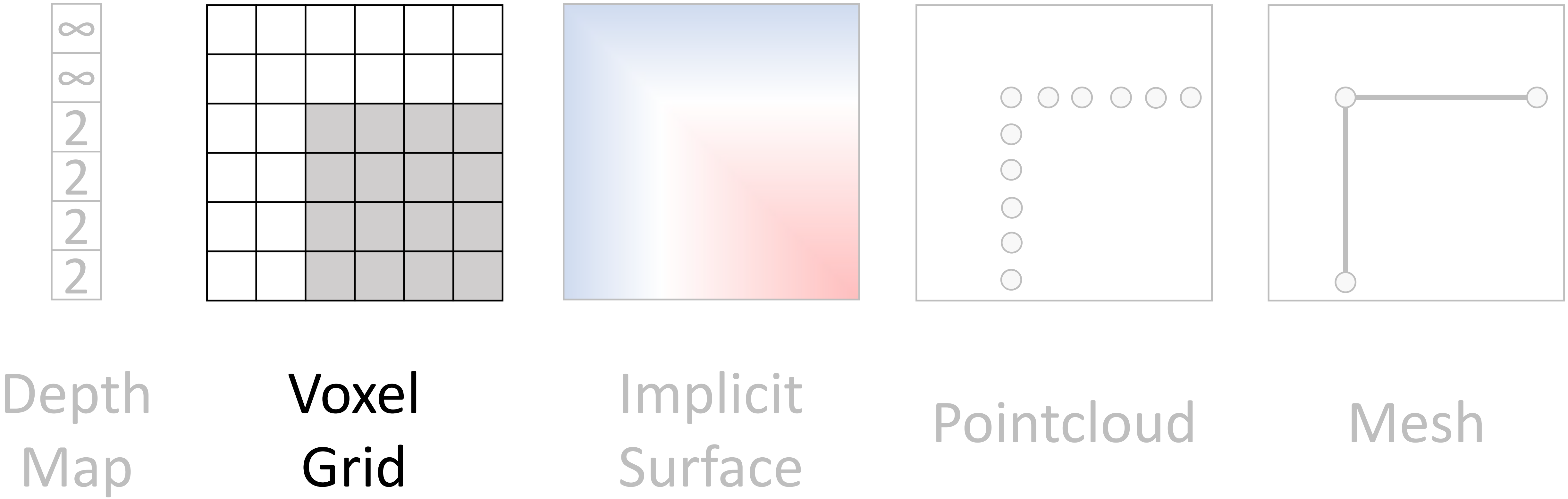
Fully Convolutional network

Predicted Normals:
3 x H x W

Recall:
$x \cdot y$
$= |x| \; |y| \cos \theta$

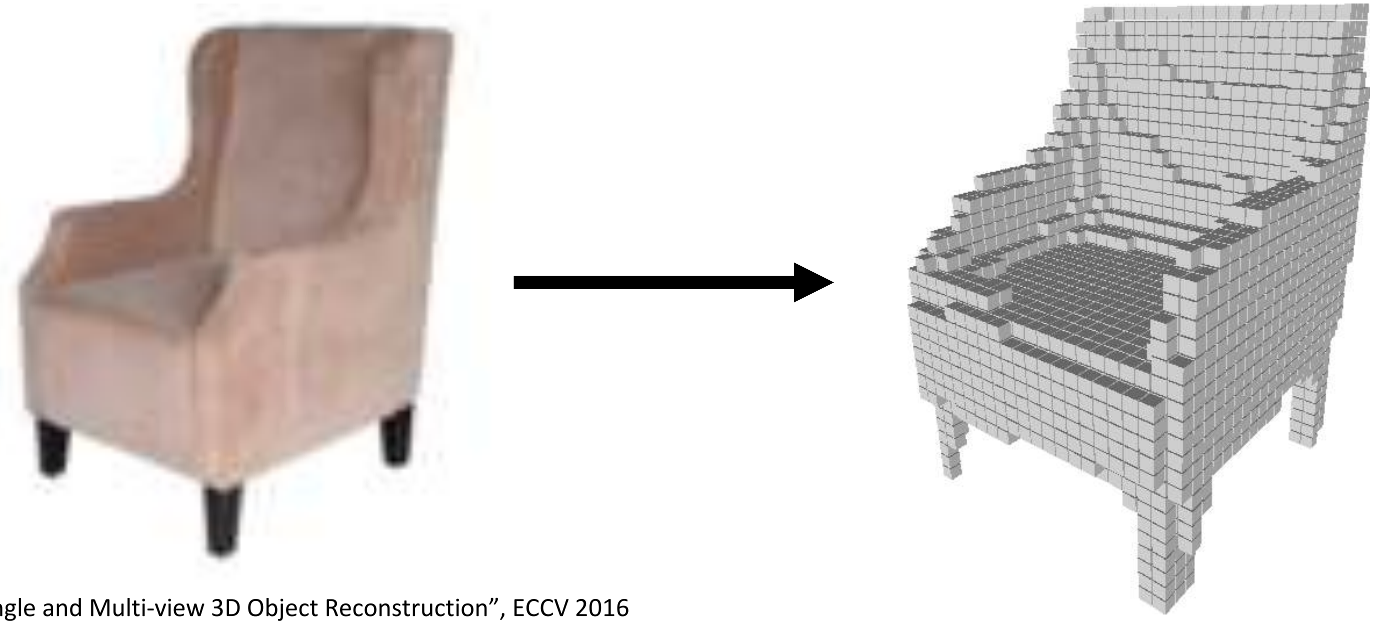
Eigen and Fergus, “Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture”, ICCV 2015

3D Shape Representations



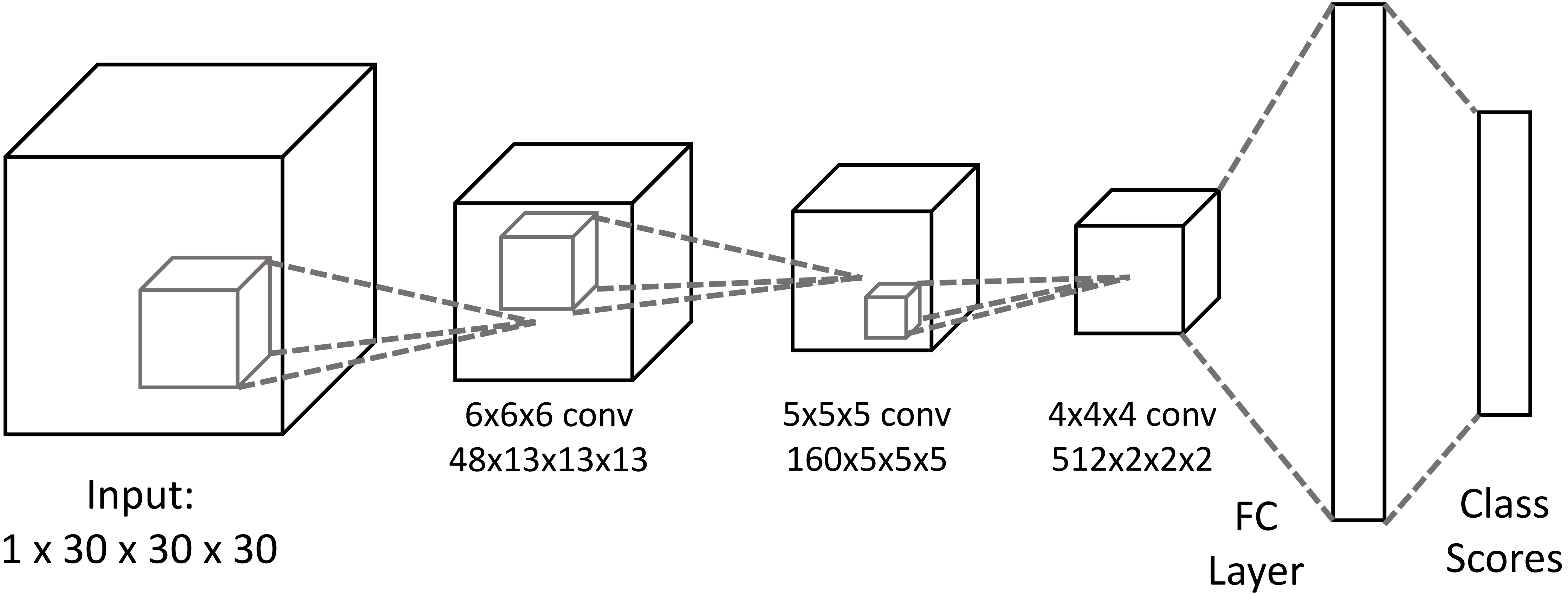
3D Shape Representations: Voxels

- Represent a shape with a V x V x V grid of occupancies
- Just like segmentation masks in Mask R-CNN, but in 3D!
- (+) Conceptually simple: just a 3D grid!
- (-) Need high spatial resolution to capture fine structures
- (-) Scaling to high resolutions is nontrivial!



Choy et al, "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction", ECCV 2016

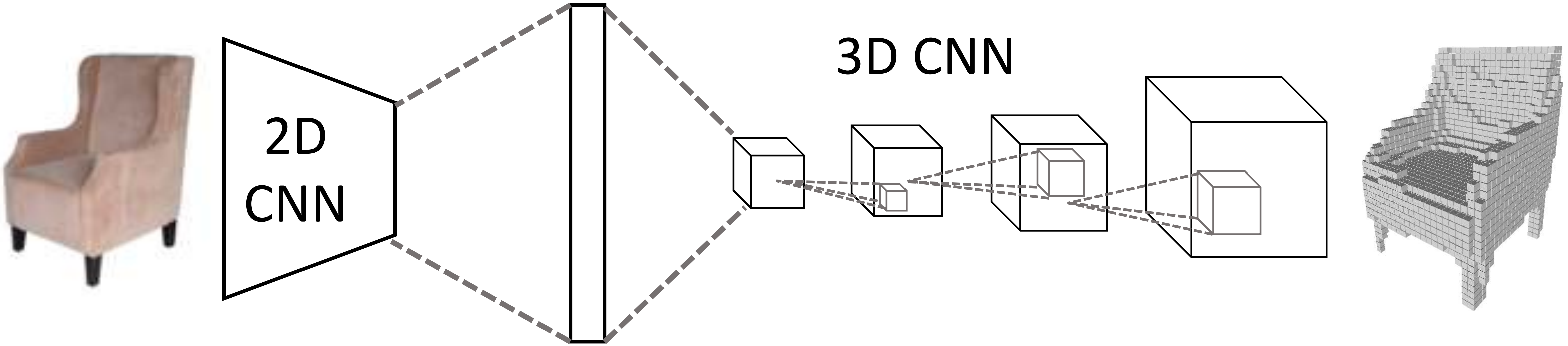
Processing Voxel Inputs: 3D Convolution



Train with classification loss

Wu et al, "3D ShapeNets: A Deep Representation for Volumetric Shapes", CVPR 2015

Generating Voxel Shapes: 3D Convolution



Input image:
3 x 112 x 112

2D Features:
C x H x W

3D Features:
C' x D' x H' x W'

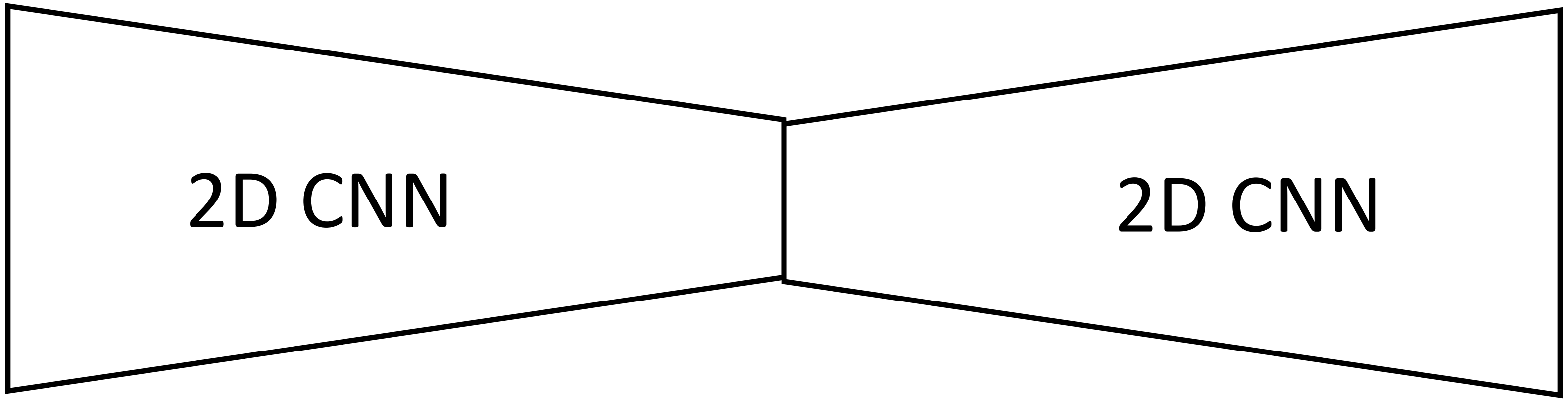
Voxels:
1 x V x V x V

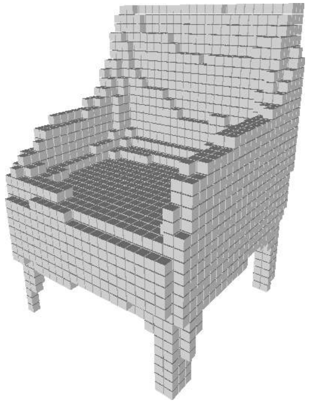
Train with per-voxel cross-entropy loss

Choy et al, "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction", ECCV 2016

Generating Voxel Shapes: ”Voxel Tubes”

Final conv layer: V filters
Interpret as a “tube” of
voxel scores





Input image:
3 x 112 x 112

2D Features:
C x H x W

3D Features:
C’ x D’ x H’ x W’

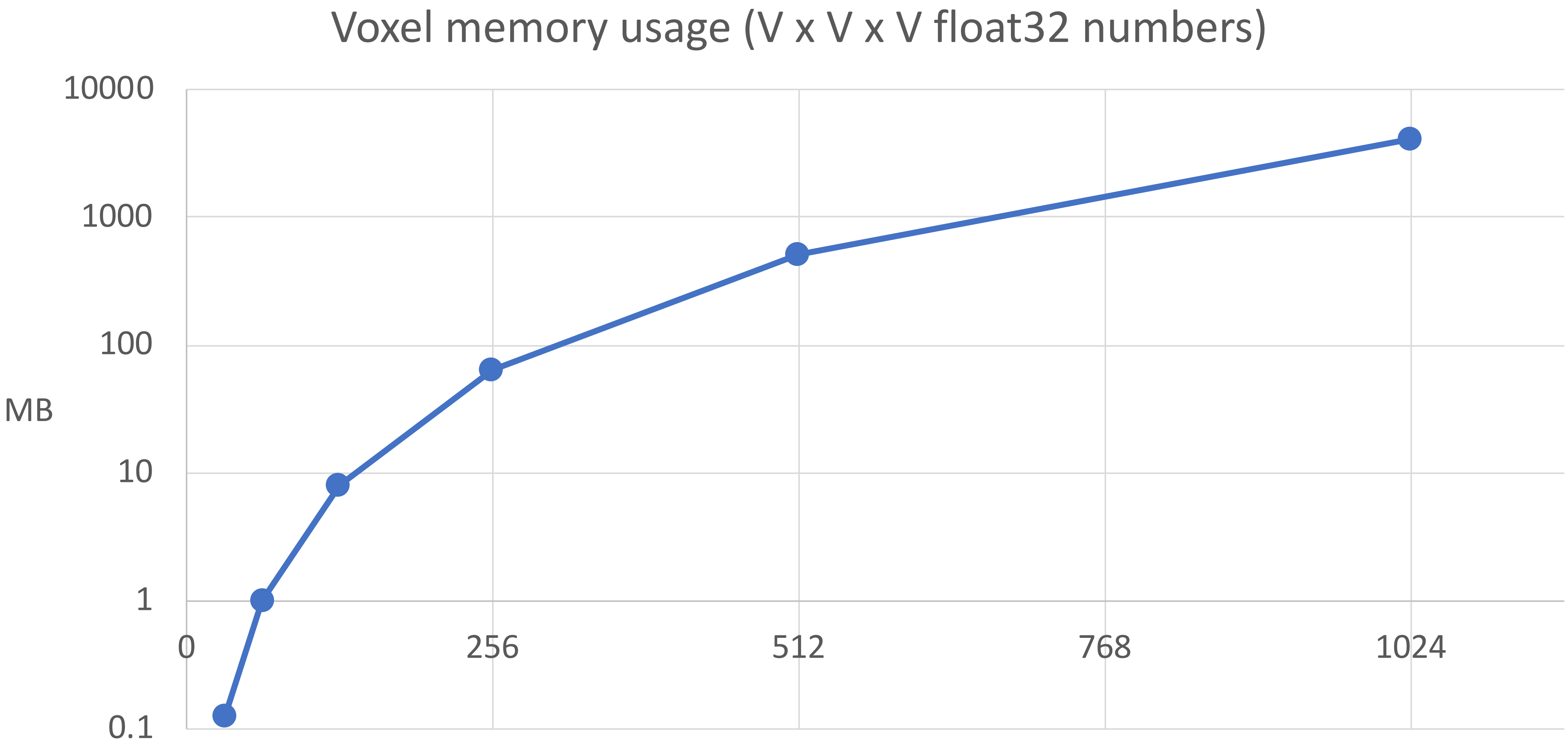
Voxels:
V x V x V

Train with per-voxel cross-entropy loss

Choy et al, “3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction”, ECCV 2016

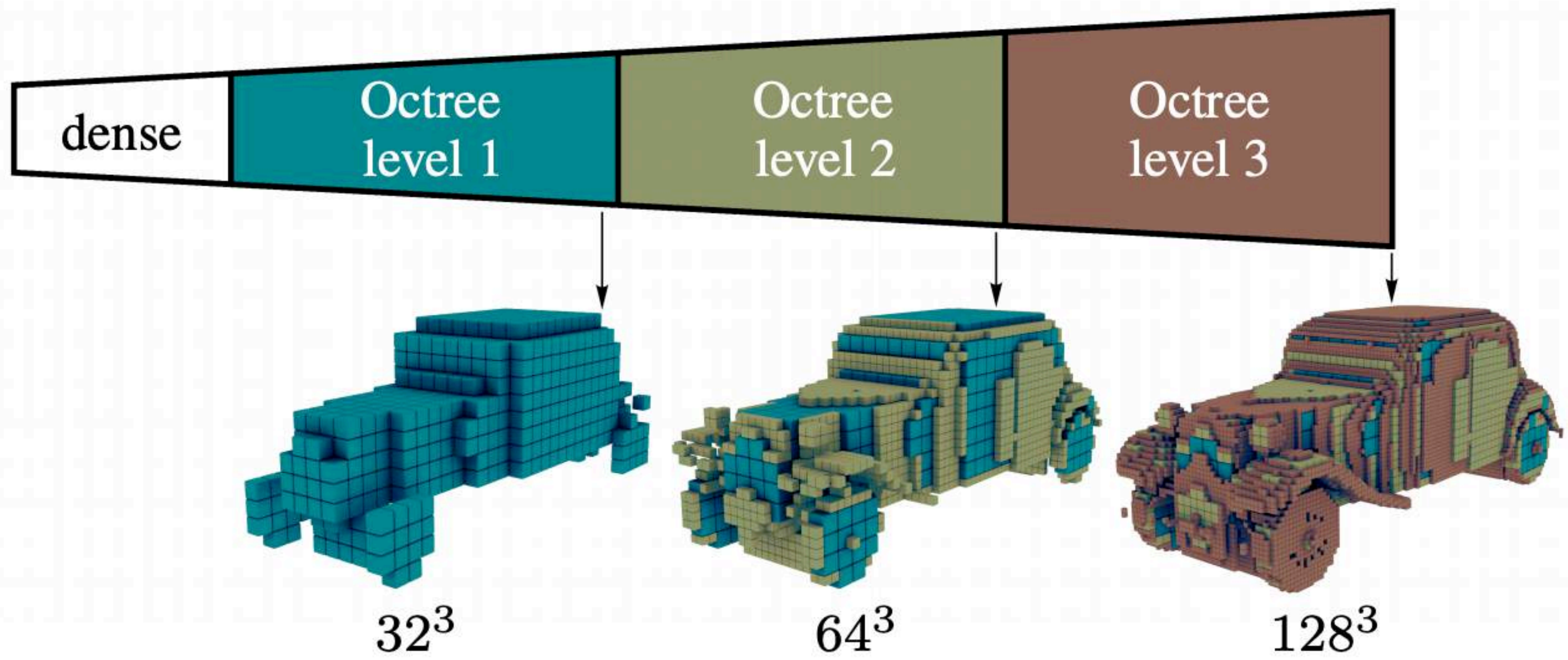
Voxel Problems: Memory Usage

Storing 1024^3 voxel grid takes 4GB of memory!



Scaling Voxels: Oct-Trees

Use voxel grids with heterogenous resolution!

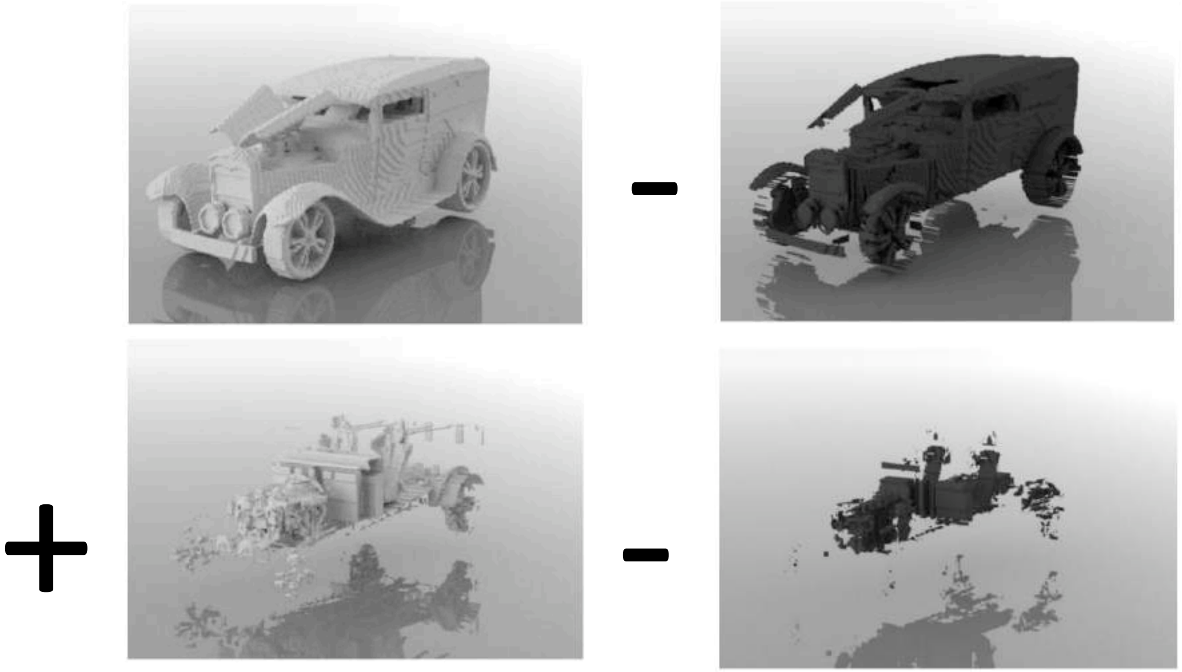


Tatarchenko et al, "Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs", ICCV 2017

Scaling Voxels: Nested Shape Layers

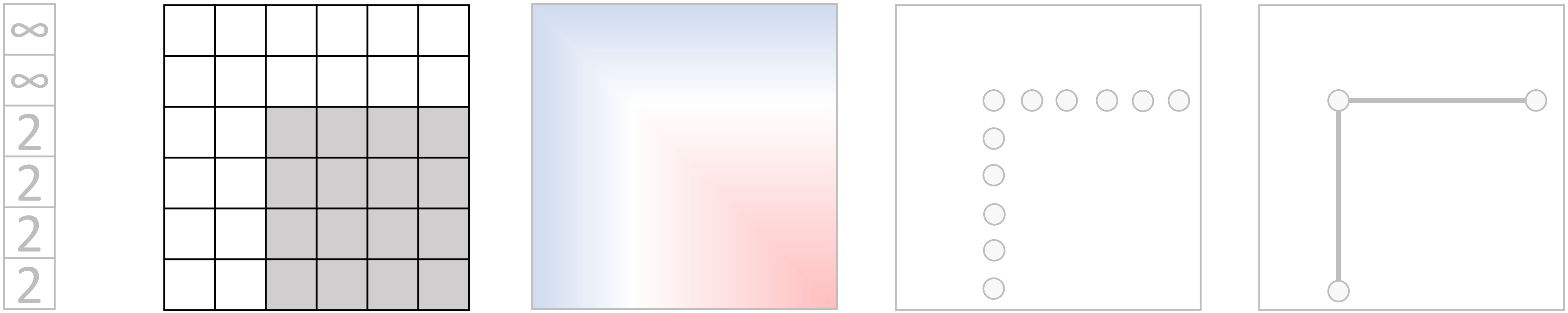
Predict shape as a composition of positive and negative spaces

=



Richter and Roth, “Matryoshka Networks: Predicting 3D Geometry via Nested Shape Layers”, CVPR 2018

3D Shape Representations



Depth Map

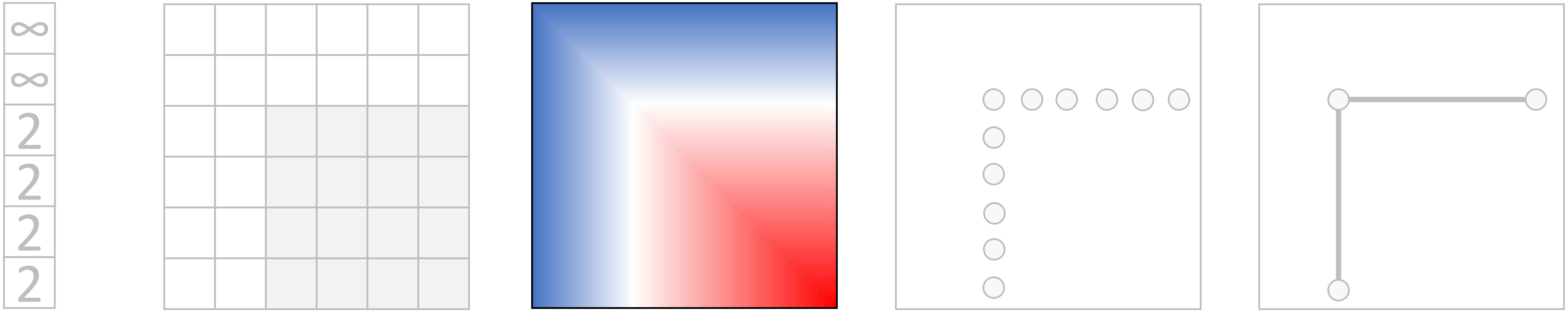
Voxel Grid

Implicit Surface

Pointcloud

Mesh

3D Shape Representations



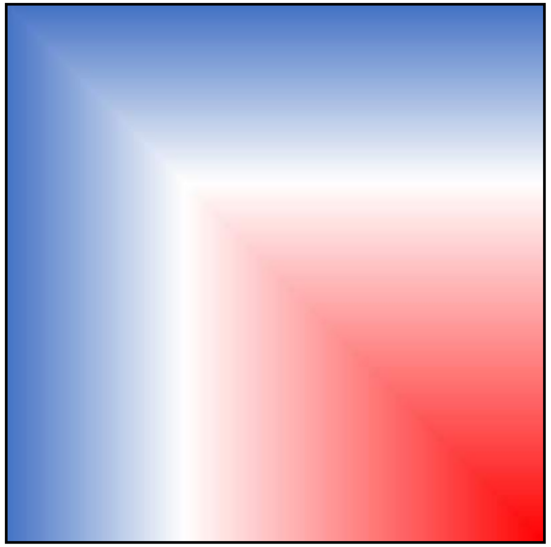
Depth Map | Voxel Grid | **Implicit Surface** | Pointcloud | Mesh

3D Shape Representations: Implicit Functions

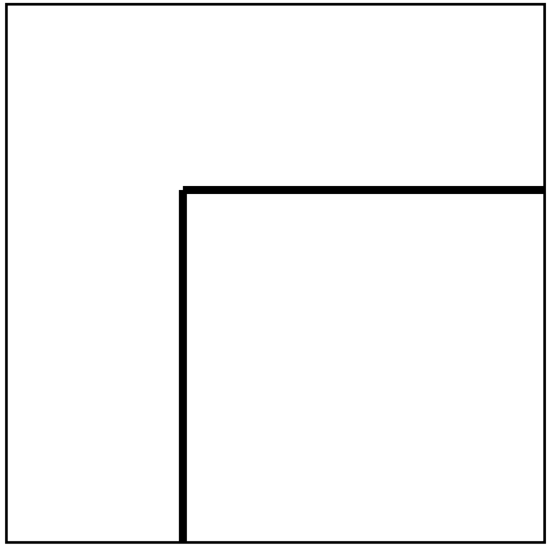
Learn a function to classify arbitrary 3D points as inside / outside the shape

$$o : \mathbb{R}^3 \rightarrow \{0, 1\}$$

The surface of the 3D object is the level set $\{x : o(x) = ½\}$



Implicit function



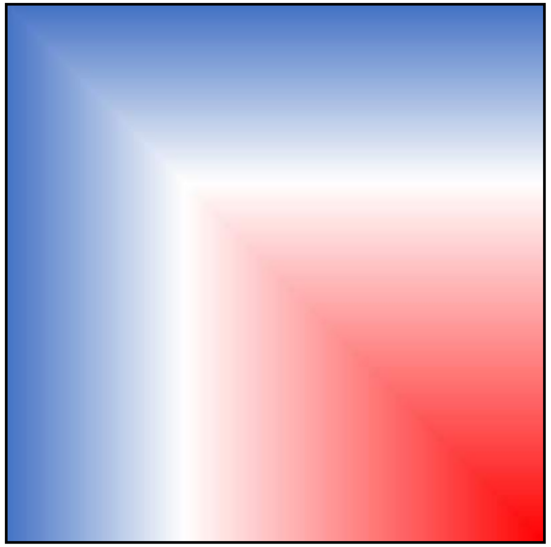
Explicit Shape

3D Shape Representations: Implicit Functions

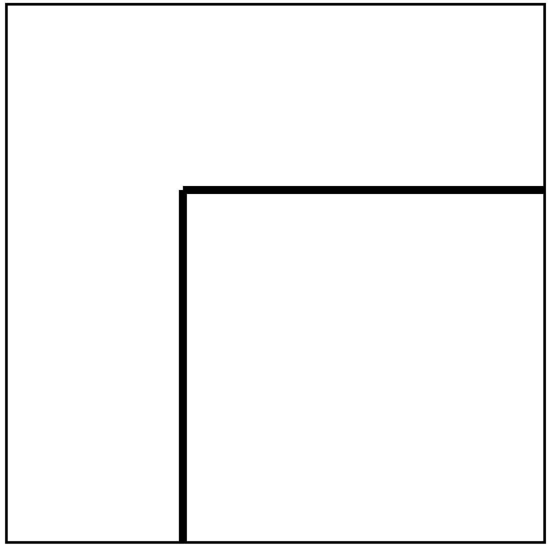
Learn a function to classify arbitrary 3D points as inside / outside the shape

$$o : \mathbb{R}^3 \to \{0, 1\}$$

The surface of the 3D object is the level set $\{x : o(x) = ½\}$



Implicit function



Explicit Shape

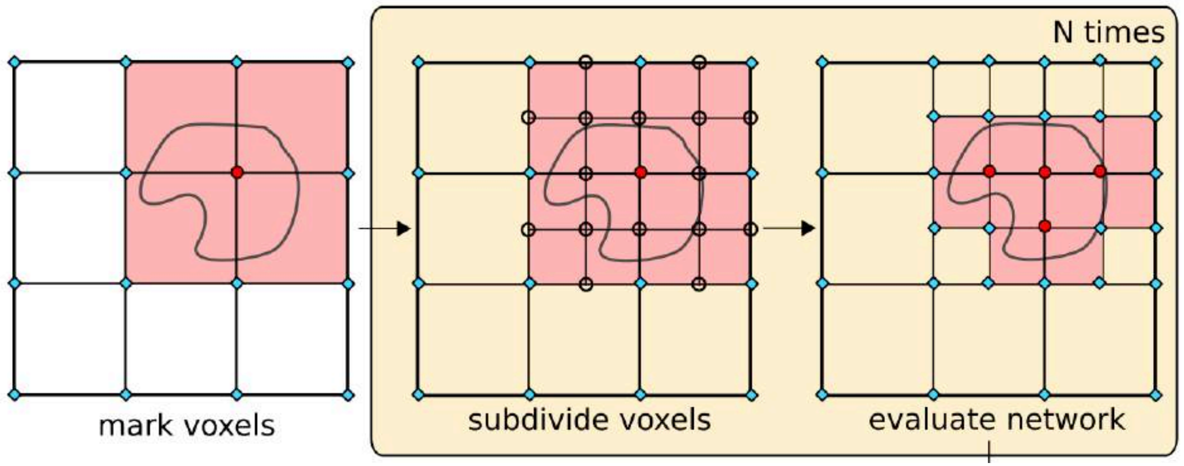
Same idea: **signed distance function (SDF)** gives the Euclidean distance to the surface of the shape; sign gives inside / outside

3D Shape Representations: Implicit Functions

Learn a function to classify arbitrary 3D points as inside / outside the shape

$$o : \mathbb{R}^3 \to \{0, 1\}$$

The surface of the 3D object is the level set $\{x : o(x) = \frac{1}{2}\}$



Allows for multiscale outputs like Oct-Trees

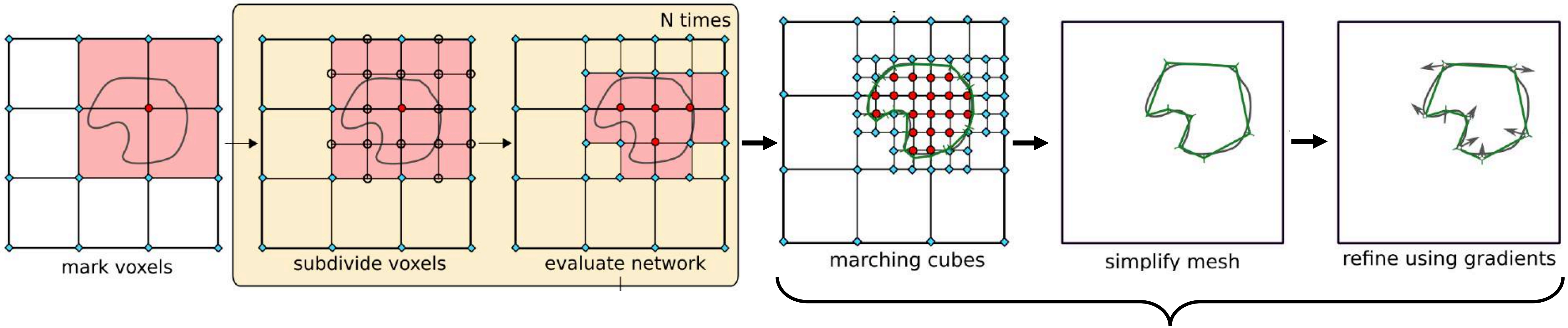
Mescheder et al, "Occupancy Networks: Learning 3D Reconstruction in Function Space", CVPR 2019

3D Shape Representations: Implicit Functions

Learn a function to classify arbitrary 3D points as inside / outside the shape

$$o : \mathbb{R}^3 \rightarrow \{0, 1\}$$

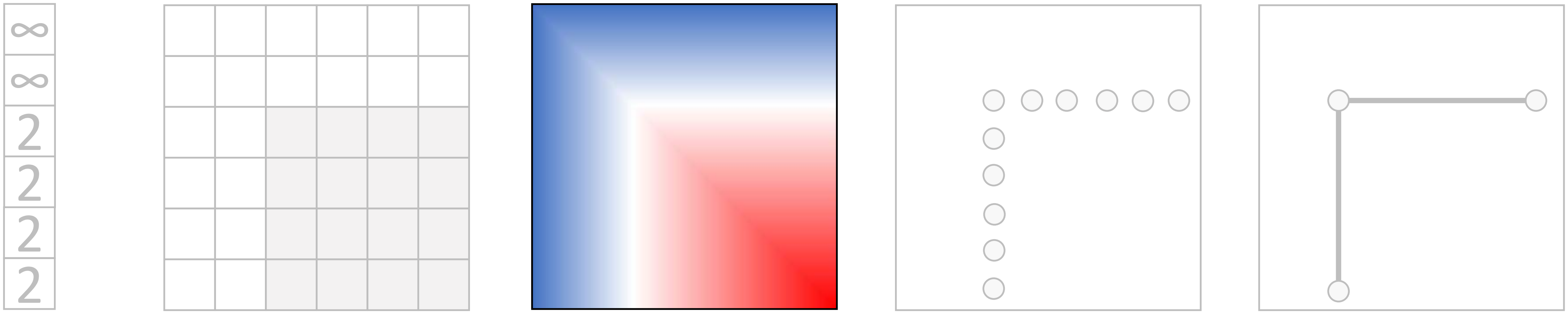
The surface of the 3D object is the level set $\{x : o(x) = ½\}$



Extracting explicit shape outputs requires post-processing

Mescheder et al, "Occupancy Networks: Learning 3D Reconstruction in Function Space", CVPR 2019

3D Shape Representations



Depth Map

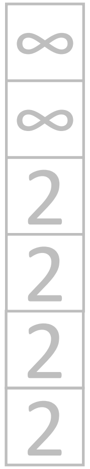
Voxel Grid

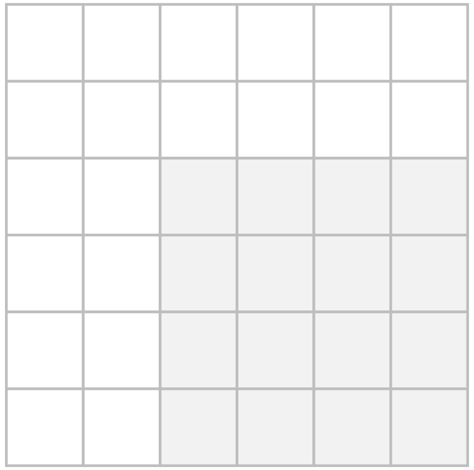
Implicit Surface

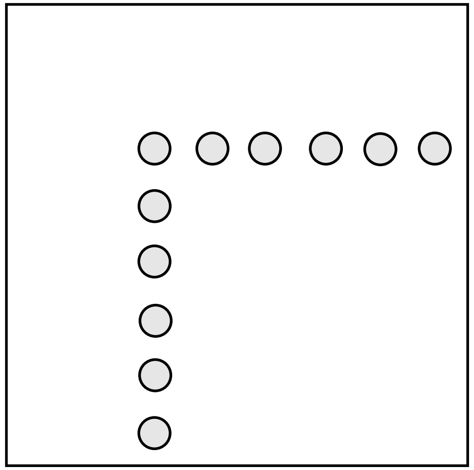
Pointcloud

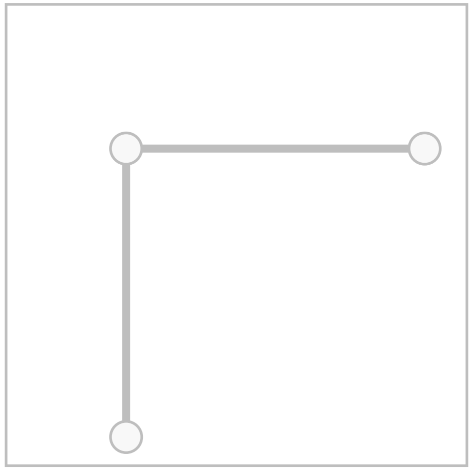
Mesh

3D Shape Representations









Depth Map

Voxel Grid

Implicit Surface

Pointcloud

Mesh

3D Shape Representations: Point Cloud

- Represent shape as a set of P points in 3D space
- (+) Can represent fine structures without huge numbers of points
- ( ) Requires new architecture, losses, etc
- (-) Doesn't explicitly represent the surface of the shape: extracting a mesh for rendering or other applications requires post-processing

Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

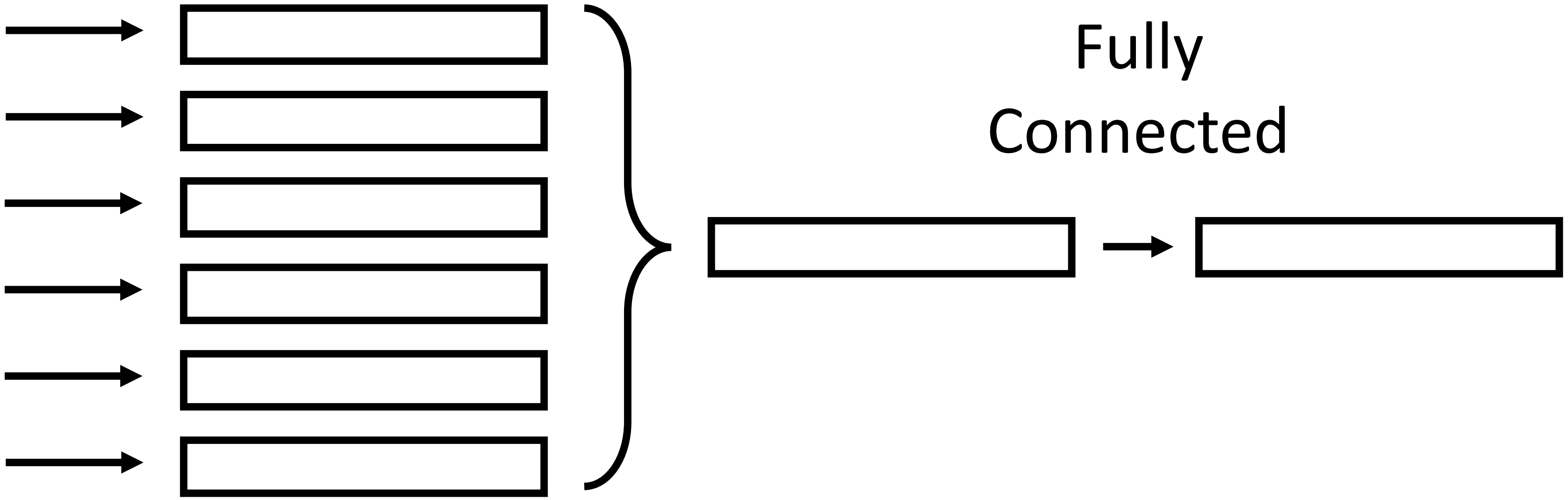
Processing Pointcloud Inputs: PointNet

Want to process pointclouds as **sets**: order should not matter

Run MLP on each point

Max-Pool

Fully Connected



Input pointcloud: P x 3

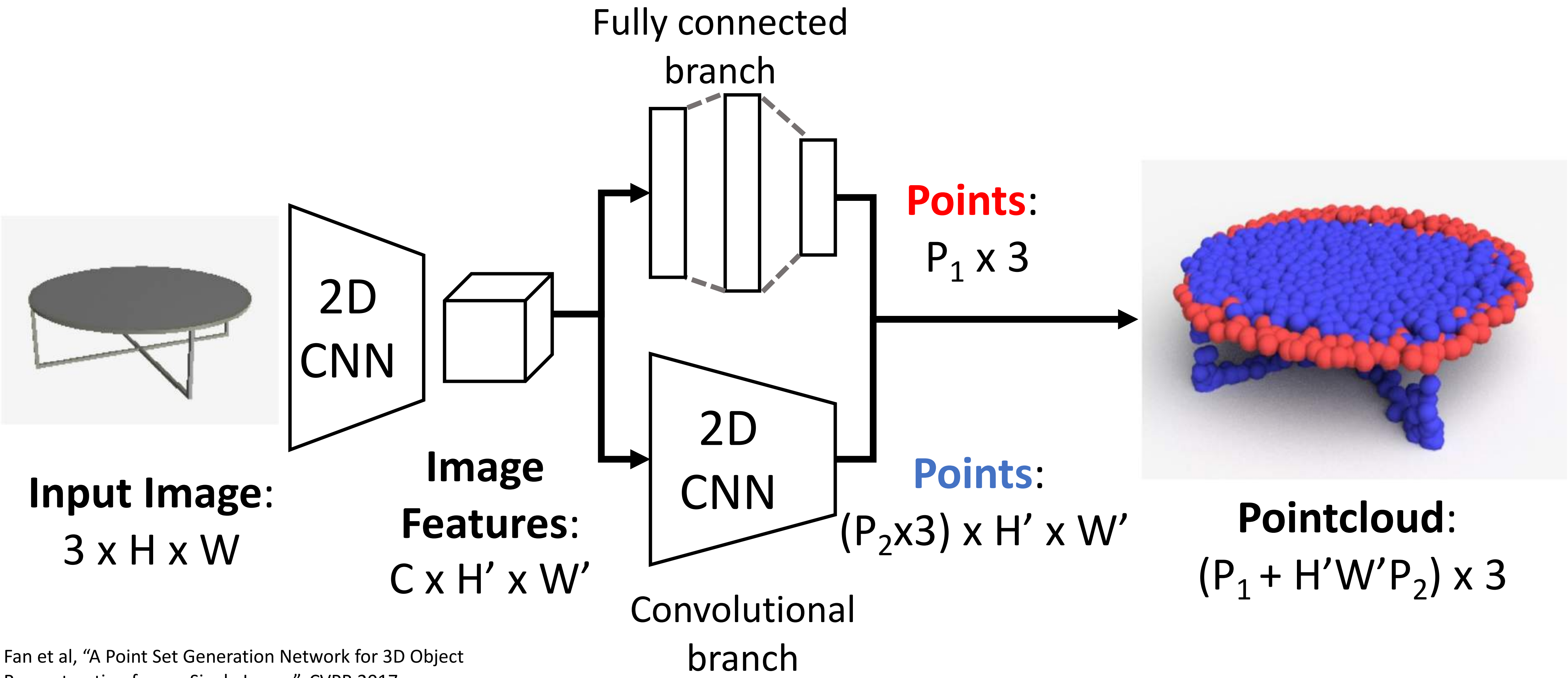
Point features: P x D

Pooled vector: D

Class score: C

Qi et al, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation", CVPR 2017
Qi et al, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space", NeurIPS 2017

Generating Pointcloud Outputs



Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

Predicting Point Clouds: Loss Function

We need a (differentiable) way to compare pointclouds **as sets**!

Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

Predicting Point Clouds: Loss Function

We need a (differentiable) way to compare pointclouds **as sets**!

Chamfer distance is the sum of L2 distance to each point's nearest neighbor in the other set

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

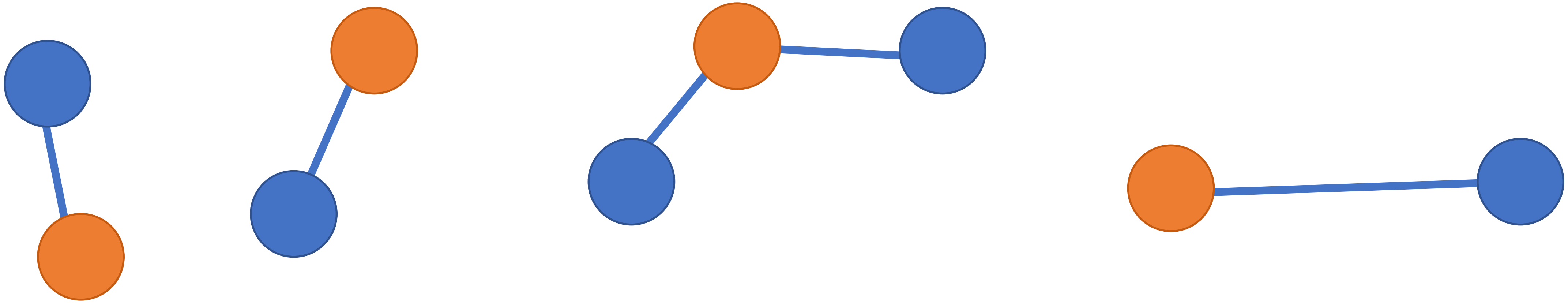
Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

Predicting Point Clouds: Loss Function

We need a (differentiable) way to compare pointclouds **as sets**!

Chamfer distance is the sum of L2 distance to each point's nearest neighbor in the other set

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



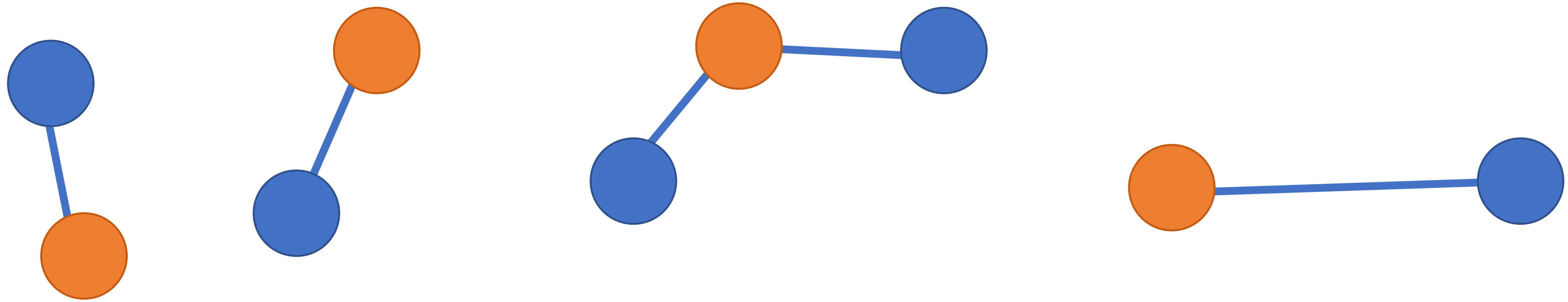
Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

Predicting Point Clouds: Loss Function

We need a (differentiable) way to compare pointclouds **as sets**!

Chamfer distance is the sum of L2 distance to each point's nearest neighbor in the other set

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



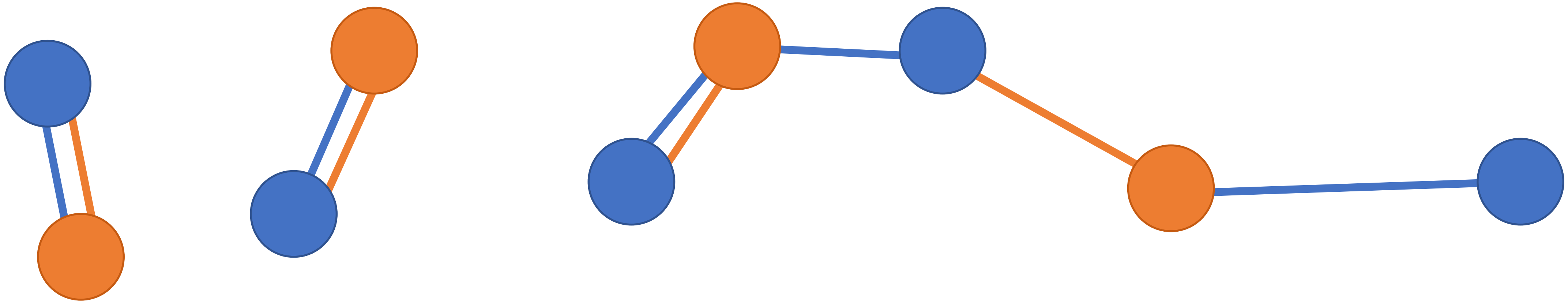
Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

Predicting Point Clouds: Loss Function

We need a (differentiable) way to compare pointclouds **as sets**!

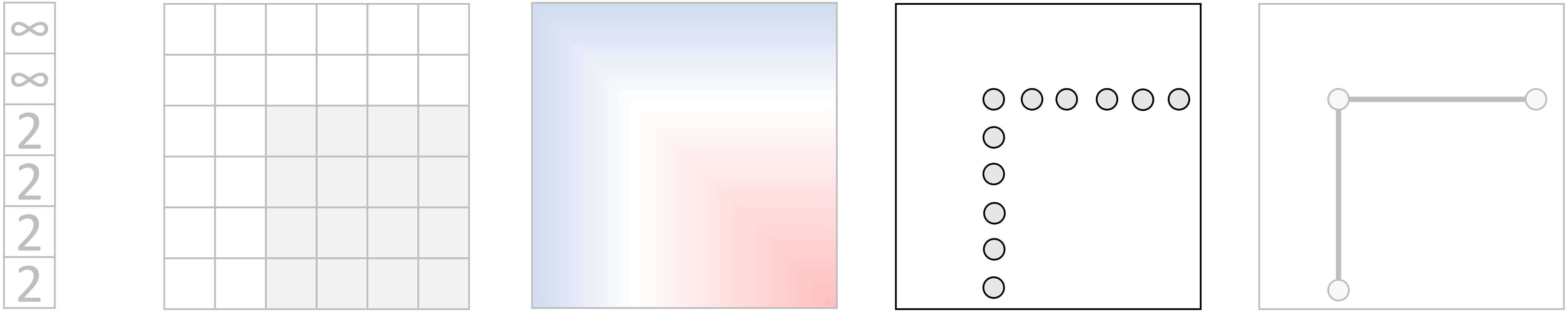
Chamfer distance is the sum of L2 distance to each point's nearest neighbor in the other set

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



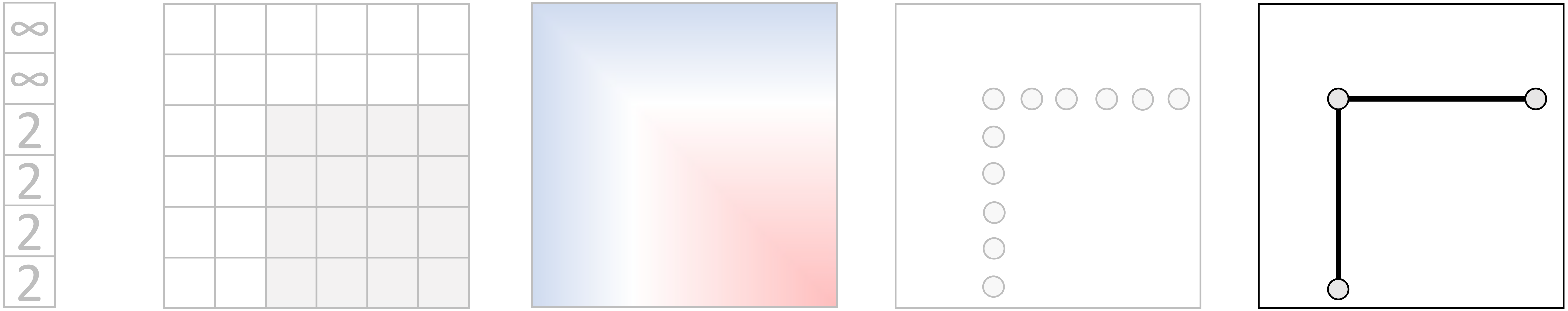
Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

3D Shape Representations



Depth Map | Voxel Grid | Implicit Surface | **Pointcloud** | Mesh

3D Shape Representations



Depth Map | Voxel Grid | Implicit Surface | Pointcloud | **Mesh**

3D Shape Representations: Triangle Mesh

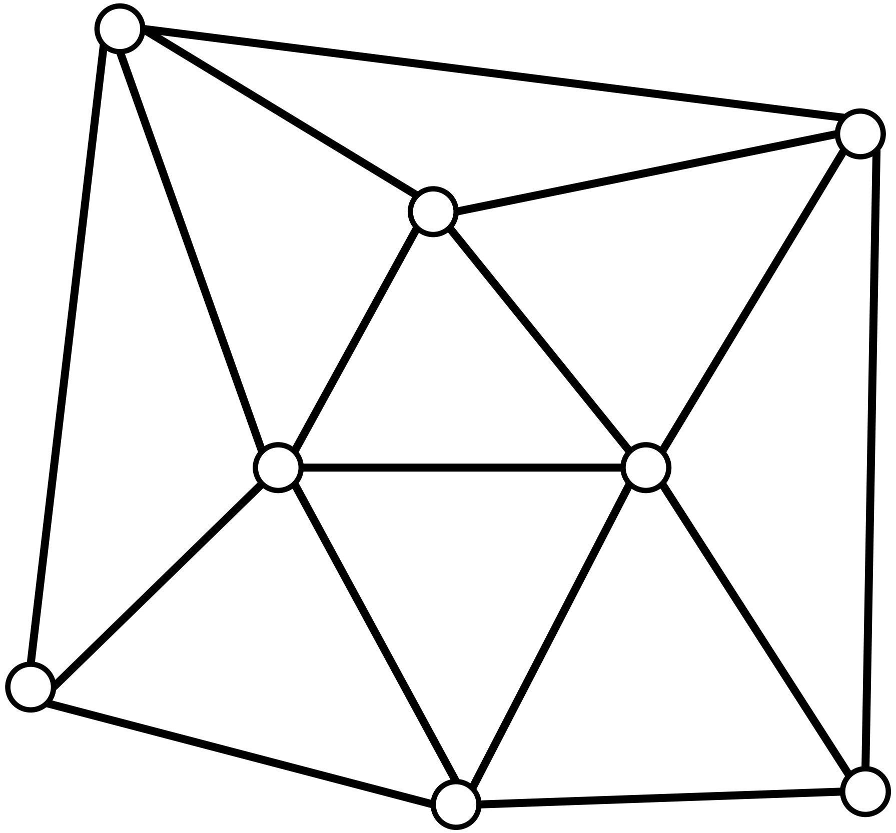
Represent a 3D shape as a set of triangles

Vertices: Set of V points in 3D space

Faces: Set of triangles over the vertices

(+) Standard representation for graphics

(+) Explicitly represents 3D shapes



3D Shape Representations: Triangle Mesh

Represent a 3D shape as a set of triangles

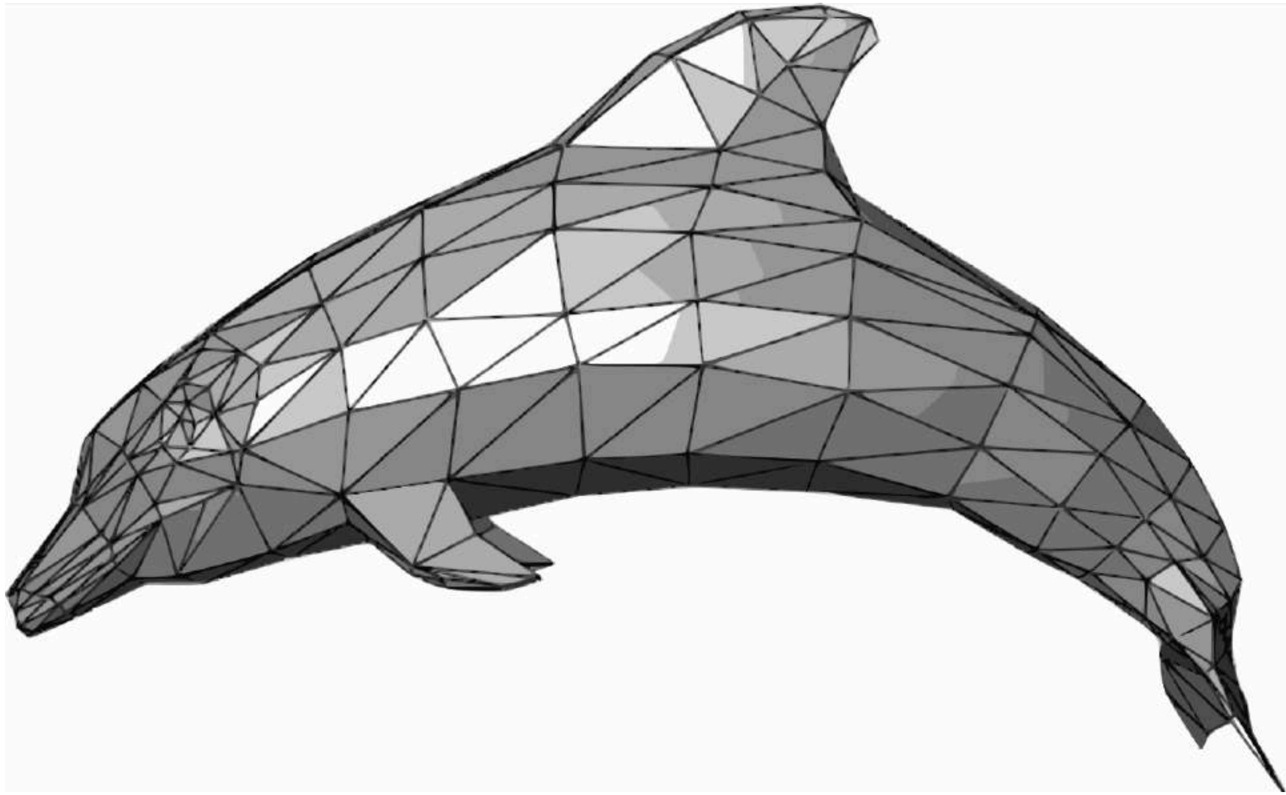
Vertices: Set of V points in 3D space

Faces: Set of triangles over the vertices

(+) Standard representation for graphics

(+) Explicitly represents 3D shapes

(+) Adaptive: Can represent flat surfaces very efficiently, can allocate more faces to areas with fine detail



Dolphin image is in the public domain

3D Shape Representations: Triangle Mesh

Represent a 3D shape as a set of triangles

Vertices: Set of V points in 3D space

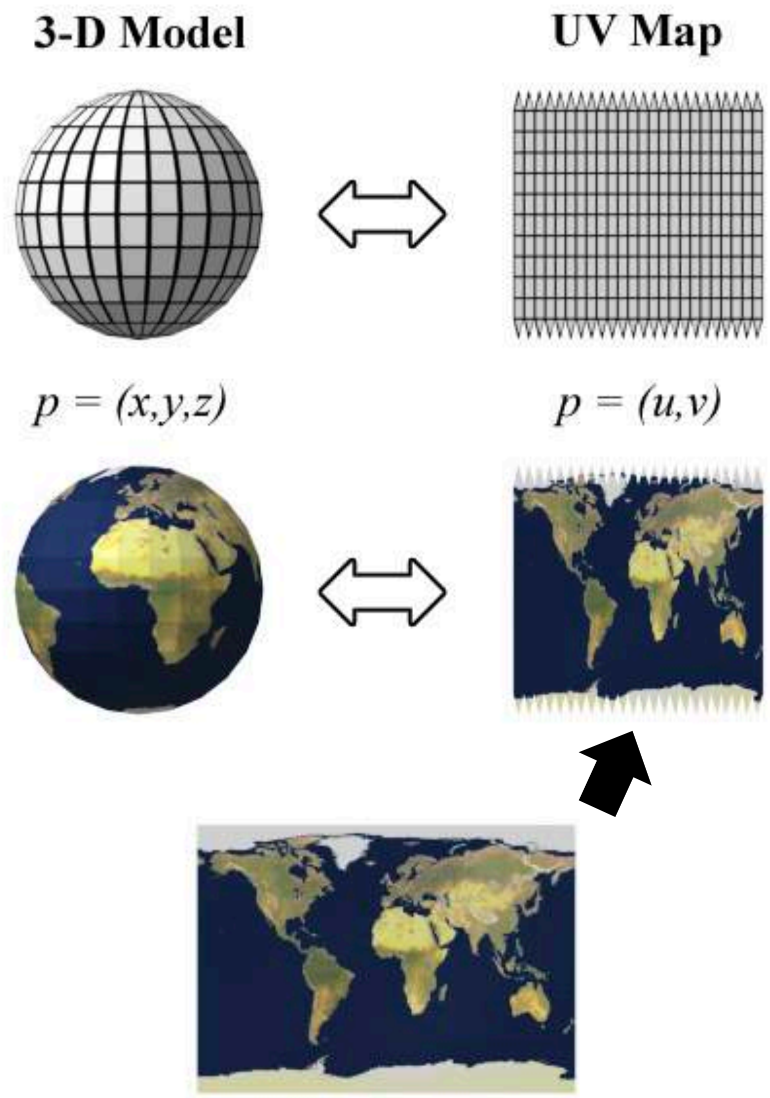
Faces: Set of triangles over the vertices

(+) Standard representation for graphics

(+) Explicitly represents 3D shapes

(+) Adaptive: Can represent flat surfaces very efficiently, can allocate more faces to areas with fine detail

(+) Can attach data on verts and interpolate over the whole surface: RGB colors, texture coordinates, normal vectors, etc.



3D Shape Representations: Triangle Mesh

Represent a 3D shape as a set of triangles

Vertices: Set of V points in 3D space

Faces: Set of triangles over the vertices

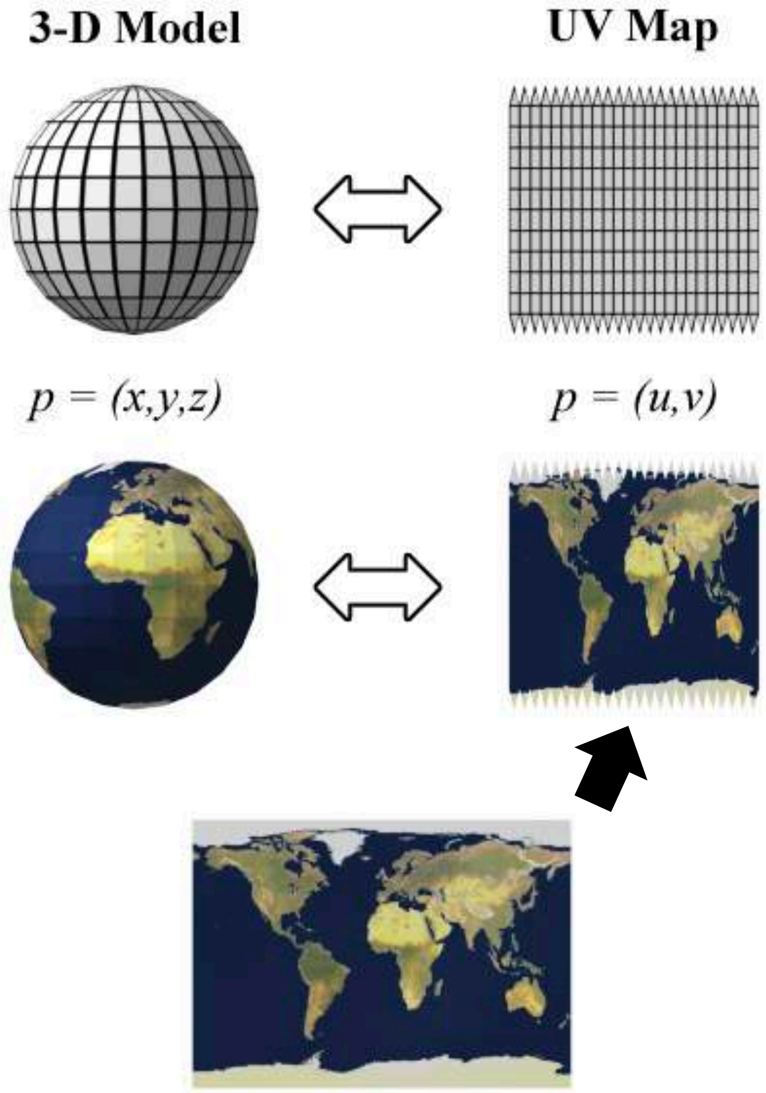
(+) Standard representation for graphics

(+) Explicitly represents 3D shapes

(+) Adaptive: Can represent flat surfaces very efficiently, can allocate more faces to areas with fine detail

(+) Can attach data on verts and interpolate over the whole surface: RGB colors, texture coordinates, normal vectors, etc.

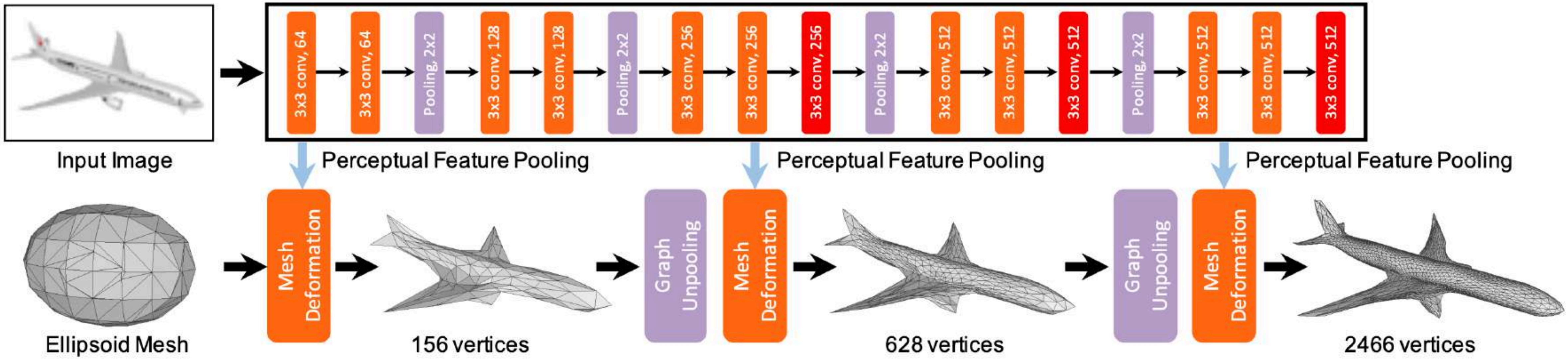
(-) Nontrivial to process with neural nets!



Predicting Meshes: Pixel2Mesh

Input: Single RGB Image of an object

Output: Triangle mesh for the object



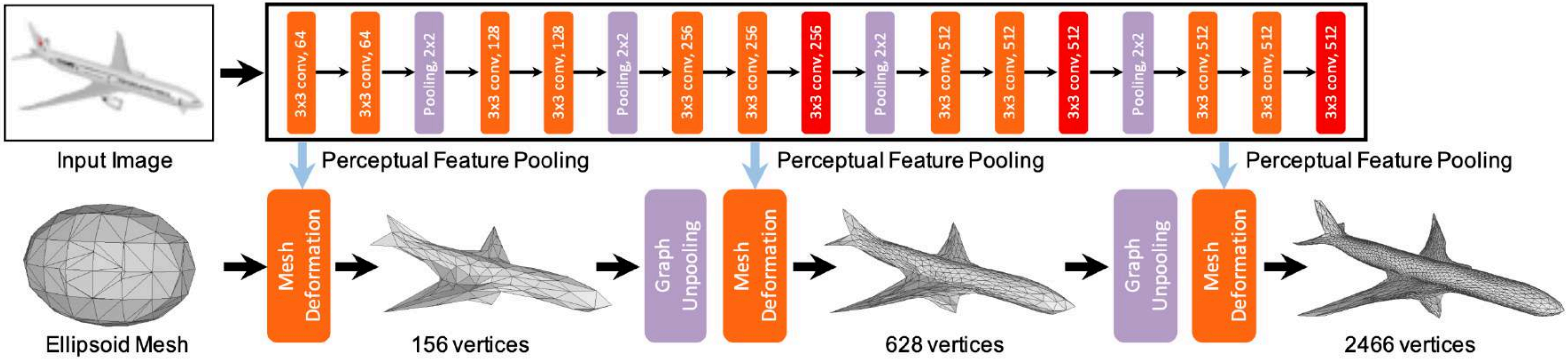
Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

Predicting Meshes: Pixel2Mesh

Input: Single RGB Image of an object

Key ideas:
Iterative Refinement
Graph Convolution
Vertex Aligned-Features
Chamfer Loss Function

Output: Triangle mesh for the object



Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

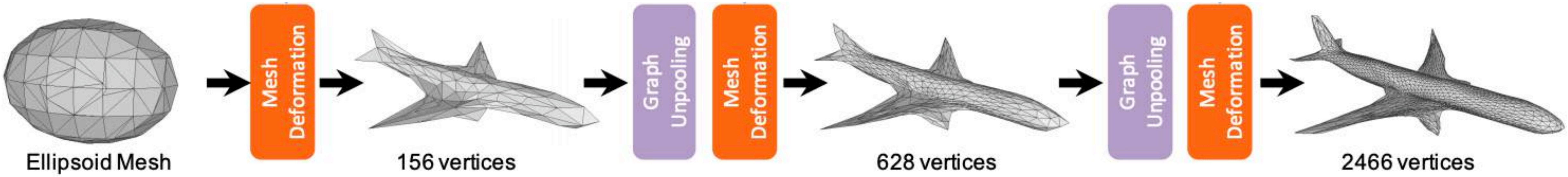
Predicting Triangle Meshes: Iterative Refinement

Idea #1: Iterative mesh refinement

Start from initial ellipsoid mesh

Network predicts offsets for each vertex

Repeat.



Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

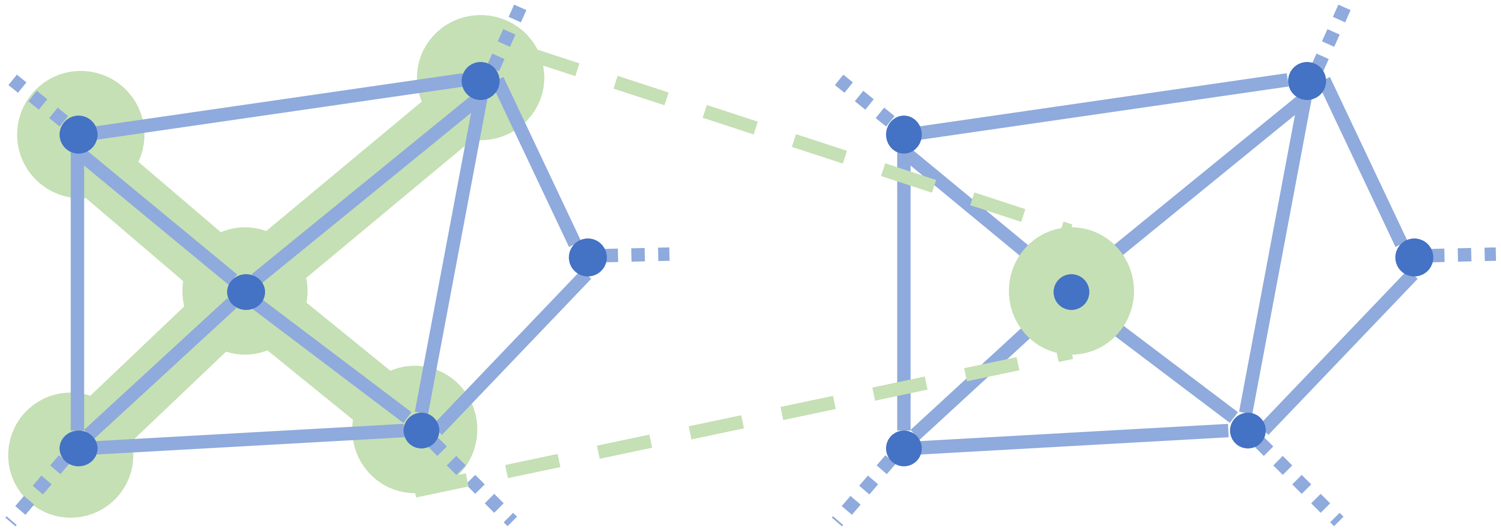
Predicting Triangle Meshes: Graph Convolution

$$f'_i = W_0 f_i + \sum_{j \in \mathcal{N}(i)} W_1 f_j$$

Vertex v_i has feature f_i

New feature f'_i for vertex vi depends on feature of neighboring vertices N(i)

Use same weights W0 and W1 to compute all outputs

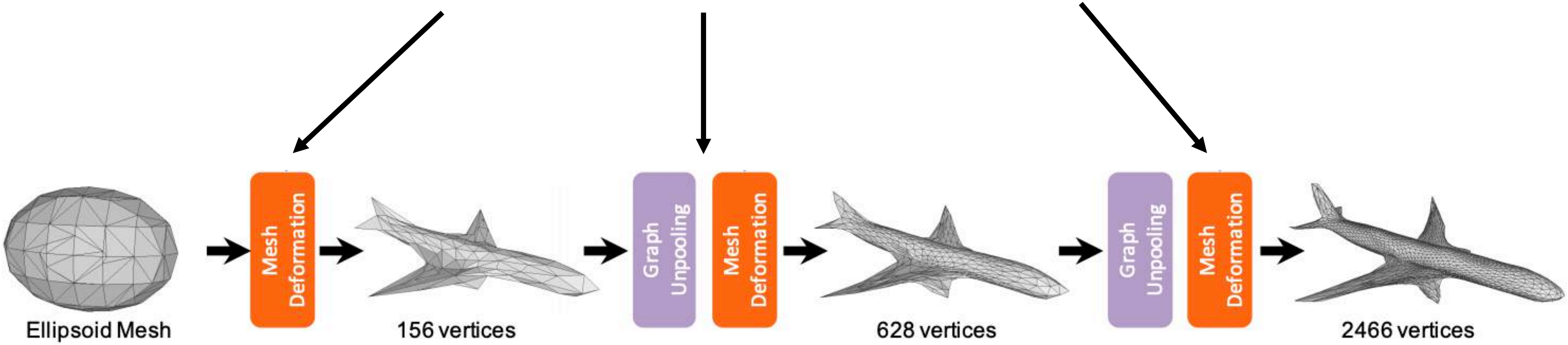


Input: Graph with a feature vector at each vertex

Output: New feature vector for each vertex

Predicting Triangle Meshes: Graph Convolution

Each of these blocks consists of a stack of **graph convolution layers** operating on edges of the mesh

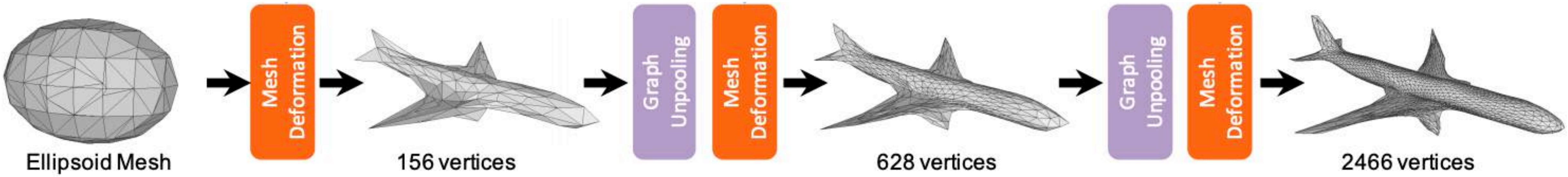


Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

Predicting Triangle Meshes: Graph Convolution

Each of these blocks consists of a stack of **graph convolution layers** operating on edges of the mesh

Problem: How to incorporate image features?



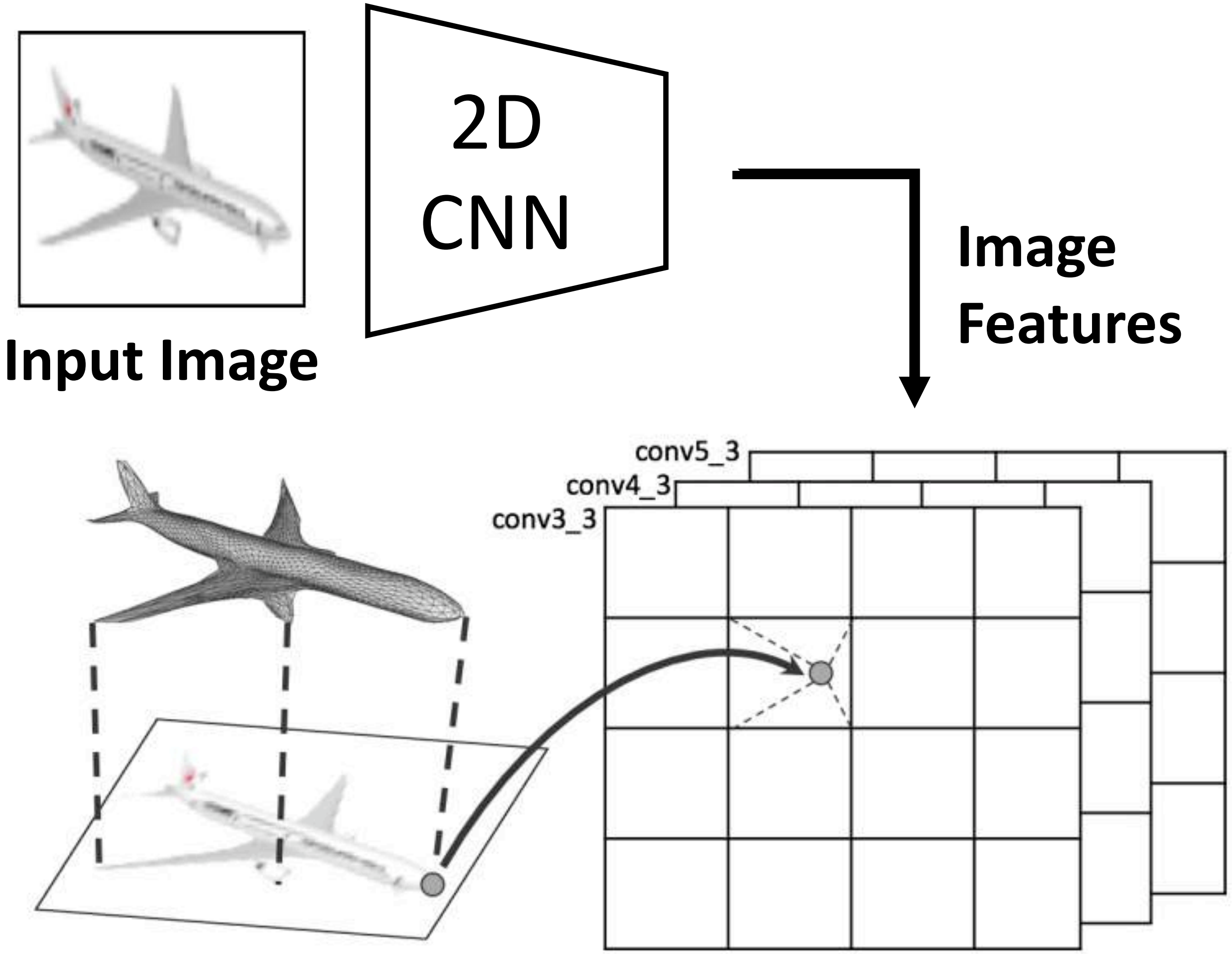
Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

Predicting Triangle Meshes: Vertex-Aligned Features

Idea #2: Aligned vertex features

For each vertex of the mesh:

- Use camera information to project onto image plane
- Use bilinear interpolation to sample a CNN feature



Wang et al, “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”, ECCV 2018

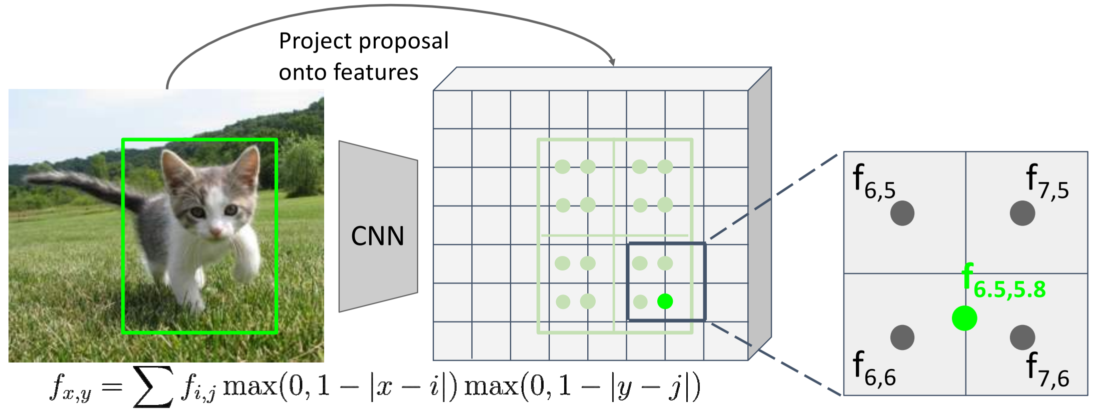
Predicting Triangle Meshes: Vertex-Aligned Features

Idea #2: Aligned vertex features

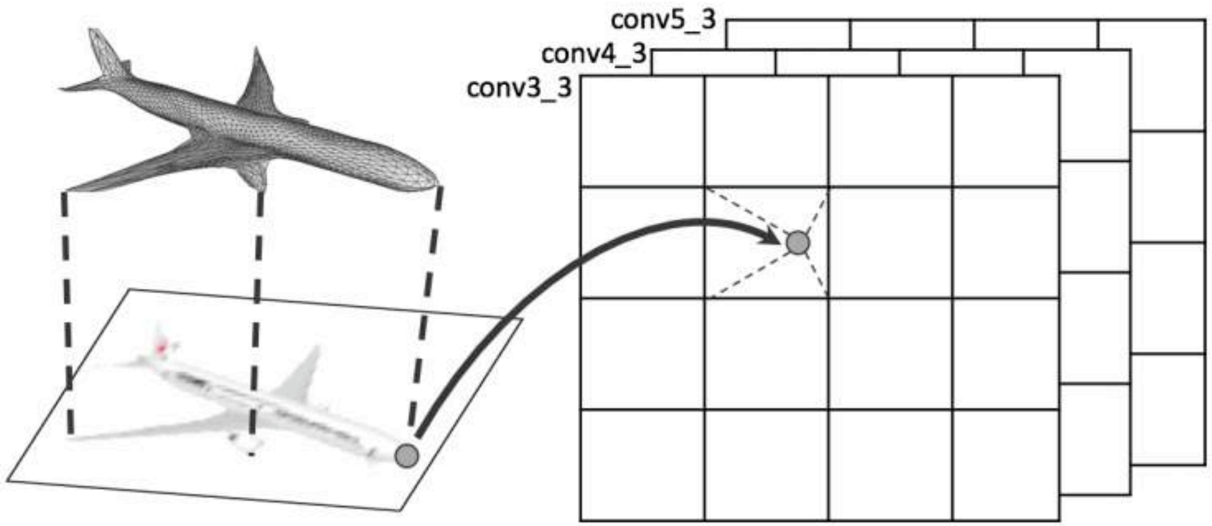
For each vertex of the mesh:

- Use camera information to project onto image plane
- Use bilinear interpolation to sample a CNN feature

Similar to RoI-Align operation from last time: maintains alignment between input image and feature vectors



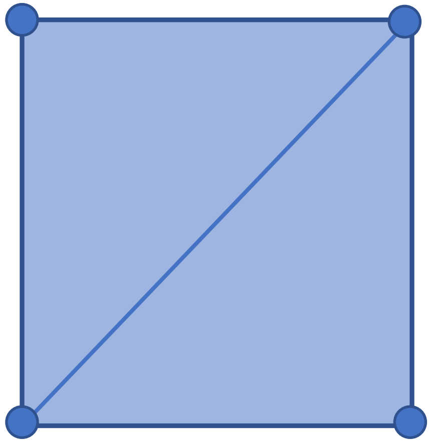
$$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1-|x-i|) \max(0, 1-|y-j|)$$



Wang et al, “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”, ECCV 2018

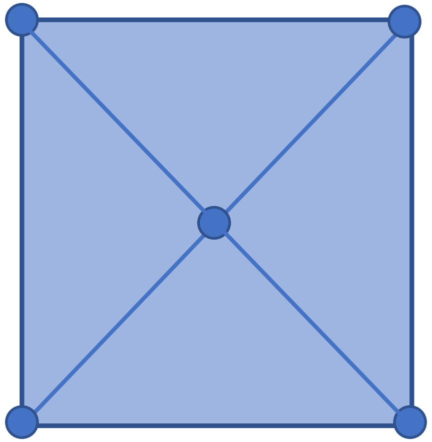
Predicting Meshes: Loss Function

The same shape can be represented with different meshes – how can we define a loss between predicted and ground-truth mesh?



Prediction

vs



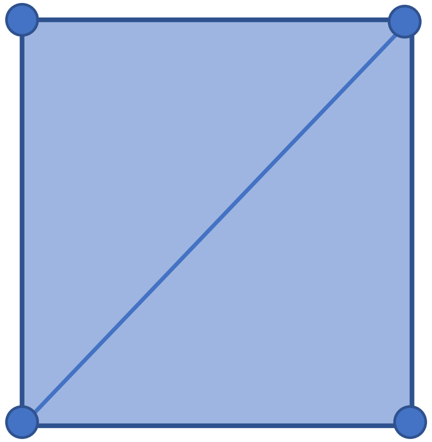
Ground-Truth

Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

Predicting Meshes: Loss Function

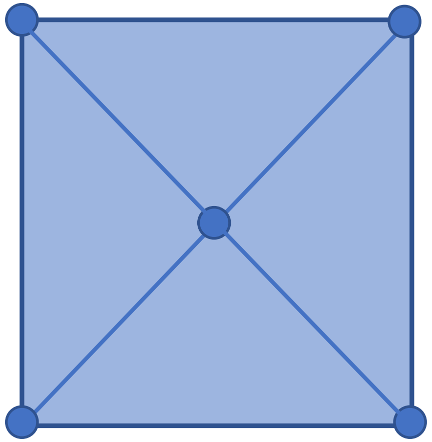
The same shape can be represented with different meshes – how can we define a loss between predicted and ground-truth mesh?

Idea: Convert meshes to pointclouds, then compute loss



Prediction

vs



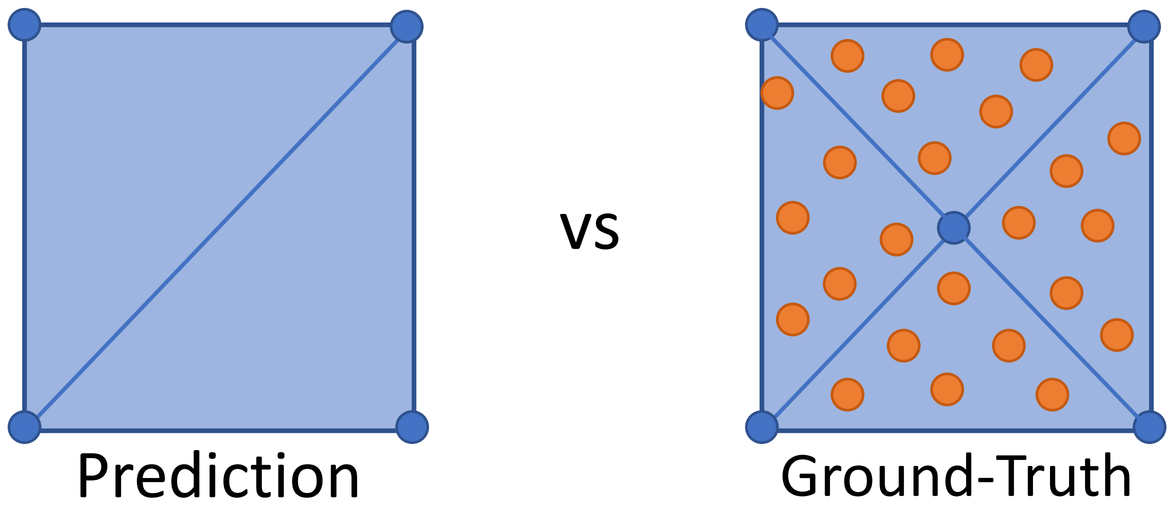
Ground-Truth

Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

Predicting Meshes: Loss Function

The same shape can be represented with different meshes – how can we define a loss between predicted and ground-truth mesh?

Idea: Convert meshes to pointclouds, then compute loss



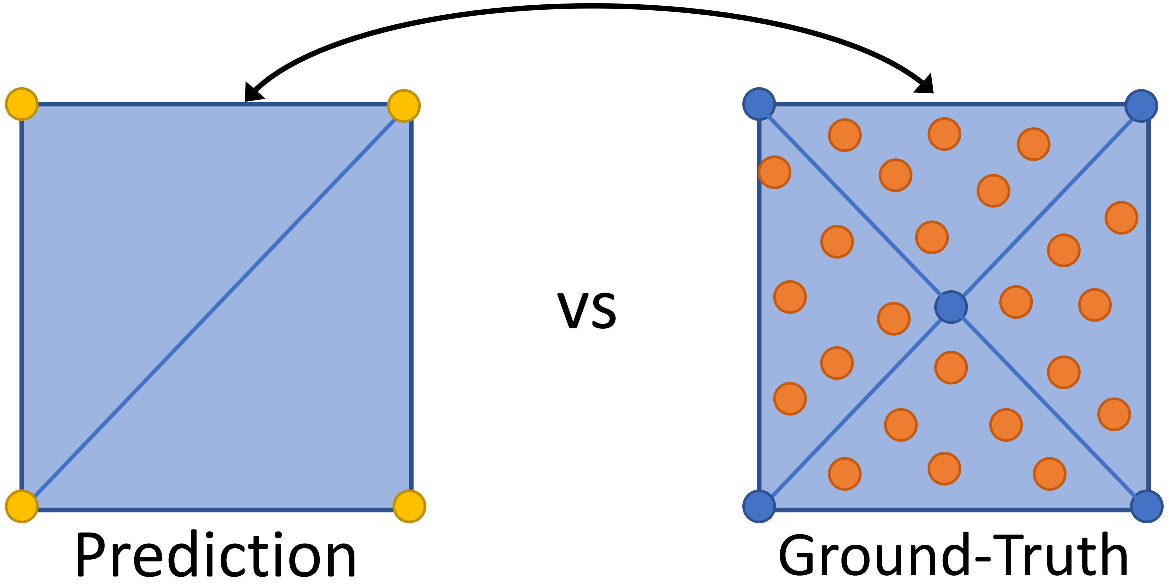
Sample points from the surface of the ground-truth mesh (offline)

Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

Predicting Meshes: Loss Function

The same shape can be represented with different meshes – how can we define a loss between predicted and ground-truth mesh?

Loss = Chamfer distance between **predicted verts** and **ground-truth samples**



Sample points from the surface of the ground-truth mesh (offline)

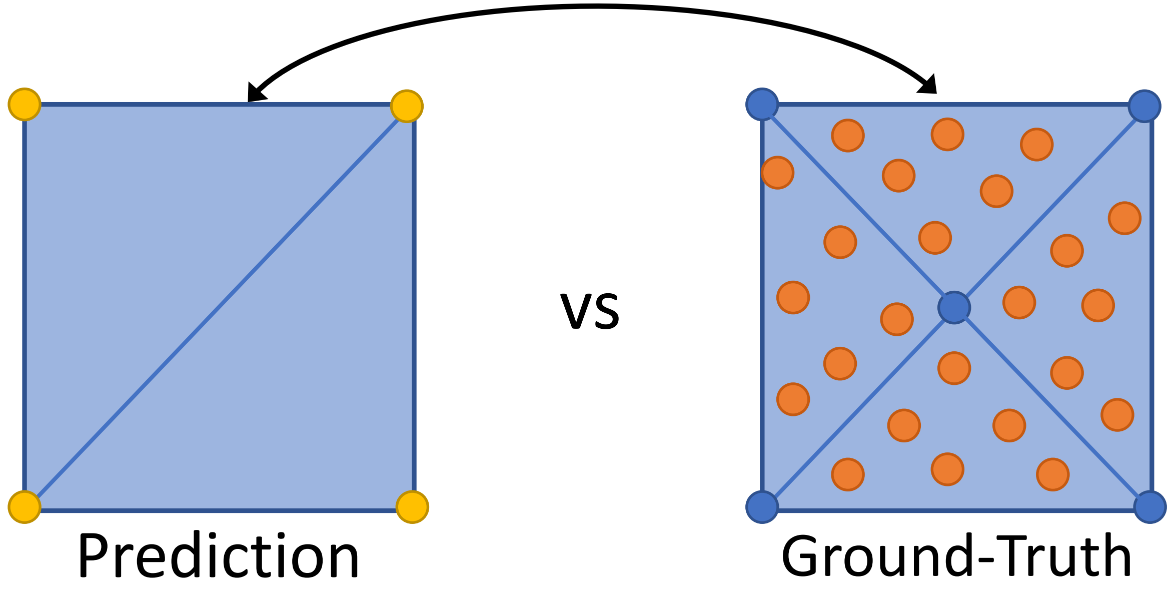
Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

Predicting Meshes: Loss Function

The same shape can be represented with different meshes – how can we define a loss between predicted and ground-truth mesh?

Loss = Chamfer distance between **predicted verts** and **ground-truth samples**

Problem: Doesn't take the interior of predicted faces into account!



Sample points from the surface of the ground-truth mesh (offline)

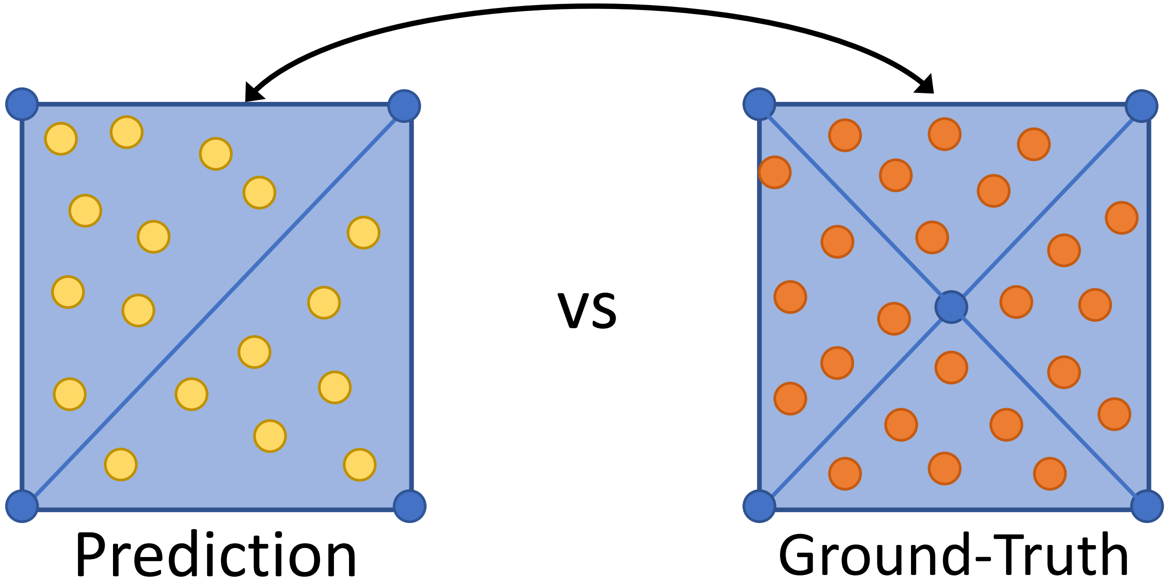
Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

Predicting Meshes: Loss Function

The same shape can be represented with different meshes – how can we define a loss between predicted and ground-truth mesh?

Loss = Chamfer distance between **predicted samples** and **ground-truth samples**

Sample points from the surface of the predicted mesh (online!)



Sample points from the surface of the ground-truth mesh (offline)

Smith et al, "GEOMetrics: Exploiting Geometric Structure for Graph-Encoded Objects", ICML 2019

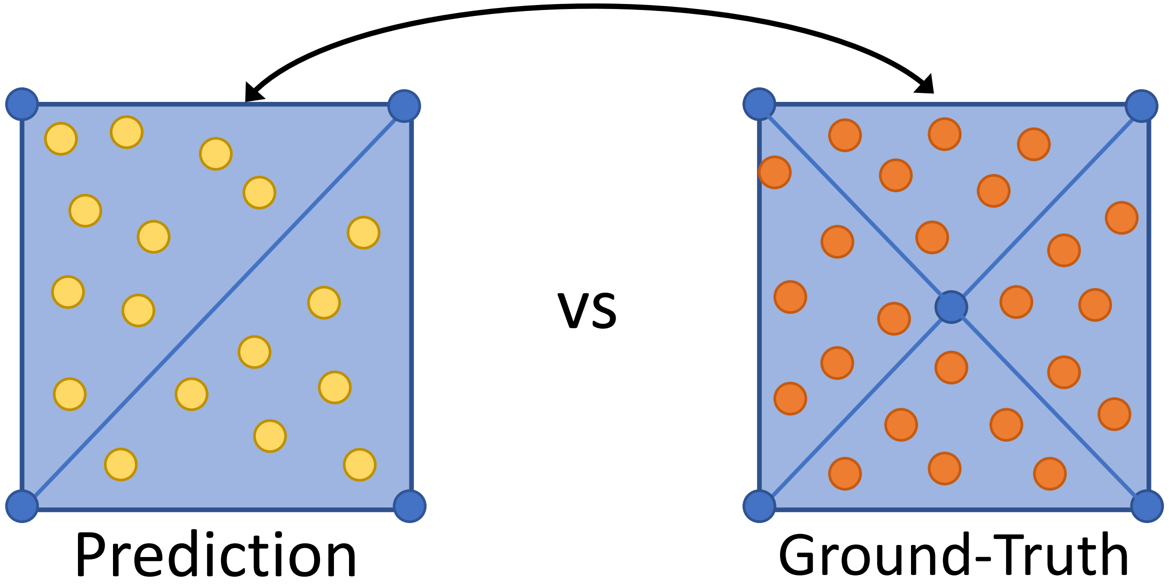
Predicting Meshes: Loss Function

Problem: Need to sample online! Must be efficient!

Problem: Need to backprop through sampling!

Loss = Chamfer distance between **predicted samples** and **ground-truth samples**

Sample points from the surface of the predicted mesh (online!)



Sample points from the surface of the ground-truth mesh (offline)

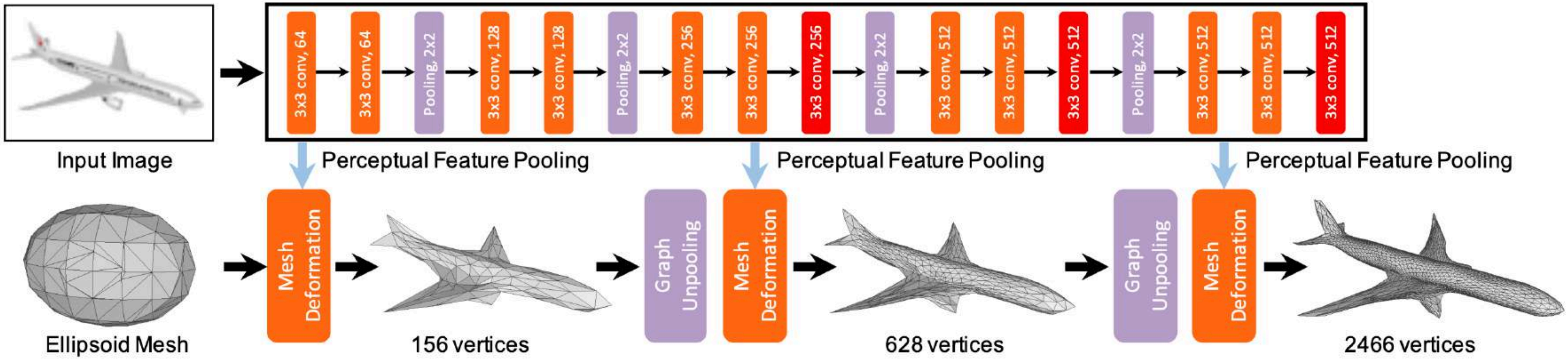
Smith et al, “GEOMetrics: Exploiting Geometric Structure for Graph-Encoded Objects”, ICML 2019

Predicting Meshes: Pixel2Mesh

Input: Single RGB Image of an object

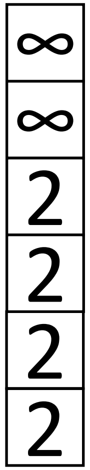
Key ideas:
Iterative Refinement
Graph Convolution
Vertex Aligned-Features
Chamfer Loss Function

Output: Triangle mesh for the object

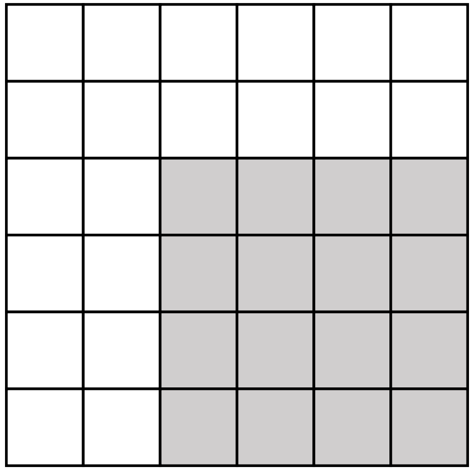


Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

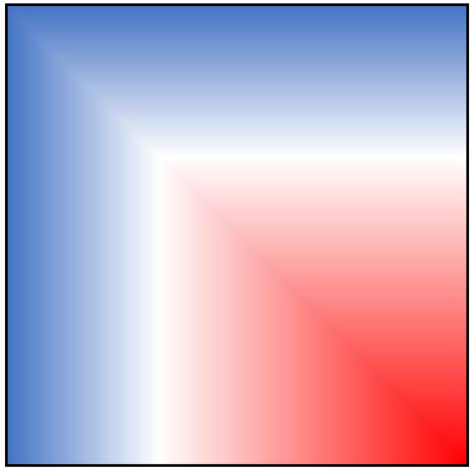
3D Shape Representations



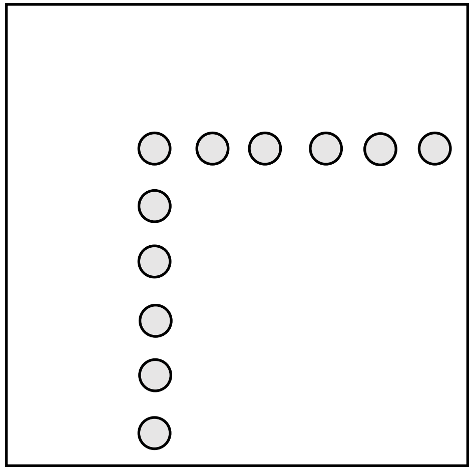
Depth Map



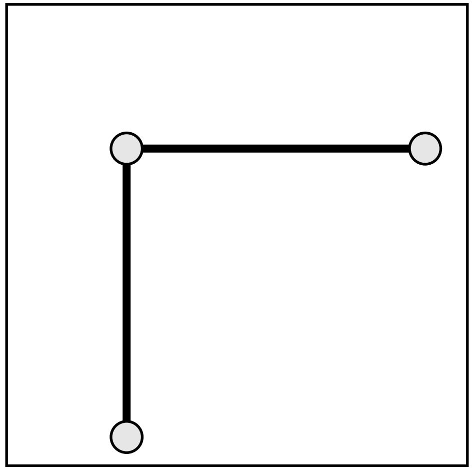
Voxel Grid



Implicit Surface



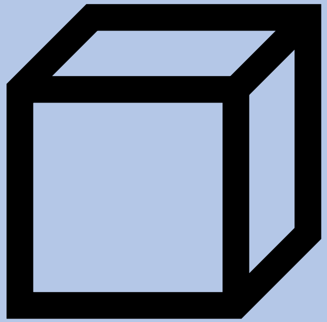
Pointcloud



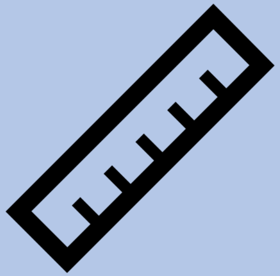
Mesh

3D Shape Prediction

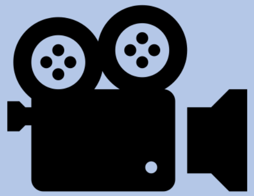
Shape Representations



Metrics



Camera Systems



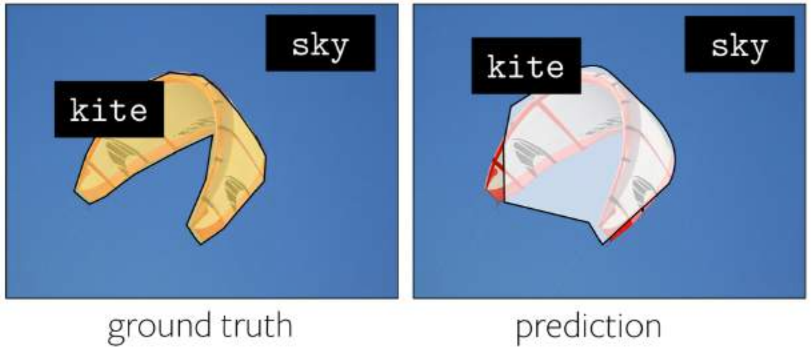
Datasets

3D Shape Prediction

Shape Representations	**Metrics**
Camera Systems	Datasets

Shape Comparison Metrics: Intersection over Union

In 2D, we evaluate boxes and segmentation masks with intersection over union (IoU):



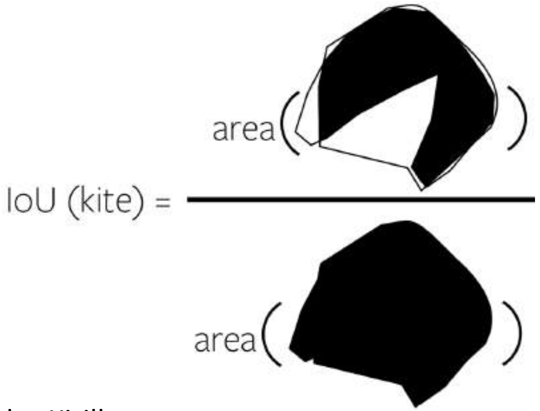
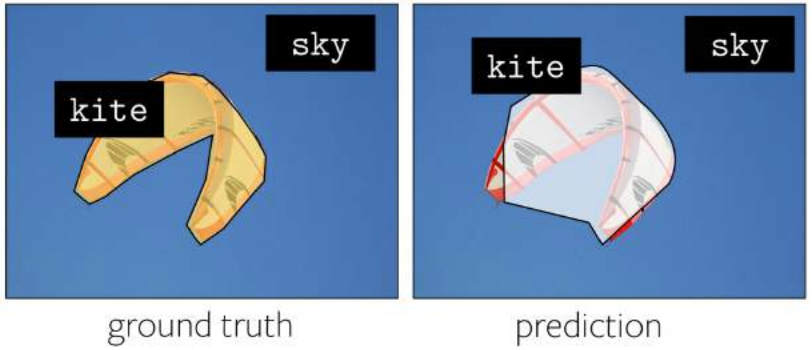
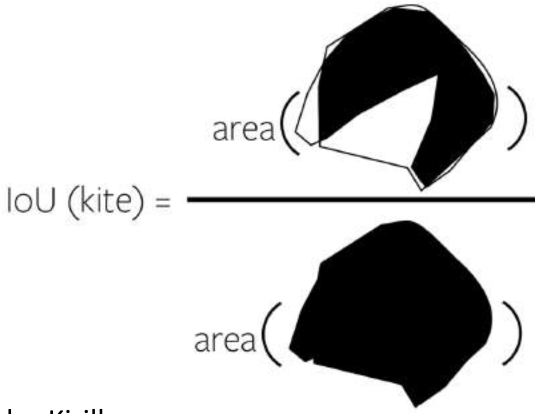


Figure credit: Alexander Kirillov

Shape Comparison Metrics: Intersection over Union

In 2D, we evaluate boxes and segmentation masks with intersection over union (IoU):





In 3D: **Voxel IoU**

Problem: Cannot capture thin structures

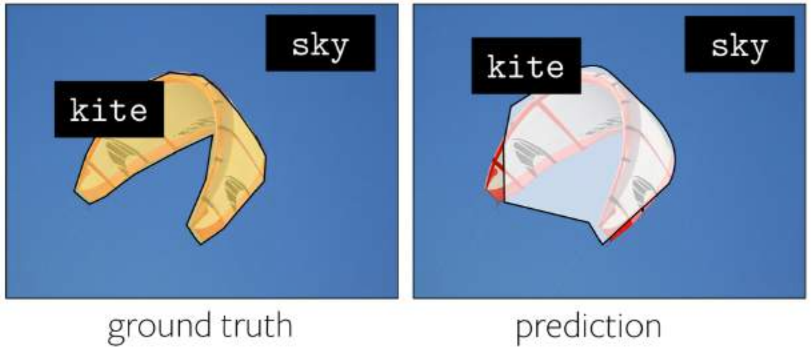
Problem: Cannot be applied to pointclouds

Problem: For meshes, need to voxelize or sample

Figure credit: Alexander Kirillov

Shape Comparison Metrics: Intersection over Union

In 2D, we evaluate boxes and segmentation masks with intersection over union (IoU):



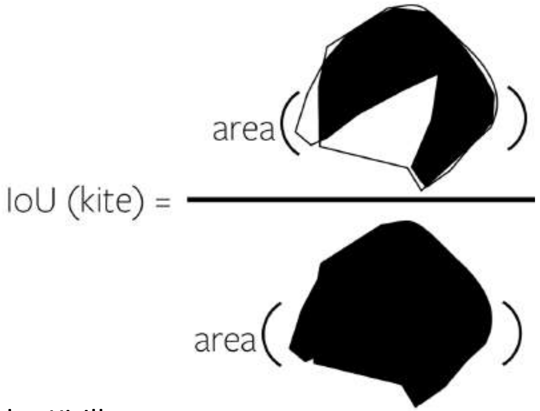


Figure credit: Alexander Kirillov

In 3D: **Voxel IoU**

Problem: Cannot capture thin structures

Problem: Cannot be applied to pointclouds

Problem: For meshes, need to voxelize or sample

Problem: Not very meaningful at low values!

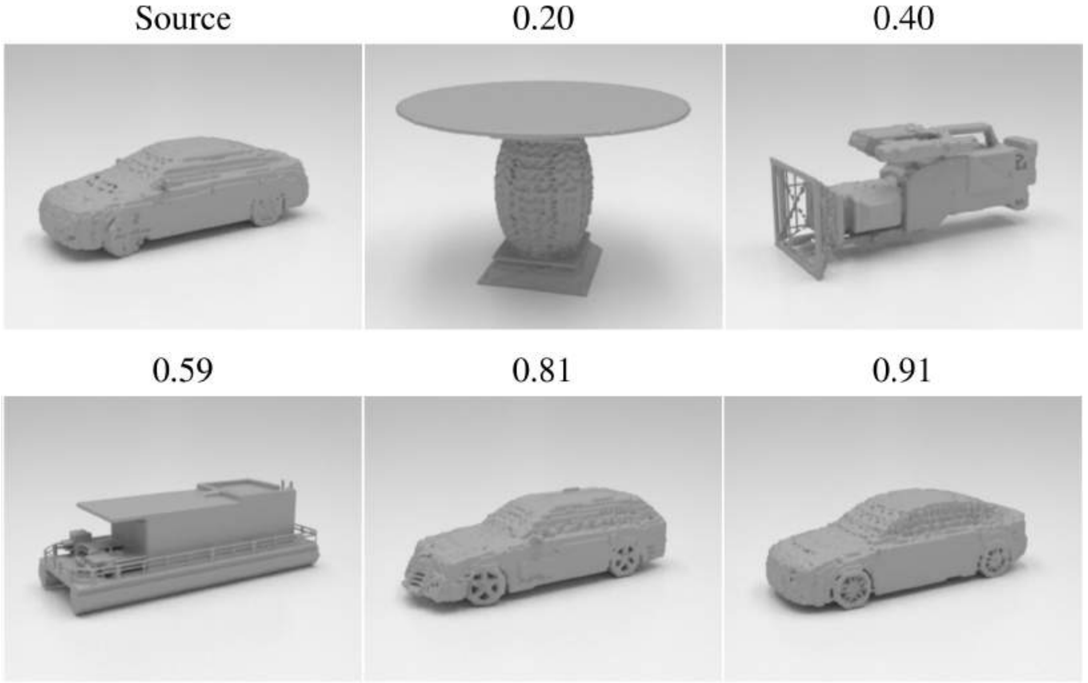
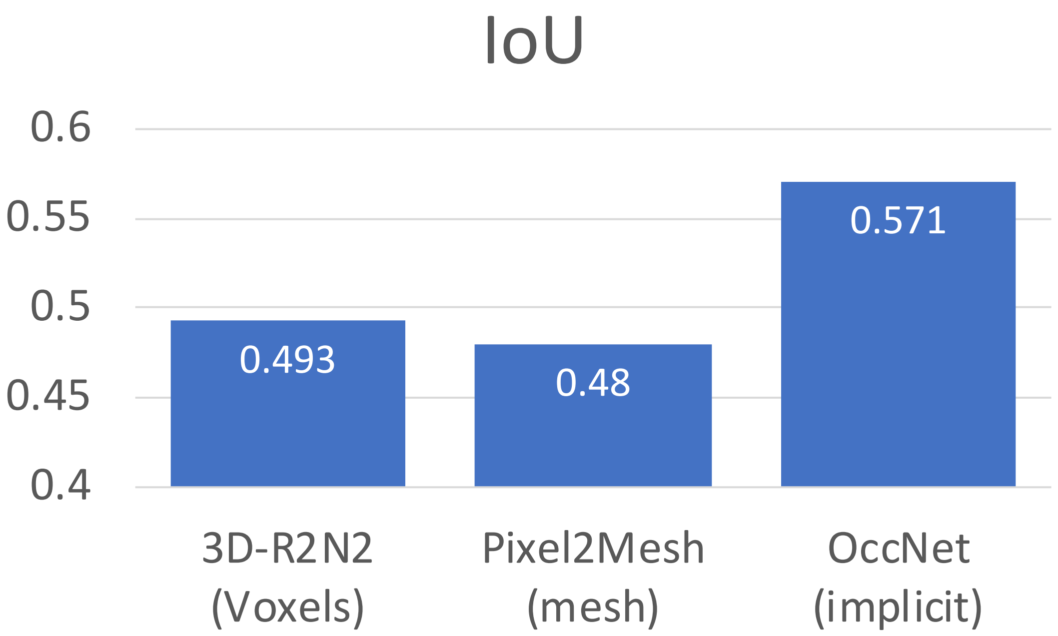


Figure credit: Tatarchenko et al, "What Do Single-view 3D Reconstruction Networks Learn?", CVPR 2019

Shape Comparison Metrics: Intersection over Union

State–of-the-art methods achieve low IoU



Results from Mescheder et al, "Occupancy Networks: Learning 3D Reconstruction in Function Space", CVPR 2019

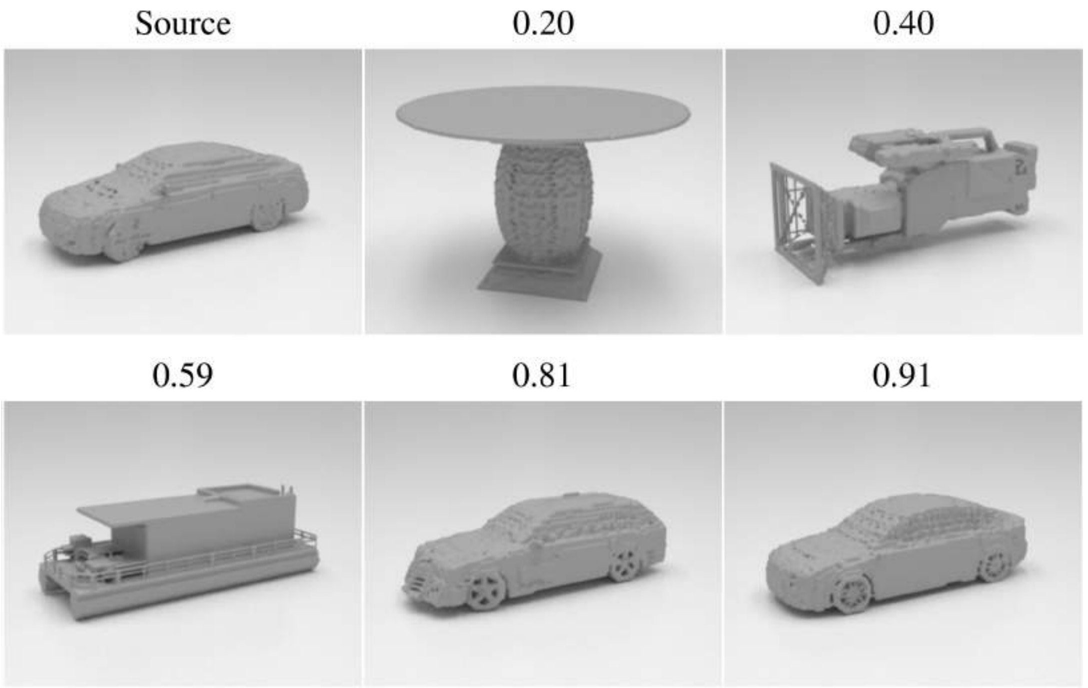
In 3D: **Voxel IoU**

Problem: Cannot capture thin structures

Problem: Cannot be applied to pointclouds

Problem: For meshes, need to voxelize or sample

Problem: Not very meaningful at low values!



Conclusion: Voxel IoU not a good metric

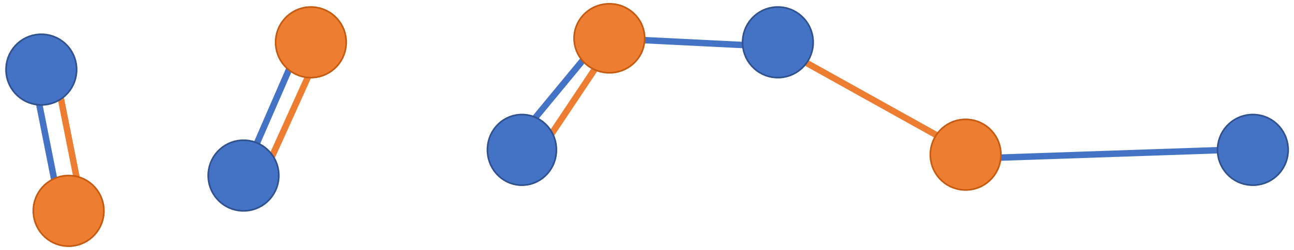
Figure credit: Tatarchenko et al, "What Do Single-view 3D Reconstruction Networks Learn?", CVPR 2019

Shape Comparison Metrics: Chamfer Distance

We've already seen another shape comparison metric: **Chamfer distance**

1. Convert your prediction and ground-truth into pointclouds via sampling
2. Compare with Chamfer distance

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



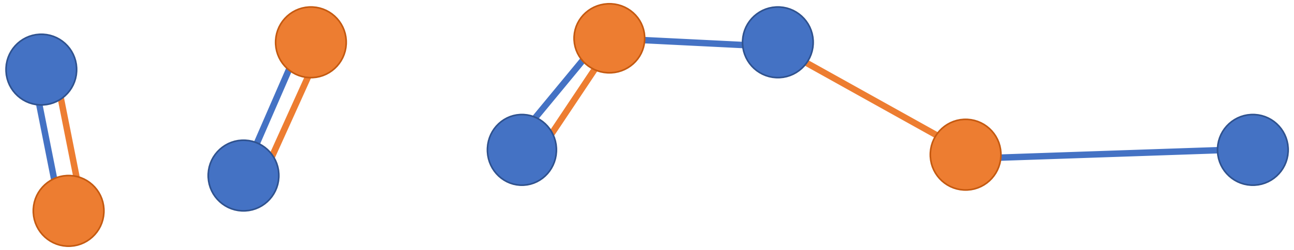
Shape Comparison Metrics: Chamfer Distance

We've already seen another shape comparison metric: **Chamfer distance**

1. Convert your prediction and ground-truth into pointclouds via sampling
2. Compare with Chamfer distance

Problem: Chamfer is very sensitive to outliers

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



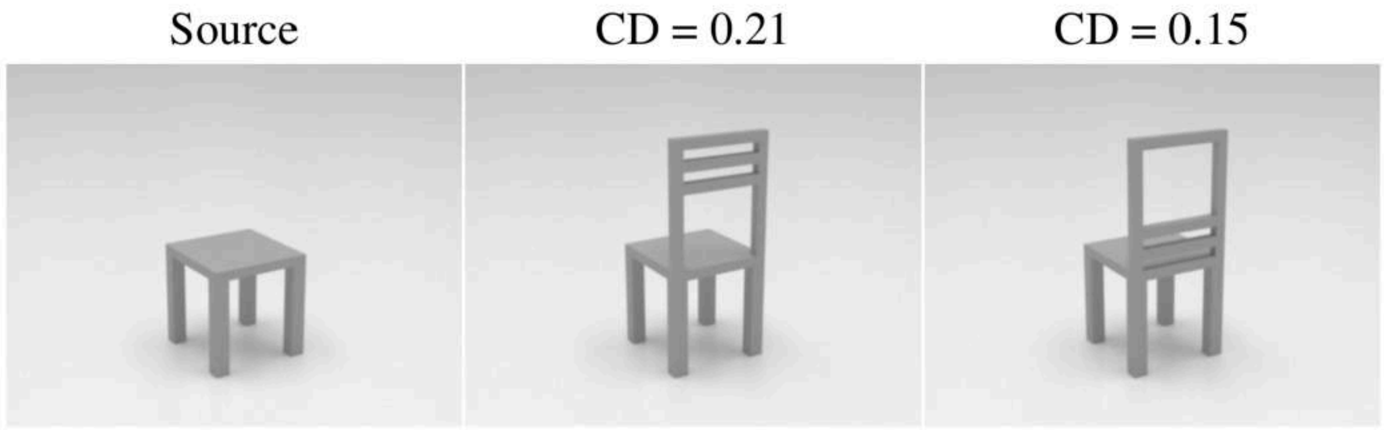
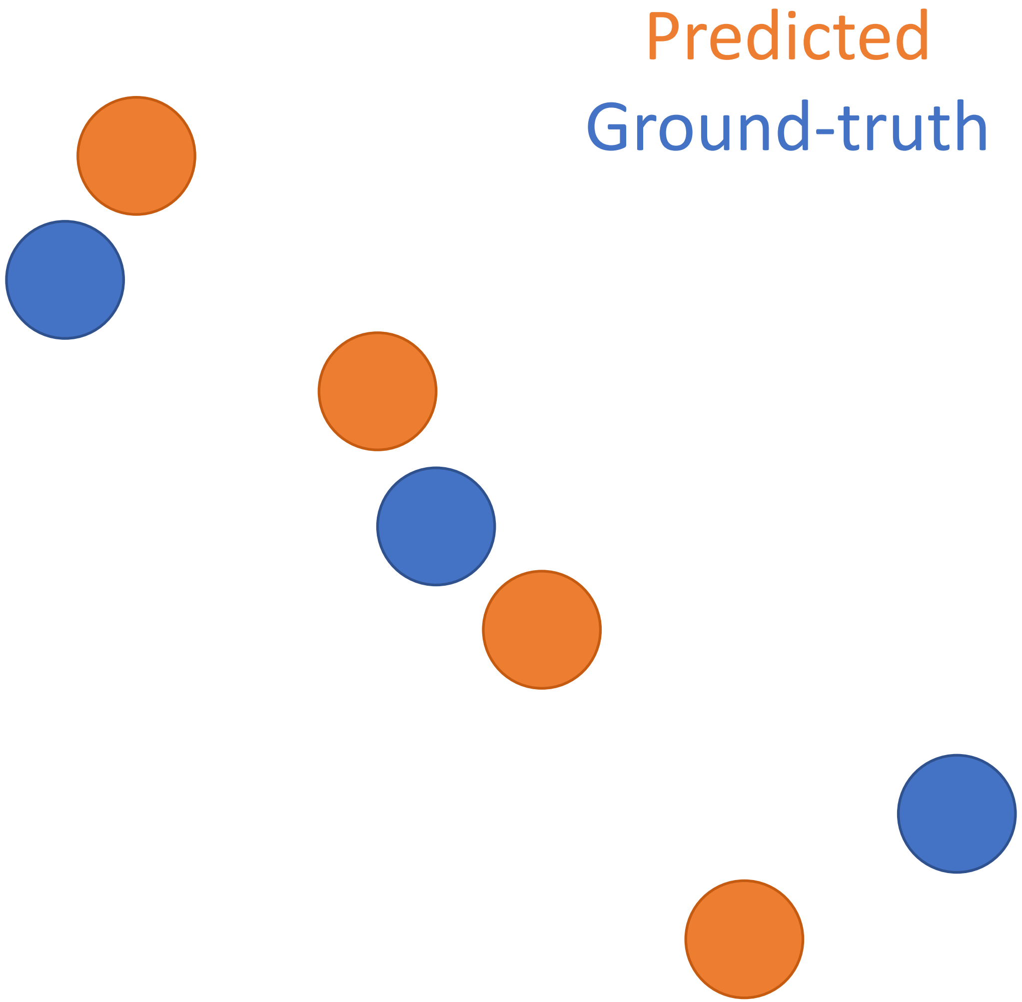


Figure credit: Tatarchenko et al, "What Do Single-view 3D Reconstruction Networks Learn?", CVPR 2019

Shape Comparison Metrics: F1 Score

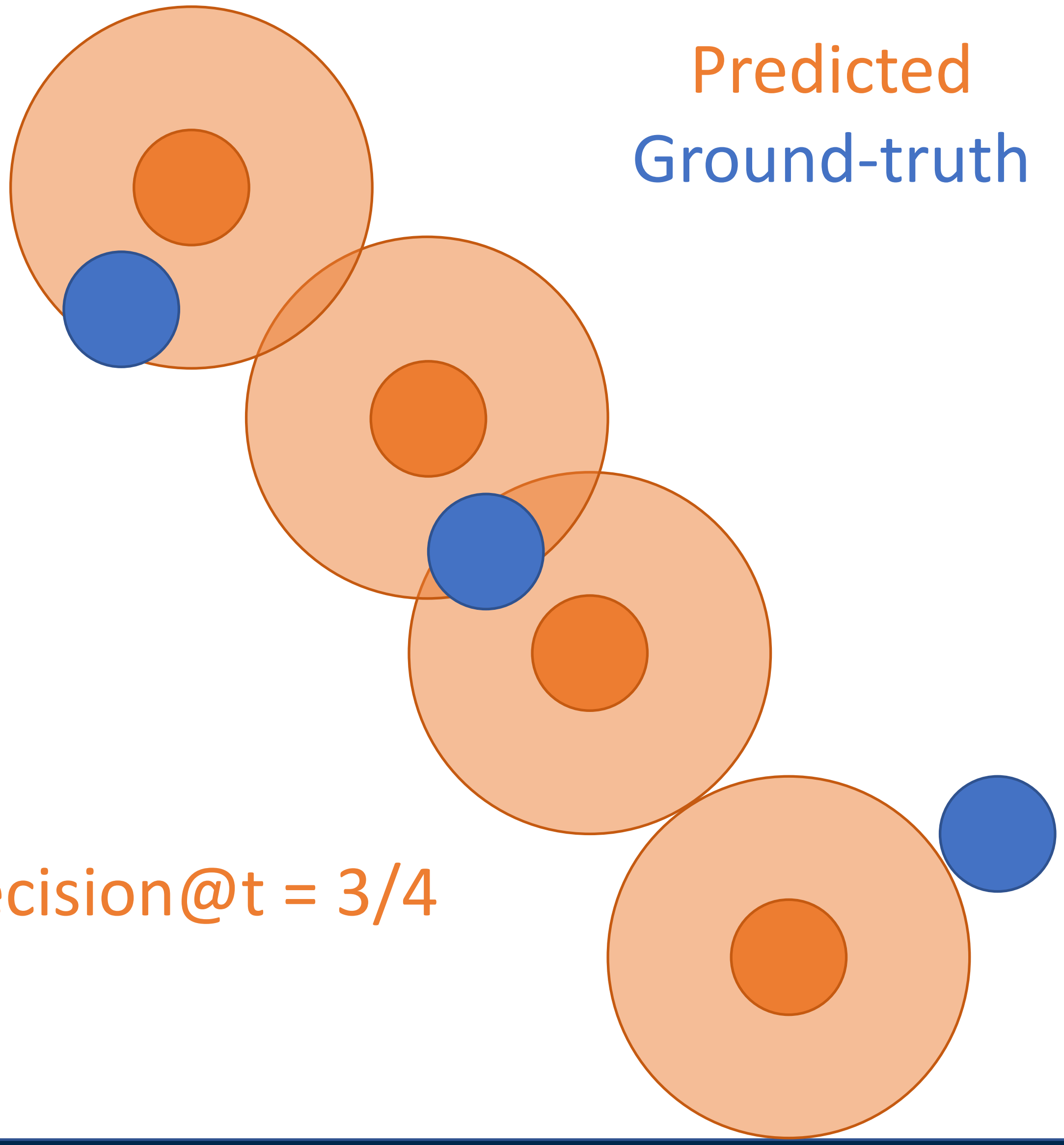
Similar to Chamfer, sample points from the surface of the prediction and the ground-truth



Shape Comparison Metrics: F1 Score

Similar to Chamfer, sample points from the surface of the prediction and the ground-truth

Precision@t = fraction of predicted points within t of some ground-truth point



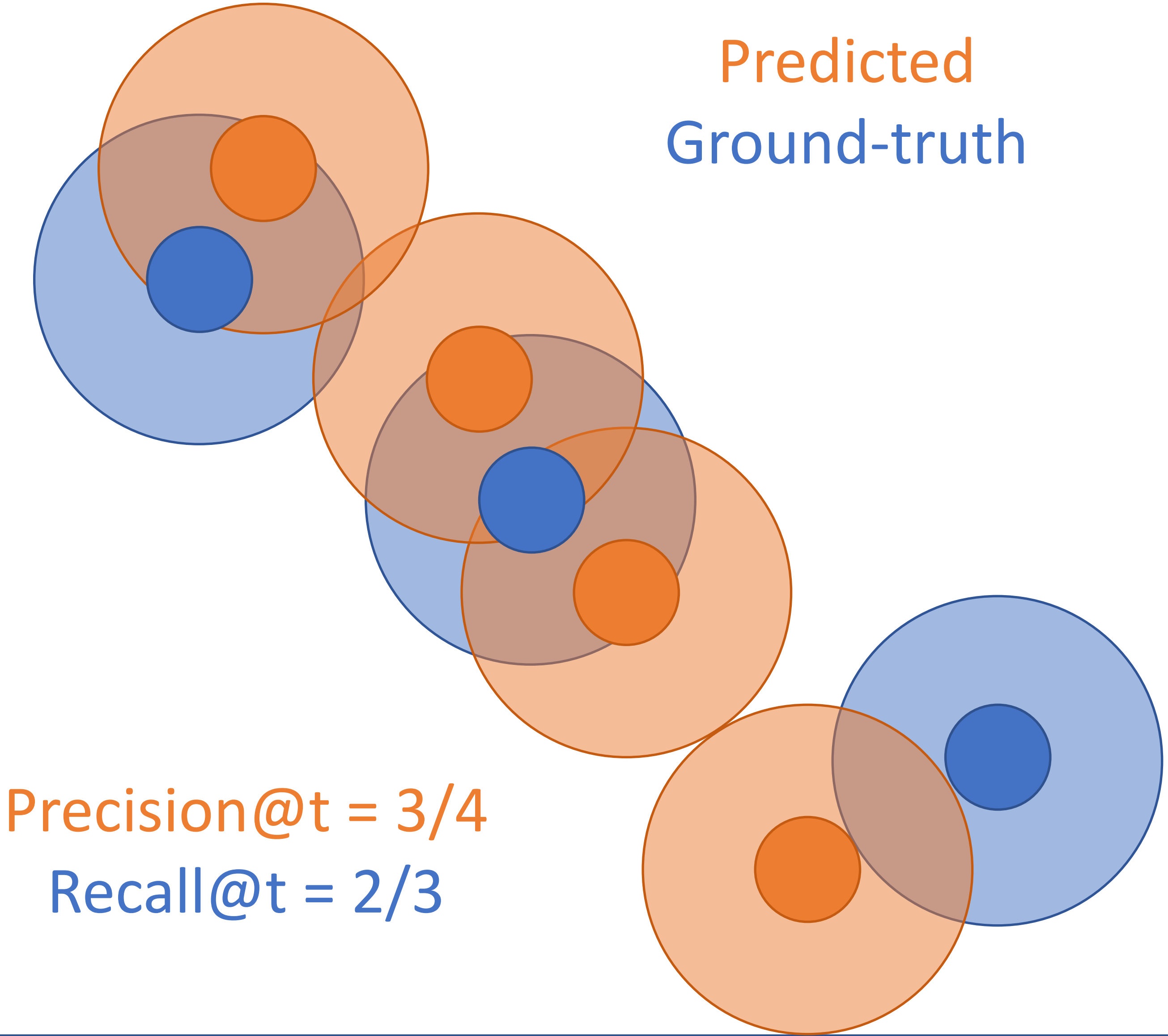
Precision@t = 3/4

Shape Comparison Metrics: F1 Score

Similar to Chamfer, sample points from the surface of the prediction and the ground-truth

Precision@t = fraction of predicted points within t of some ground-truth point

Recall@t = fraction of ground-truth points within t of some predicted point



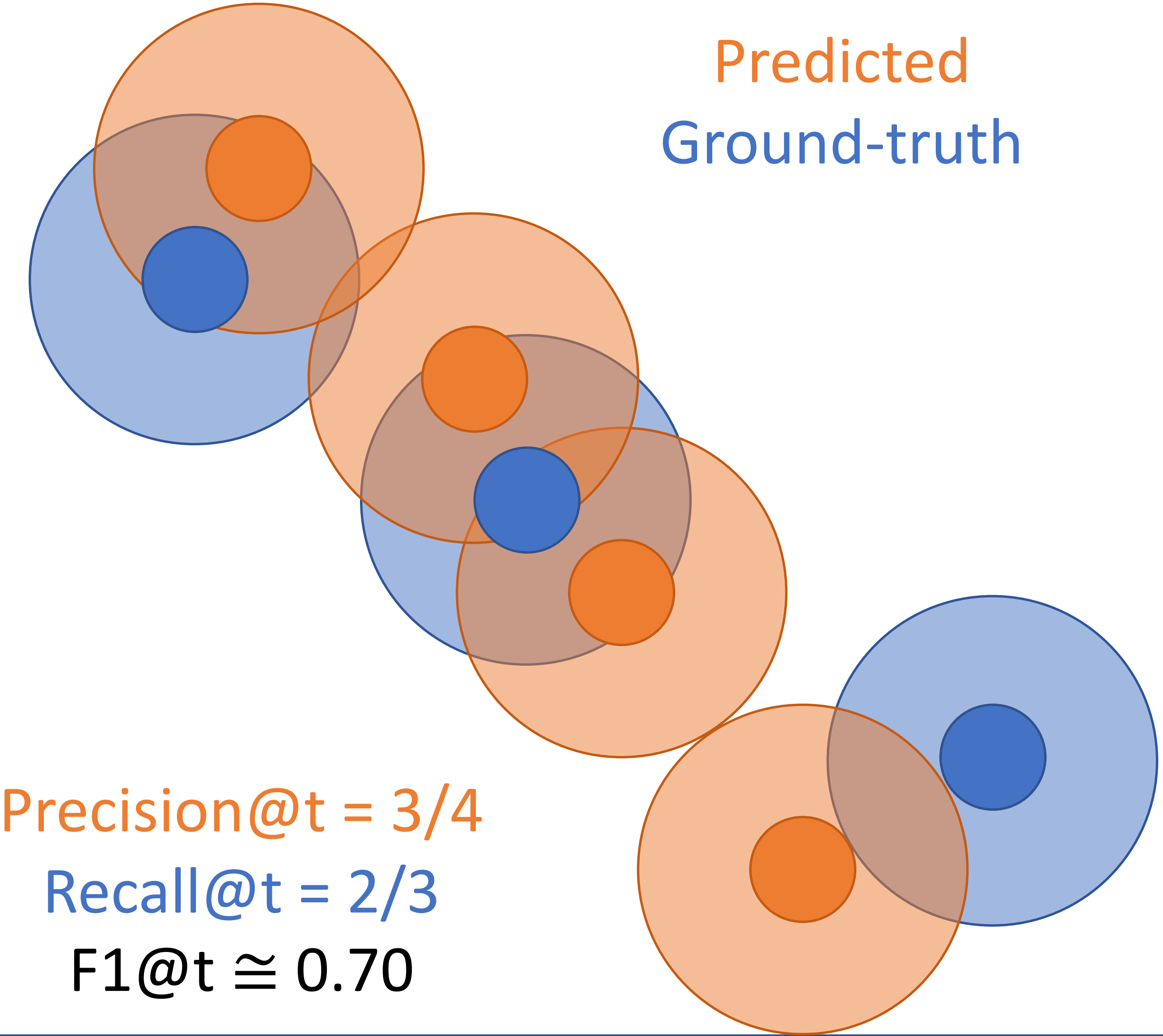
Shape Comparison Metrics: F1 Score

Similar to Chamfer, sample points from the surface of the prediction and the ground-truth

Precision@t = fraction of predicted points within t of some ground-truth point

Recall@t = fraction of ground-truth points within t of some predicted point

F1@t = $2 * \frac{Precision@t * Recall@t}{Precision@t + Recall@t}$



Predicted

Ground-truth

Precision@t = 3/4

Recall@t = 2/3

F1@t ≅ 0.70

Shape Comparison Metrics: F1 Score

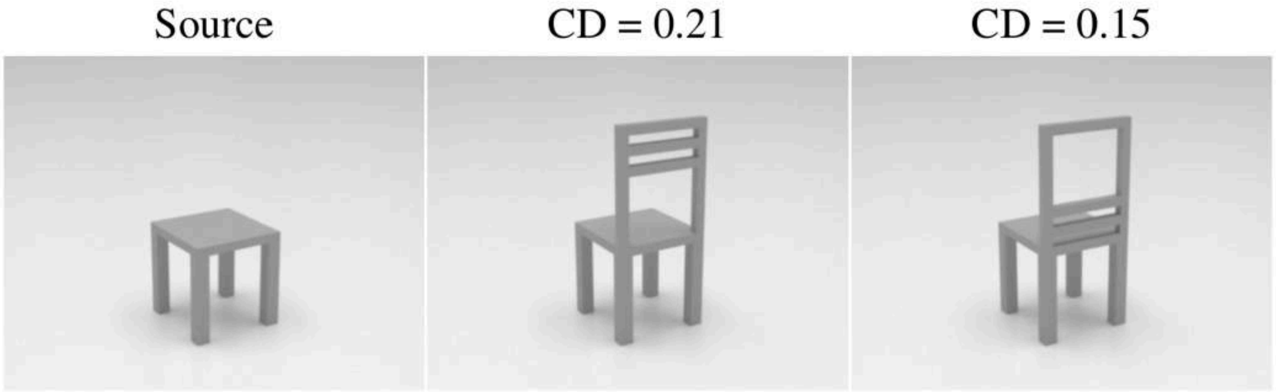
Similar to Chamfer, sample points from the surface of the prediction and the ground-truth

Precision@t = fraction of predicted points within t of some ground-truth point

Recall@t = fraction of ground-truth points within t of some predicted point

F1@t = $2 * \frac{Precision@t * Recall@t}{Precision@t + Recall@t}$

F1 score is robust to outliers!

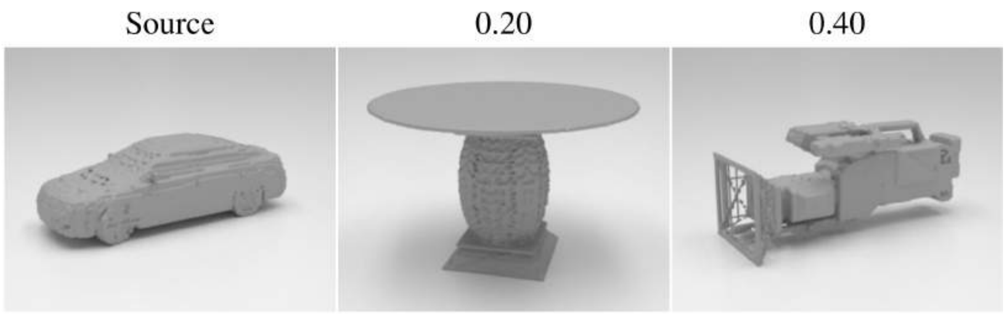


Conclusion: F1 score is probably the best shape prediction metric in common use

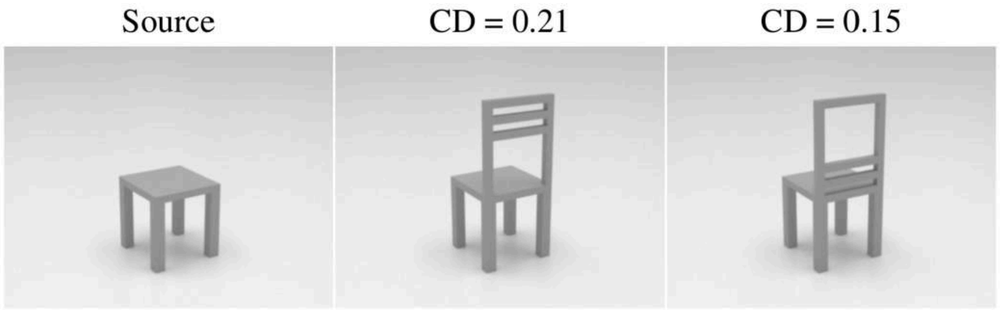
Figure credit: Tatarchenko et al, "What Do Single-view 3D Reconstruction Networks Learn?", CVPR 2019

Shape Comparison Metrics: Summary

Intersection over Union:
Doesn't capture fine structure,
not meaningful at low values



Chamfer Distance:
Very sensitive to outliers
Can be directly optimized



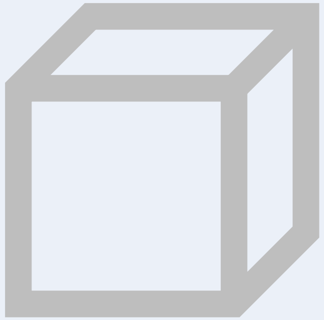
F1 score:
Robust to outliers, but need to
look at different threshold values
to capture details at different scales

3D Shape Prediction

Shape Representations	**Metrics**
Camera Systems	Datasets

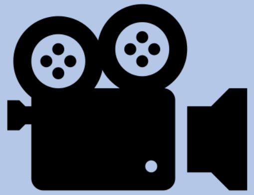
3D Shape Prediction

Shape Representations



Metrics

Camera Systems



Datasets

Cameras: Canonical vs View Coordinates

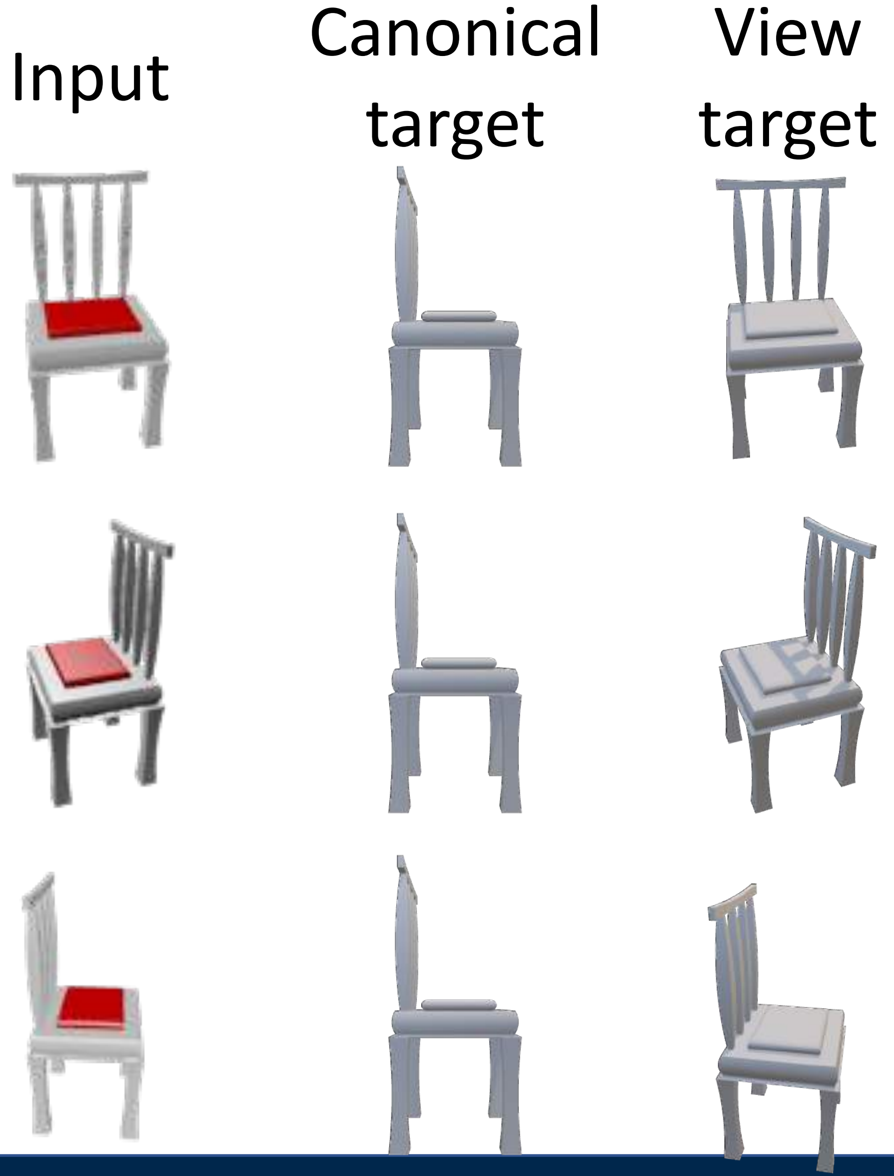
Canonical Coordinates: Predict 3D shape in a canonical coordinate system (e.g. front of chair is +z) regardless of the viewpoint of the input image

Cameras: Canonical vs View Coordinates

Canonical Coordinates: Predict 3D shape in a canonical coordinate system (e.g. front of chair is +z) regardless of the viewpoint of the input image

View Coordinates: Predict 3D shape aligned to the viewpoint of the camera

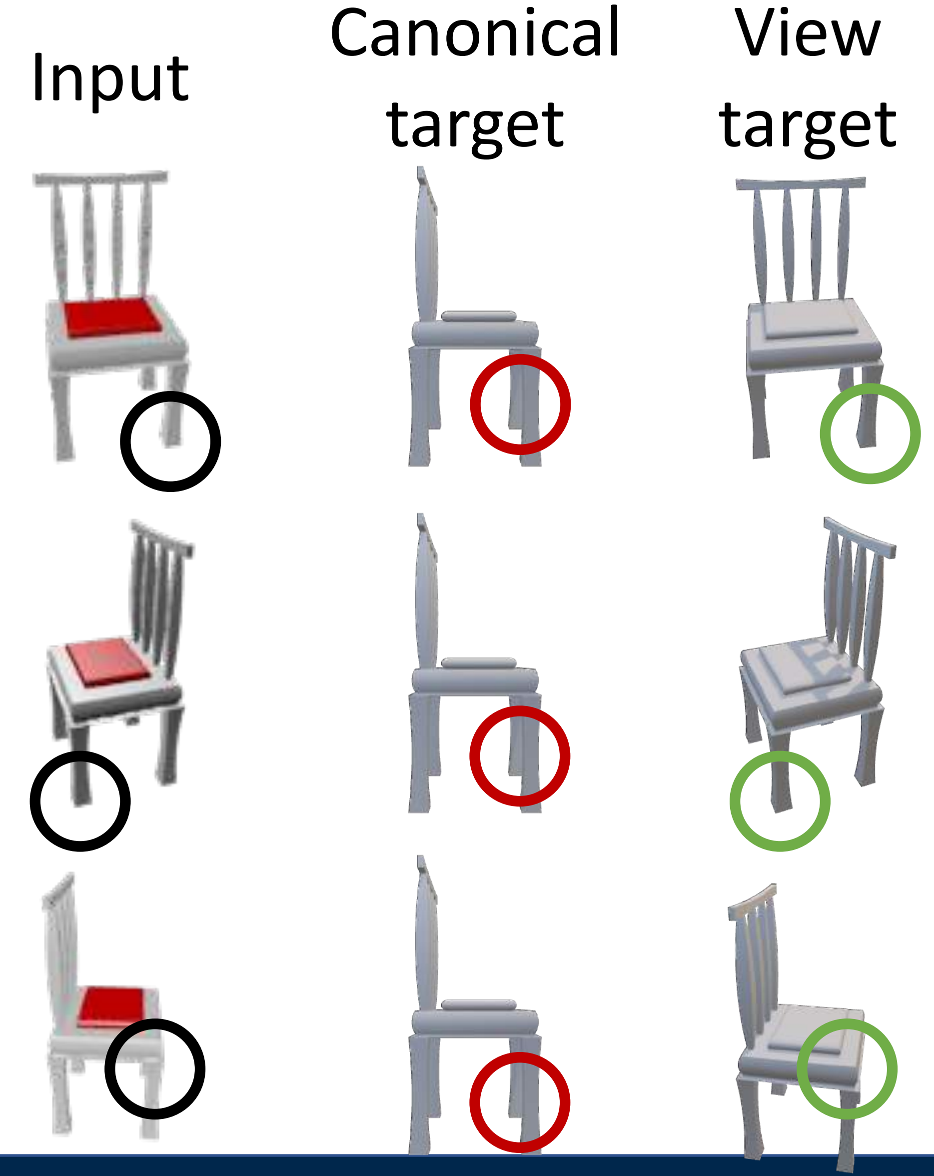
Many papers predict in canonical coordinates – easier to load data



Cameras: Canonical vs View Coordinates

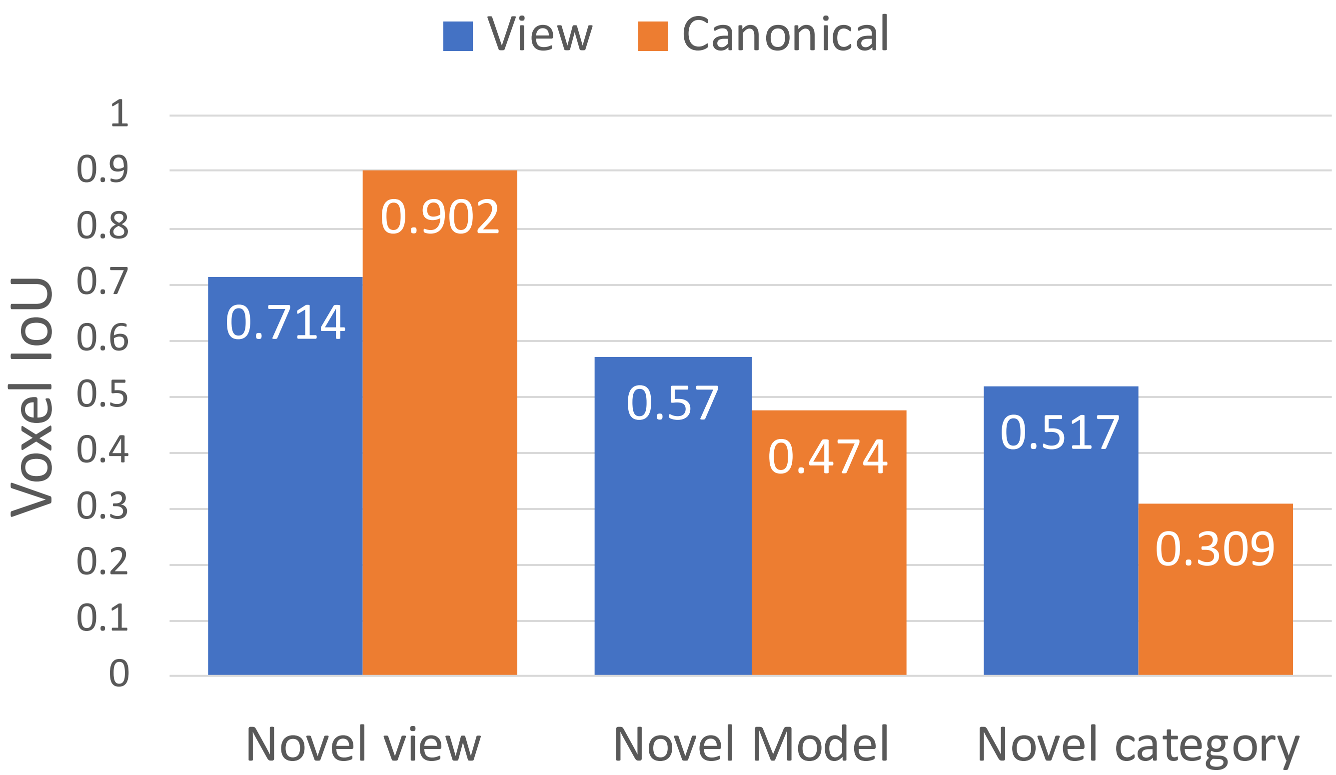
Problem: Canonical view breaks the “principle of feature alignment”: Predictions should be aligned to inputs

View coordinates maintain alignment between inputs and predictions!

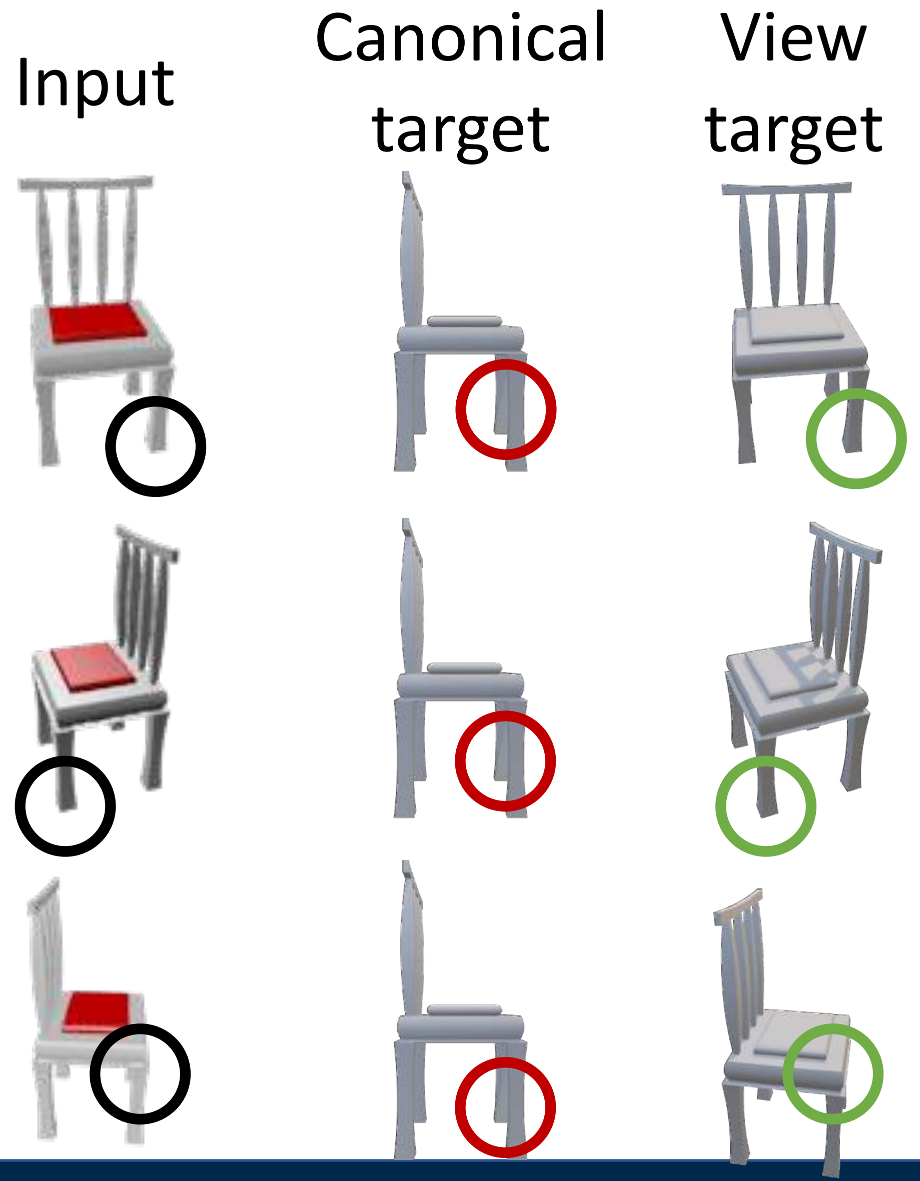


Cameras: Canonical vs View Coordinates

Problem: Canonical view overfits to training shapes:
Better generalization to new views of known shapes
Worse generalization to new shapes or new categories

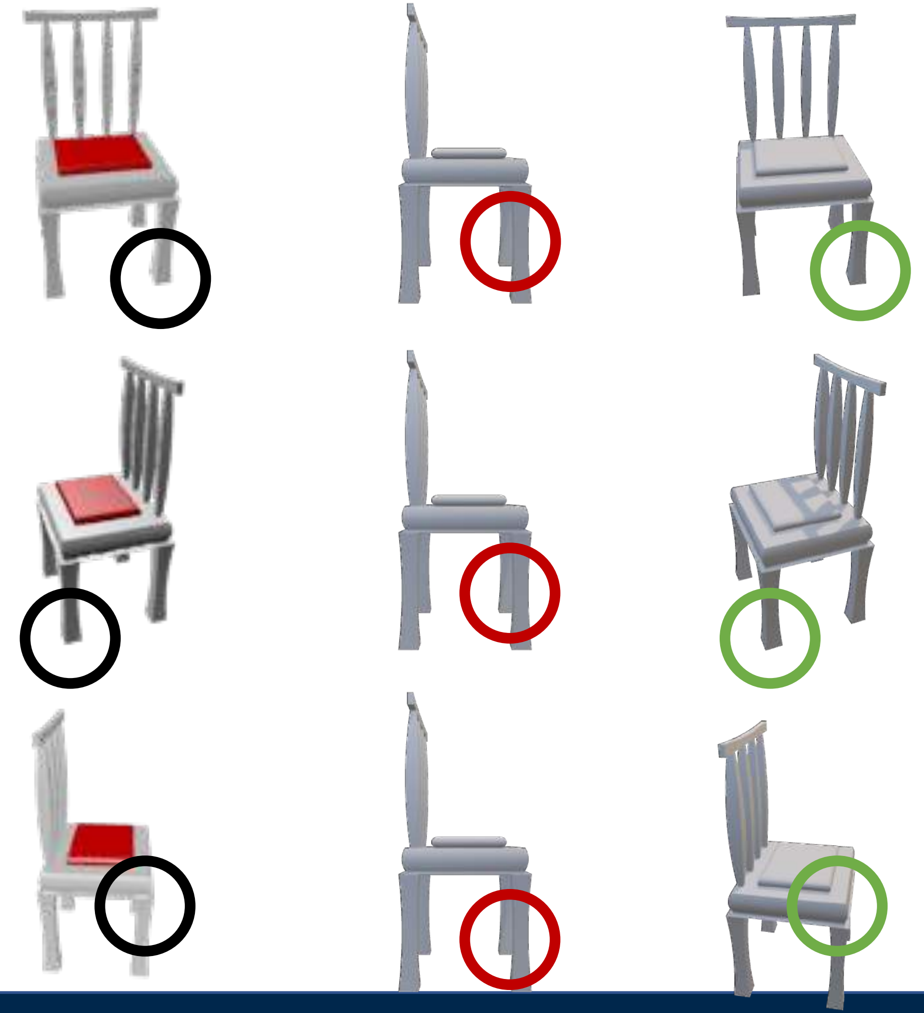


Shin et al, "Pixels, voxels, and views: A study of shape representations for single view 3D object shape prediction", CVPR 2018



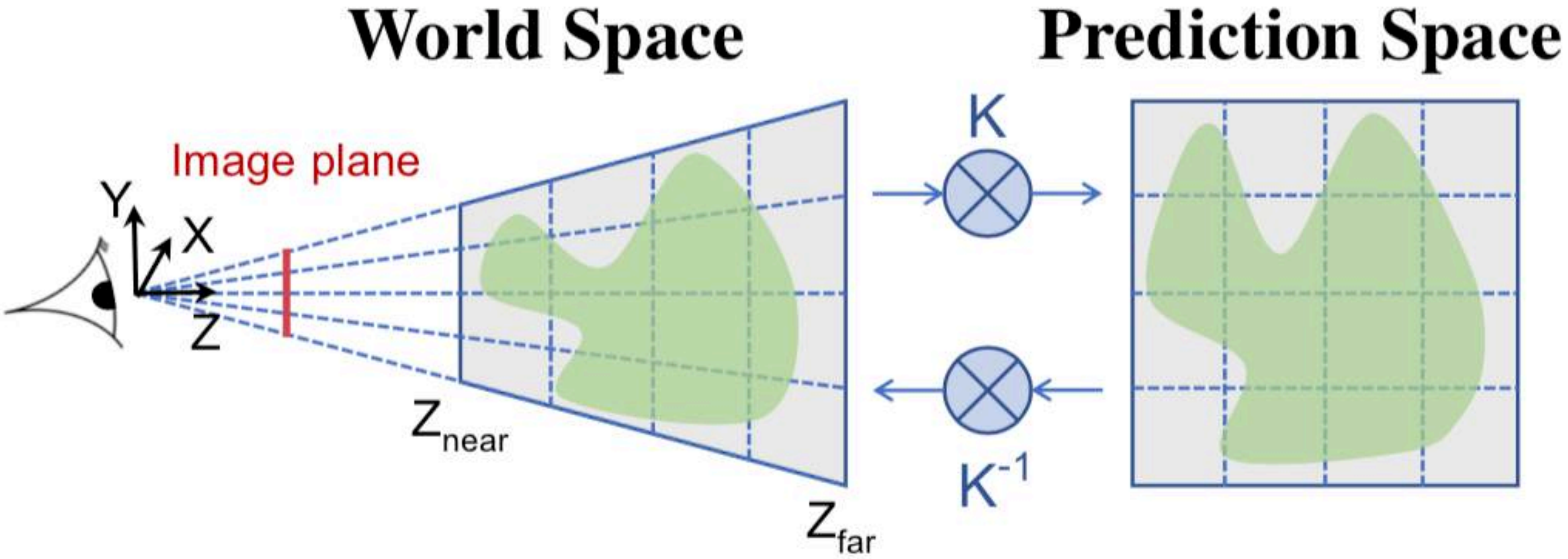
Cameras: Canonical vs View Coordinates

Problem: Canonical view overfits to training shapes:
Better generalization to new views of known shapes
Worse generalization to new shapes or new categories



Conclusion: Prefer view coordinate system

View-Centric Voxel Predictions

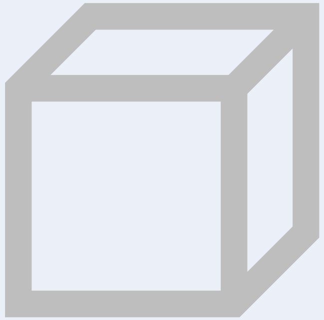


View-centric predictions! Voxels take perspective camera into account, so our "voxels" are actually frustums

Gkioxari, Malik, and Johnson, "Mesh R-CNN", ICCV 2019

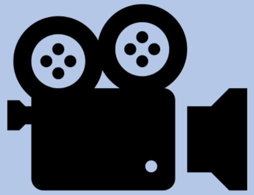
3D Shape Prediction

Shape Representations



Metrics

Camera Systems



Datasets

3D Shape Prediction

Shape Representations

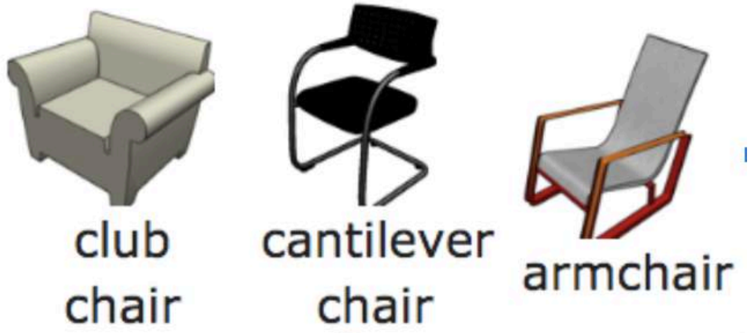
Metrics

Camera Systems

Datasets

3D Datasets: Object-Centric

ShapeNet



~50 categories, ~50k 3D CAD models

Standard split has 13 categories, ~44k models, 25 rendered images per model

Many papers show results here

(-) Synthetic, isolated objects; no context

(-) Lots of chairs, cars, airplanes

Chang et al, "ShapeNet: An Information-Rich 3D Model Repository", arXiv 2015
Choy et al, "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction", ECCV 2016

3D Datasets: Object-Centric

ShapeNet

~50 categories, ~50k 3D CAD models

Standard split has 13 categories, ~44k models, 25 rendered images per model

Many papers show results here

(-) Synthetic, isolated objects; no context

(-) Lots of chairs, cars, airplanes

Pix3D

9 categories, 219 3D models of IKEA furniture aligned to ~17k real images

Some papers train on ShapeNet and show qualitative results here, but use ground-truth segmentation masks

(+) Real images! Context!

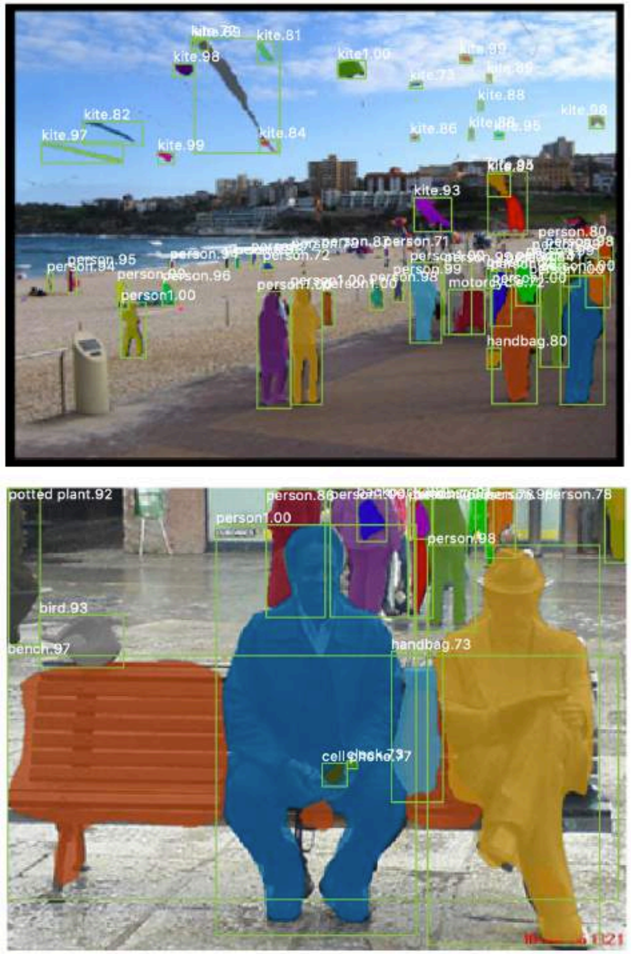
(-) Small, partial annotations – only 1 obj/image

Chang et al, "ShapeNet: An Information-Rich 3D Model Repository", arXiv 2015
Choy et al, "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction", ECCV 2016

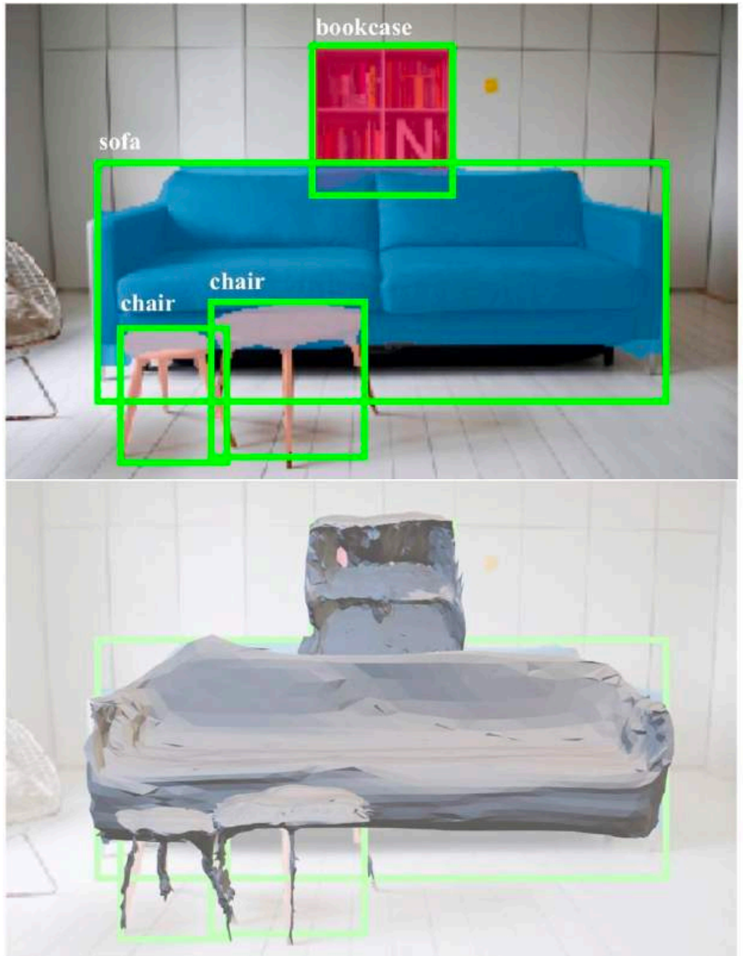
Sun et al, "Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling", CVPR 2018

3D Shape Prediction: Mesh R-CNN

Mask R-CNN:
2D Image -> 2D shapes



Mesh R-CNN:
2D Image -> **Triangle Meshes**



He, Gkioxari, Dollár, and Girshick, "Mask R-CNN", ICCV 2017

Gkioxari, Malik, and Johnson, "Mesh R-CNN", ICCV 2019

Mesh R-CNN: Task

Input: Single RGB image

Output:

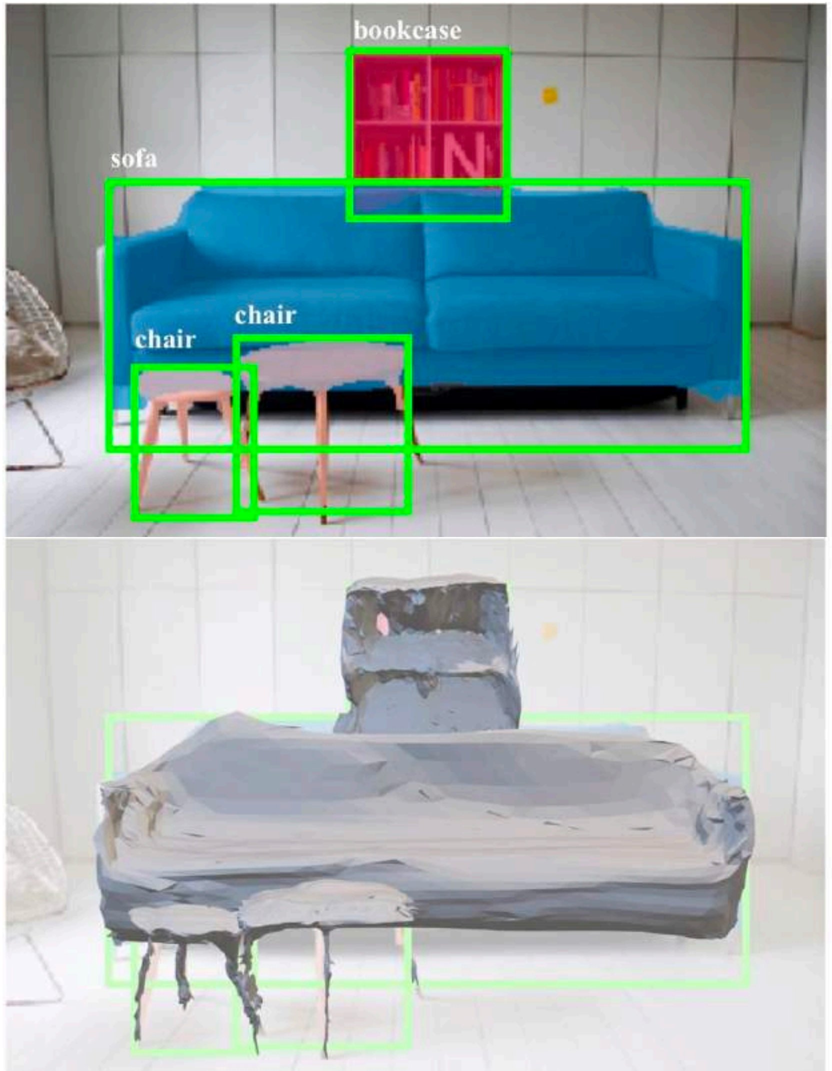
A set of detected objects

For each object:

- Bounding box
- Category label
- Instance segmentation
- 3D triangle mesh

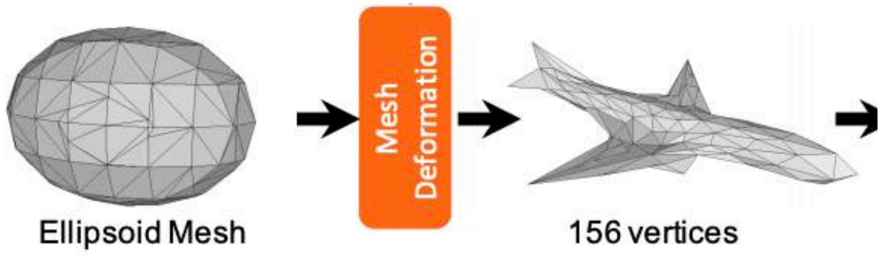
Mask R-CNN

Mesh head



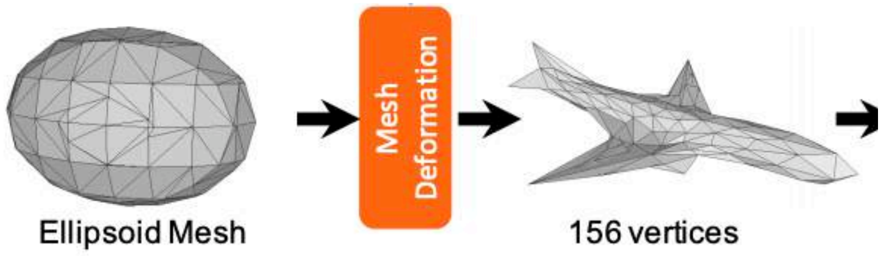
Mesh R-CNN: Hybrid 3D shape representation

Mesh deformation gives good results, but the topology (verts, faces, genus, connected components) fixed by the initial mesh

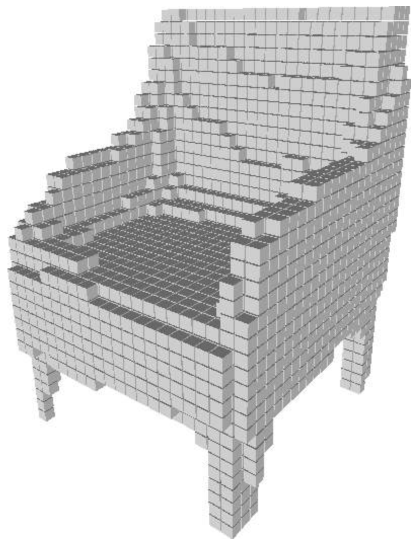


Mesh R-CNN: Hybrid 3D shape representation

Mesh deformation gives good results, but the topology (verts, faces, genus, connected components) fixed by the initial mesh



Our approach: Use voxel predictions to create initial mesh prediction!



Mesh R-CNN Pipeline

Input image

Mesh R-CNN Pipeline

Input image

→

2D object recognition

Mesh R-CNN Pipeline

Input image

2D object recognition

3D object voxels

Mesh R-CNN Pipeline

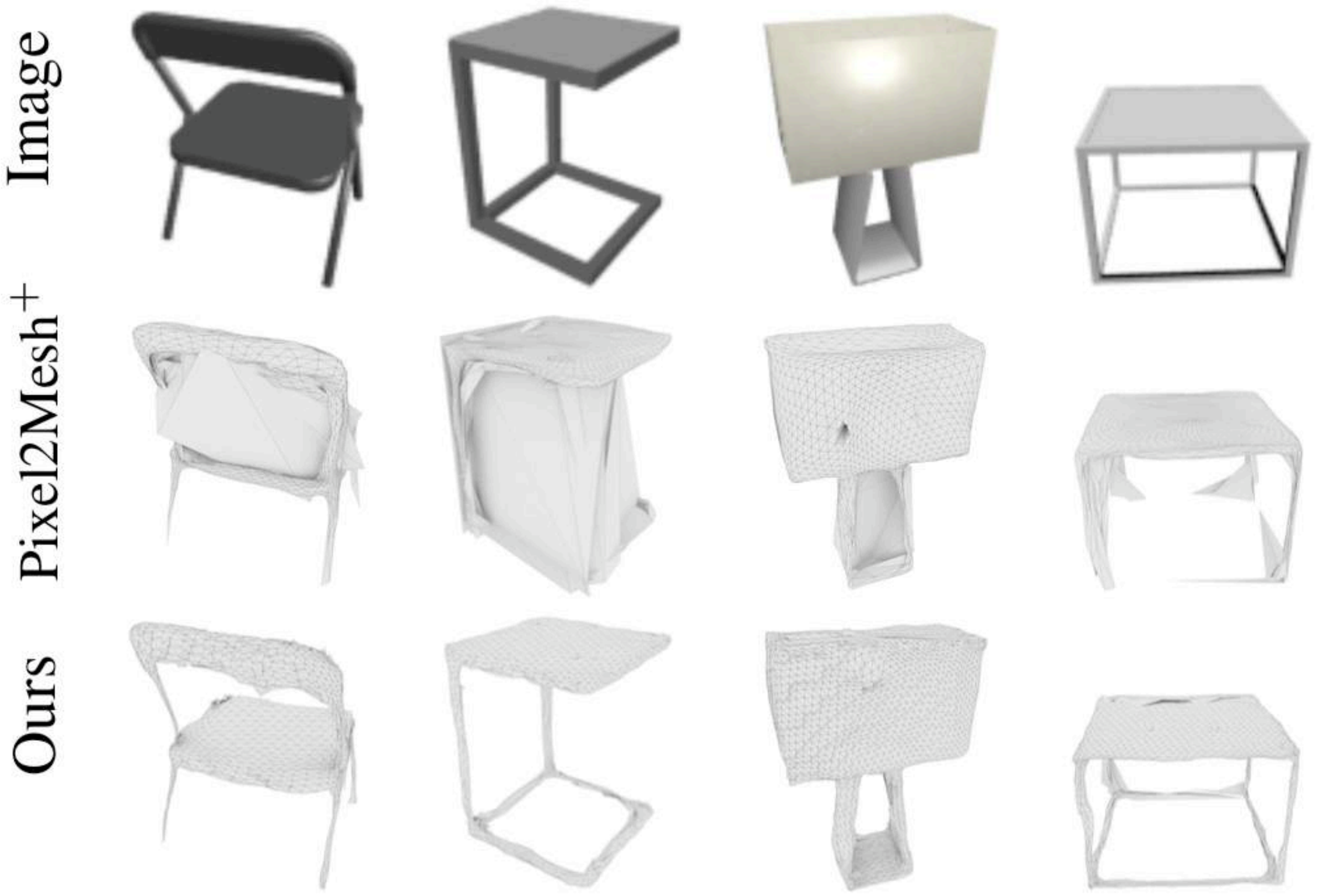
Input image

2D object recognition

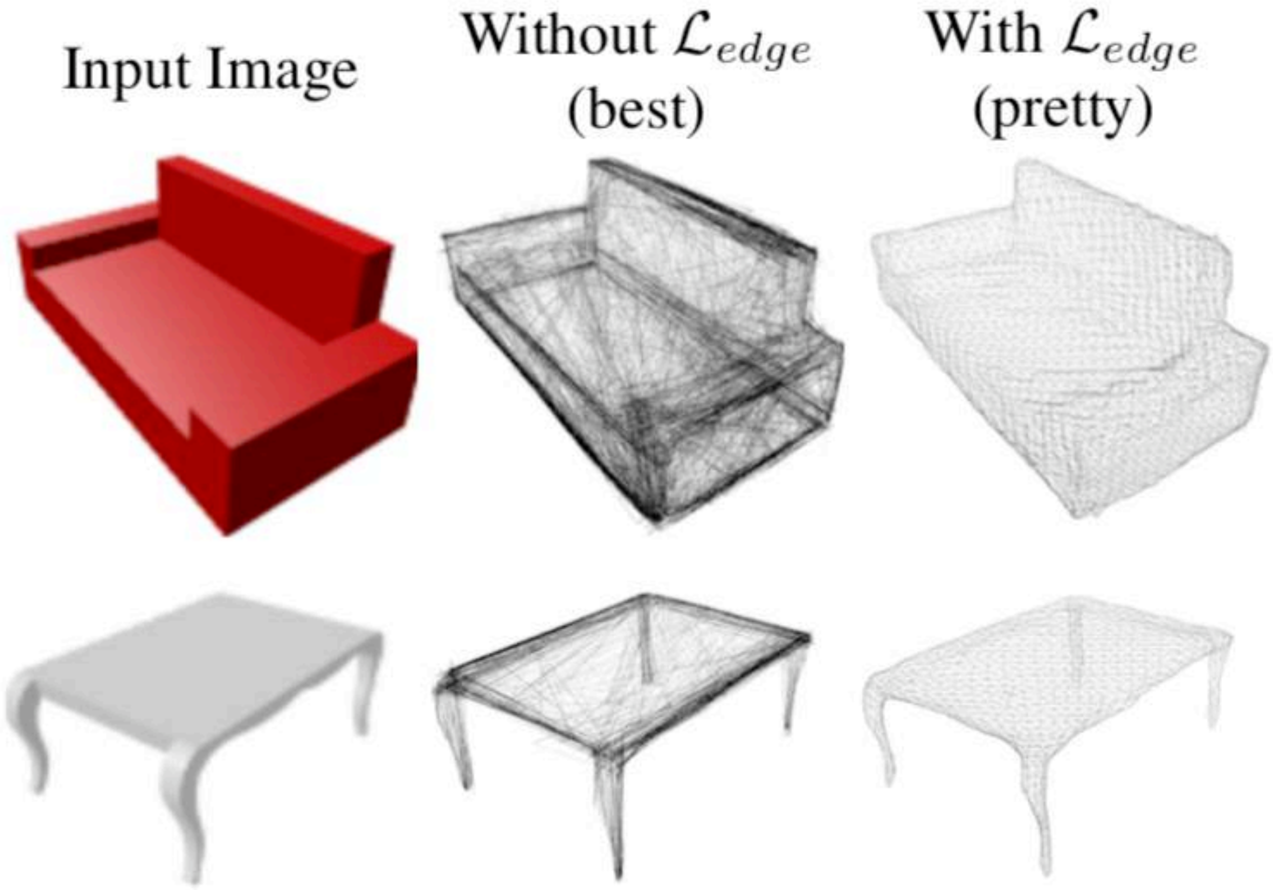
3D object meshes

3D object voxels

Mesh R-CNN: ShapeNet Results



Mesh R-CNN: Shape Regularizers



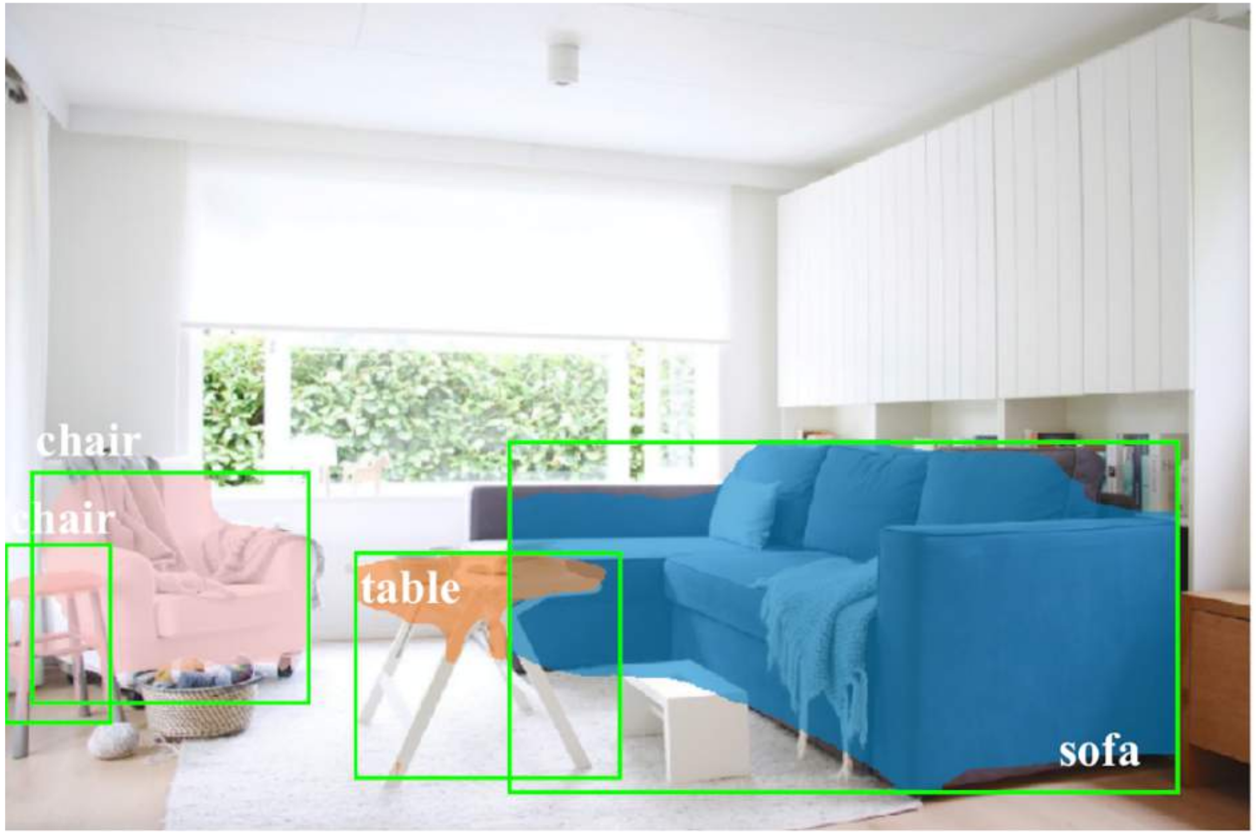
Using Chamfer as only mesh loss gives degenerate meshes. Also need ”mesh regularizer” to encourage nice predictions:

L_{edge} = minimize L2 norm of edges in the predicted mesh

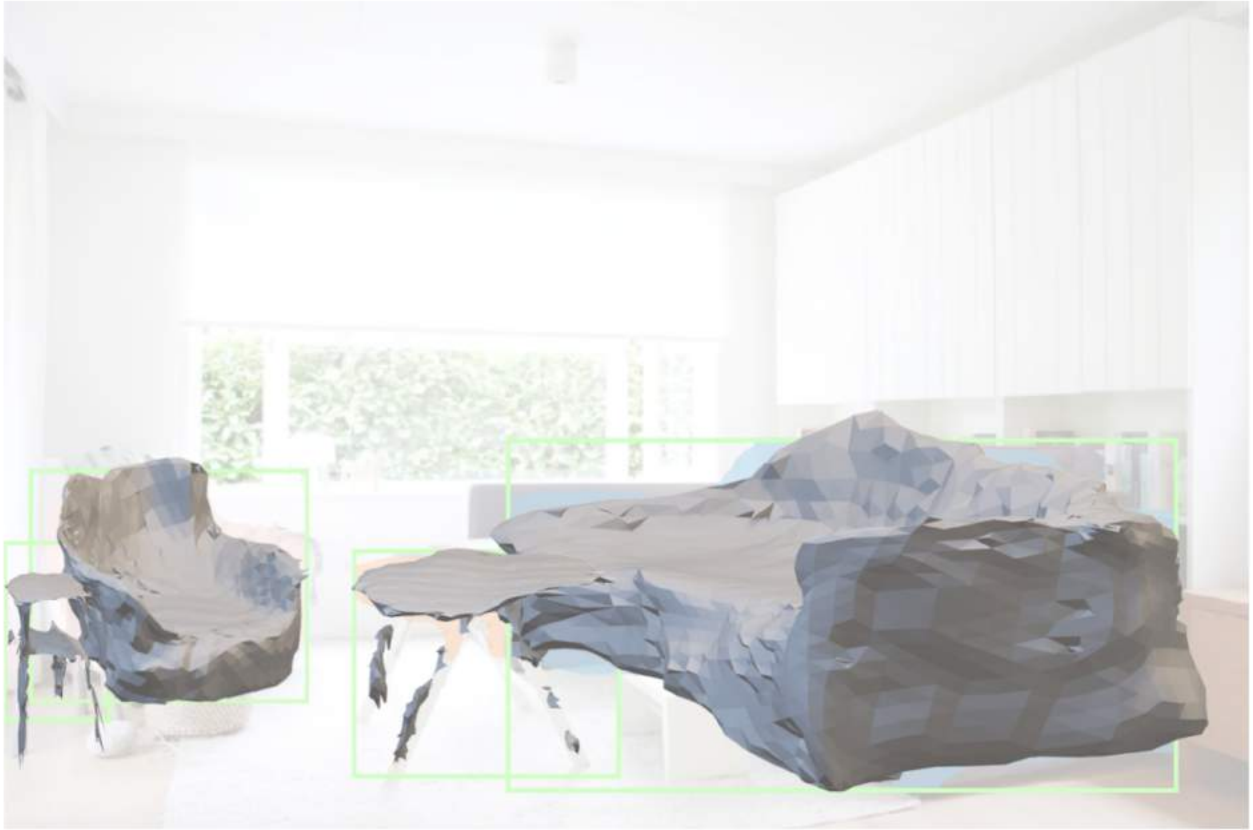
Mesh R-CNN: Pix3D Results

Mesh R-CNN: Pix3D Results

Predicting many objects per scene



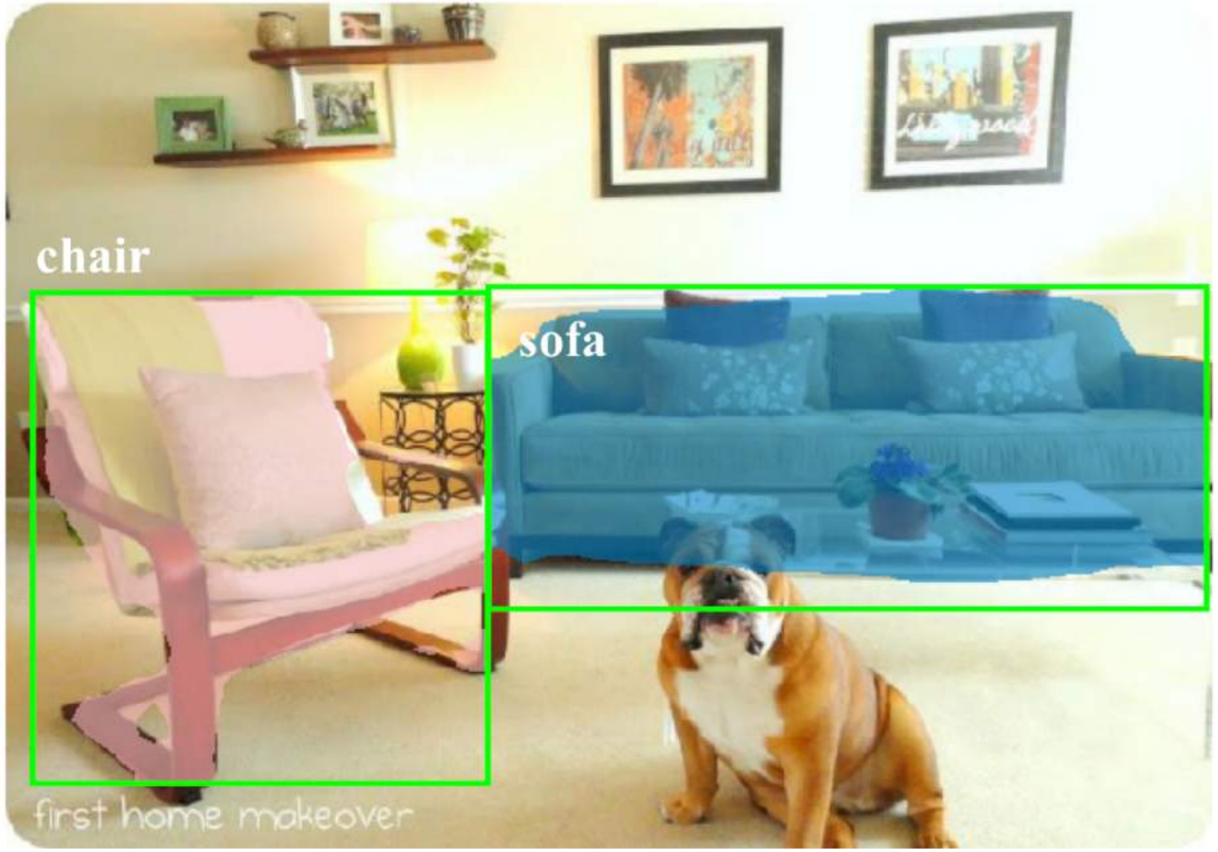
Box & Mask Predictions



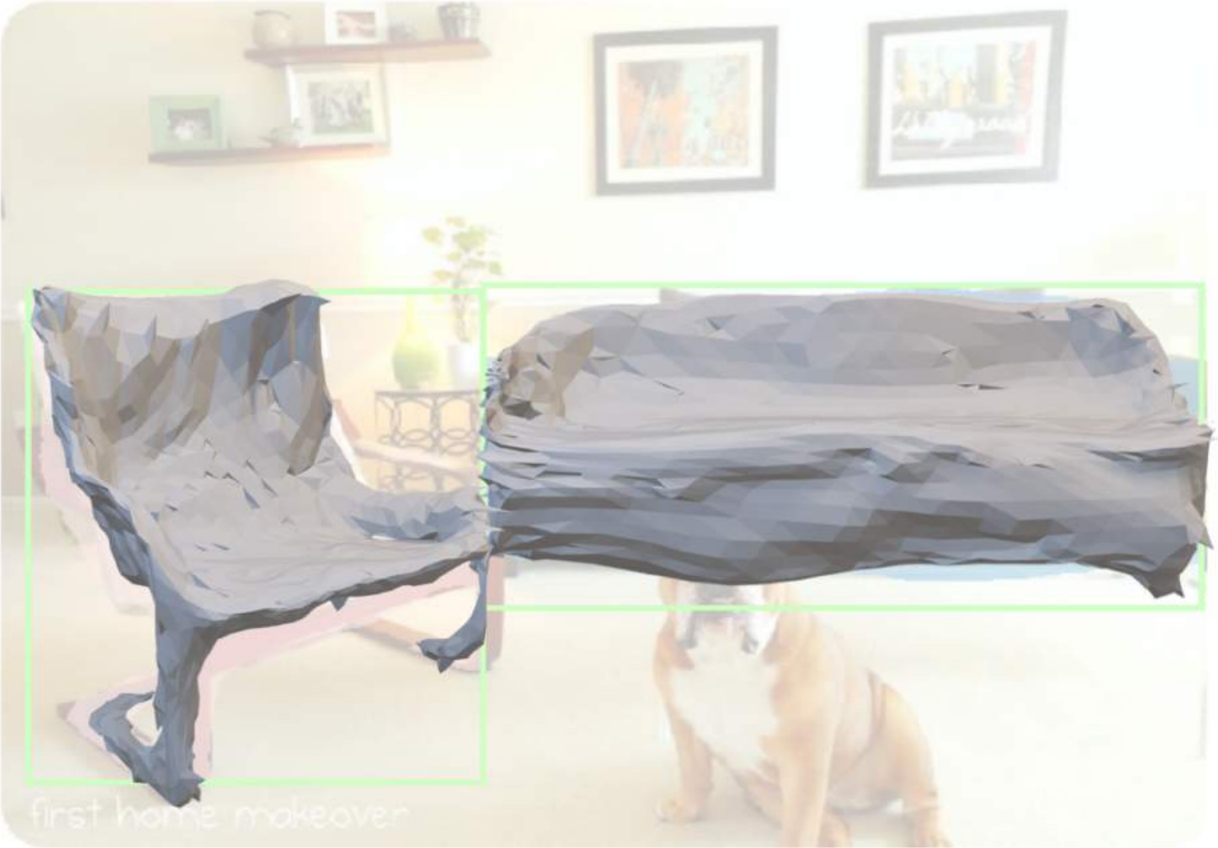
Mesh Predictions

Mesh R-CNN: Pix3D Results

Amodal completion: predict occluded parts of objects



Box & Mask Predictions



Mesh Predictions

Mesh R-CNN: Pix3D Results

Segmentation failures propagate to meshes

Box & Mask Predictions

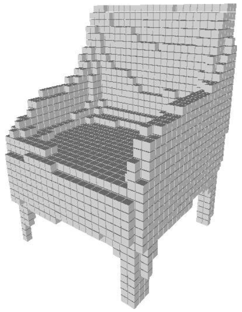
Mesh Predictions

Recap

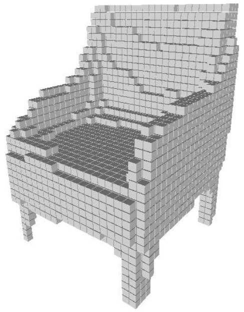
Predicting 3D Shapes from single image

Processing 3D input data

Input Image



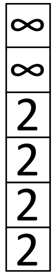
3D Shape



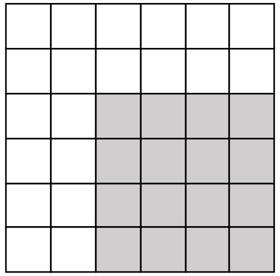
3D Shape

Chair

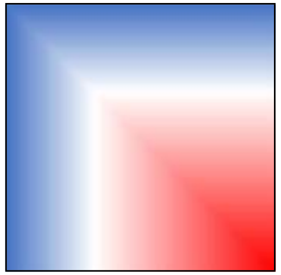
3D Shape Representations



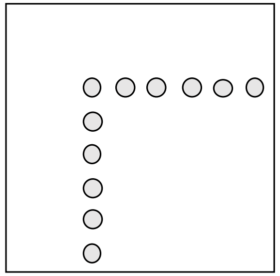
Depth Map



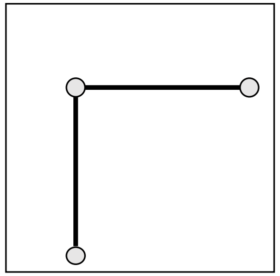
Voxel Grid



Implicit Surface



Pointcloud



Mesh

Next Time:
Videos