

# 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  
Semantic Segmentation  
Object Detection  
Instance Segmentation

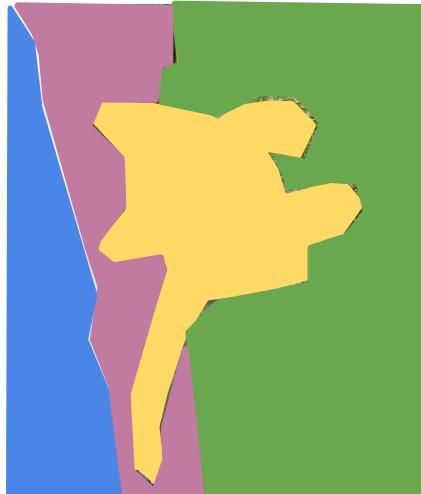


CAT

GRASS, CAT, TREE,  
SKY

No spatial extent

No objects, just pixels



GRASS, CAT, TREE,  
SKY



DOG, DOG, CAT

DOG, DOG, CAT

Multiple Objects



DOG, DOG, CAT

This image is CC0 public domain

# Today: Predicting 3D Shapes of Objects

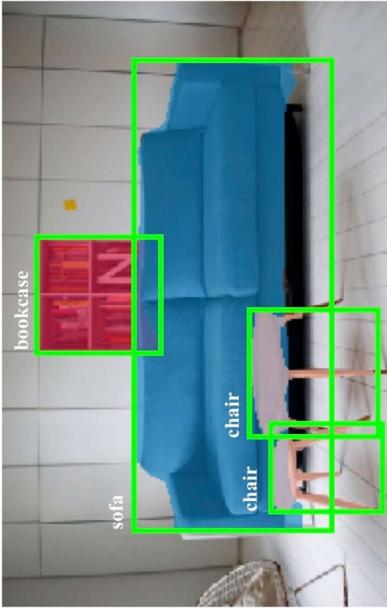
## Mask R-CNN:

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

Justin Johnson

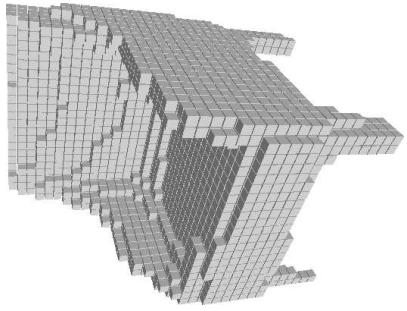
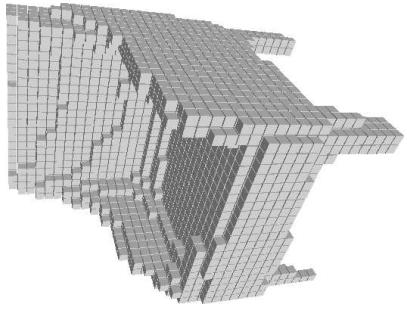
Lecture 17 - 5

November 13, 2019

# Focus on Two Problems today

Predicting 3D Shapes  
from single image

Processing 3D  
input data



Input Image

3D Shape

3D Shape



Chair

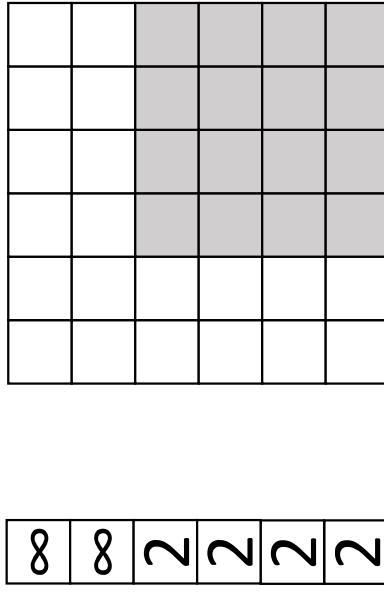
Many more topics in 3D Vision!

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

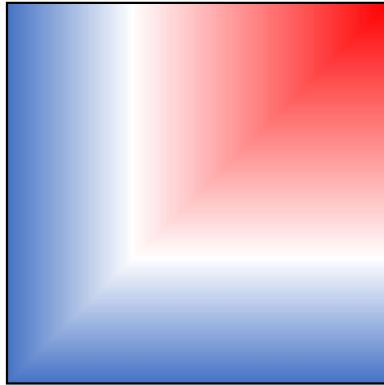
Many non-Deep Learning methods alive and well in 3D!

# 3D Shape Representations

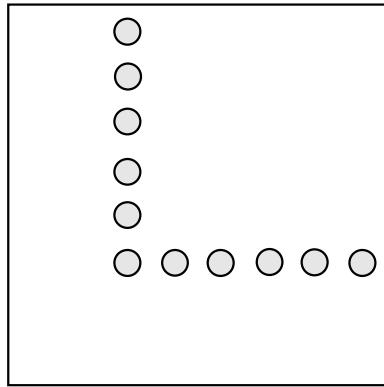
Depth Map



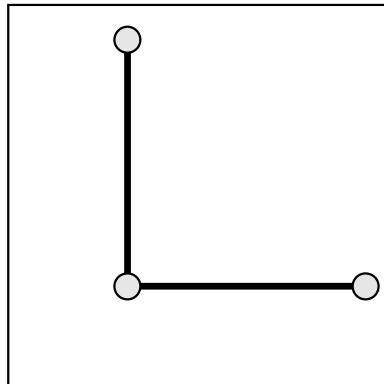
Voxel Grid



Implicit Surface



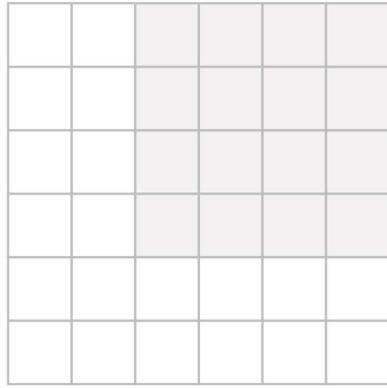
Pointcloud



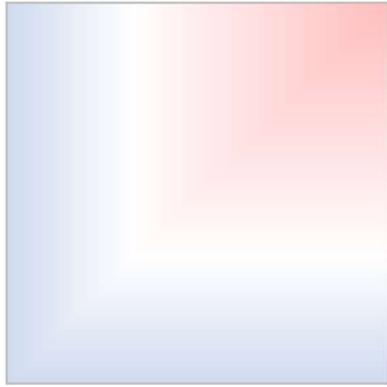
Mesh

# 3D Shape Representations

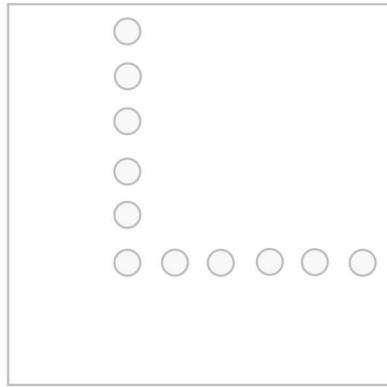
Depth Map



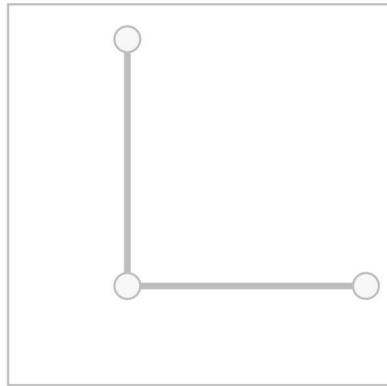
Voxel Grid



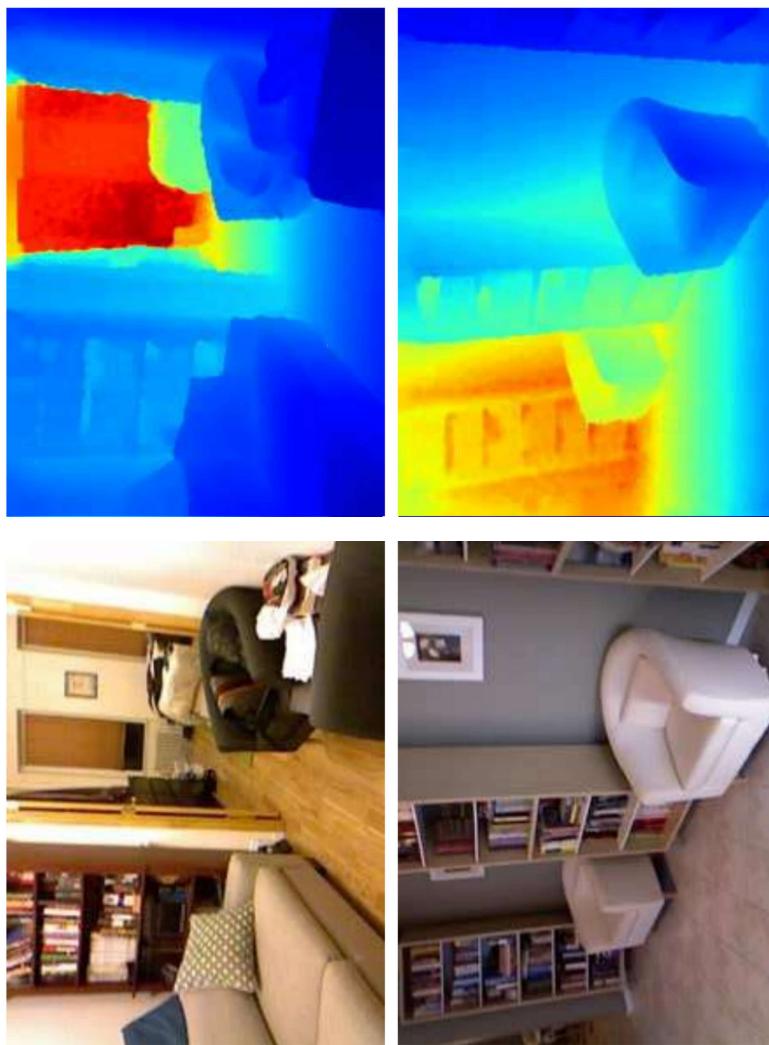
Pointcloud



Mesh



# 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:  $3 \times H \times W$  Depth Map:  $H \times W$

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

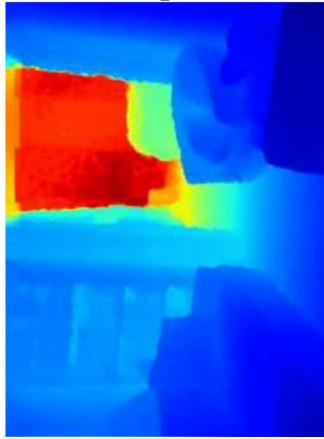
Justin Johnson

Lecture 17 - 10

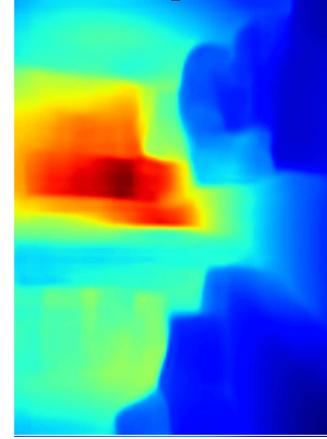
November 13, 2019

# Predicting Depth Maps

**Predicted Depth Image:**  
 $1 \times H \times W$

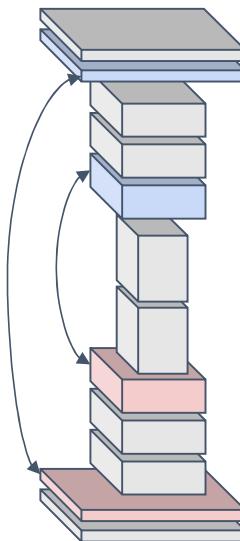


**Per-Pixel Loss**  
(L2 Distance)



**Predicted Depth Image:**  
 $1 \times H \times W$

**Fully Convolutional network**



**RGB Input Image:**  
 $3 \times H \times W$

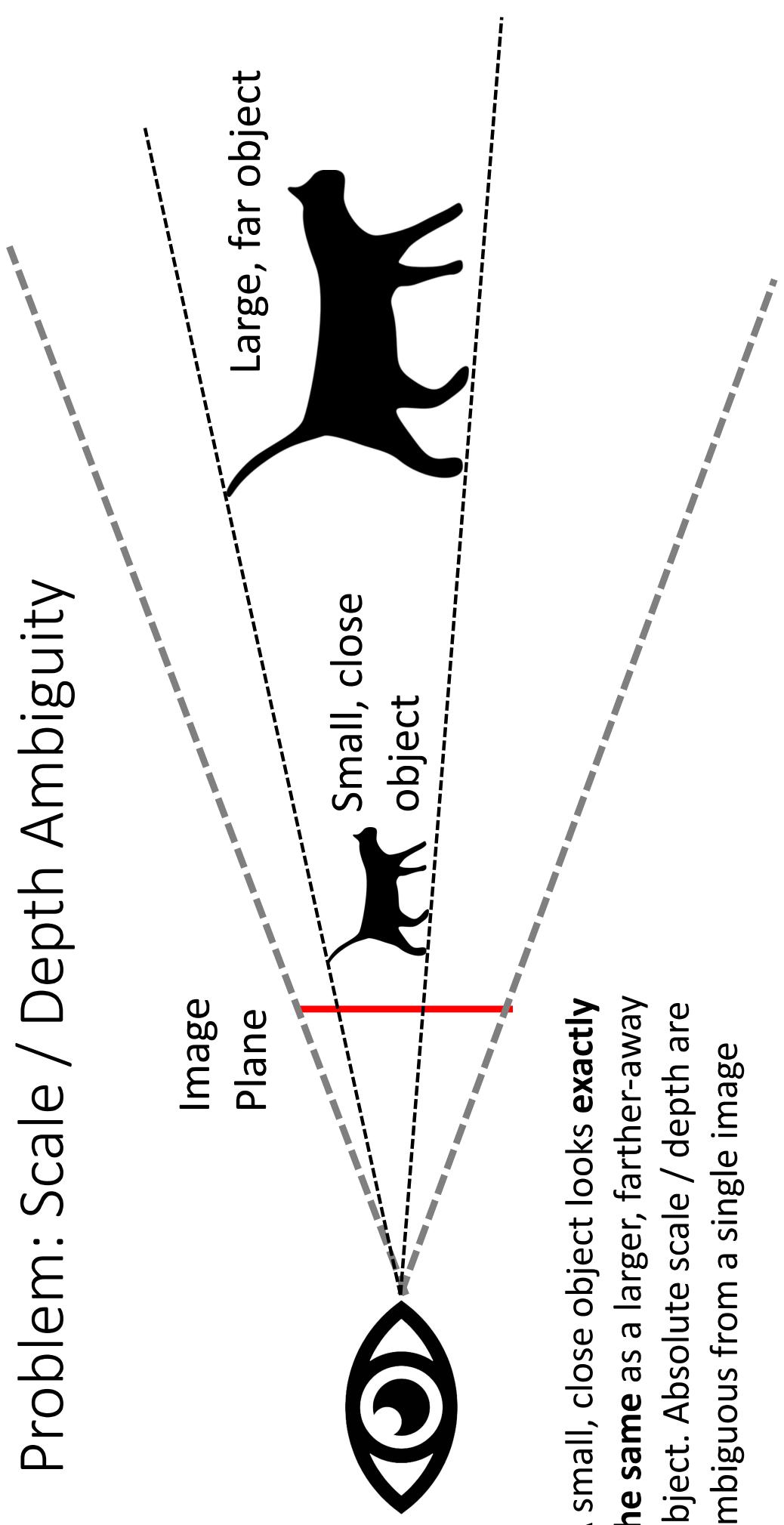
Eigen, Puhrsch, and Fergus, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network", NeurIPS 2014  
Eigen and Fergus, "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture", ICCV 2015

Justin Johnson

Lecture 17 - 11

November 13, 2019

# Problem: Scale / Depth Ambiguity



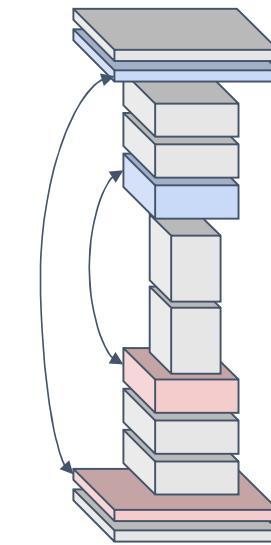
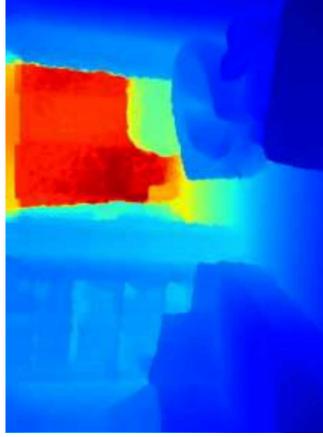
A small, close object looks **exactly the same** as a larger, farther-away object. Absolute scale / depth are ambiguous from a single image

## Predicting Depth Maps

**Predicted Depth Image:**  
 $1 \times H \times W$

### Scale invariant loss

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



**RGB Input Image:**  
 $3 \times H \times W$

**Fully Convolutional network**  
**Predicted Depth Image:**  
 $1 \times H \times W$

Eigen, Puhrsch, 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



Normals:  $3 \times H \times W$

RGB Image:  $3 \times H \times W$

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

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

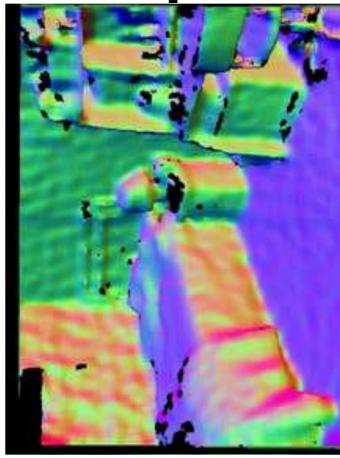
Justin Johnson

Lecture 17 - 14

November 13, 2019

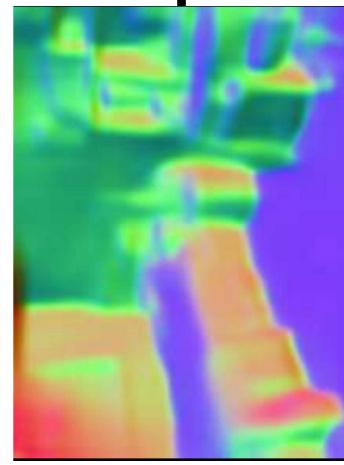
# Predicting Normals

Ground-truth Normals:  
 $3 \times H \times W$



Per-Pixel Loss:

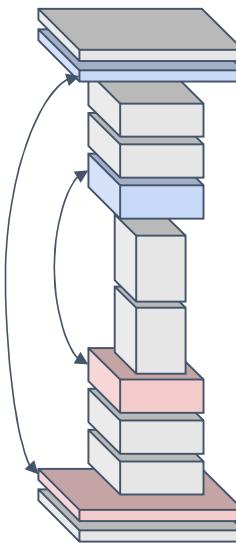
$$(x \cdot y) / (|x| |y|)$$



Recall:

$$\begin{aligned} x \cdot y \\ = |x| |y| \cos \theta \end{aligned}$$

Predicted Normals:  
 $3 \times H \times W$



Fully Convolutional  
network

RGB Input Image:  
 $3 \times H \times W$

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

Justin Johnson

Lecture 17 - 15

November 13, 2019

# 3D Shape Representations

Depth  
Map

Voxel  
Grid

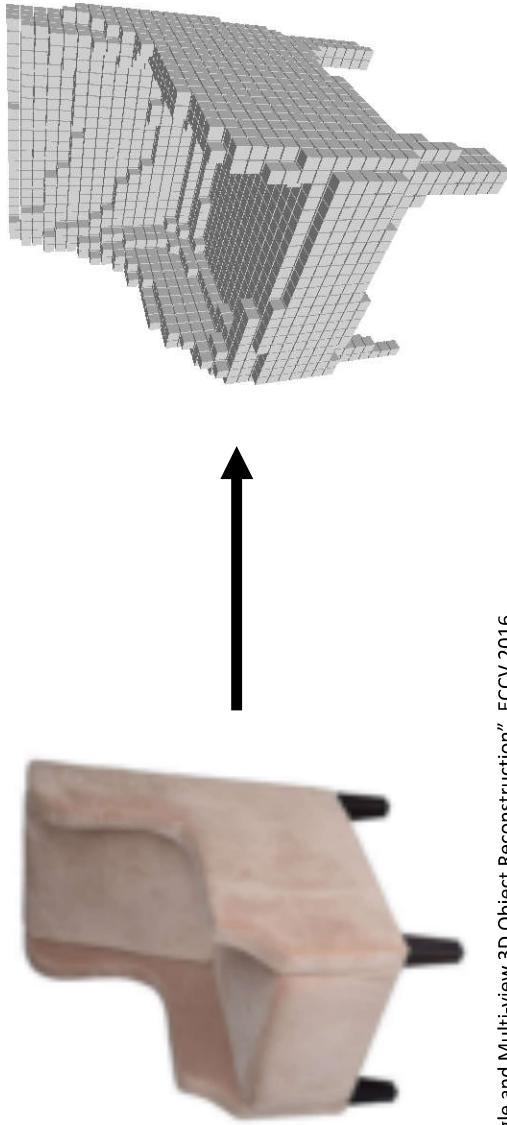
Implicit  
Surface

Pointcloud

Mesh

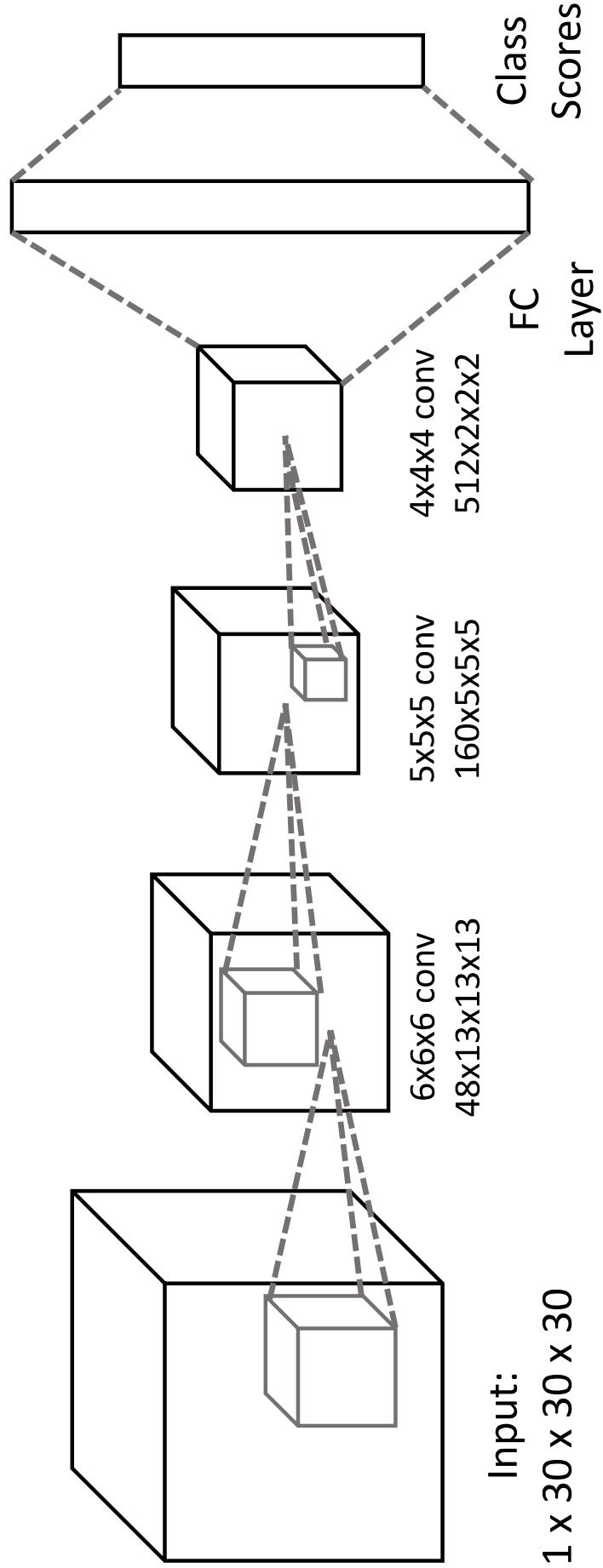
# 3D Shape Representations: VoxelS

- Represent a shape with a  $V \times V \times V$  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



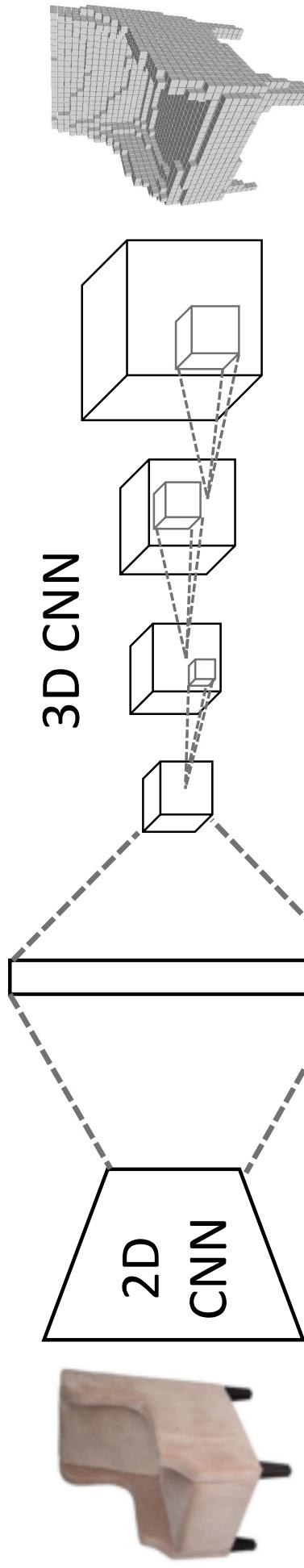
Wu et al, "3D ShapeNets: A Deep Representation for Volumetric Shapes", CVPR 2015

Justin Johnson

Lecture 17 - 18

November 13, 2019

# Generating Voxel Shapes: 3D Convolution



Input image:  
 $3 \times 112 \times 112$

2D Features:  
 $C \times H \times W$

3D Features:  
 $C' \times D' \times H' \times W'$

Voxels:  
 $1 \times V \times V \times V$

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

Train with per-voxel cross-entropy loss

# Generating Voxel Shapes: "Voxel Tubes"

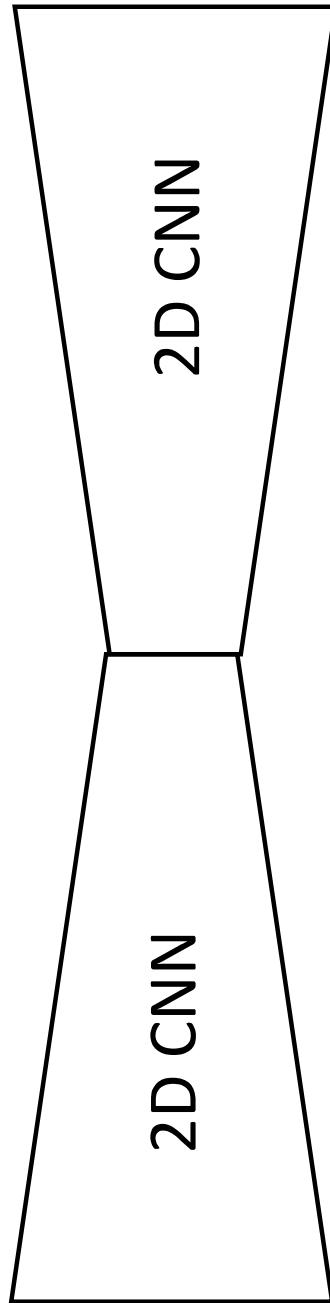
Input image:  
 $3 \times 112 \times 112$



2D Features:  
 $C \times H \times W$

Voxels:  
 $V \times V \times V$

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



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

Train with per-voxel cross-entropy loss

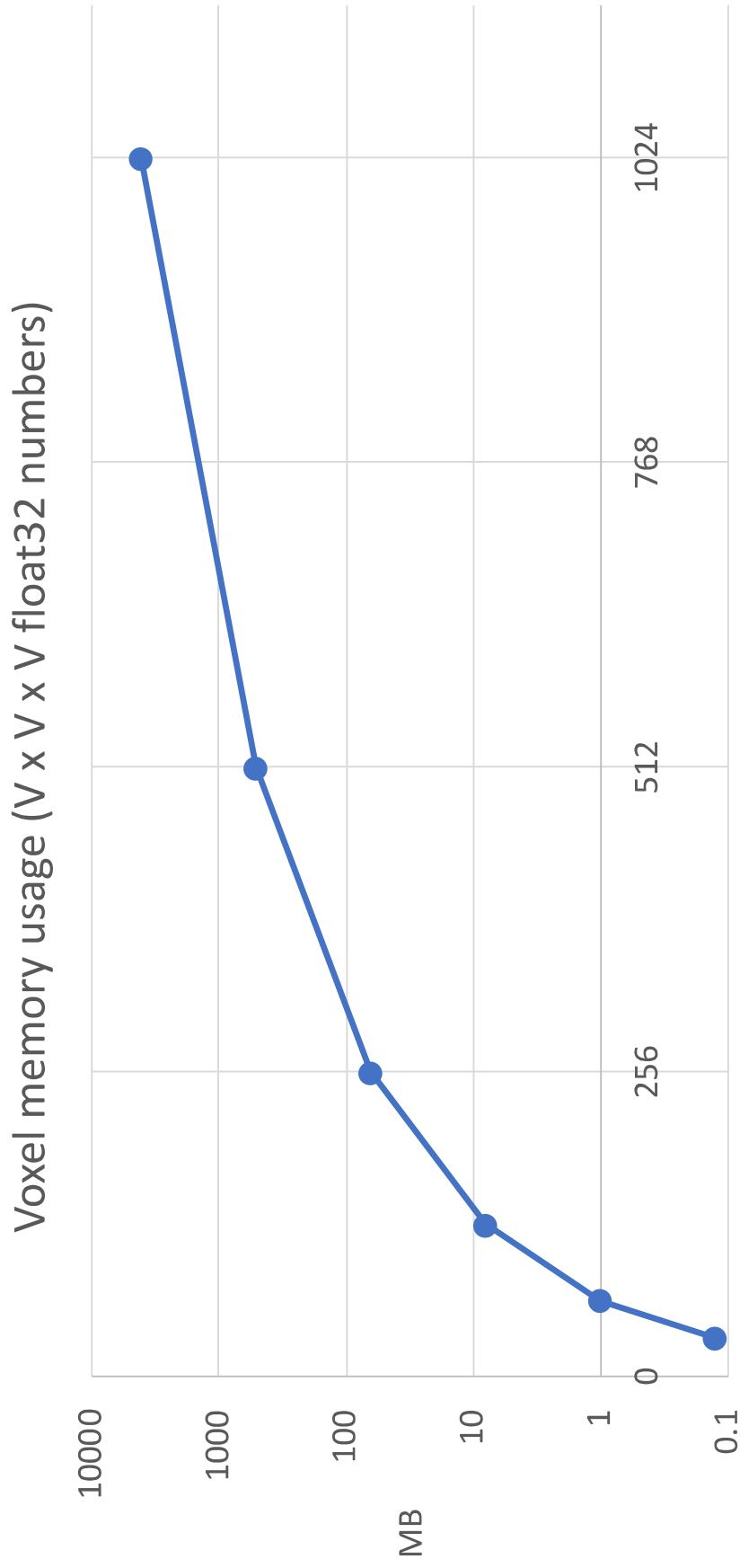
Justin Johnson

Lecture 17 - 20

November 13, 2019

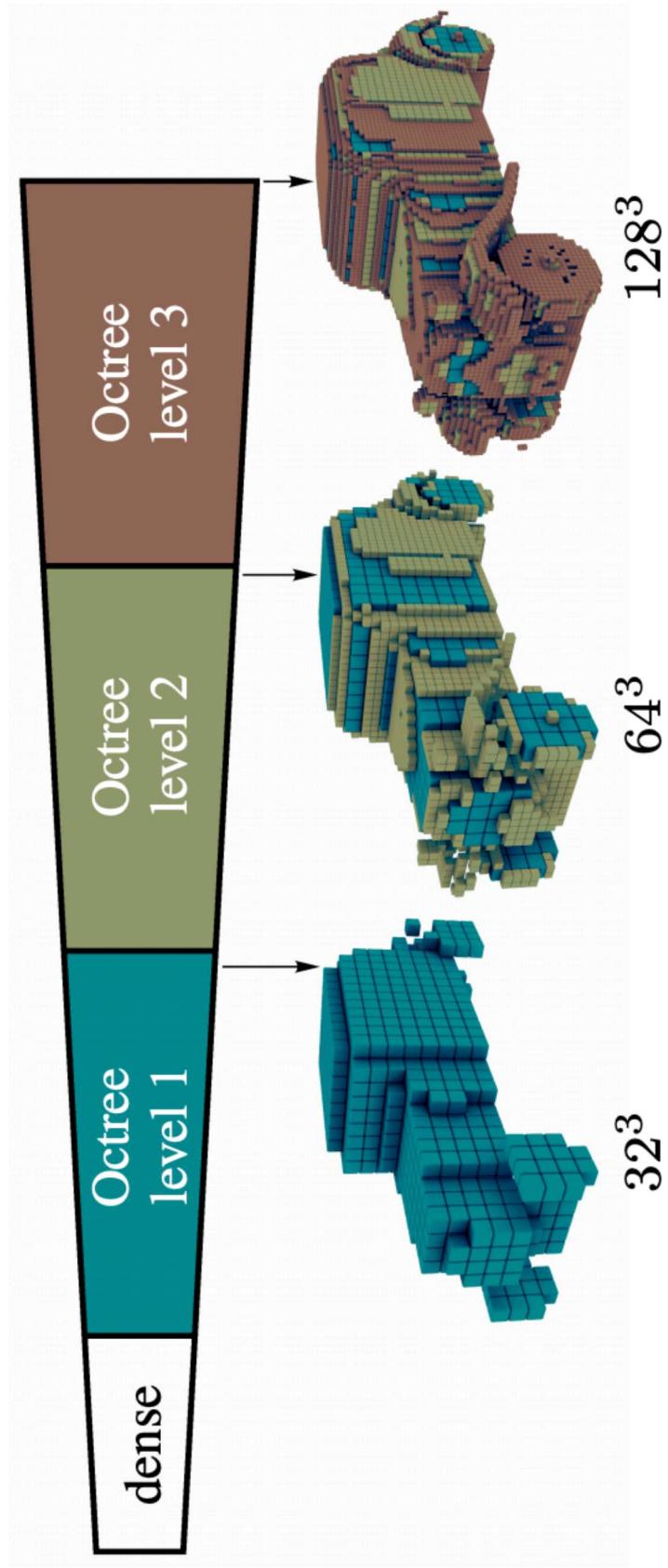
# Voxel Problems: Memory Usage

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



# Scaling Voxels: Oct-Trees

Use voxel grids with heterogeneous resolution!



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

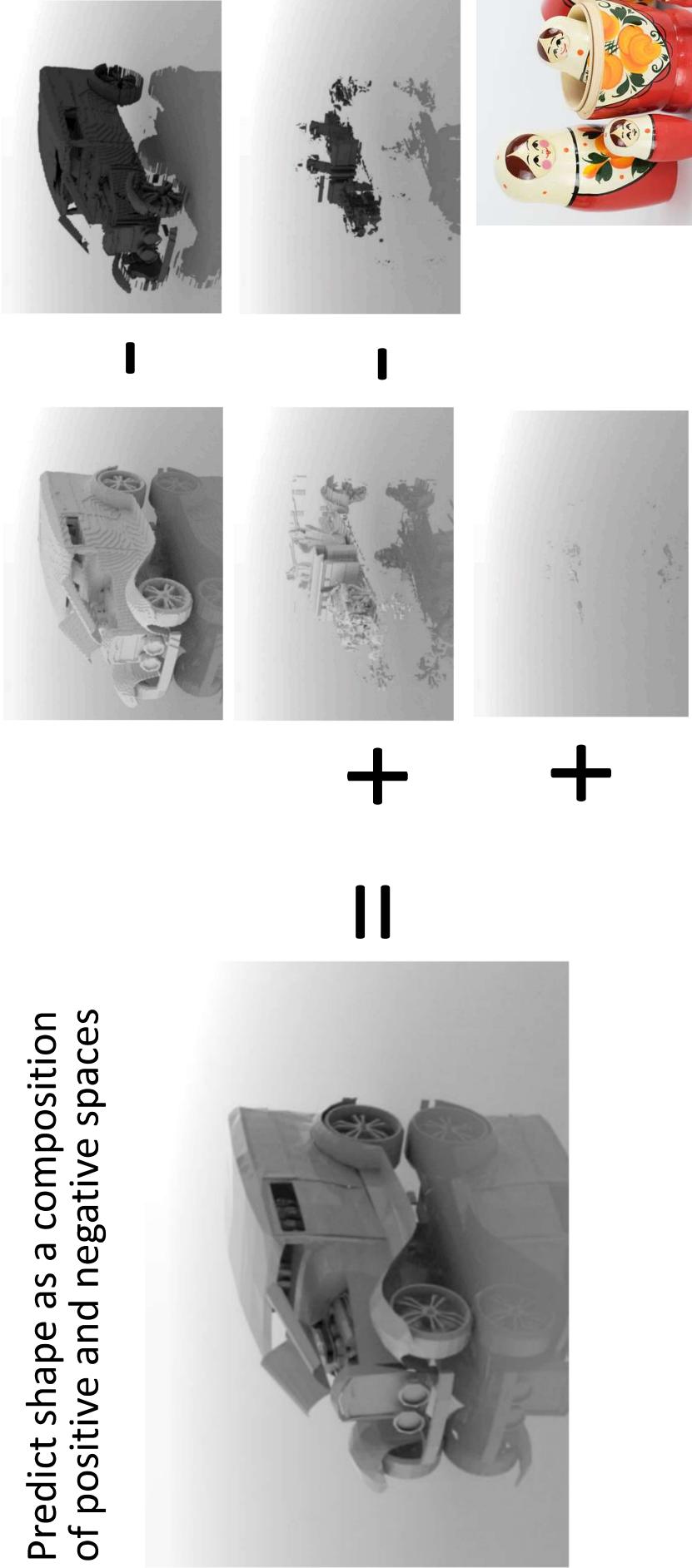
Justin Johnson

Lecture 17 - 22

November 13, 2019

# 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

Doll image is licensed under CC-BY 2.0.

Justin Johnson

Lecture 17 - 23

November 13, 2019

# 3D Shape Representations

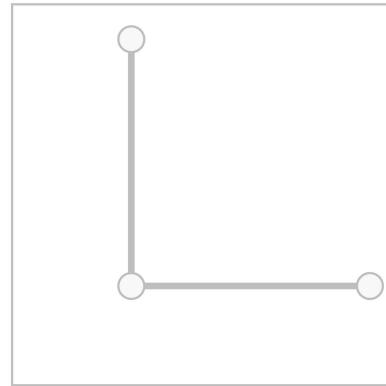
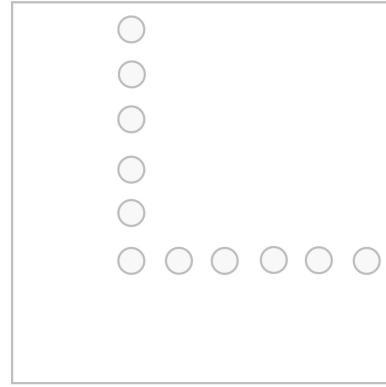
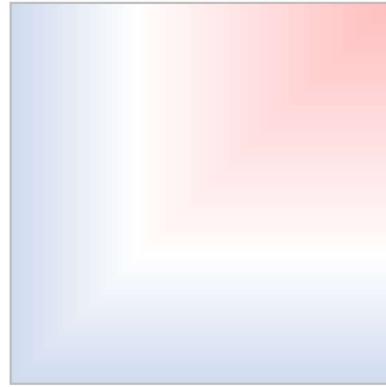
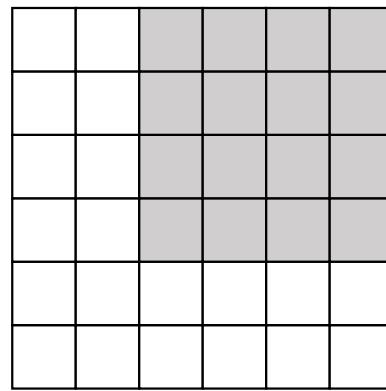
Depth  
Map

Voxel  
Grid

Implicit  
Surface

Pointcloud

Mesh



∞ ∞ 2 2 2 2

# 3D Shape Representations

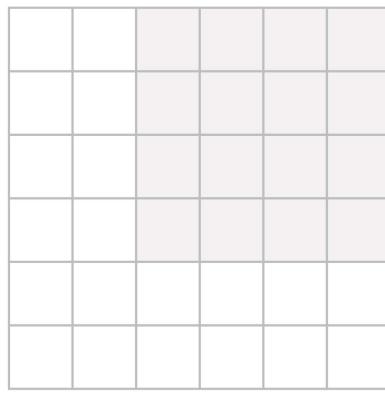
Depth Map

Voxel Grid

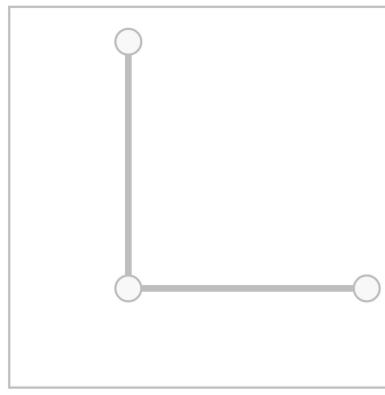
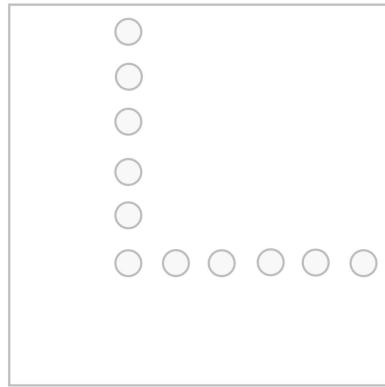
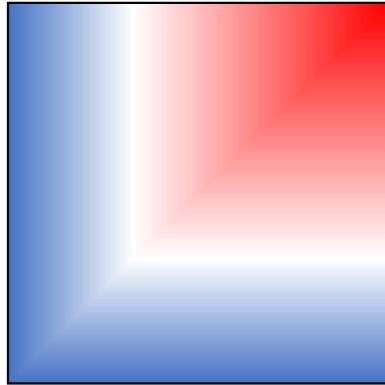
Implicit Surface

Pointcloud

Mesh



∞	∞	2	2	2	2
---	---	---	---	---	---

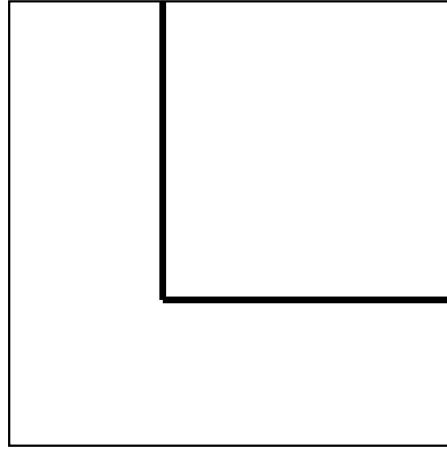
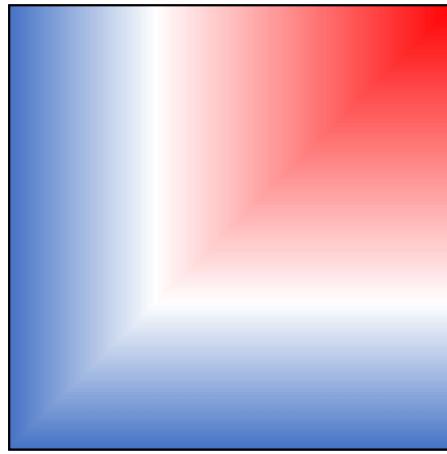


# 3D Shape Representations: Implicit Functions

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

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

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



Implicit function

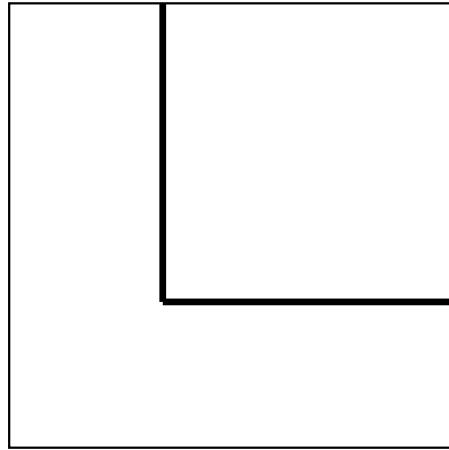
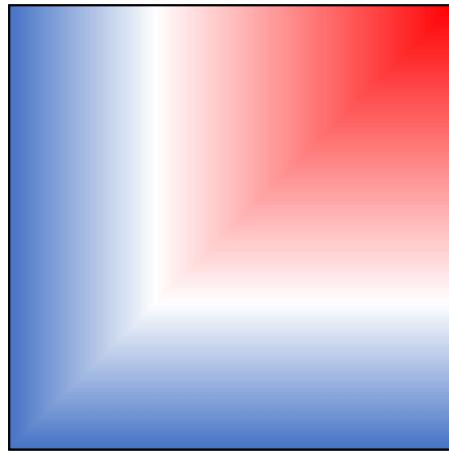
Explicit Shape

# 3D Shape Representations: Implicit Functions

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

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

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



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

Implicit function

Explicit Shape

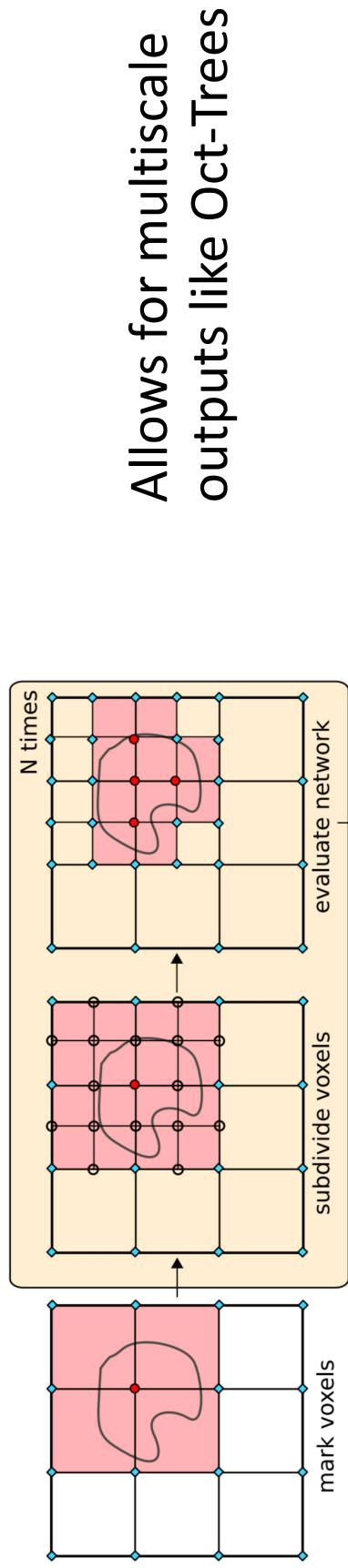
# 3D Shape Representations: Implicit Functions

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

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

The surface of the 3D object is the level set

$$\{x : \phi(x) = \frac{1}{2}\}$$



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

Justin Johnson

Lecture 17 - 28

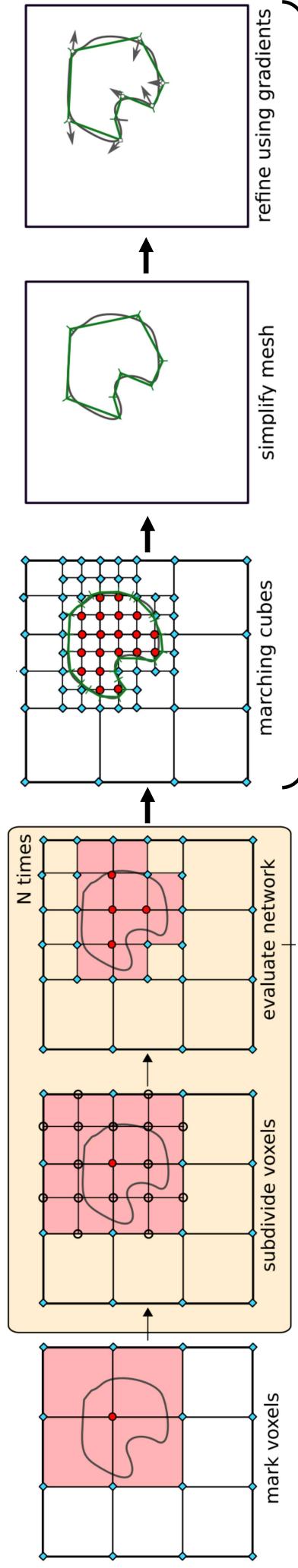
November 13, 2019

# 3D Shape Representations: Implicit Functions

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

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

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



Extracting explicit shape outputs  
requires post-processing

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

Justin Johnson

Lecture 17 - 29

November 13, 2019

# 3D Shape Representations

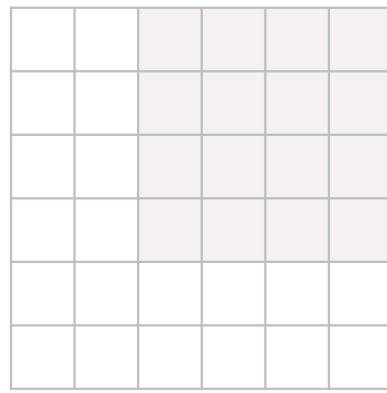
Depth Map

Voxel Grid

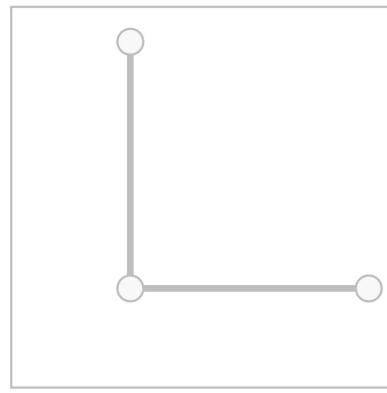
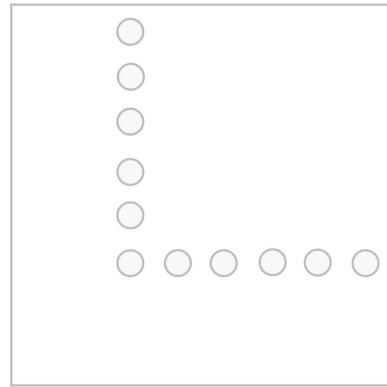
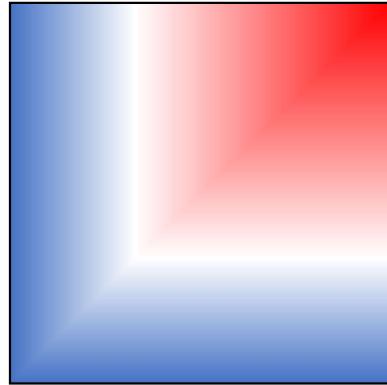
Implicit Surface

Pointcloud

Mesh



∞	∞	2	2	2	2
---	---	---	---	---	---



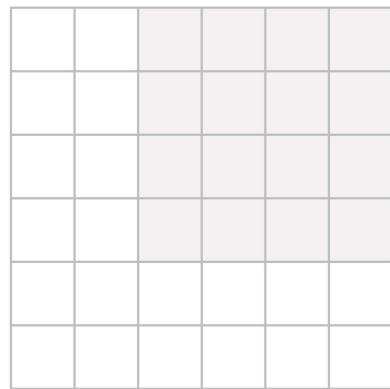
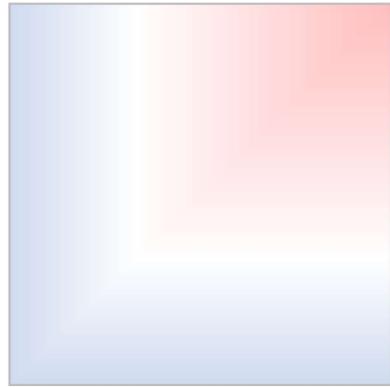
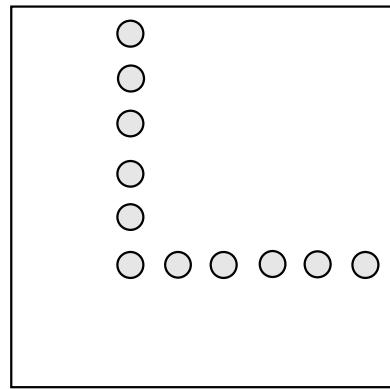
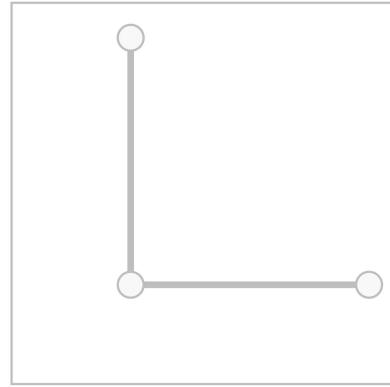
# 3D Shape Representations

Depth Map

Voxel Grid

Pointcloud

Mesh



8	8	2	2	2	2
---	---	---	---	---	---

# 3D Shape Representations: Point Cloud

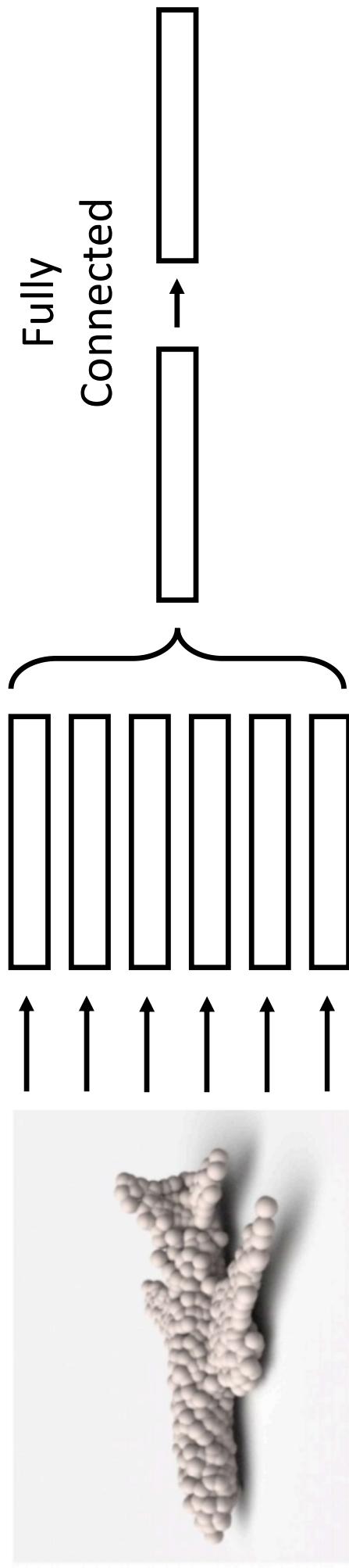
- Represent shape as a set of  $P$  points in 3D space
- (+) Can represent fine structures without huge numbers of points
- ( ) Requires new architecture, losses, etc
- (-) Doesn't explicitly represent the surface of the shape: extracting a mesh for rendering or other applications requires post-processing



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

# Processing Pointcloud Inputs: PointNet

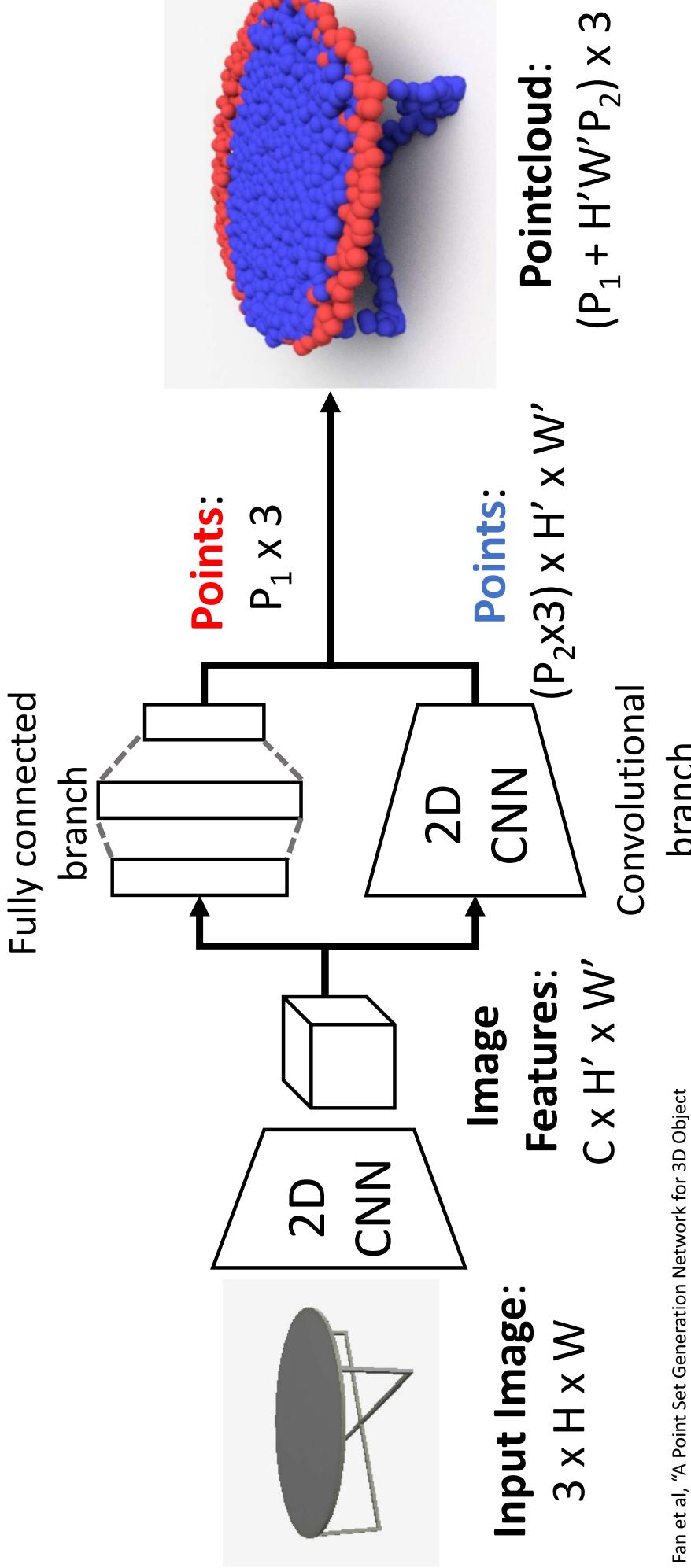
Run MLP on each point  
Max-Pool



**Input pointcloud:**  $P \times 3$   
**Point features:**  $P \times D$   
**Pooled vector:**  $P \times D$   
**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**!

# 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

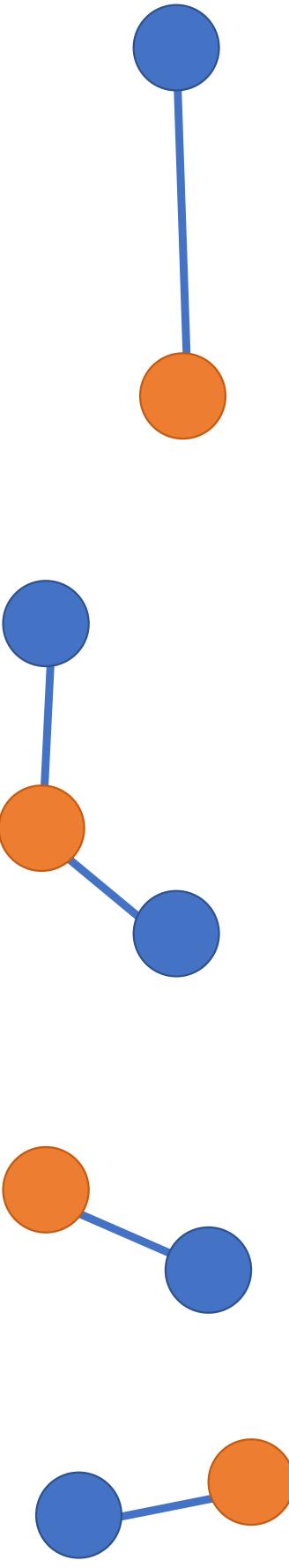
$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$

# 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

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



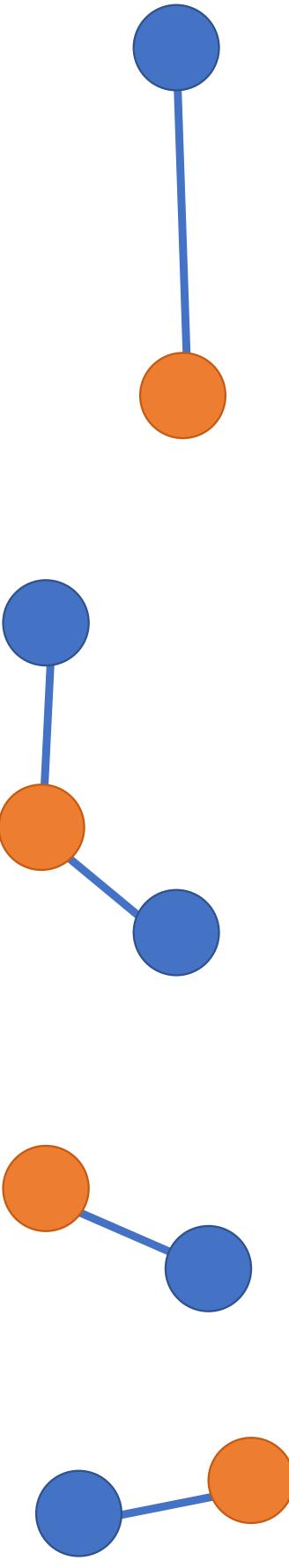
Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

# 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

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



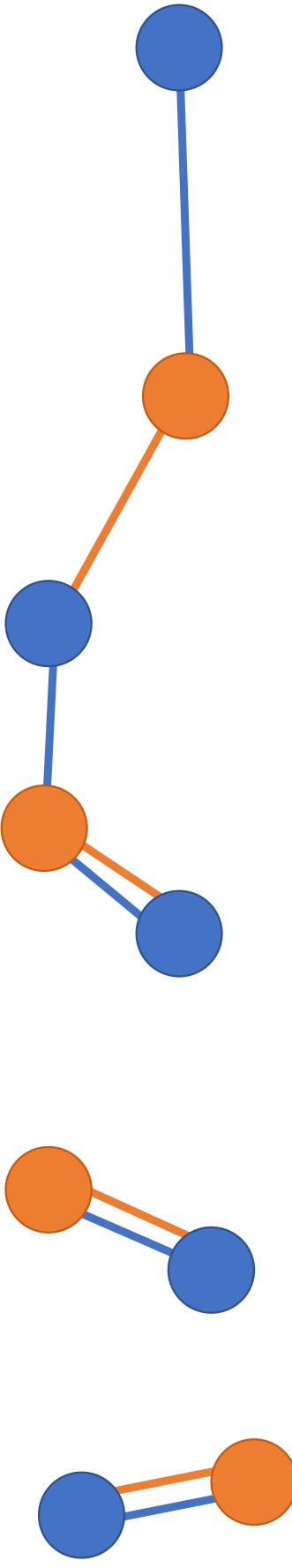
Fan et al., "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

# 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

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



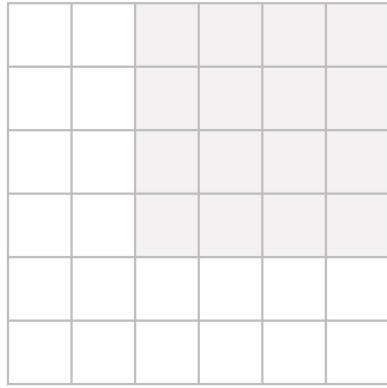
Fan et al., "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

# 3D Shape Representations

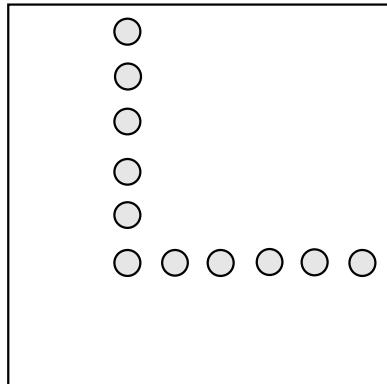
Depth Map



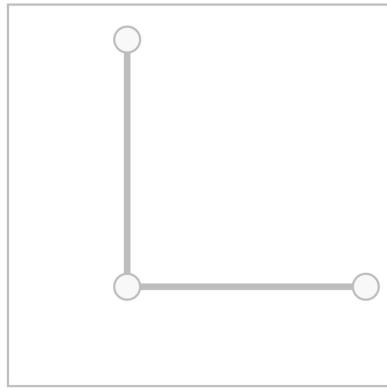
Voxel Grid



Pointcloud



Mesh



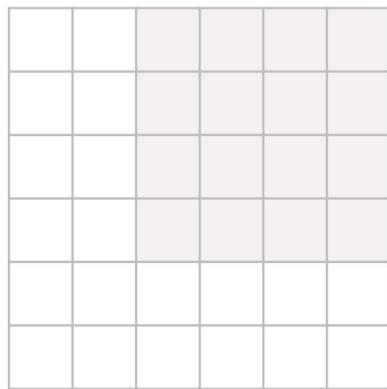
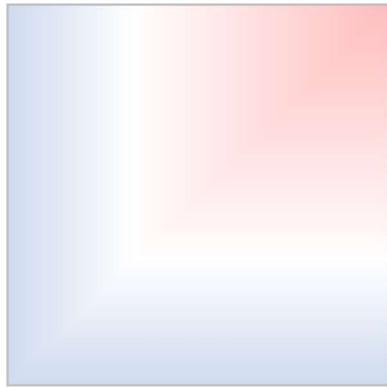
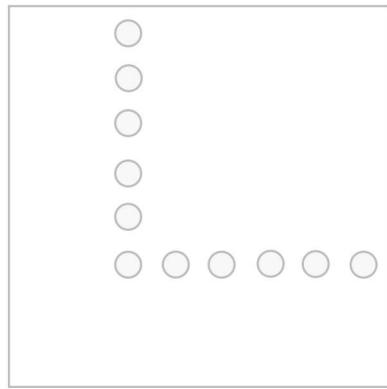
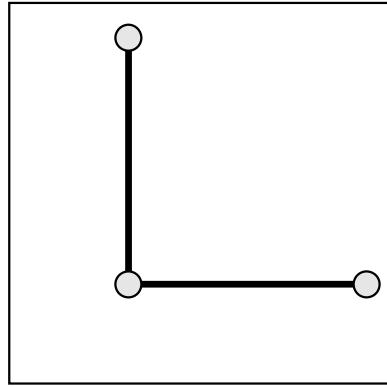
# 3D Shape Representations

Depth Map

Voxel Grid

Pointcloud

Mesh



# 3D Shape Representations: Triangle Mesh

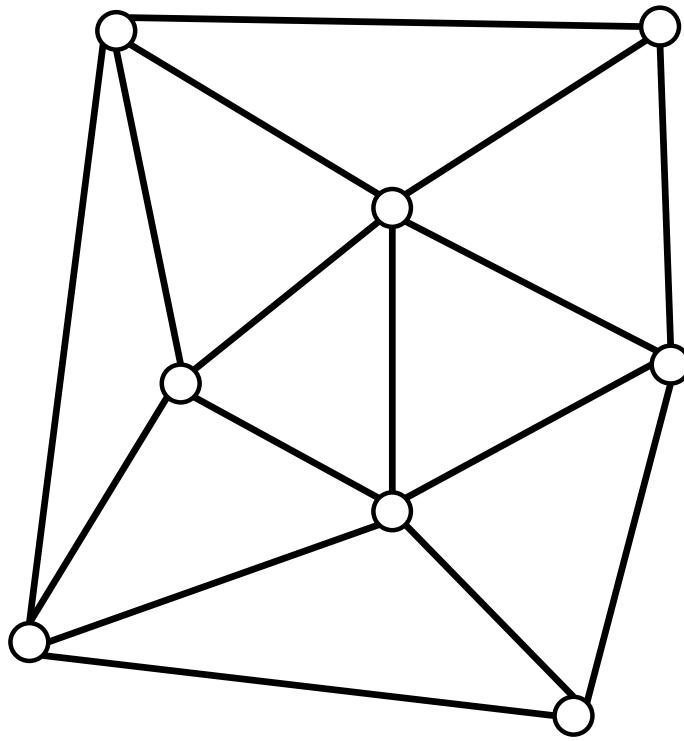
Represent a 3D shape as a set of triangles

**Vertices:** Set of  $V$  points in 3D space

**Faces:** Set of triangles over the vertices

(+) Standard representation for graphics

(+) Explicitly represents 3D shapes



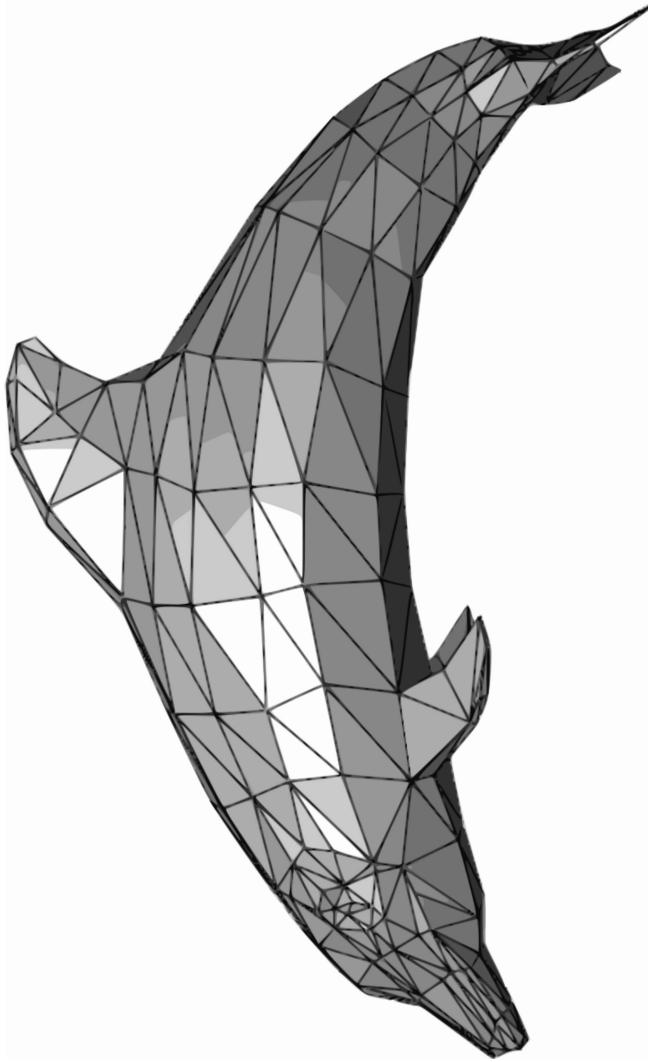
# 3D Shape Representations: Triangle Mesh

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

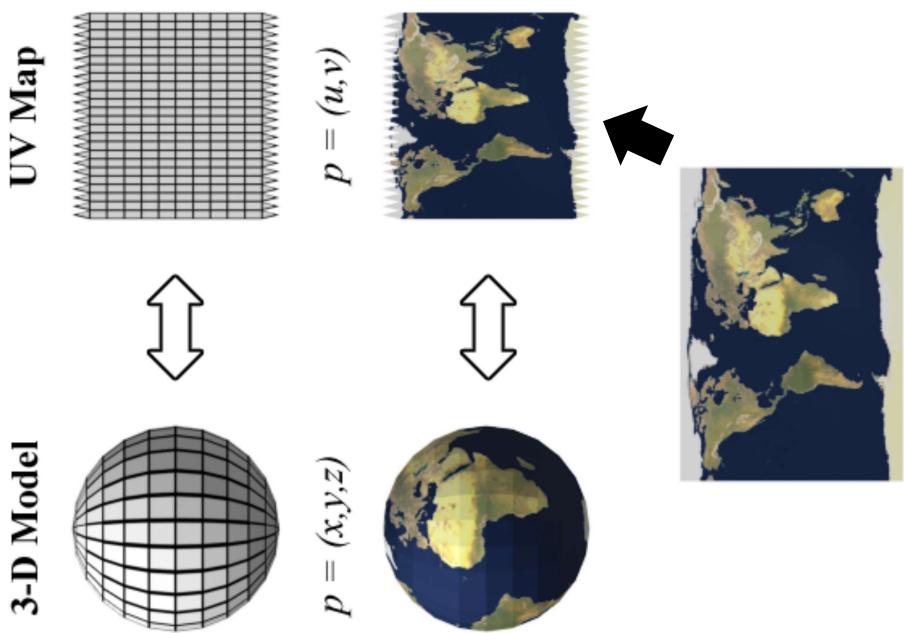
**Faces:** Set of triangles over the vertices

(+) 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.



UV mapping figure is licensed under CC BY-SA 3.0. Figure slightly reorganized.

# 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

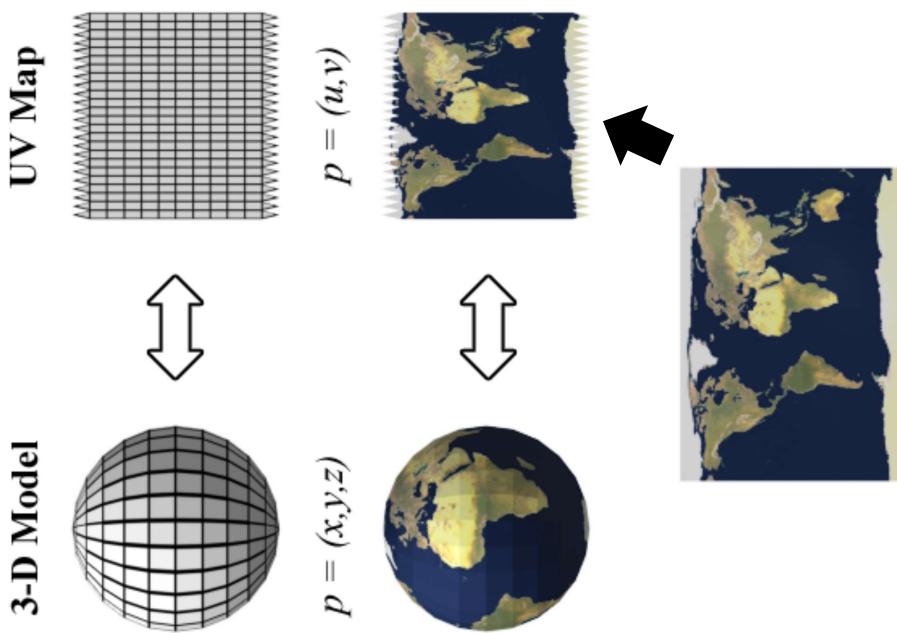
(+) Standard representation for graphics

(+) 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!

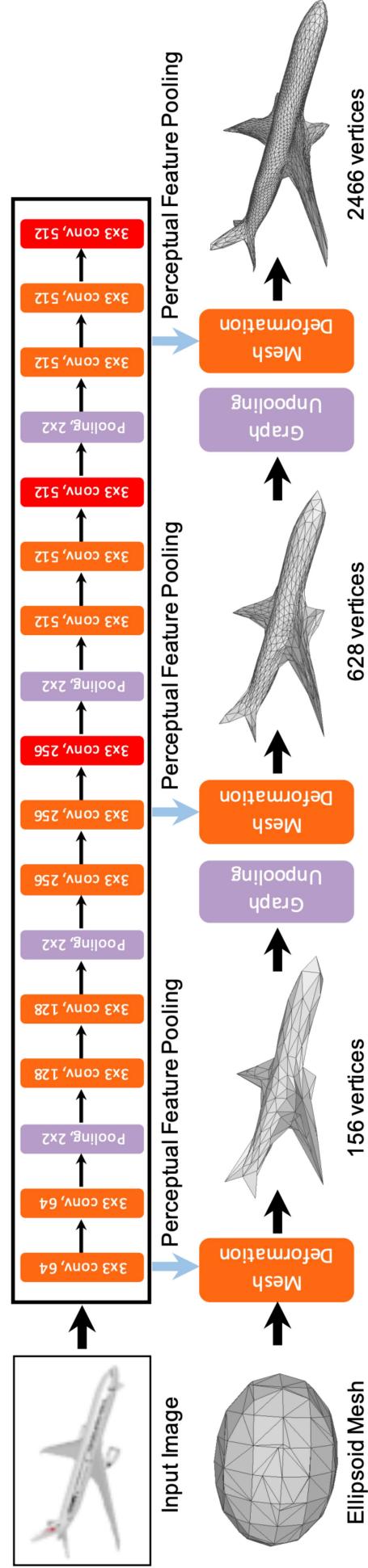


UV mapping figure is licensed under CC BY-SA 3.0. Figure slightly reorganized.

# 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

Justin Johnson

Lecture 17 - 46

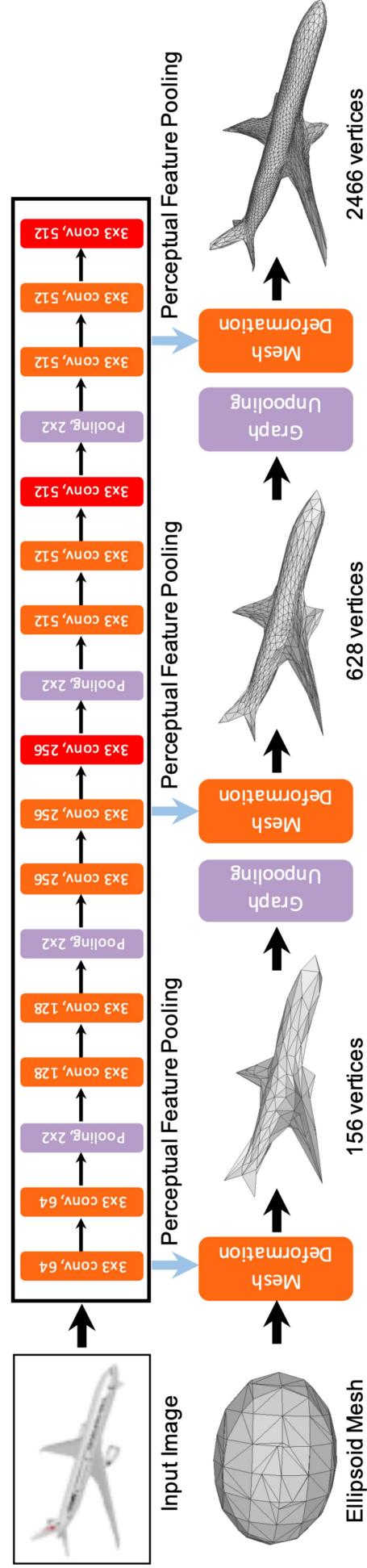
November 13, 2019

# Predicting Meshes: Pixel2Mesh

## Key ideas:

**Input:** Single RGB Image of an object  
**Output:** Triangle mesh for the object

- Iterative Refinement
- Graph Convolution
- Vertex Aligned-Features
- Chamfer Loss Function



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

Justin Johnson

Lecture 17 - 47

November 13, 2019

# 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

Justin Johnson

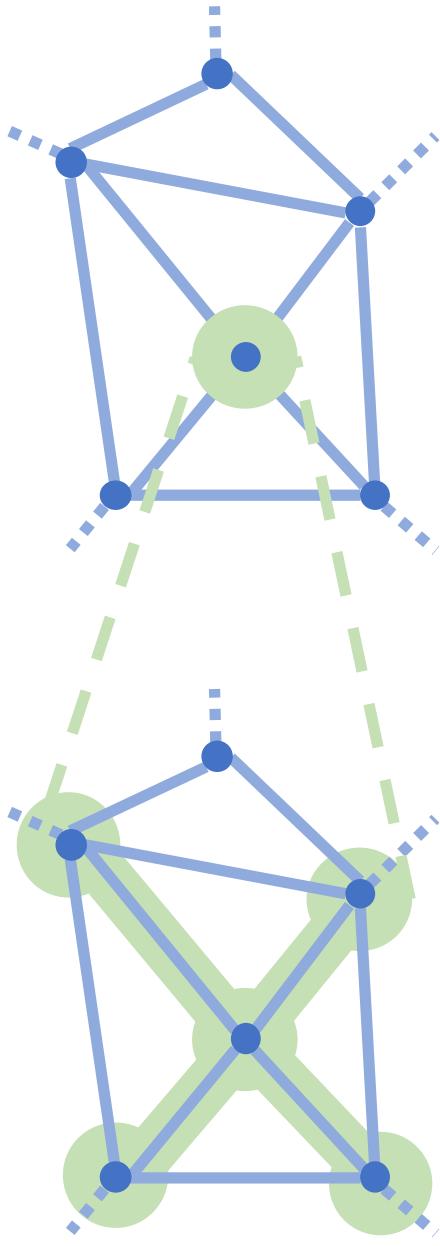
Lecture 17 - 48

November 13, 2019

# 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  $v_i$  depends on feature of neighboring vertices  $N(i)$

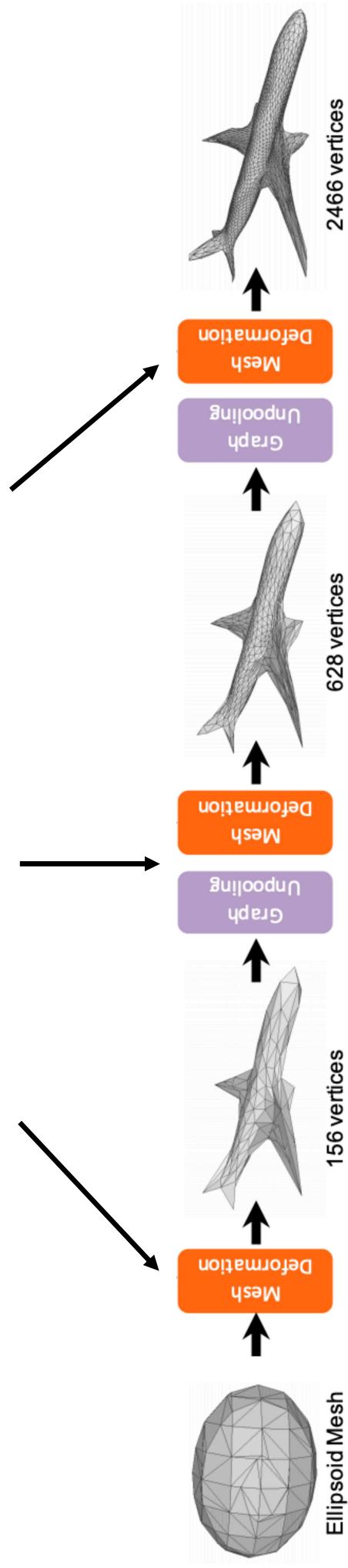
Use same weights  $W_0$  and  $W_1$  to compute all outputs

**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

Justin Johnson

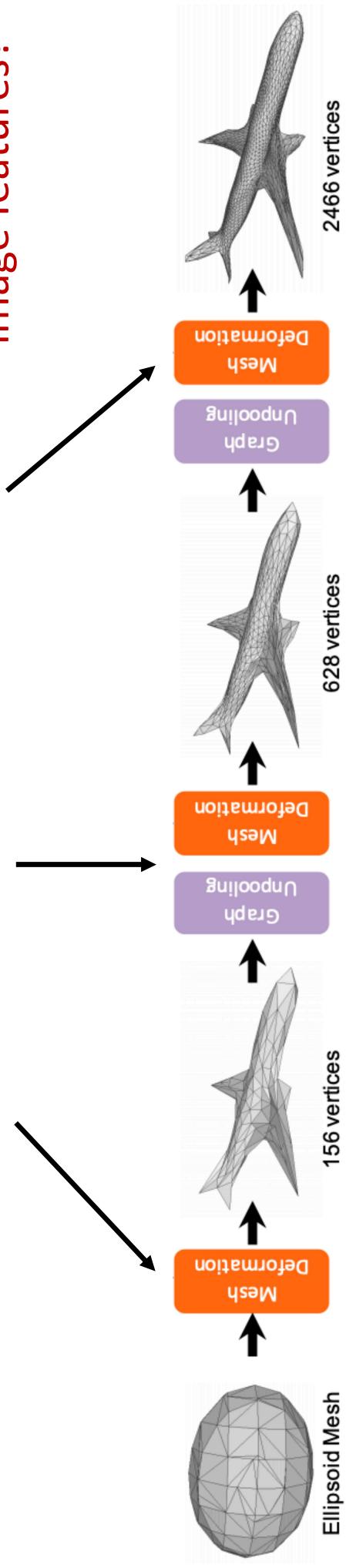
Lecture 17 - 50

November 13, 2019

# 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

Justin Johnson

Lecture 17 - 51

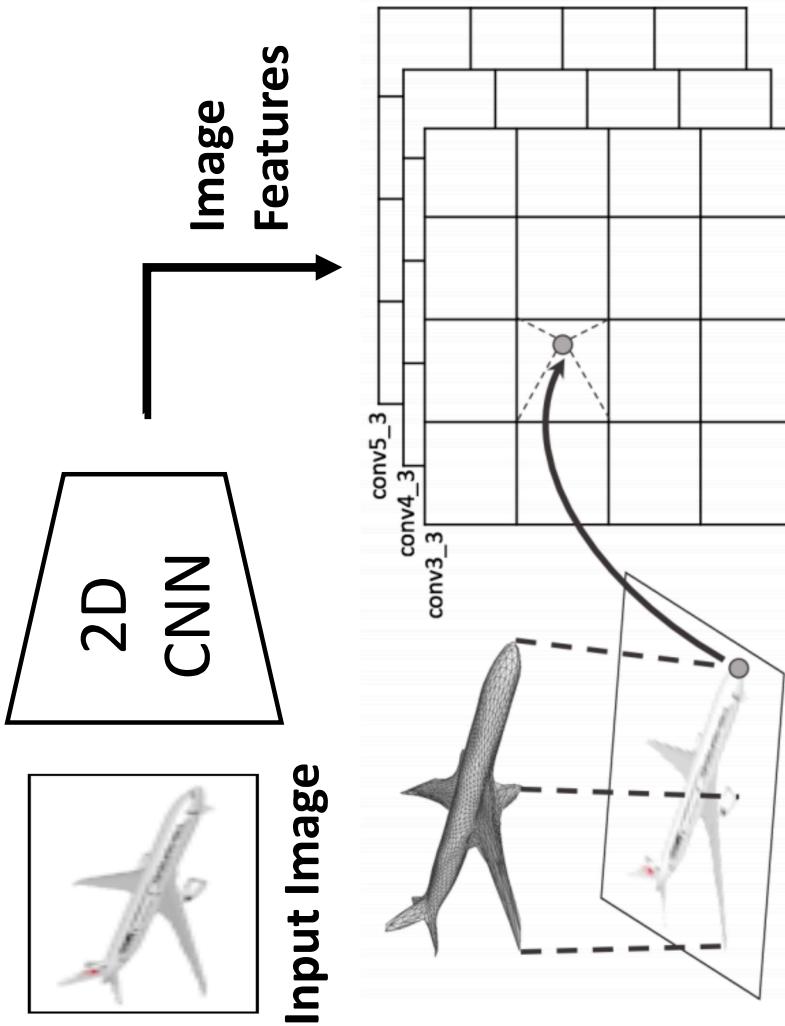
November 13, 2019

# 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

Justin Johnson

Lecture 17 - 52

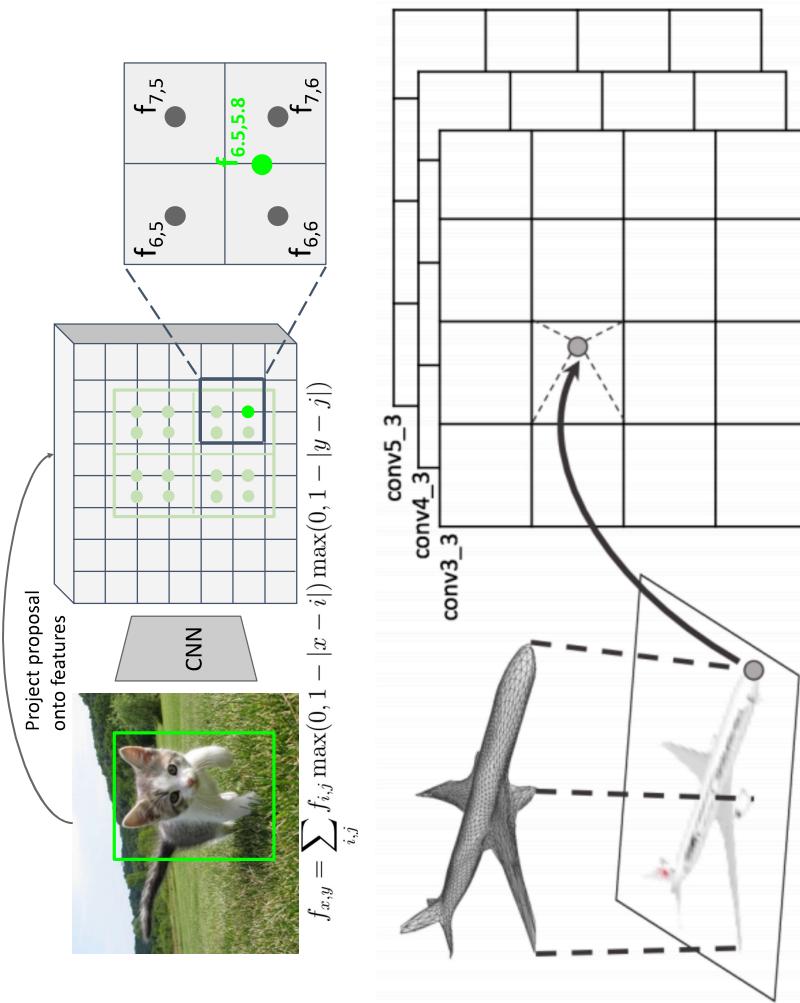
November 13, 2019

# 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



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

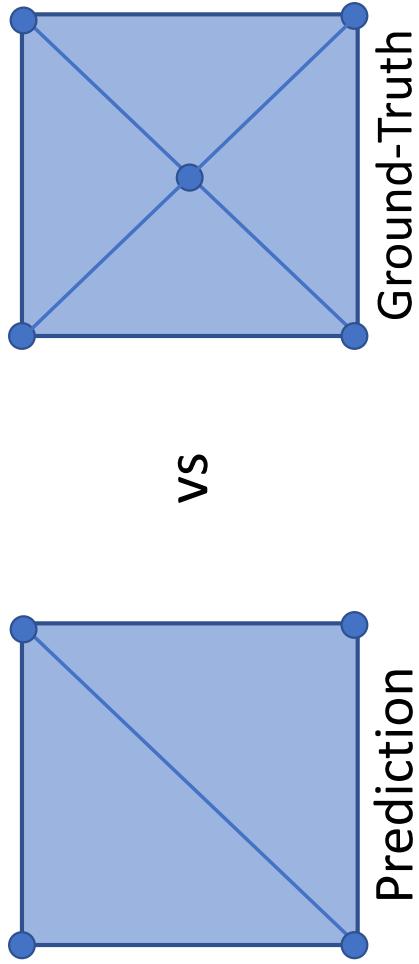
Justin Johnson

Lecture 17 - 53

November 13, 2019

# 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?



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

Justin Johnson

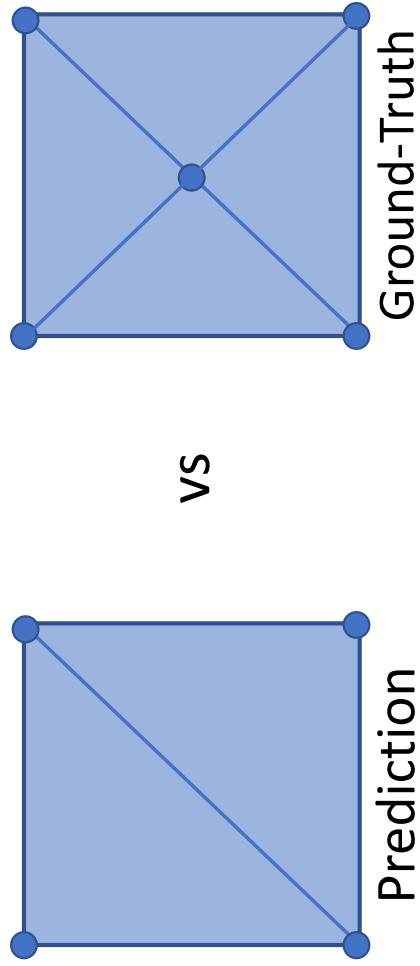
Lecture 17 - 54

November 13, 2019

# 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



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

Justin Johnson

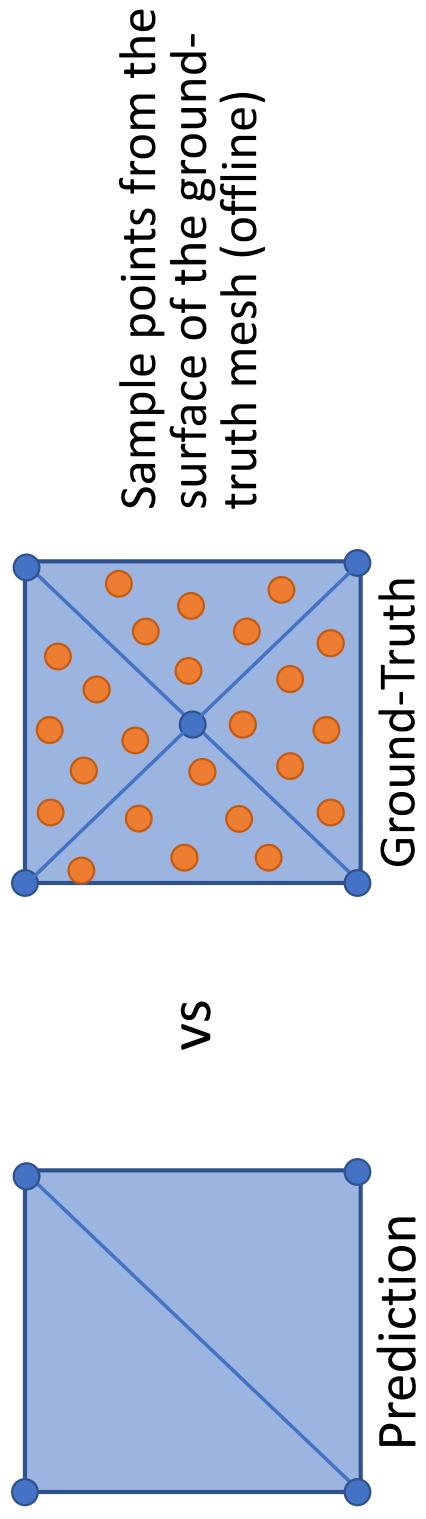
Lecture 17 - 55

November 13, 2019

# 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



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

Justin Johnson

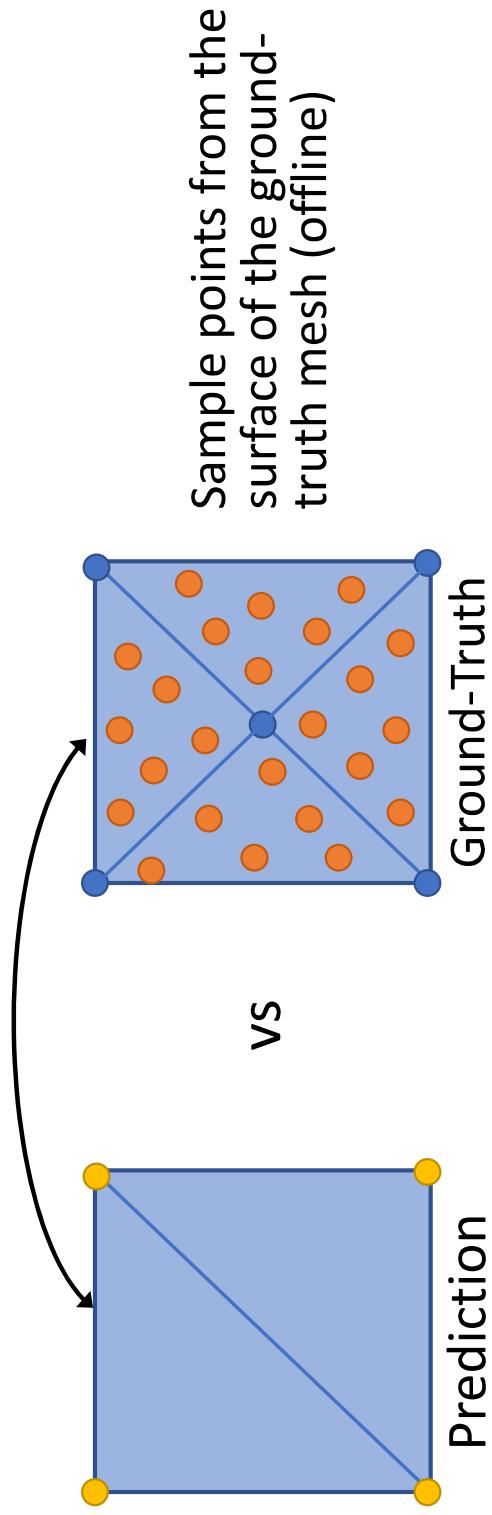
Lecture 17 - 56

November 13, 2019

# 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**



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

Justin Johnson

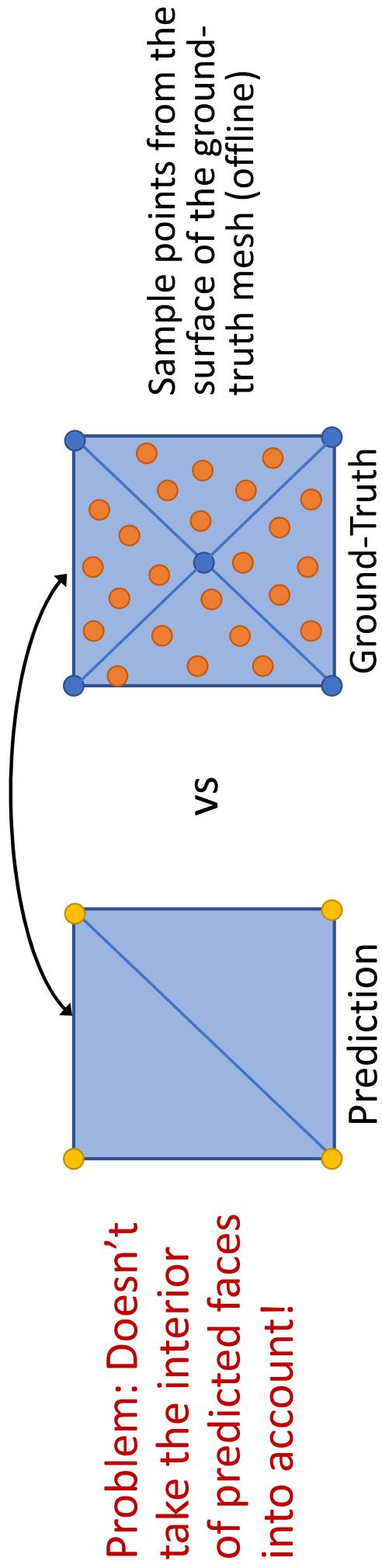
Lecture 17 - 57

November 13, 2019

# 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**



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

Justin Johnson

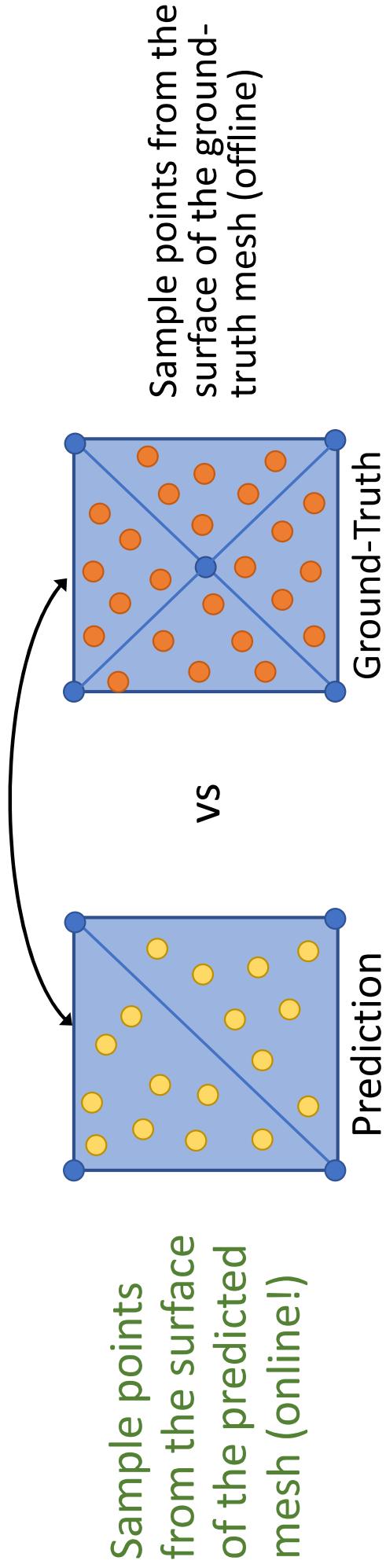
Lecture 17 - 58

November 13, 2019

# 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?

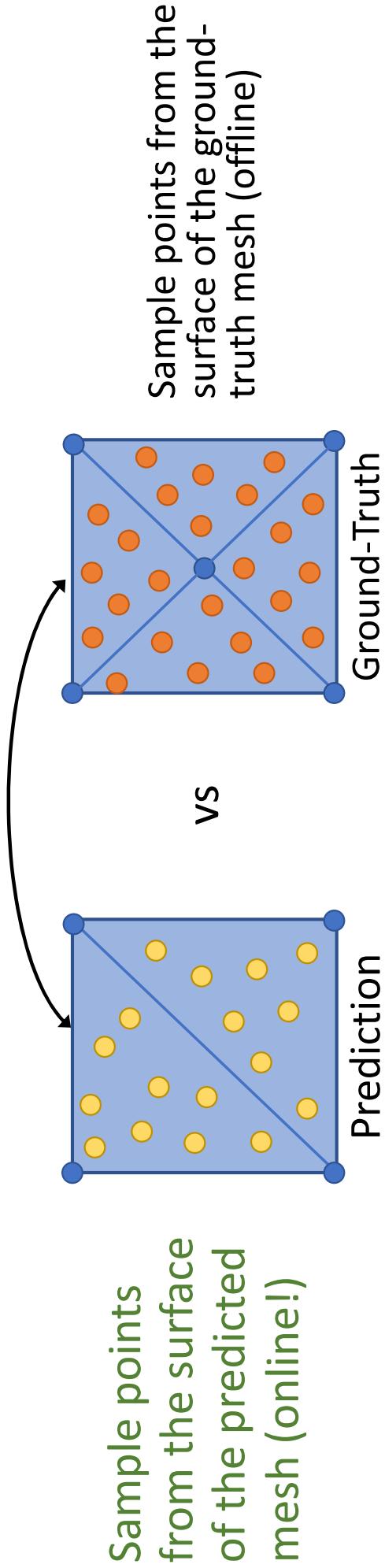
Loss = Chamfer distance between **predicted samples** and **ground-truth samples**



# 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**

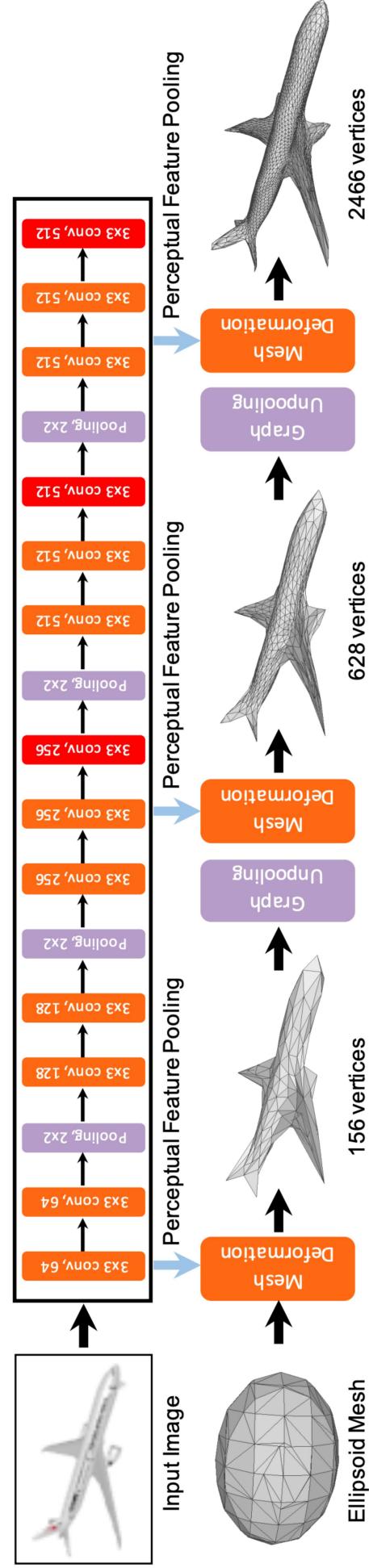


# Predicting Meshes: Pixel2Mesh

## Key ideas:

**Input:** Single RGB Image of an object  
**Output:** Triangle mesh for the object

- Iterative Refinement
- Graph Convolution
- Vertex Aligned-Features
- Chamfer Loss Function



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

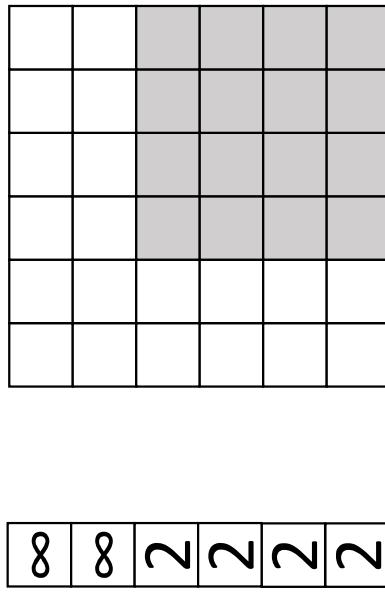
Justin Johnson

Lecture 17 - 61

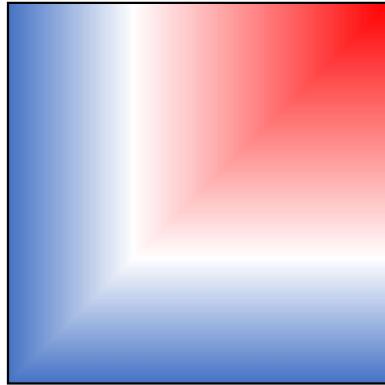
November 13, 2019

# 3D Shape Representations

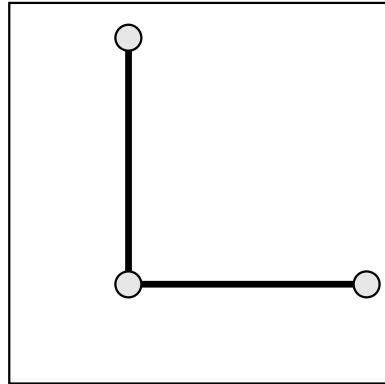
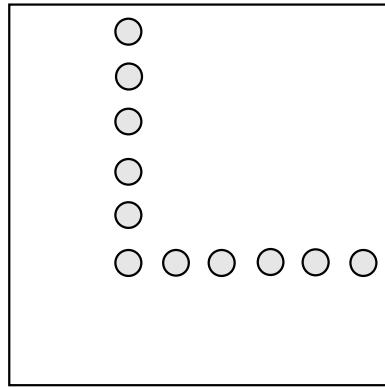
Depth Map



Voxel Grid



Implicit Surface

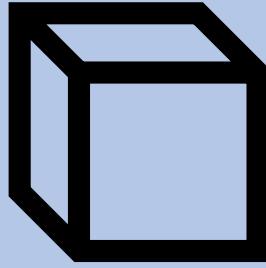


Pointcloud

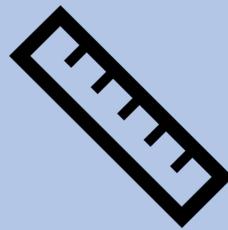
Mesh

# 3D Shape Prediction

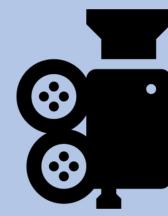
Shape Representations



Metrics



Camera Systems

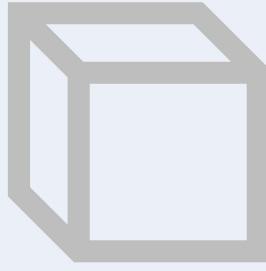


Datasets

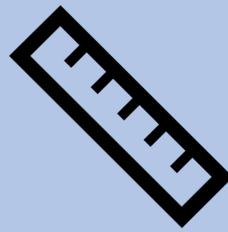


# 3D Shape Prediction

Shape Representations



Metrics



Camera Systems

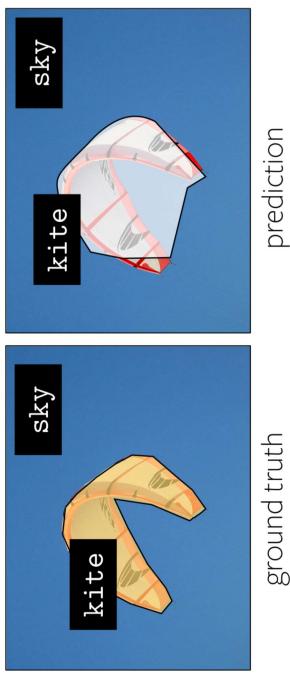


Datasets



# Shape Comparison Metrics: Intersection over Union

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



ground truth prediction

$$\text{IoU}(\text{kite}) = \frac{\text{area}(\text{kite} \cap \text{prediction})}{\text{area}(\text{kite} \cup \text{prediction})}$$

Figure credit: Alexander Kirillov

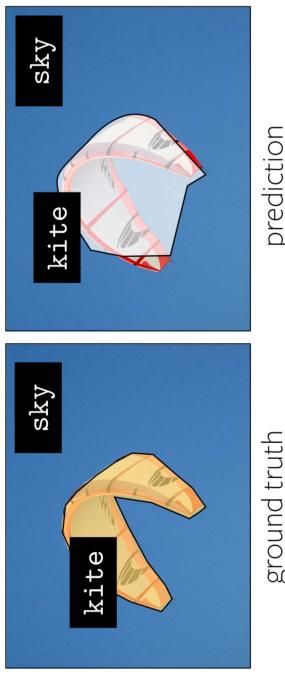
Justin Johnson

Lecture 17 - 65

November 13, 2019

# 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

$$\text{IoU}(\text{kite}) = \frac{\text{area}(\text{kite} \cap \text{sky})}{\text{area}(\text{kite} \cup \text{sky})}$$

Figure credit: Alexander Kirillov

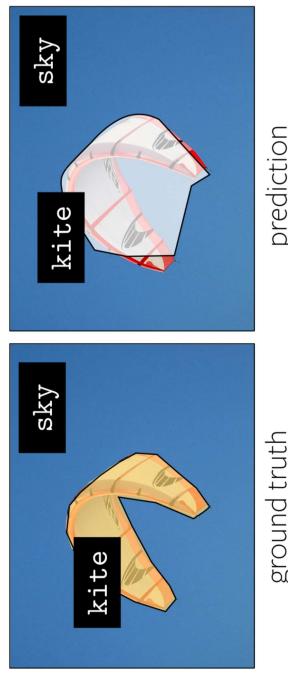
Justin Johnson

Lecture 17 - 66

November 13, 2019

# Shape Comparison Metrics: Intersection over Union

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



$$\text{IoU}(\text{kite}) = \frac{\text{area}(\text{kite} \cap \text{prediction})}{\text{area}(\text{kite} \cup \text{prediction})}$$

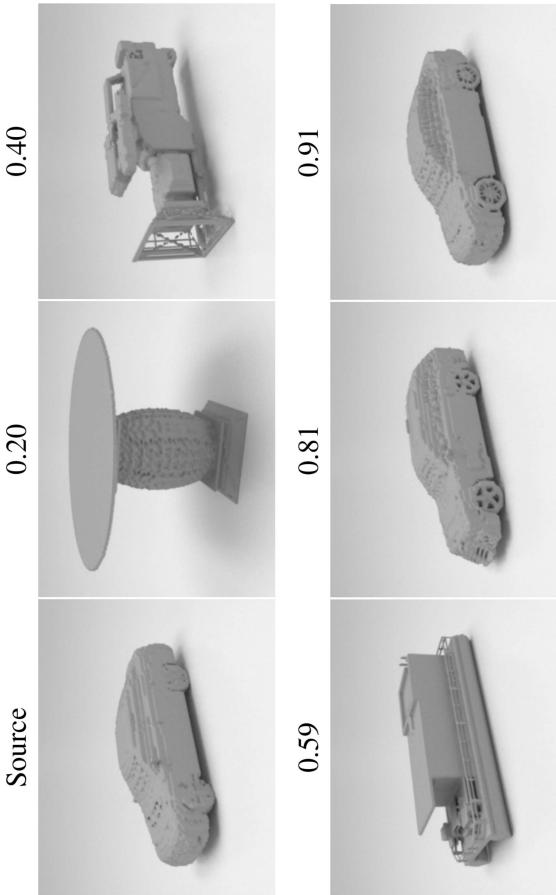


Figure credit: Alexander Kirillov

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

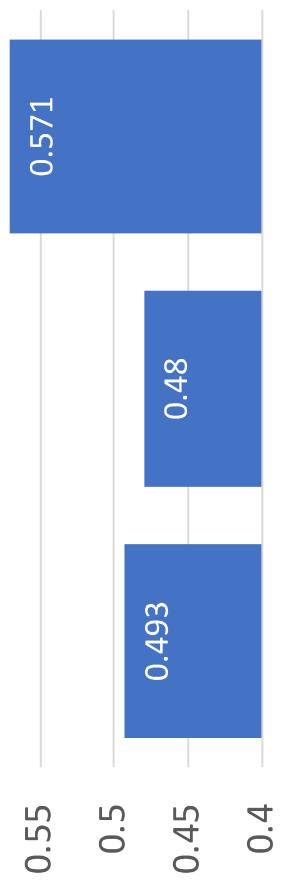
In 3D: Voxel IoU

**Problem:** Cannot capture thin structures

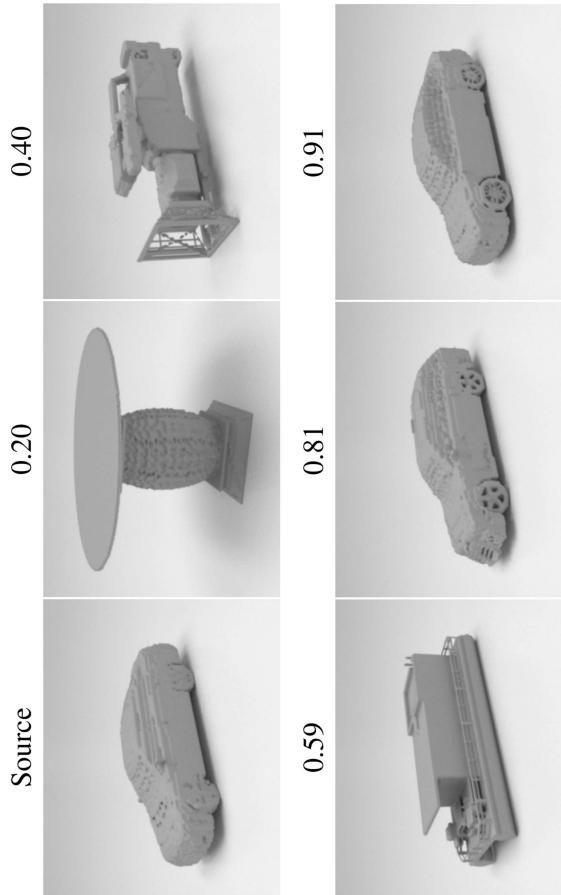
**Problem:** Cannot be applied to pointclouds

**Problem:** For meshes, need to voxelize or sample

**Problem:** Not very meaningful at low values!



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



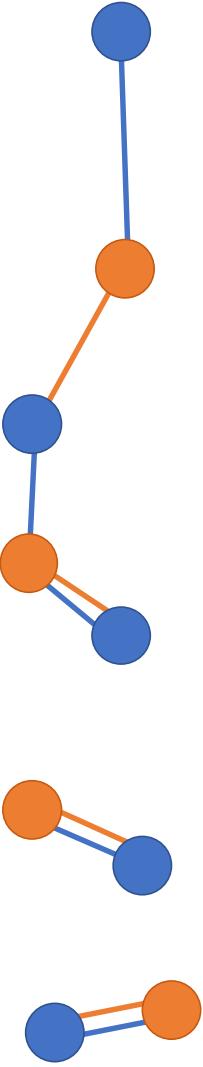
**Conclusion:** Voxel IoU not a good metric

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**

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

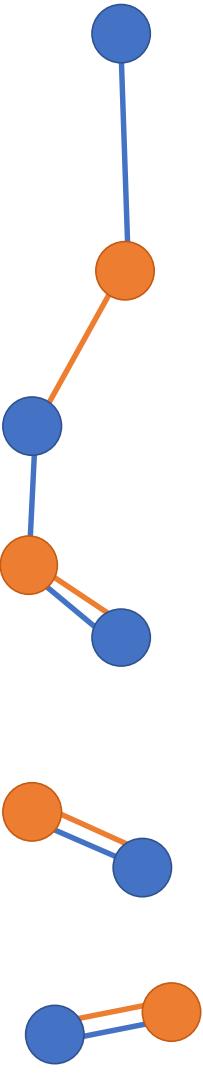


1. Convert your prediction and ground-truth into pointclouds via sampling
2. Compare with Chamfer distance

# Shape Comparison Metrics: Chamfer Distance

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

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



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

**Problem:** Chamfer is very sensitive to outliers

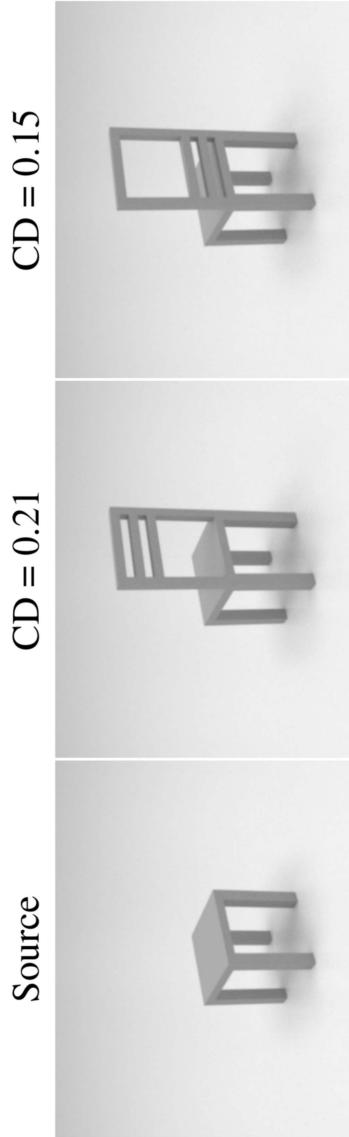
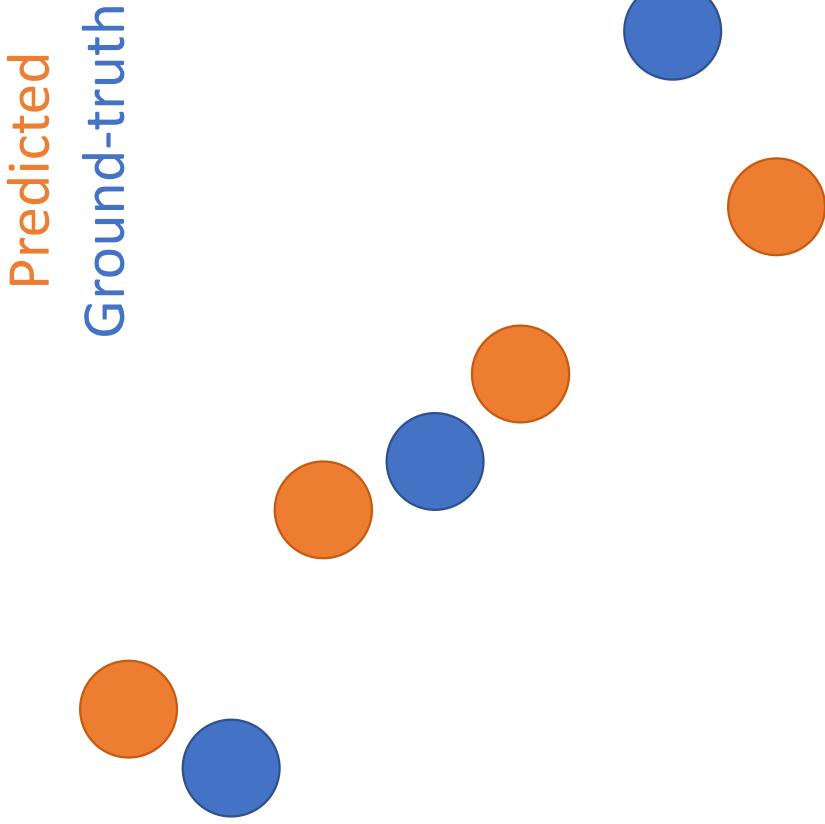


Figure credit: Tatarchenko et al, "What Do Single-view 3D Reconstruction Networks Learn?", CVPR 2019

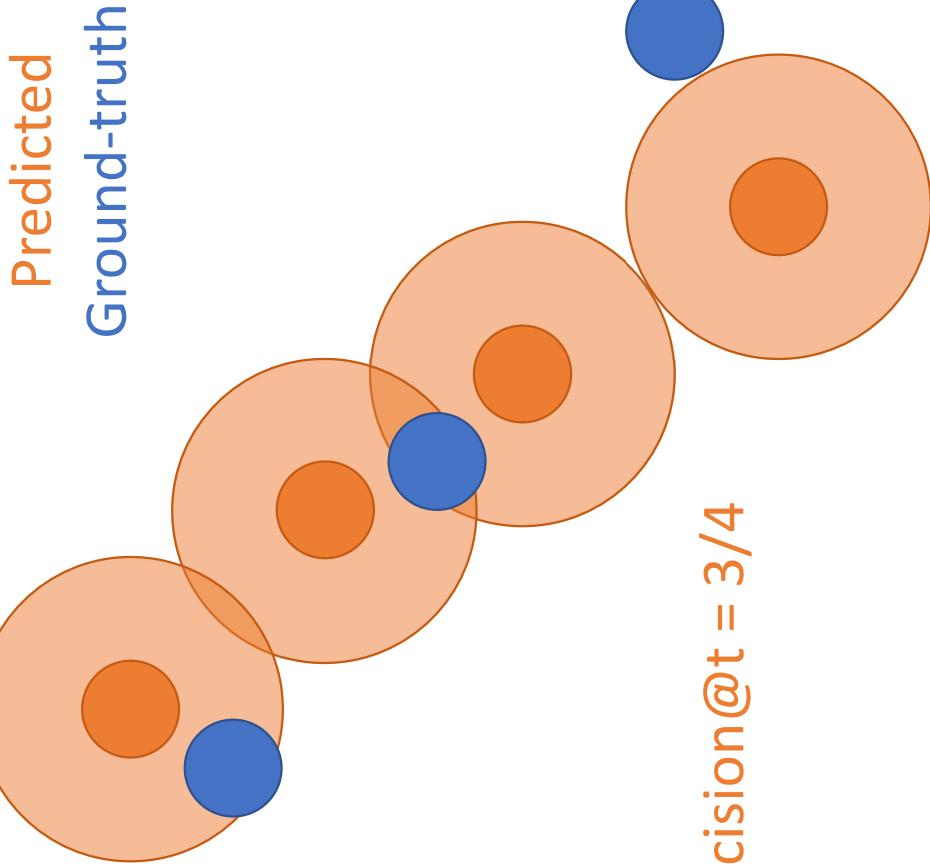
## Shape Comparison Metrics: F1 Score

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

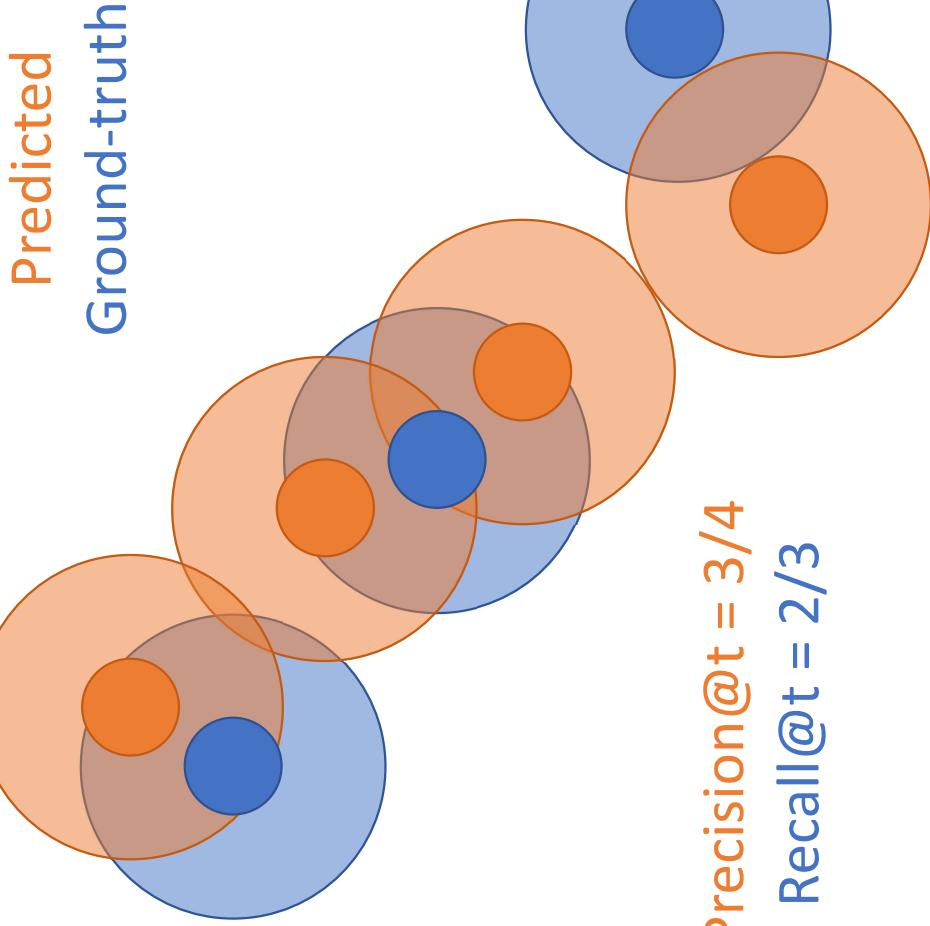


Precision@ $t$  = fraction of predicted points within  $t$  of some ground-truth point

$$\text{Precision}@t = 3/4$$

# Shape Comparison Metrics: F1 Score

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



$\text{Precision}@t$  = fraction of predicted points within  $t$  of some ground-truth point

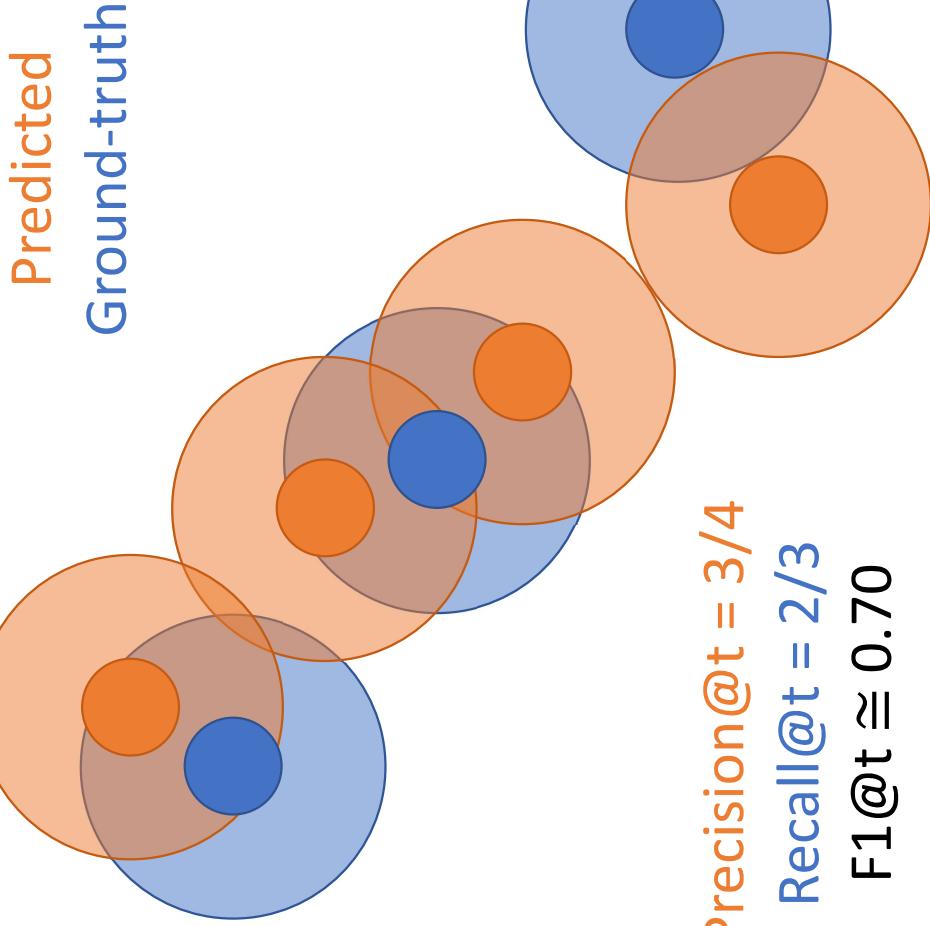
$\text{Recall}@t$  = fraction of ground-truth points within  $t$  of some predicted point

$$\text{Precision}@t = 3/4$$

$$\text{Recall}@t = 2/3$$

# Shape Comparison Metrics: F1 Score

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



$\text{Precision}@t$  = fraction of predicted points within  $t$  of some ground-truth point

$\text{Recall}@t$  = fraction of ground-truth points within  $t$  of some predicted point

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

$\text{Precision}@t = 3/4$

$\text{Recall}@t = 2/3$

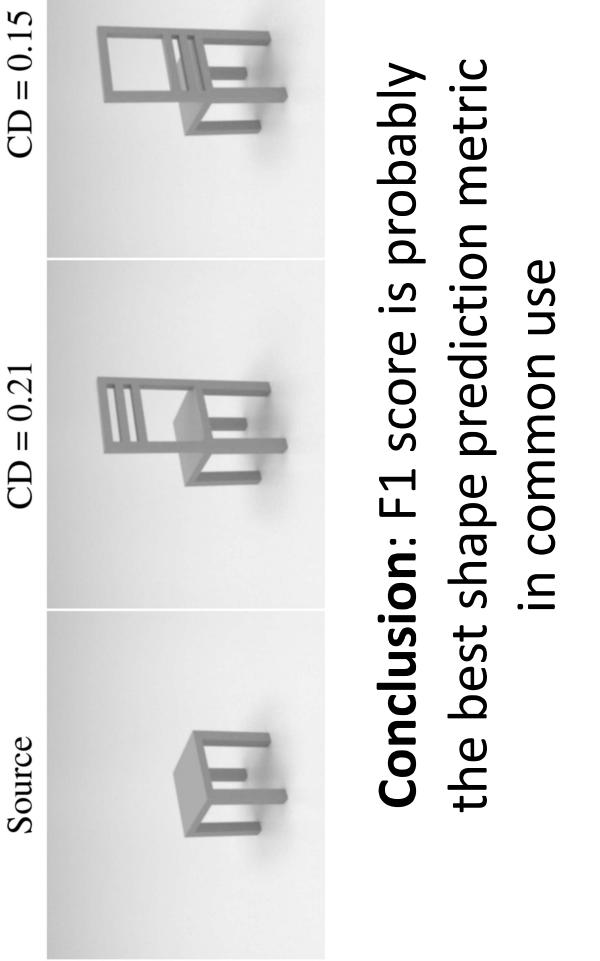
$\text{F1}@t \cong 0.70$

# Shape Comparison Metrics: F1 Score

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

F1 score is robust to outliers!

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

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

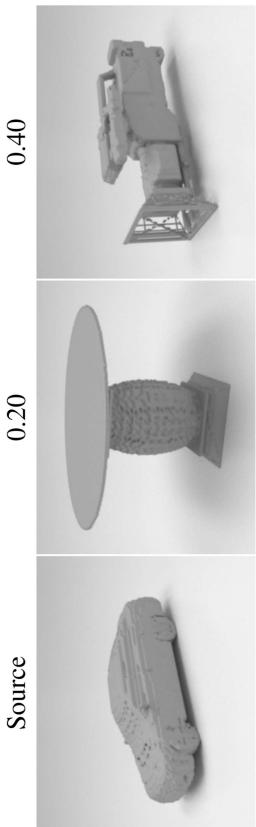
$$F1@t = 2 * \frac{Precision@t * Recall@t}{Precision@t + Recall@t}$$

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



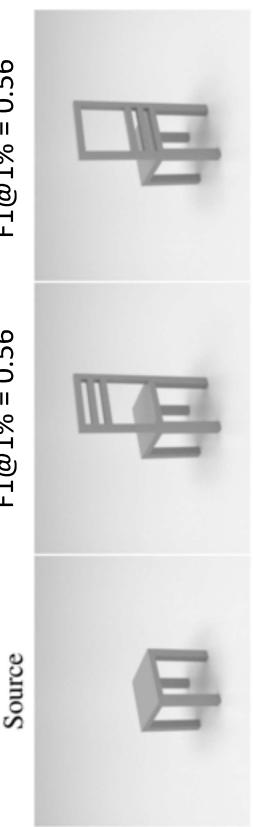
## Chamfer Distance:

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

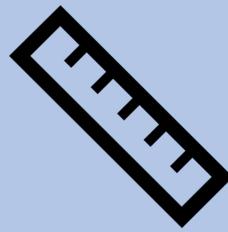


# 3D Shape Prediction

Shape Representations



Metrics



Camera Systems

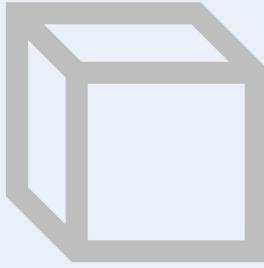


Datasets



# 3D Shape Prediction

Shape Representations



Metrics



Camera Systems



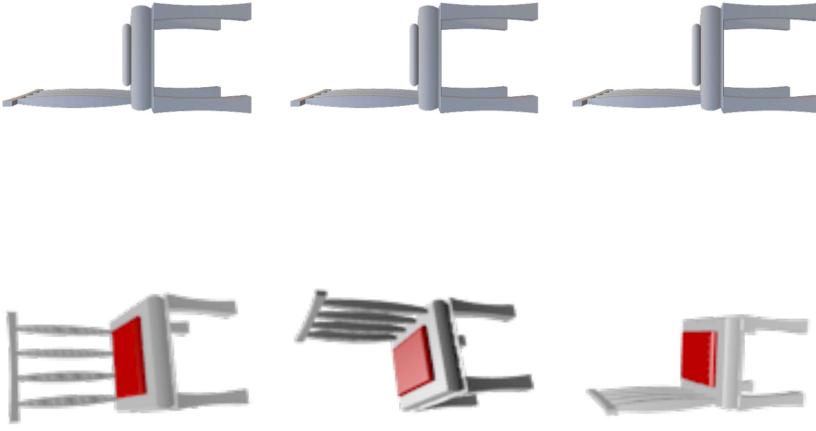
Datasets



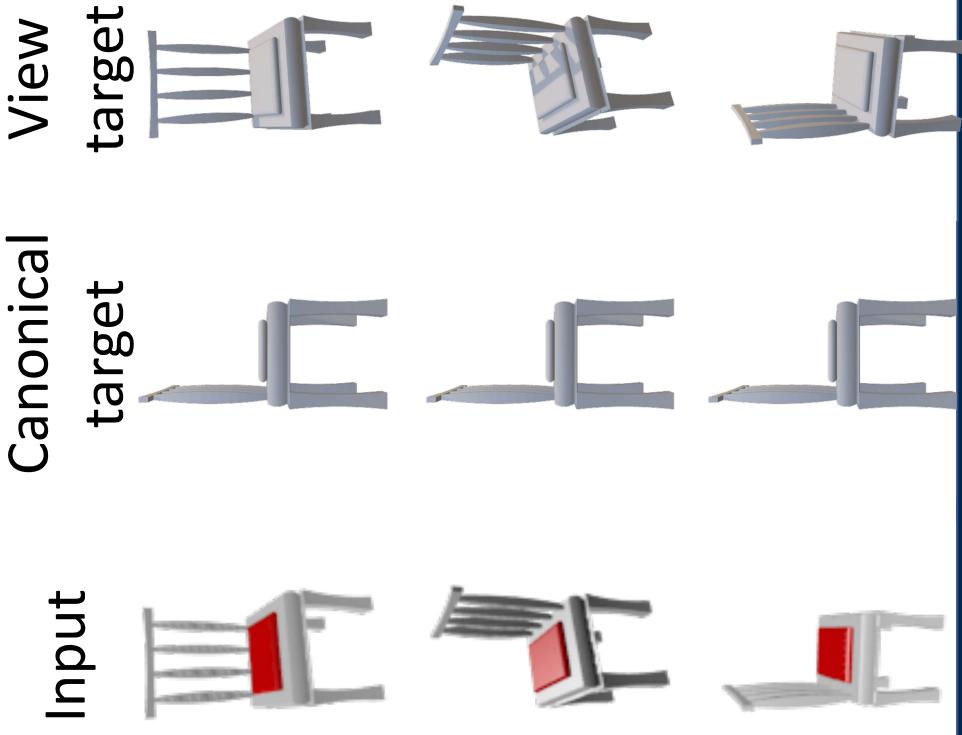
# Cameras: Canonical vs View Coordinates

Canonical  
Input  
target

**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

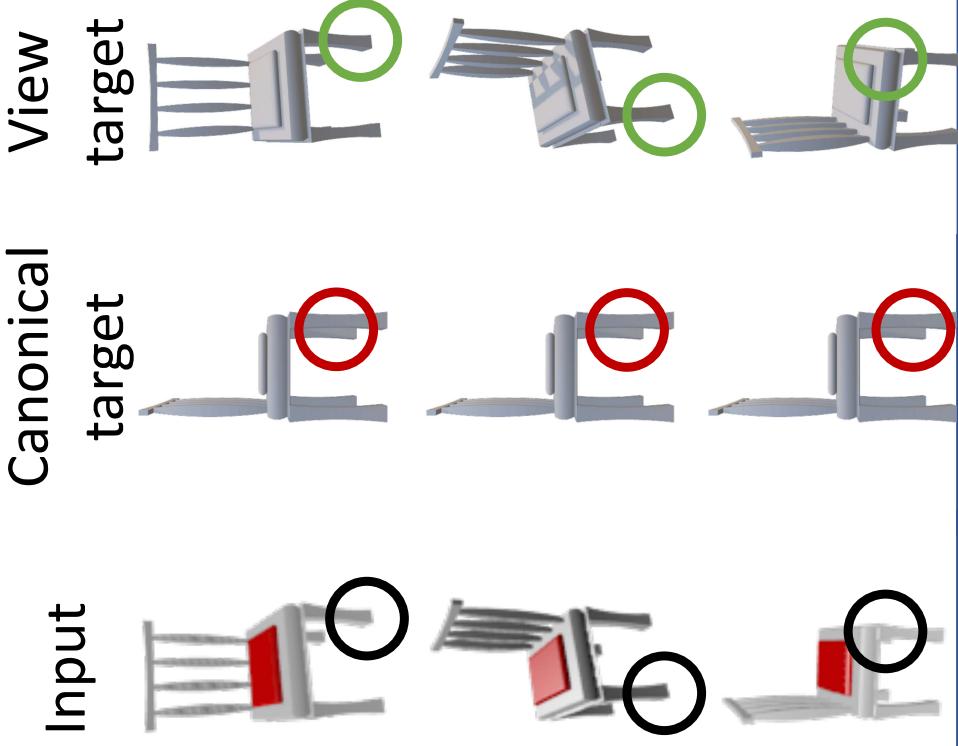


**Canonical Coordinates:** Predict 3D shape in a canonical coordinate system (e.g. front of chair is +z) regardless of the viewpoint of the input image

**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



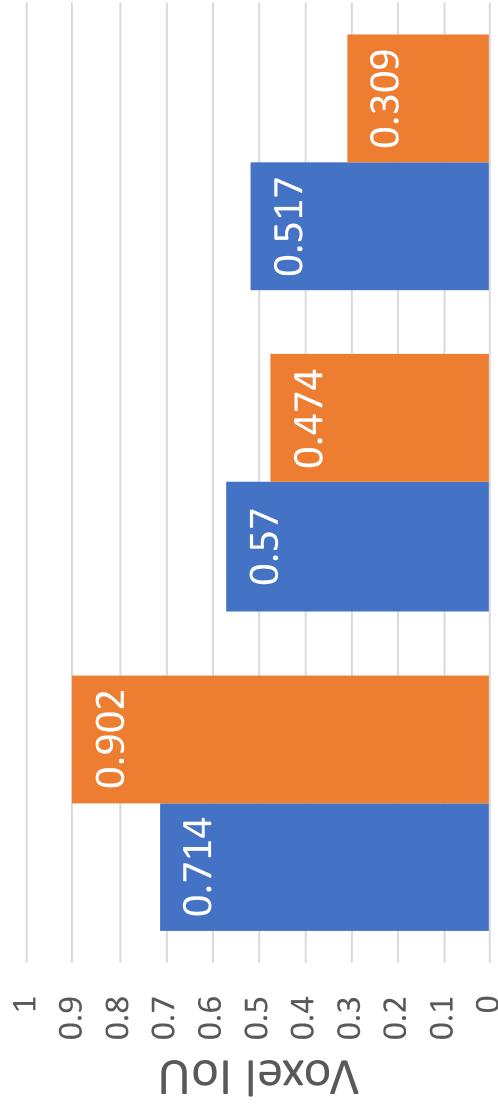
**Problem:** Canonical view breaks  
the “principle of feature  
alignment”: Predictions should be  
aligned to inputs

View coordinates maintain  
alignment between inputs and  
predictions!

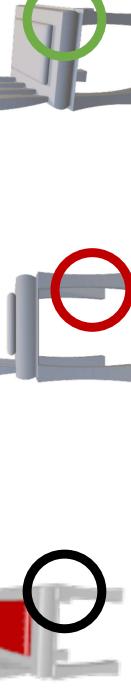
# Cameras: Canonical vs View Coordinates

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

■ View ■ Canonical

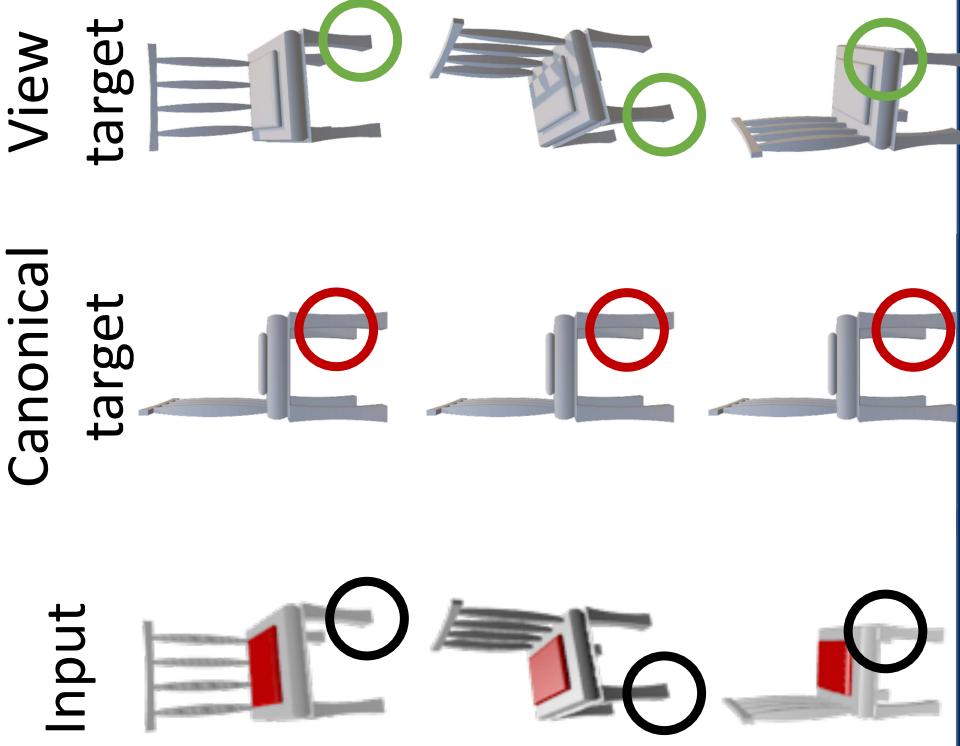


Novel view  
Shin et al, "Pixels, voxels, and views: A study of shape representations for single view 3D object shape prediction", CVPR 2018



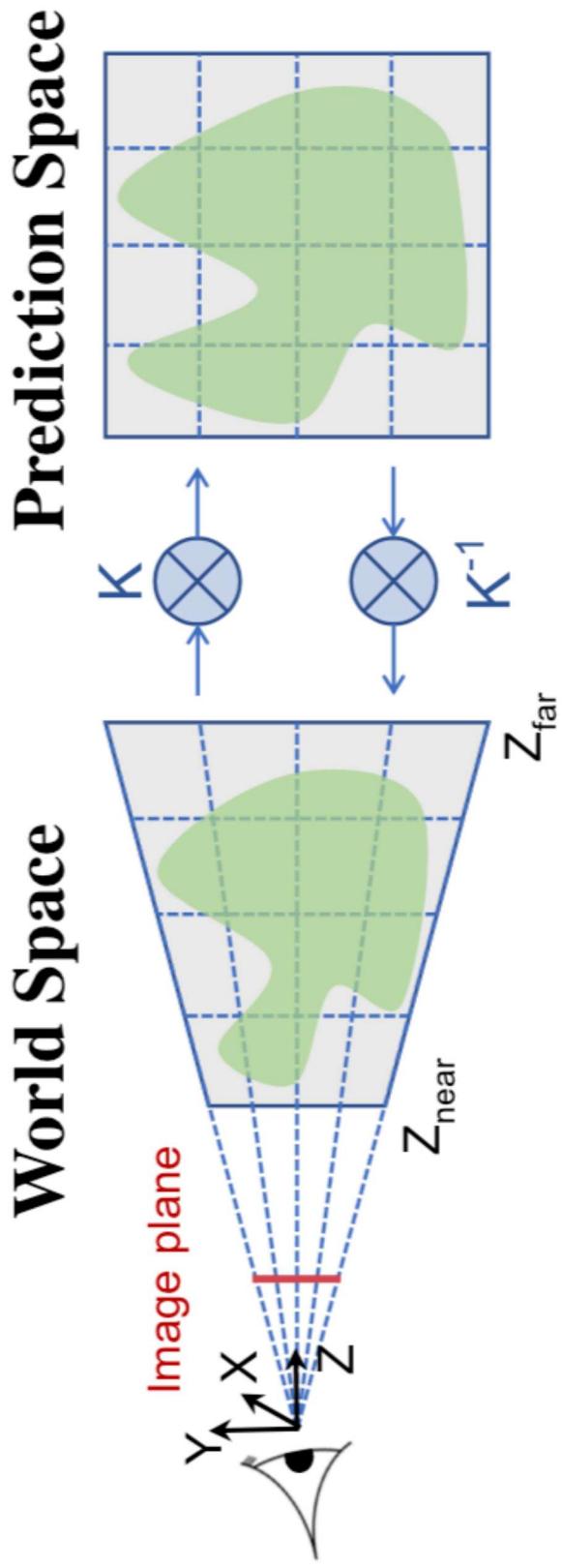
# Cameras: Canonical vs View Coordinates

**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

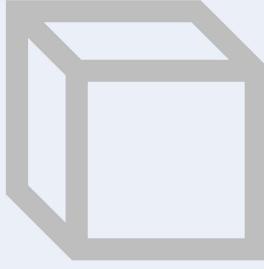
# View-Centric Voxel Predictions



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

# 3D Shape Prediction

Shape Representations



Metrics



Camera Systems

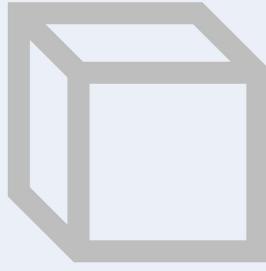


Datasets



# 3D Shape Prediction

Shape Representations



Metrics



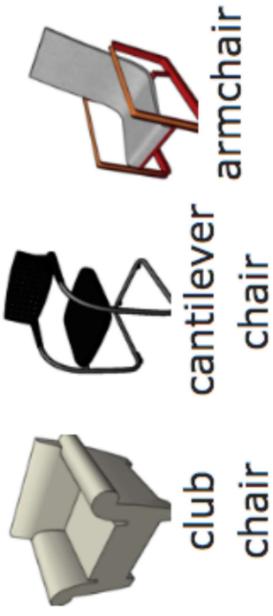
Camera Systems



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

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

Justin Johnson

Lecture 17 - 88

## 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

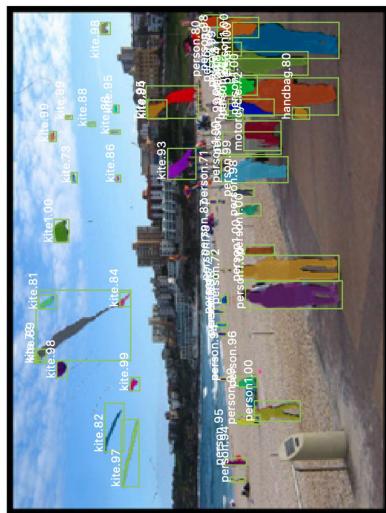
Sun et al, "Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling", CVPR 2018

November 13, 2019

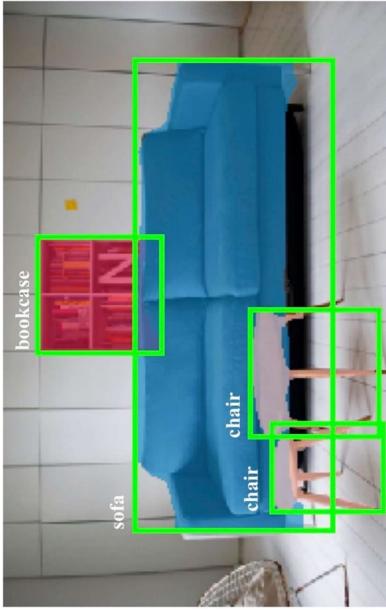
# 3D Shape Prediction: Mesh R-CNN

Mask R-CNN:

2D Image -> 2D shapes



2D Image -> Triangle Meshes



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

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

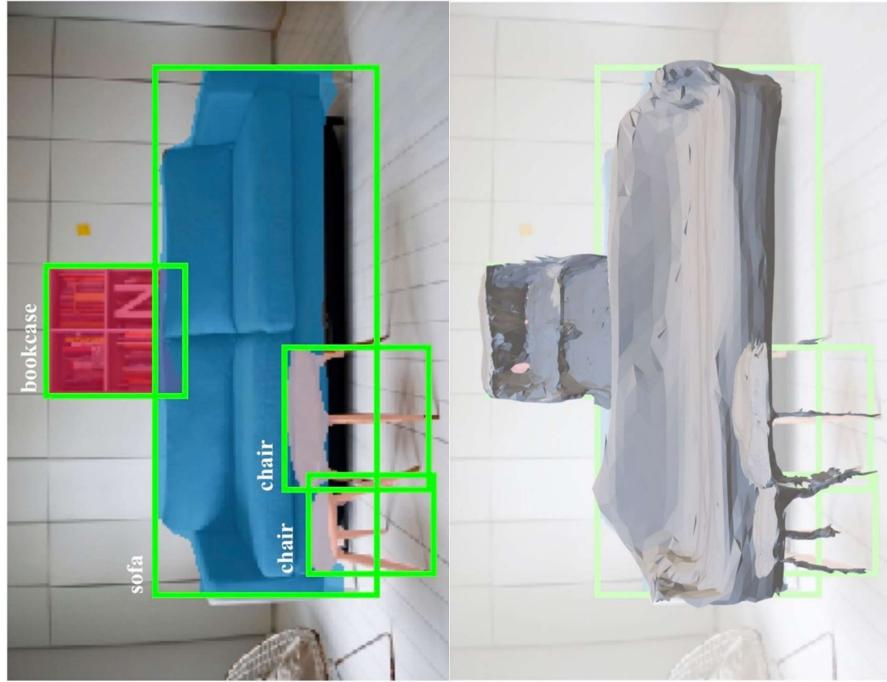
Justin Johnson

Lecture 17 - 89

November 13, 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

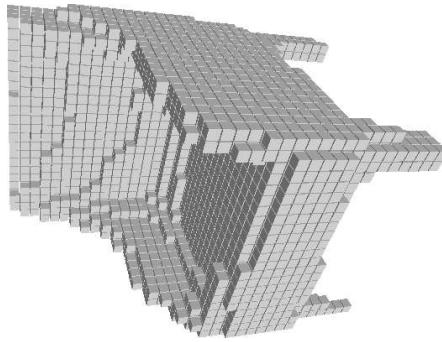
# Mesh R-CNN: Hybrid 3D shape representation

**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

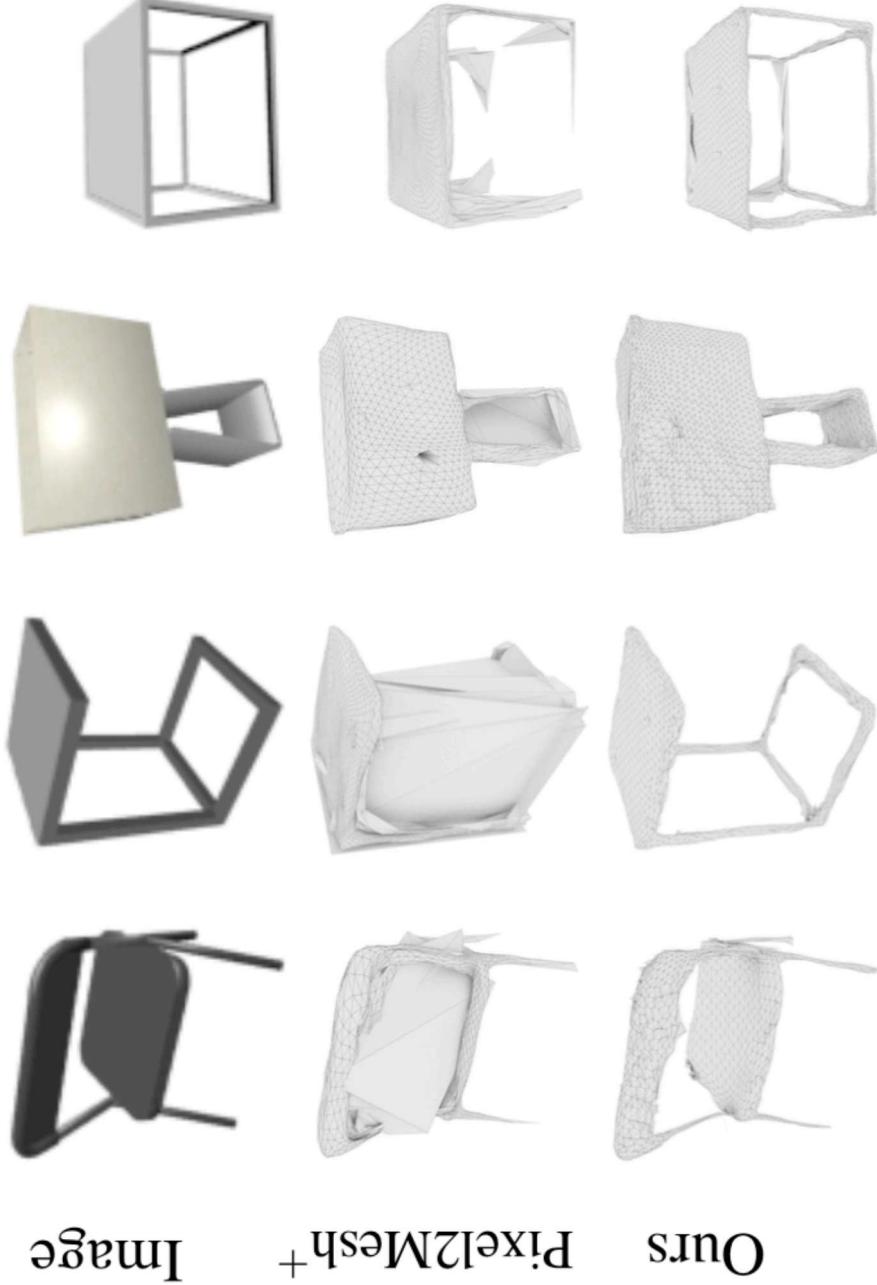


3D object meshes

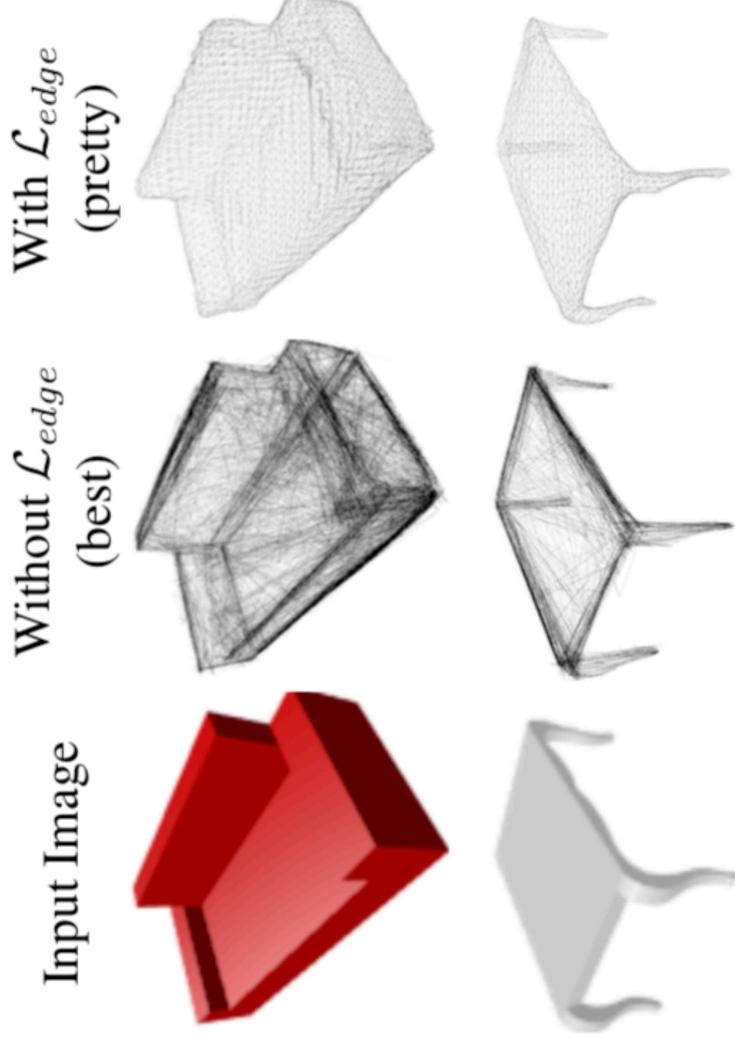
3D object voxels



# Mesh R-CNN: ShapeNet Results

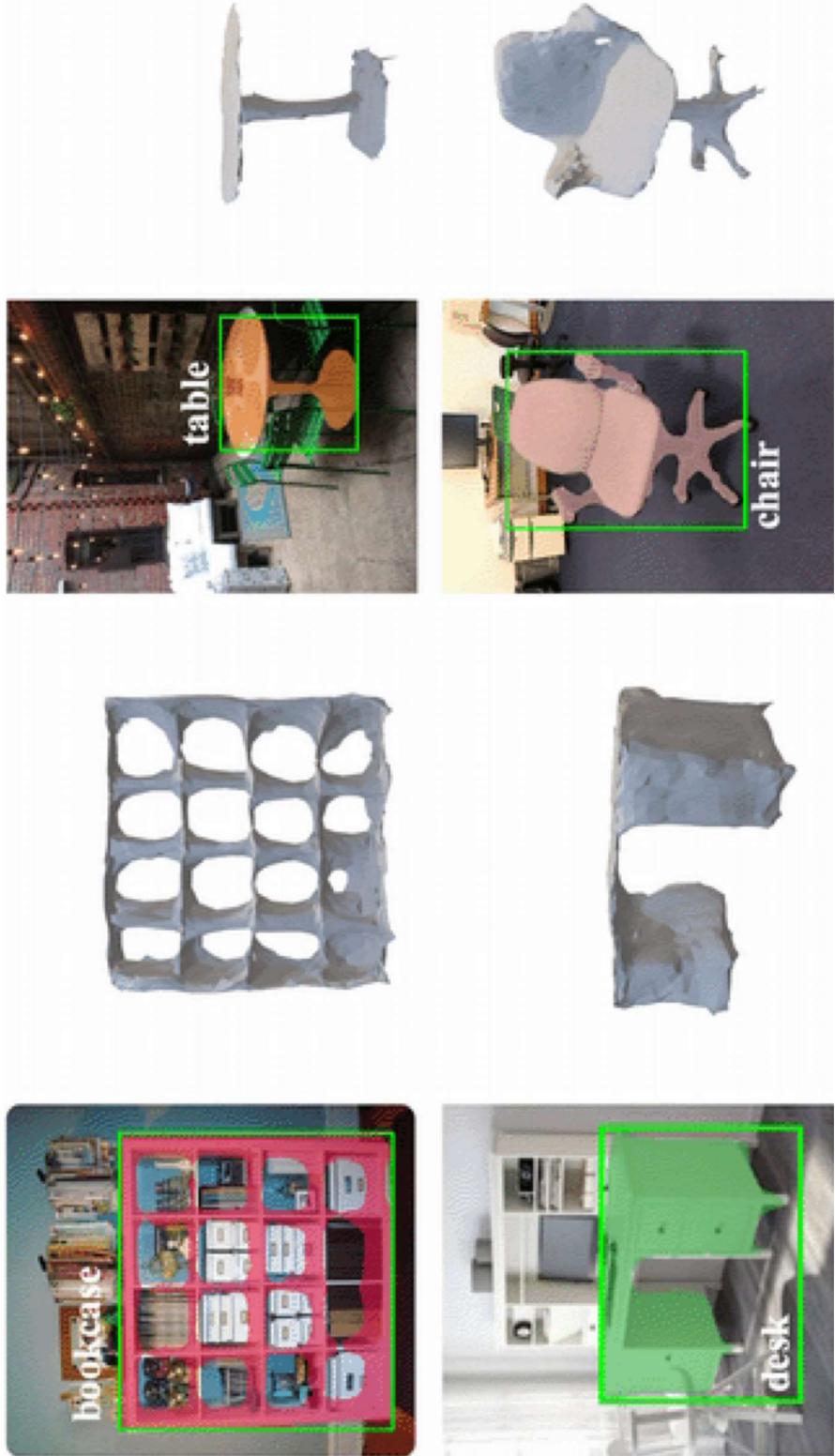


# Mesh R-CNN: Shape Regularizers



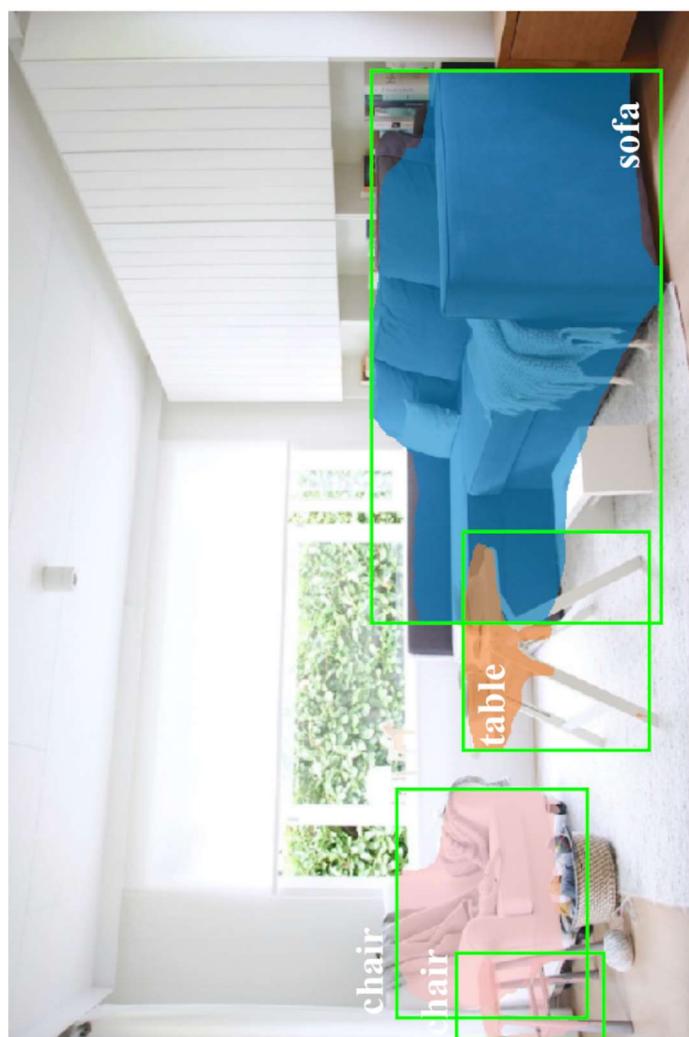
Using Chamfer as only mesh loss gives degenerate meshes.  
Also need "mesh regularizer" to encourage nice predictions:  
 $\mathcal{L}_{edge}$  = minimize L2 norm of edges in the predicted mesh

# Mesh R-CNN: Pix3D Results



# Mesh R-CNN: Pix3D Results

Predicting many objects per scene



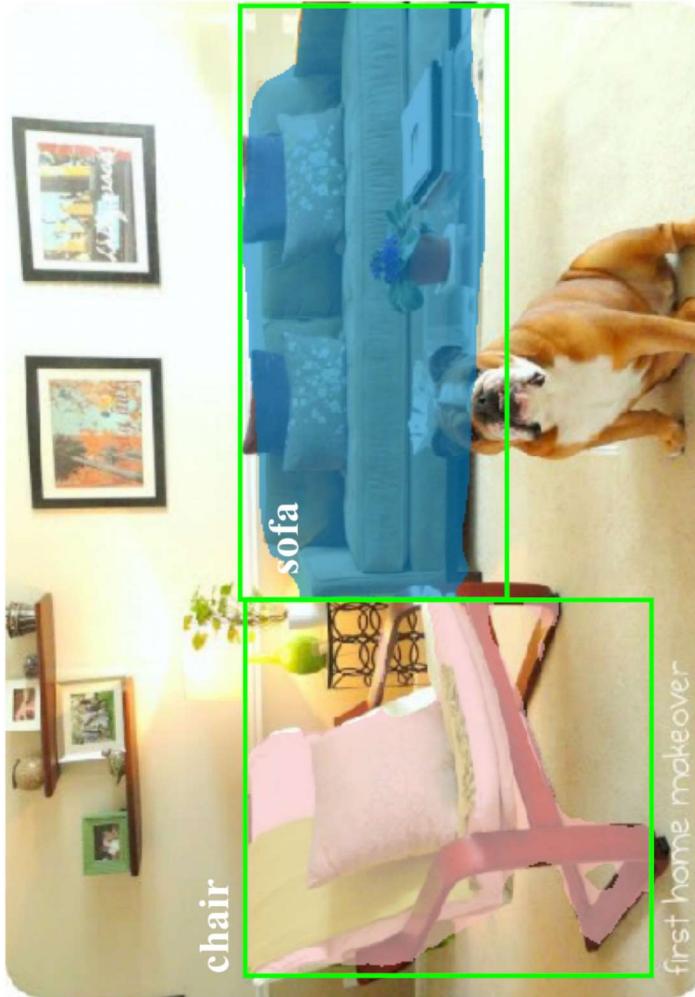
Box & Mask Predictions

Mesh Predictions



# Mesh R-CNN: Pix3D Results

Amodal completion: predict  
occluded parts of objects



Box & Mask Predictions

Justin Johnson

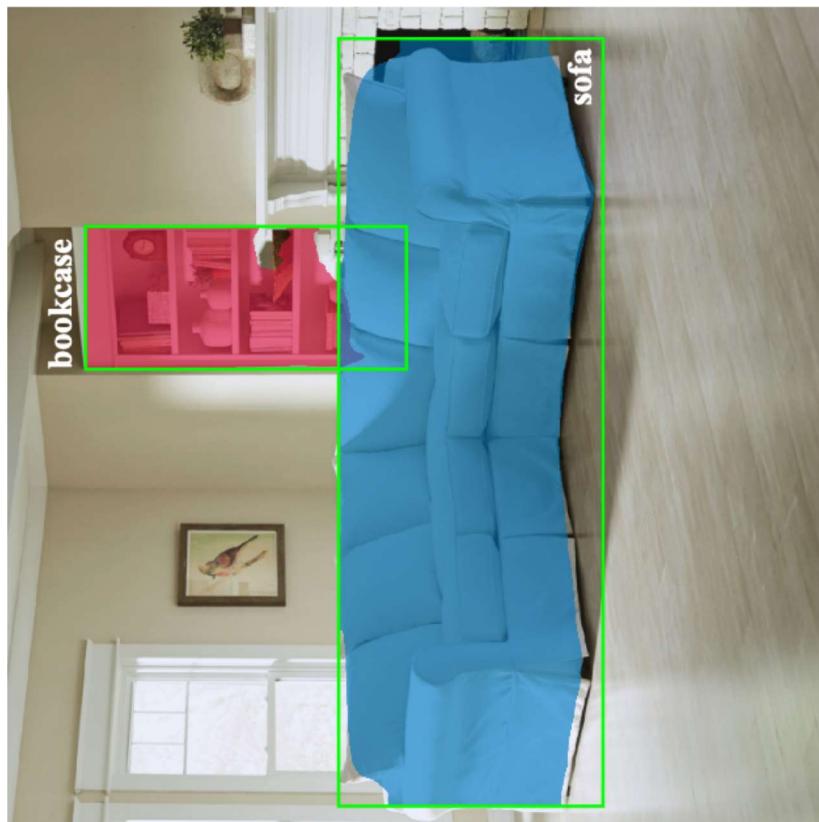
Lecture 17 - 101

November 13, 2019

Mesh Predictions

# Mesh R-CNN: Pix3D Results

Segmentation failures  
propagate to meshes



Box & Mask Predictions

Mesh Predictions



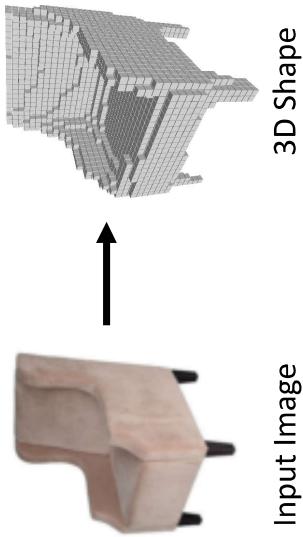
Justin Johnson

Lecture 17 - 102

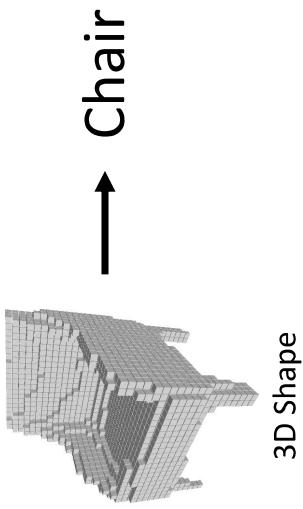
November 13, 2019

# Recap

Predicting 3D Shapes  
from single image



Processing 3D  
input data



## 3D Shape Representations

Depth Map	Voxel Grid	Implicit Surface	Pointcloud	Mesh

Next Time:  
Videos