# 3D Vision

Slides credit: CS231N at Stanford, Justin Johnson

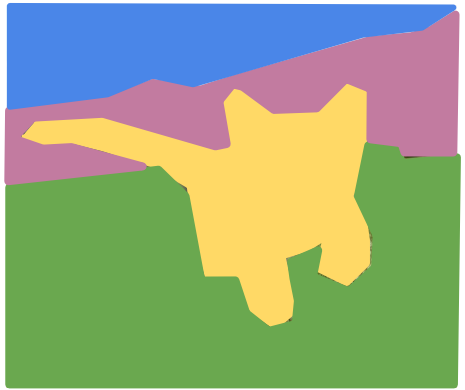
# **Recall**: 2D Detection and Segmentation

**Classification**

**CAT**

No spatial extent

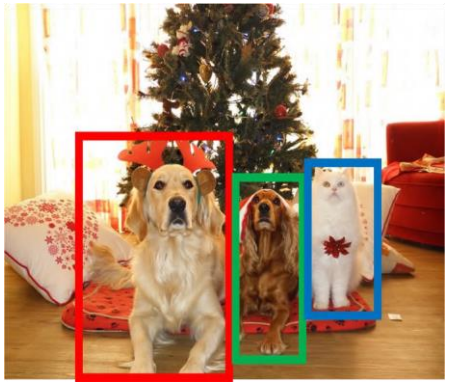
**Semantic Segmentation**



**GRASS**, **CAT**, **TREE**, **SKY**

No objects, just pixels

**Object Detection**



**DOG**, **DOG**, **CAT**

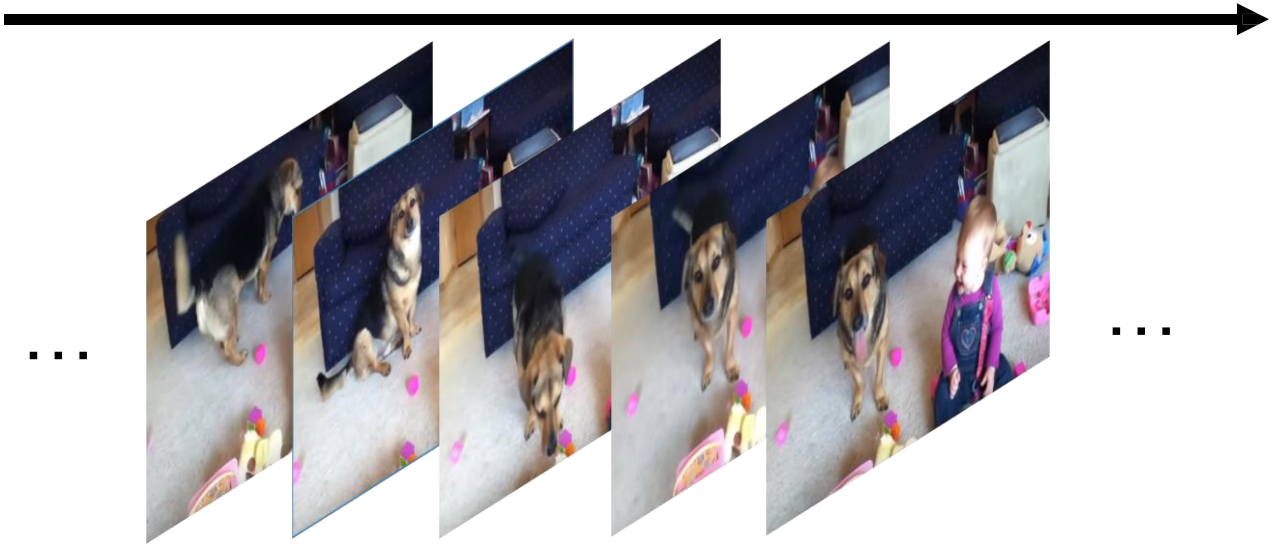
**Instance Segmentation**

**DOG**, **DOG**, **CAT**

Multiple Objects

# **Recall**: **Video** = 2D + Time

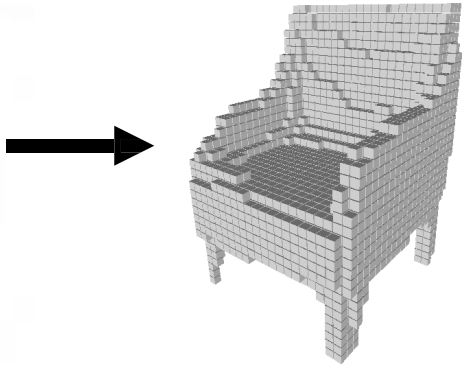
A video is a **sequence** of images
4D tensor: T x 3 x H x W
(or 3 x T x H x W)



# Focus on Two Problems today

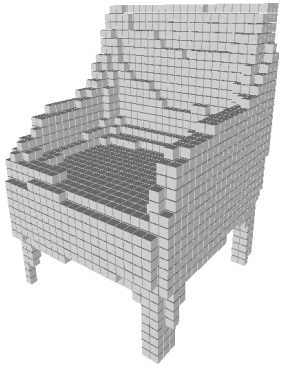
Predicting 3D Shapes from single image

Input Image



3D Shape

Processing 3D input data



3D Shape

Chair

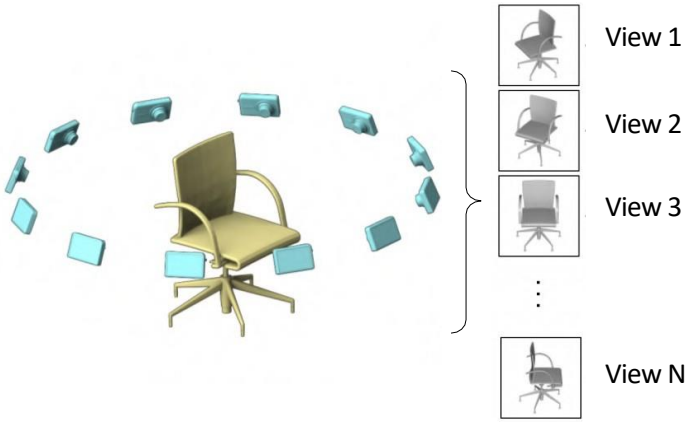
# Many more topics in 3D Vision!

3D Representations
Computing Correspondences
Multi-view stereo
Structure from Motion
Simultaneous Localization and Mapping (SLAM)
View Synthesis
Differentiable Graphics
3D Sensors
……

# Multi-View CNN

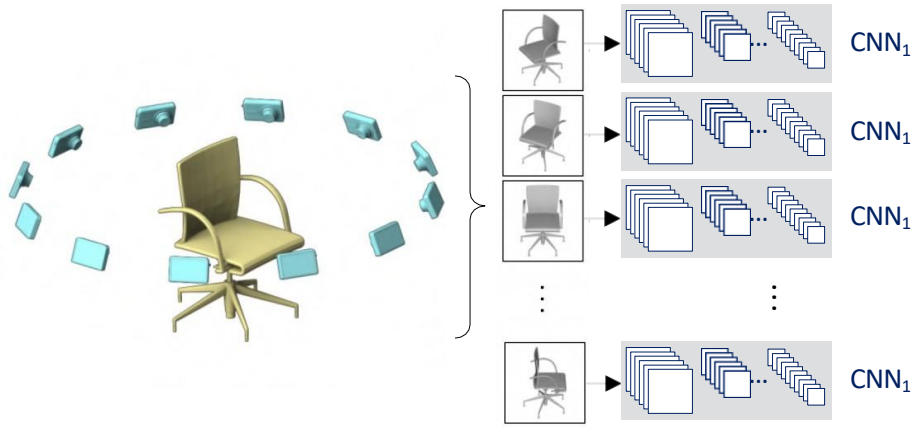
Su et al. ICCV 2015

# Multi-View CNN




Su et al. ICCV 2015

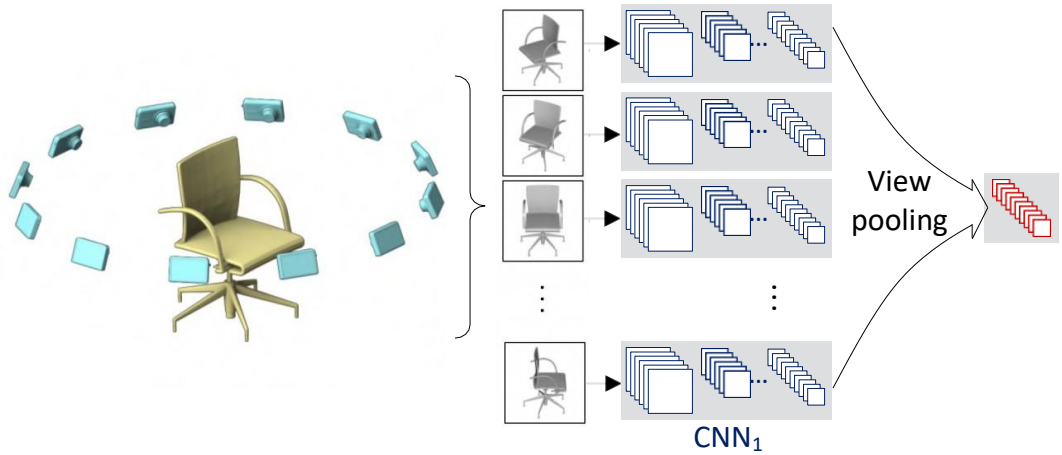
# Multi-View CNN




$CNN_1$: a ConvNet extracting image features

Su et al. ICCV 2015

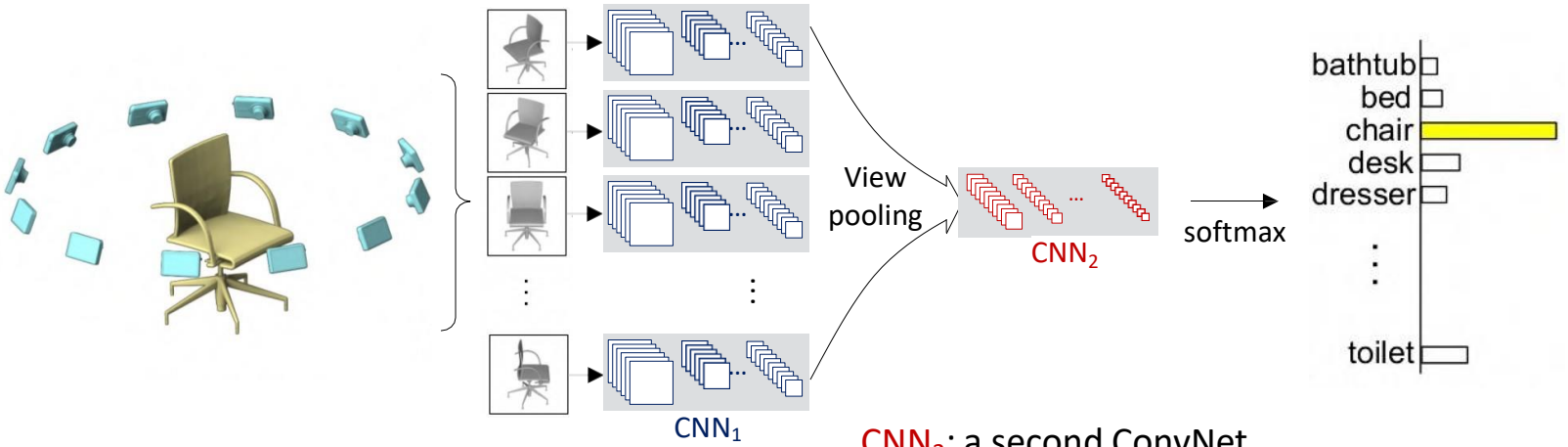
# Multi-View CNN




View pooling: element-wise max-pooling across all views

Su et al. ICCV 2015

# Multi-View CNN




$CNN_2$: a second ConvNet producing shape descriptors

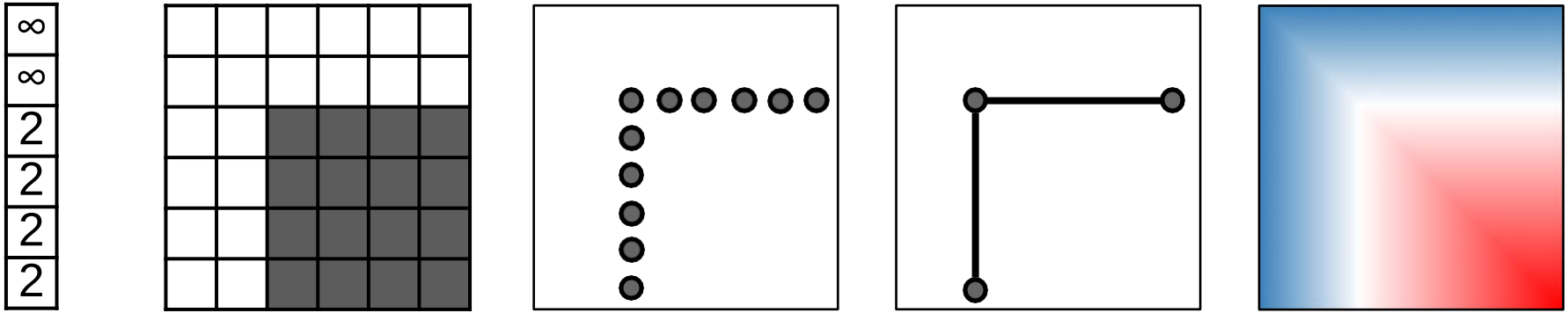
Su et al. ICCV 2015

# Experiments – Classification & Retrieval

| | Method | Classification (Accuracy) | Retrieval (mAP) |
|---|---|---|---|
| Non-deep | SPH | 68.2% | 33.3% |
| | LFD | 75.5% | 40.9% |
| | 3D ShapeNets | 77.3% | 49.2% |
| | FV, 12 views | 84.8% | 43.9% |
| | CNN, 12 views | 88.6% | 62.8% |
| | MVCNN, 12 views | **89.9%** | 70.1% |
| | MVCNN+metric, 12 views | 89.5% | **80.2%** |
| | MVCNN, 80 views | 90.1% | 70.4% |
| | MVCNN+metric, 80 views | **90.1%** | **79.5%** |

On ModelNet 40

# 3D Shape Representations




Depth Map

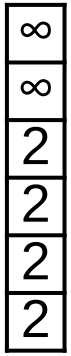
Voxel Grid

Pointcloud

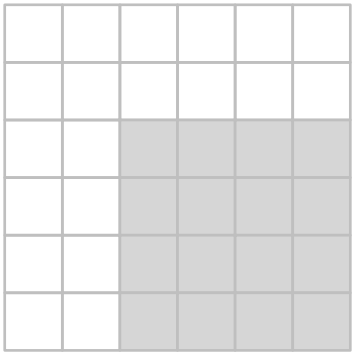
Mesh

Implicit Surface

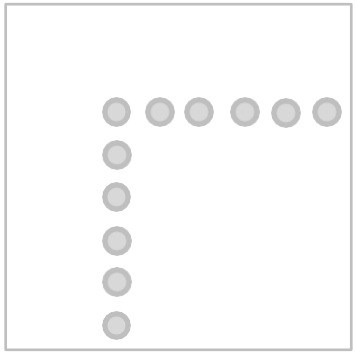
# 3D Shape Representations



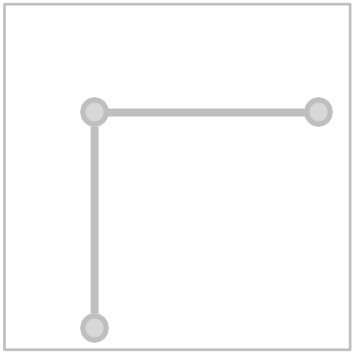

Depth Map



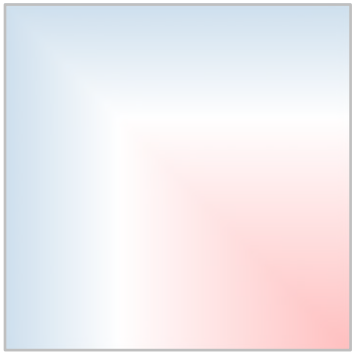
Voxel Grid



Pointcloud



Mesh



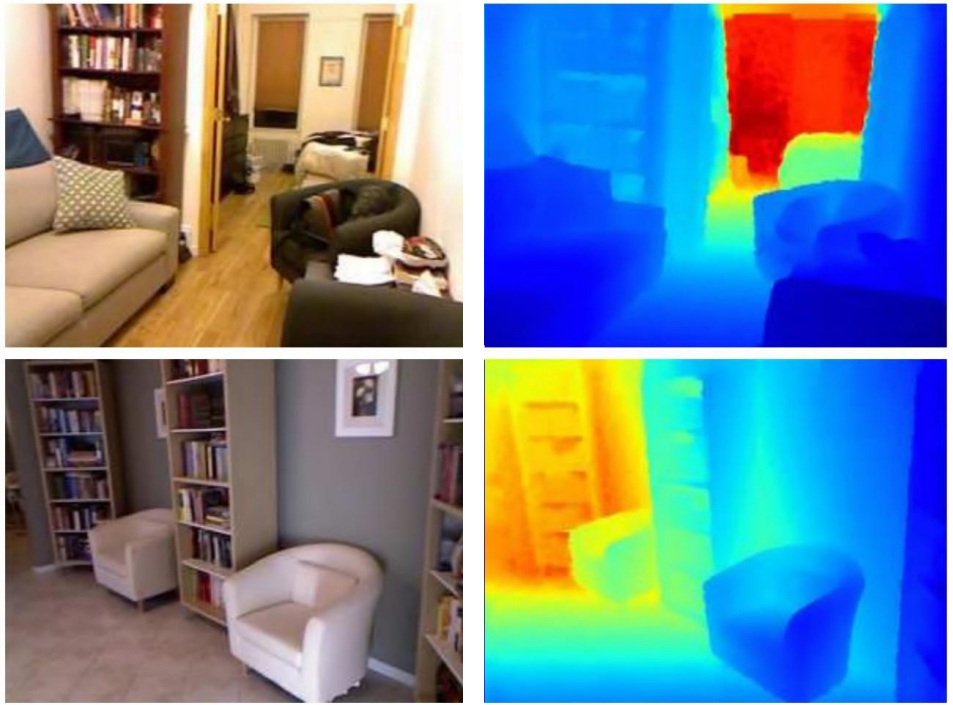
Implicit Surface

# 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)

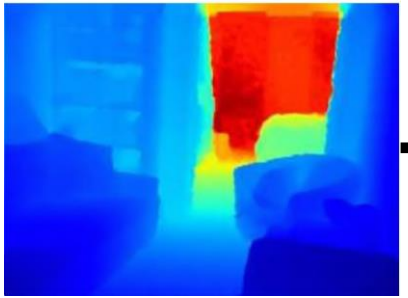


RGB Image: 3 x H x W  Depth Map: H x W

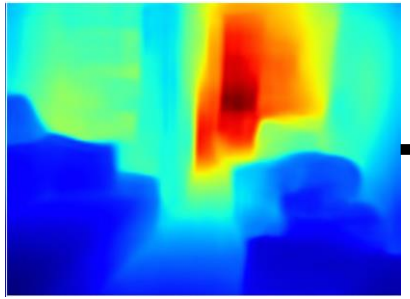
Eigen and Fergus, “Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture”, ICCV 2015

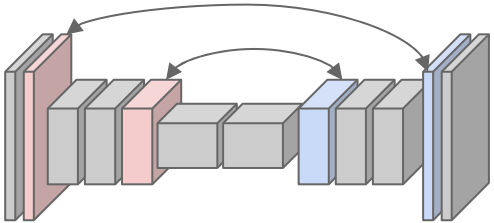
# Predicting Depth Maps

**Predicted Depth Image:**
1 x H x W



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





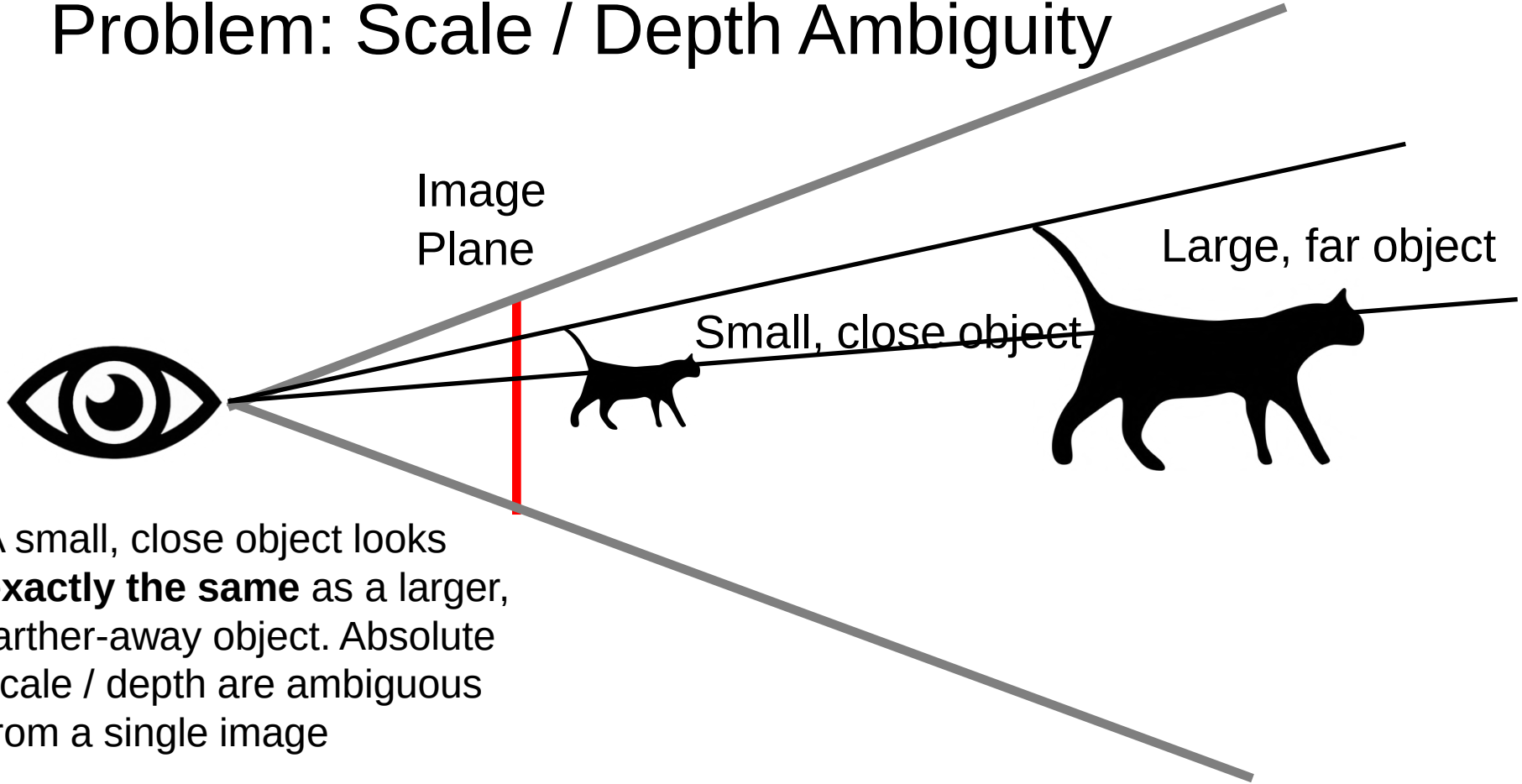
**RGB Input Image:**
3 x H x W

**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

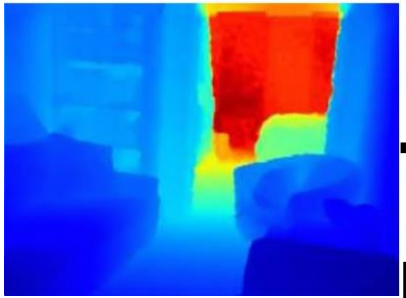
# Problem: Scale / Depth Ambiguity




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

**Predicted Depth Image:**
1 x H x W

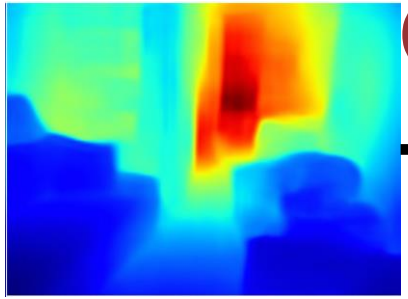


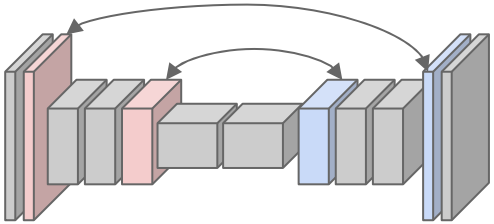
**Scale invariant loss**

$$D(y, y^*) = \frac{1}{2n^2} \sum_{i,j} \left( (\log y_i - \log y_j) - (\log y_i^* - \log y_j^*) \right)^2$$

$$= \frac{1}{n} \sum_i d_i^2 - \frac{1}{n^2} \sum_{i,j} d_i d_j = \frac{1}{n} \sum_i d_i^2 - \frac{1}{n^2} \left( \sum_i d_i \right)^2$$

**Per-Pixel Loss**
**(Scale invariant)**





**RGB Input Image:**
3 x H x W

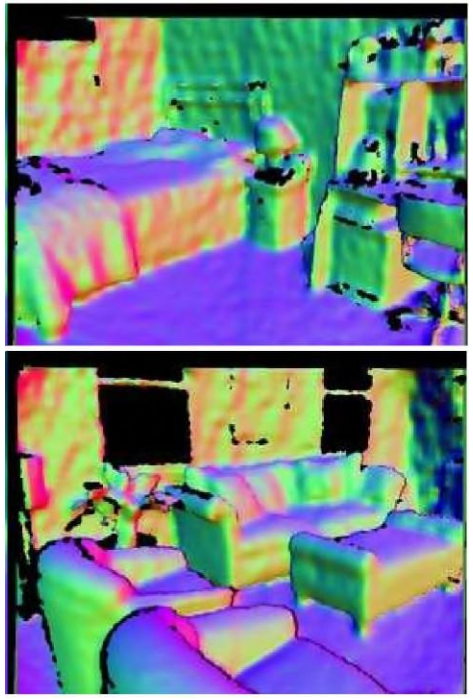
**Fully Convolutional network**

**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



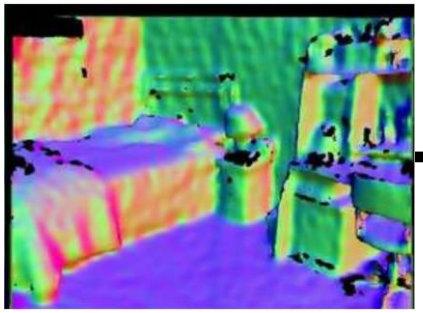
RGB Image: 3 x H x W

Normals: 3 x H x W

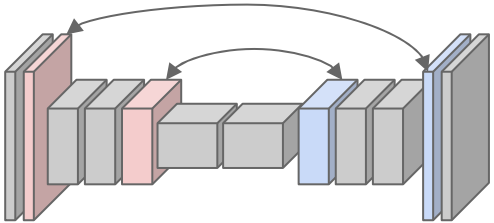
Eigen and Fergus, “Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture”, ICCV 2015

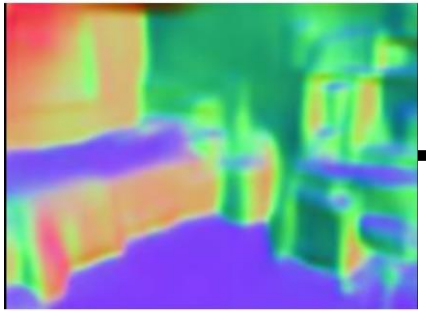
# Predicting Normals

**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**

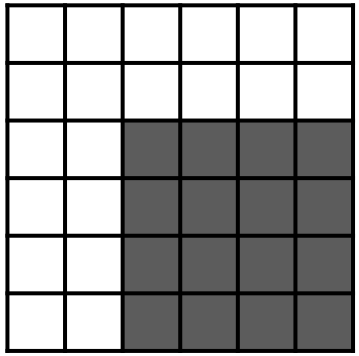
**Predicted Normals:**
3 x H x W

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

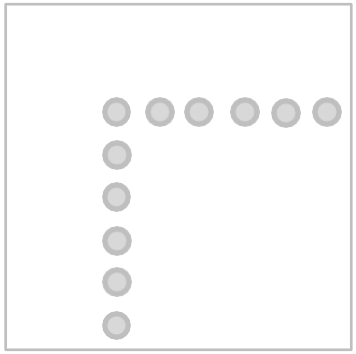
Eigen and Fergus, “Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture”, ICCV 2015

# 3D Shape Representations

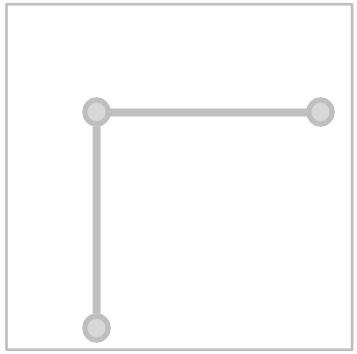

Depth Map



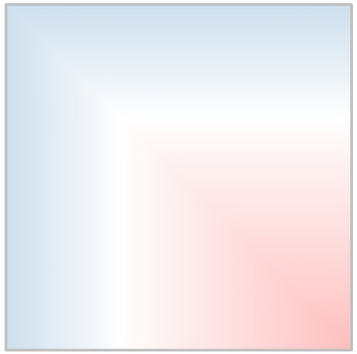
Voxel Grid



Pointcloud



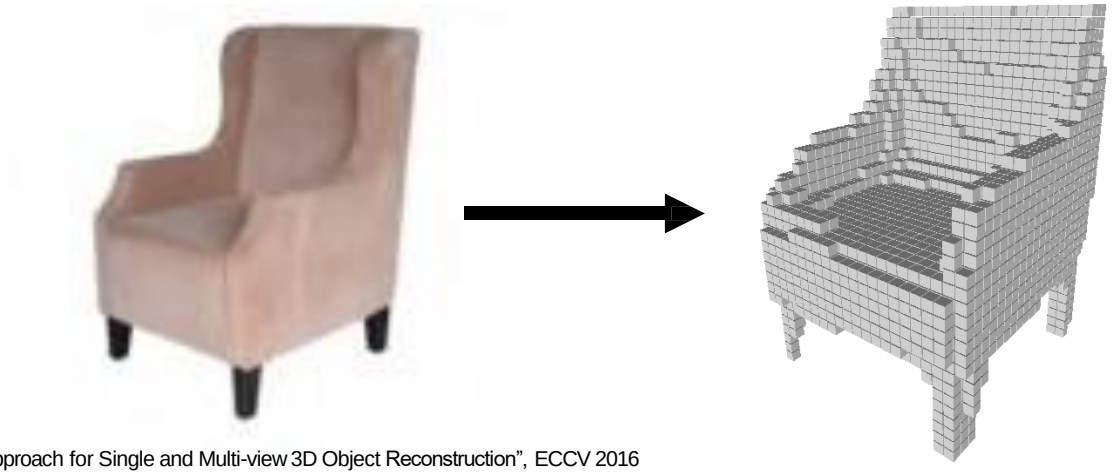
Mesh



Implicit Surface

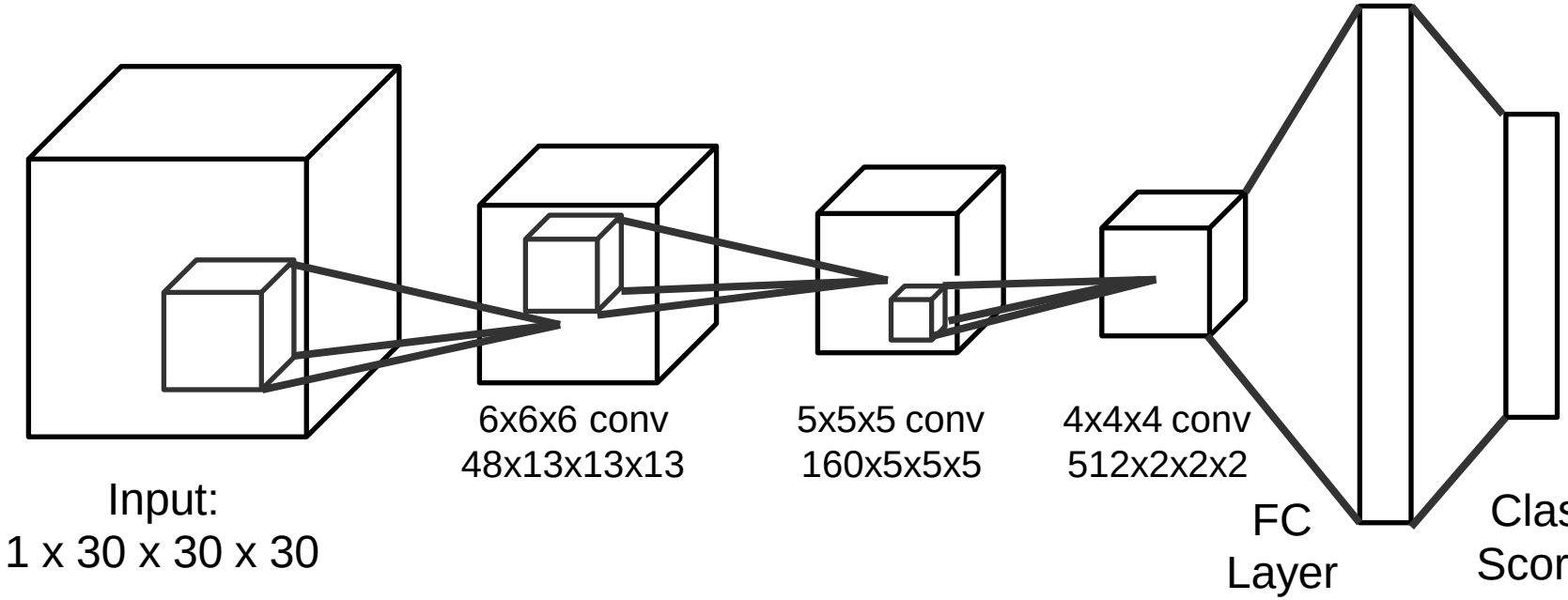
# 3D Shape Representations: Voxels

- 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

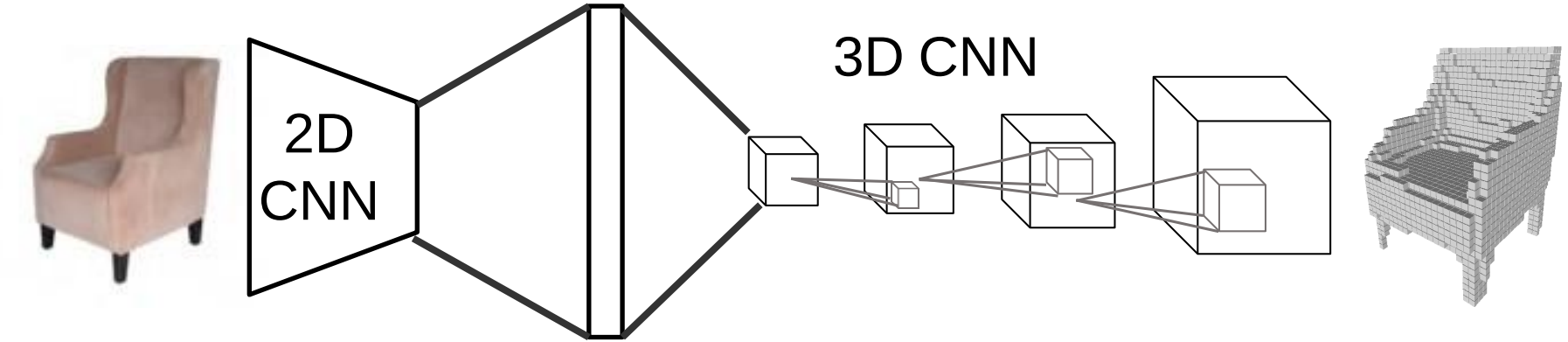
# Processing Voxel Inputs: 3D Convolution




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'

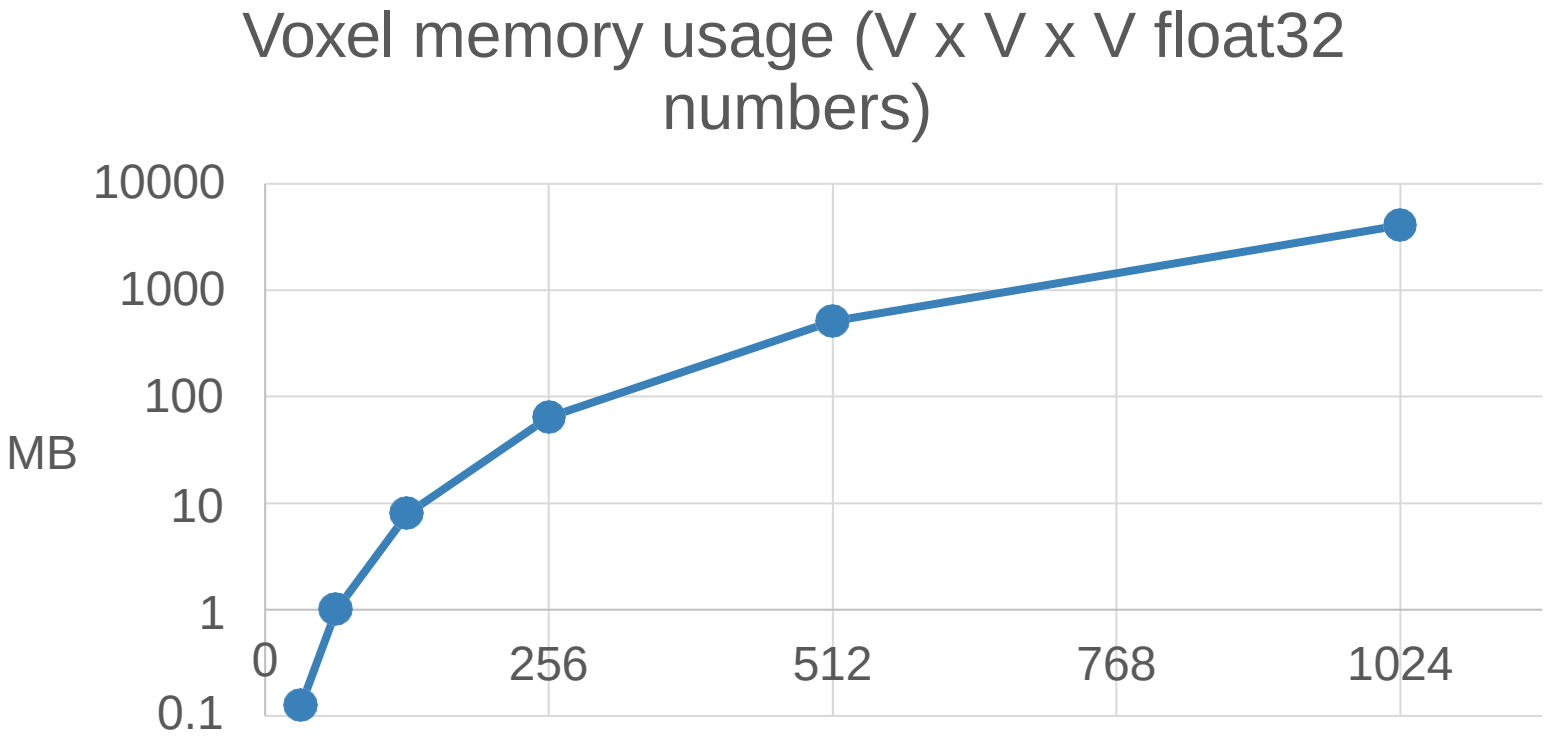
Voxels:
1 x V x V x V

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

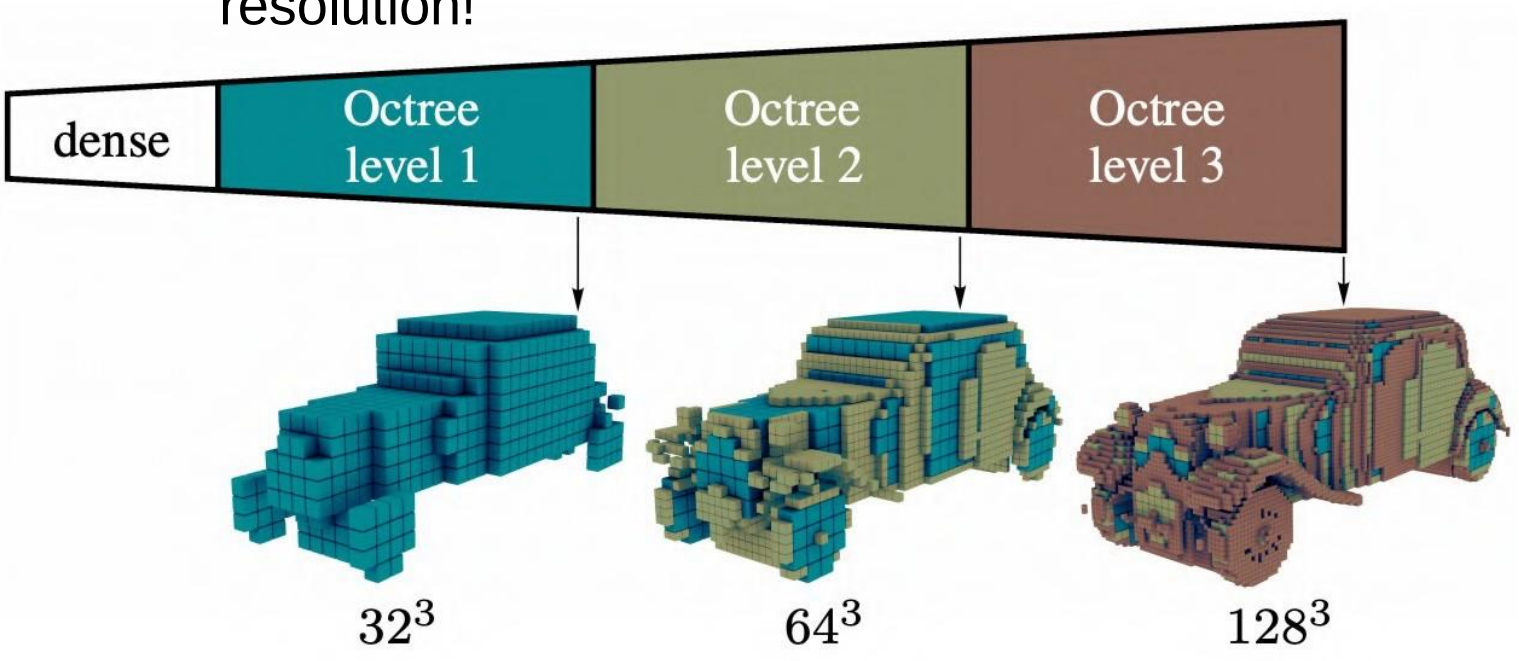
# Voxel Problems: Memory Usage

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



# Scaling Voxels: Oct-Trees

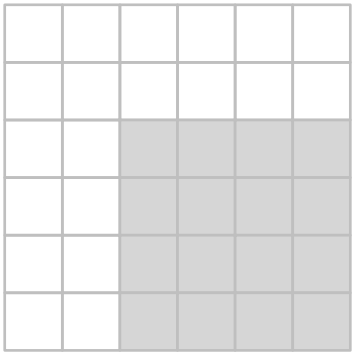
Use voxel grids with heterogenous resolution!



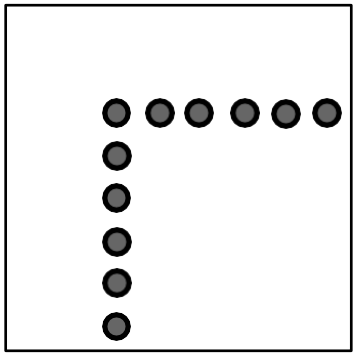

Tatarchenko et al, “Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs”, ICCV 2017

# 3D Shape Representations

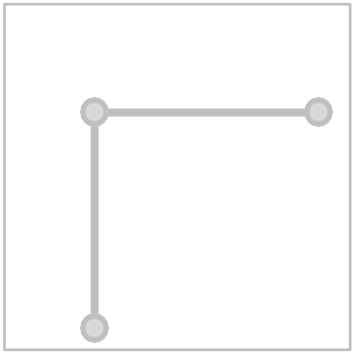

Depth Map



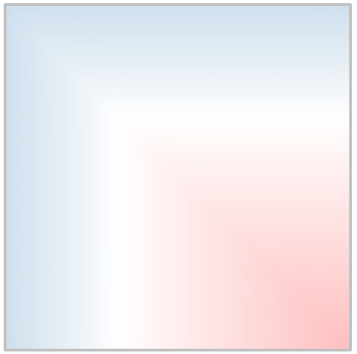
Voxel Grid



Pointcloud



Mesh



Implicit Surface

# 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

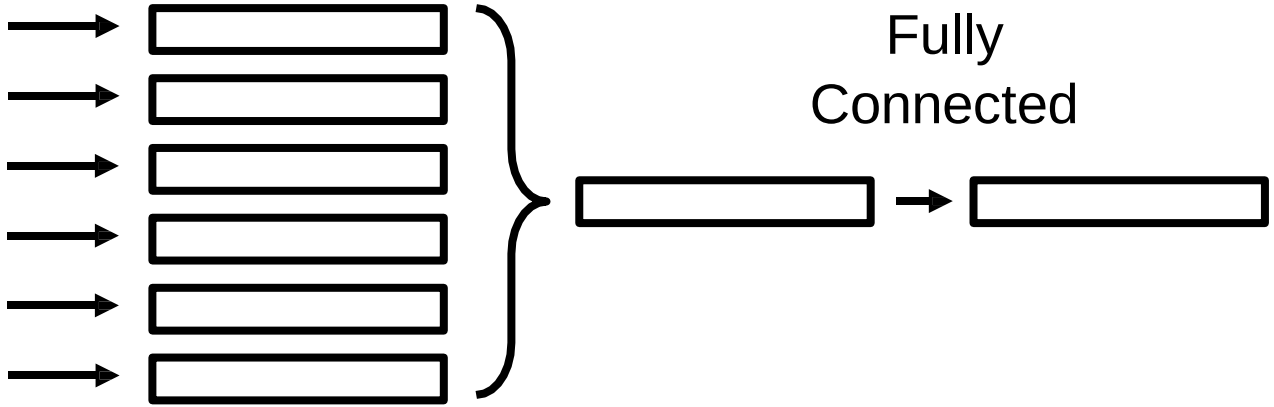
Fan et al, “A Point Set Generation Network for 3D Object Reconstruction from a Single Image”, CVPR 2017

# Processing Pointcloud Inputs: PointNet

Run MLP on each point

Max-Pool

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




| **Input pointcloud:** | **Point features:** | **Pooled vector:** | **Class score:** |
|---|---|---|---|
| P x 3 | P x D | D | 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

**Input Image**:
3 x H x W

2D CNN

**Image Features**:
C x H' x W'

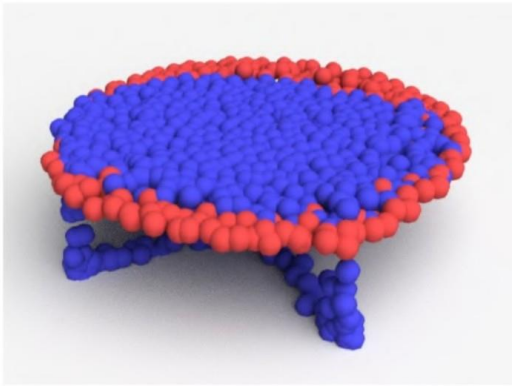
Fully connected branch

**Points**:
$P_1$ x 3

2D CNN

Convolutional branch

**Points**:
($P_2$x3) x H' x W'



**Pointcloud**:
($P_1$ + H'W'$P_2$) x 3

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

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

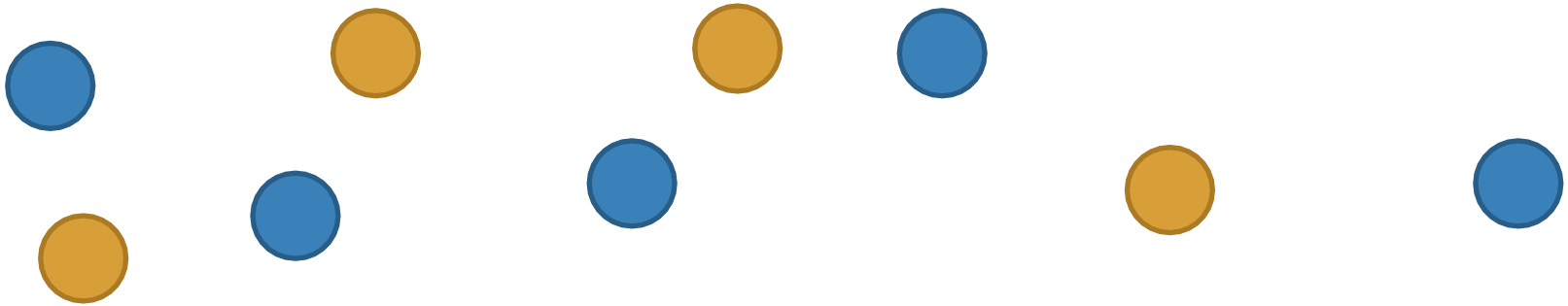
Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

# 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

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



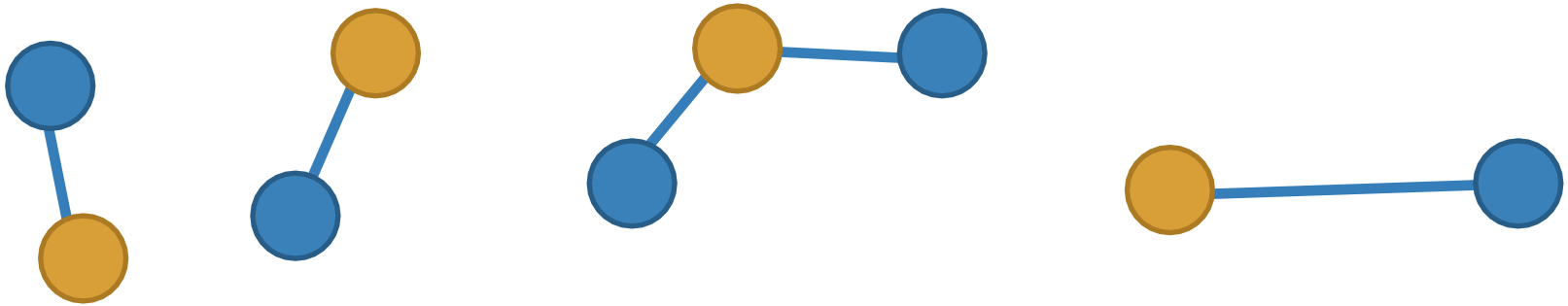
Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

# 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

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



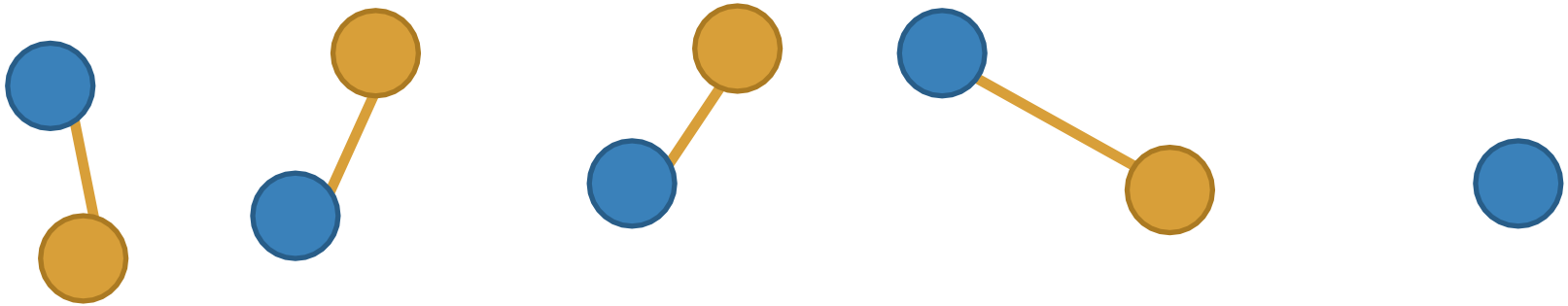
Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

# 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

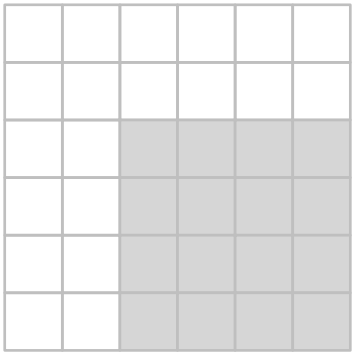
$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$



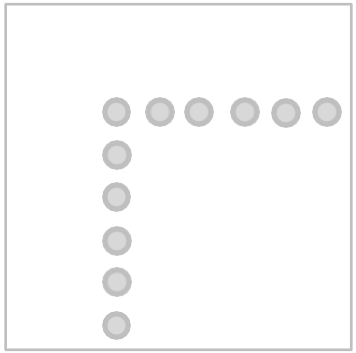
Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

# 3D Shape Representations

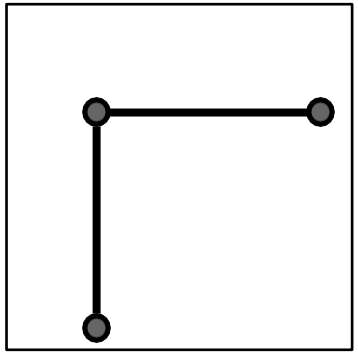

Depth Map



Voxel Grid



Pointcloud



Mesh

Implicit Surface

# 3D Shape Representations: Triangle Mesh

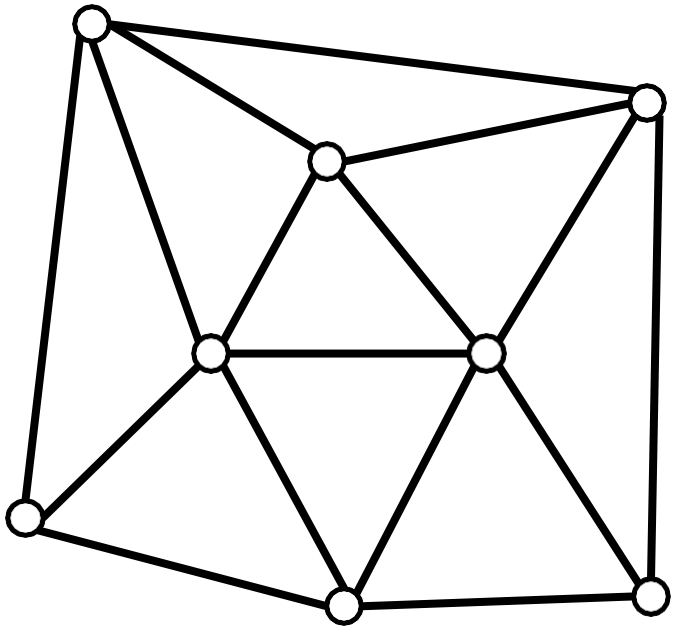
Represent a 3D shape as a set of triangles

**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

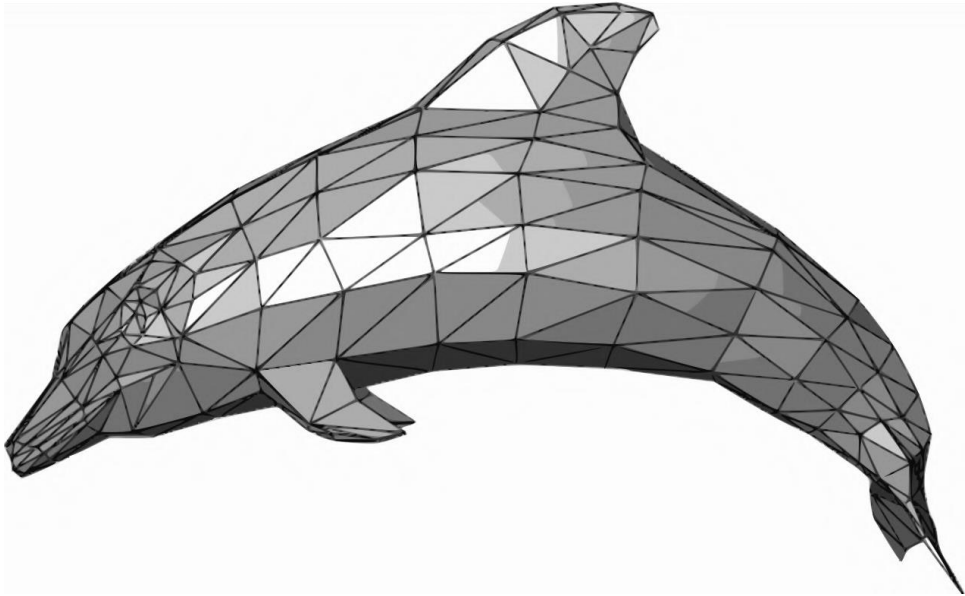
**Vertices**: Set of V points in 3D space

**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



# 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

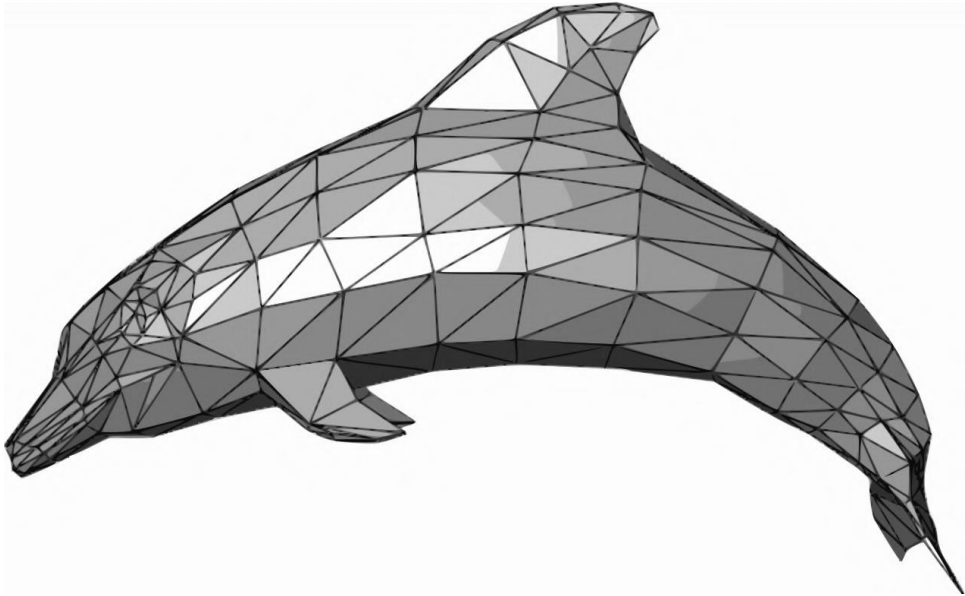
(+) Standard representation for graphics

(+) 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.

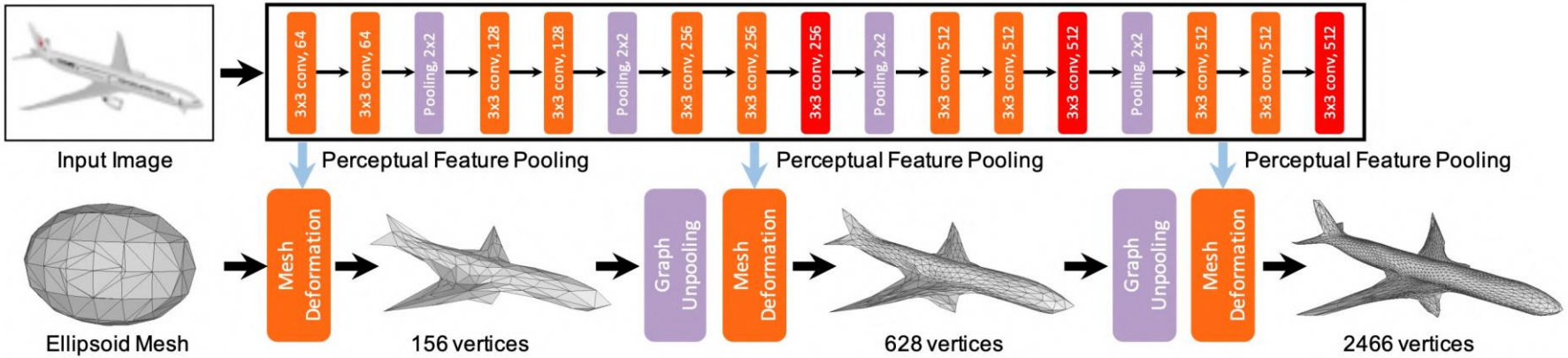
(-) Nontrivial to process with neural networks



# Predicting Meshes: Pixel2Mesh

**Input**: Single RGB Image of an object

**Output**: Triangle mesh for the object



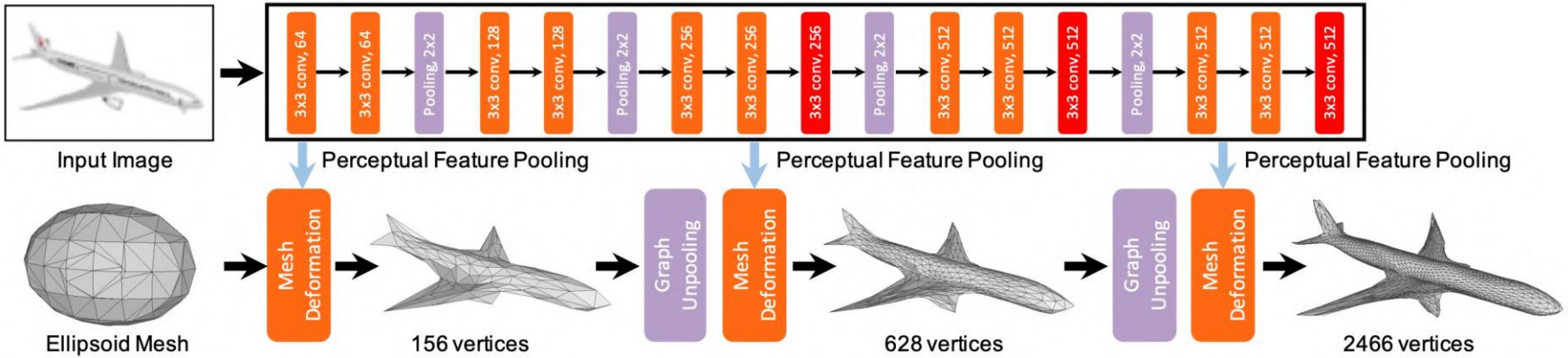

Wang et al, “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”, ECCV 2018

# 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

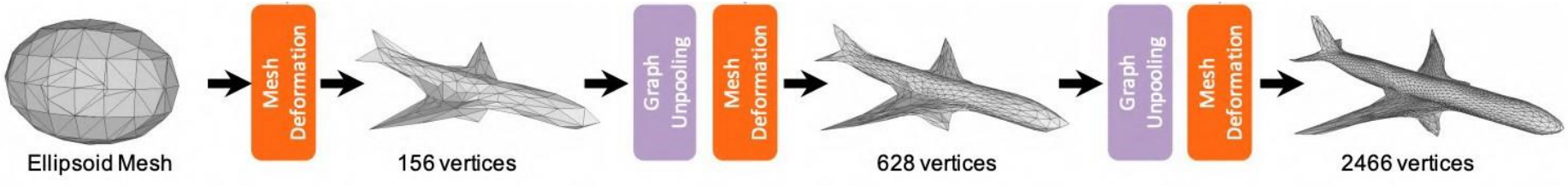
# Predicting Triangle Meshes: Iterative Refinement

**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

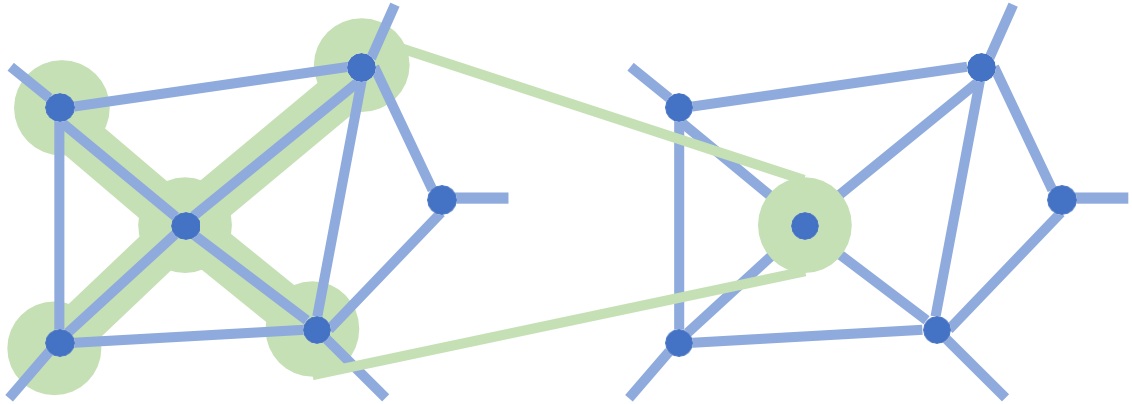
# Predicting Triangle Meshes: Graph Convolution

$$f_{)}^{*} = W_{+} f_{)} + \% \underset{\mathrm{j} \in N()}{} W_{1} f_{\mathrm{j}}$$

Vertex $v_{\mathrm{i}}$ has feature $f_{\mathrm{i}}$

New feature $f_{\mathrm{i}}^{"}$ for vertex $v_{\mathrm{i}}$ depends on feature of neighboring vertices $N(i)$

Use same weights $W_{\#}$ and $W_{\$}$ to compute all outputs

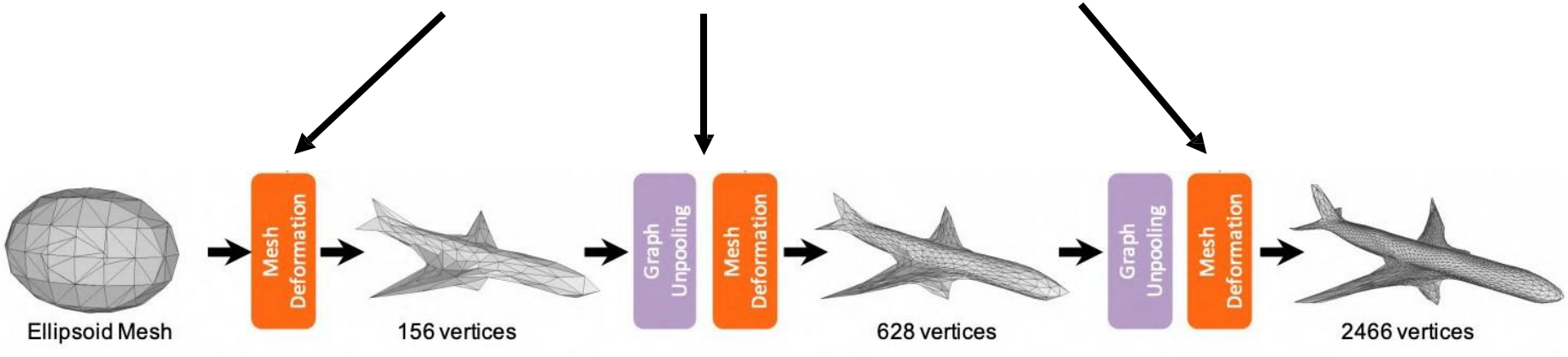


**Input**: Graph with a feature vector at each vertex

**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

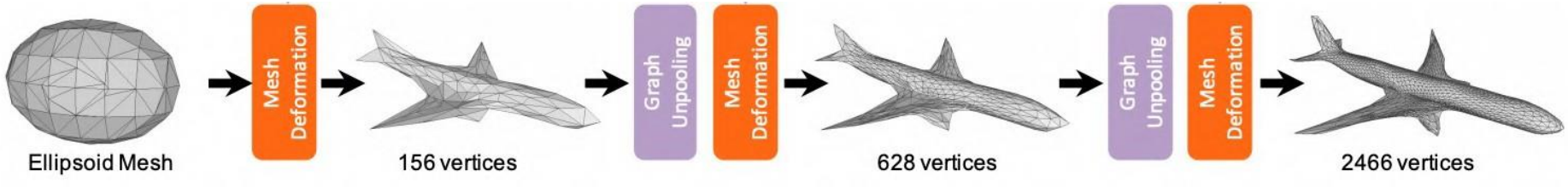



Wang et al, “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”, ECCV 2018

# 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?



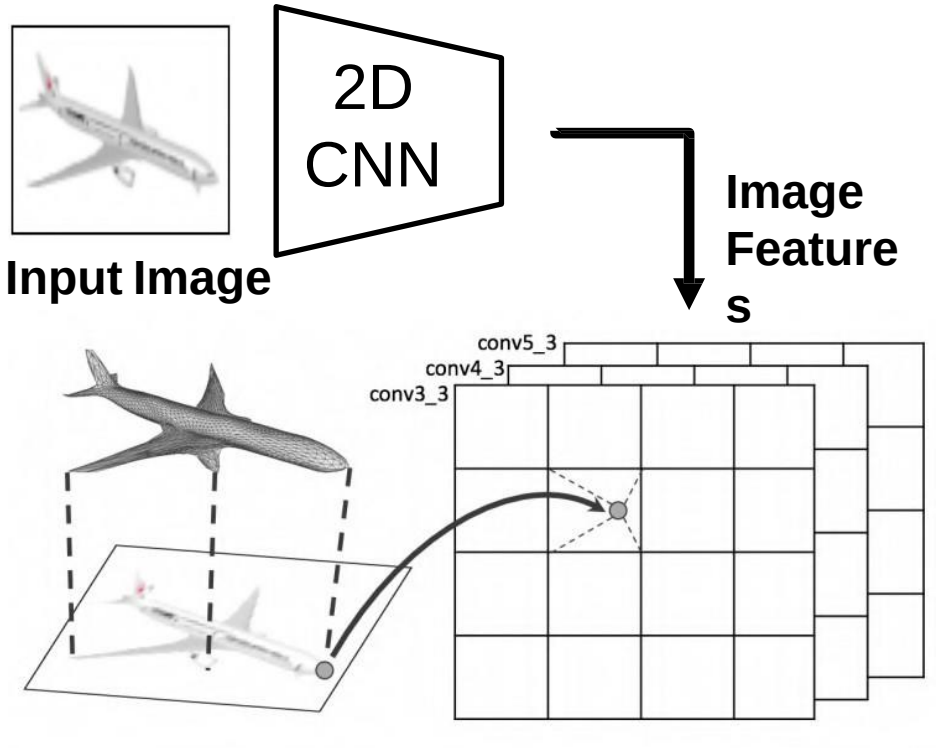

Wang et al, “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”, ECCV 2018

# 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

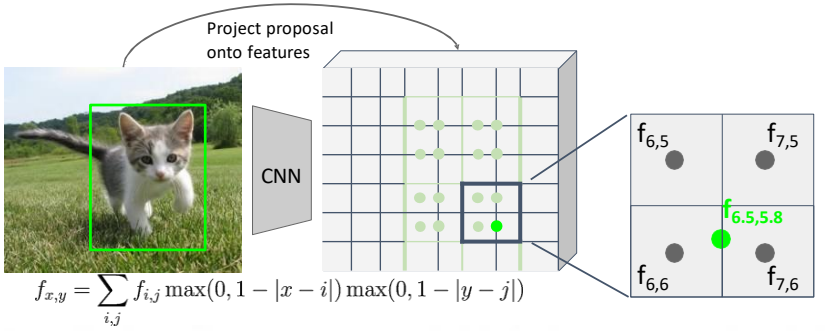
# Predicting Triangle Meshes: Vertex-Aligned Features

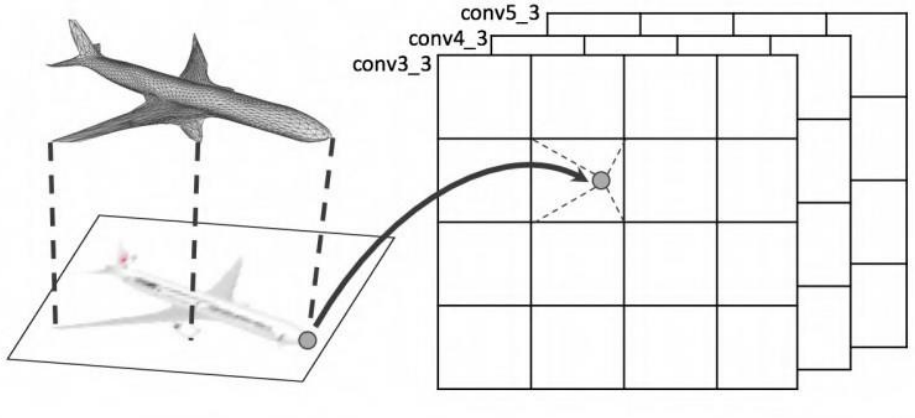
**Idea #2**: Aligned vertex features

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 detection: maintains alignment between input image and feature vectors

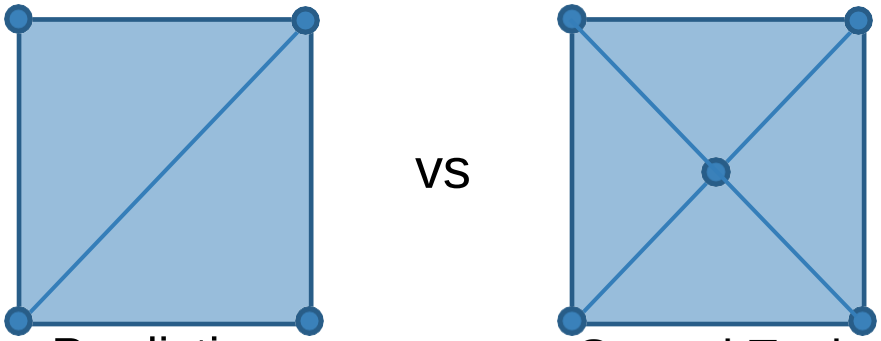



Wang et al, “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”, ECCV 2018

# 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?

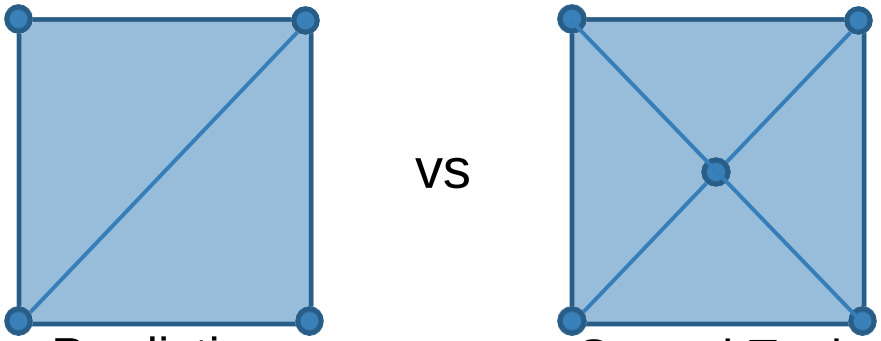



Wang et al, “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”, ECCV 2018

# 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

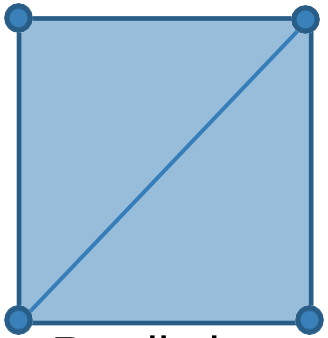



Wang et al, “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”, ECCV 2018

# Predicting Meshes: Loss Function

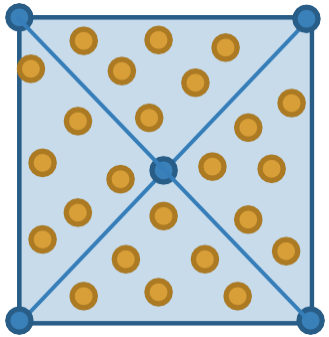
The same shape can be represented with different meshes – how can we define a loss between predicted and ground-truth mesh?

**Idea:** Convert meshes to pointclouds, then compute loss



Prediction

vs



Ground-Truth

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

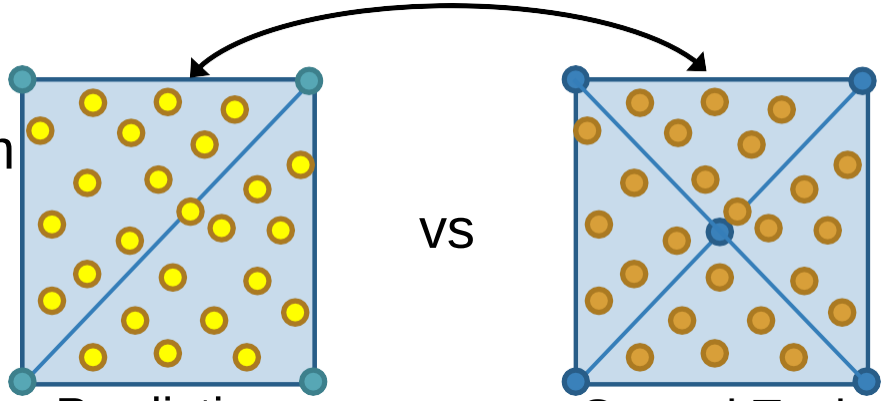
Wang et al, “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”, ECCV 2018

# 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 predicted mesh (online)




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

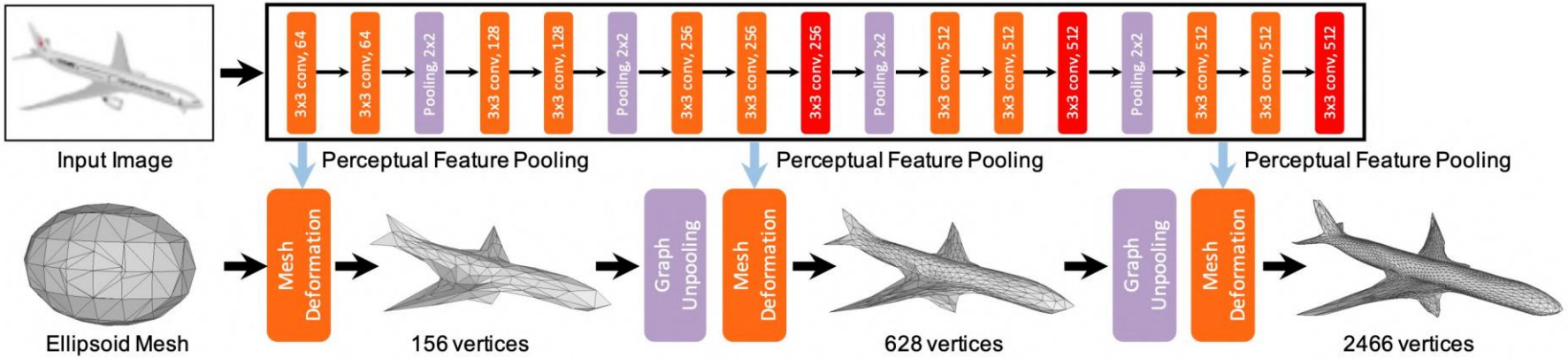
Wang et al, “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”, ECCV 2018

# Predicting Meshes: Pixel2Mesh

**Input**: Single RGB Image of an object

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

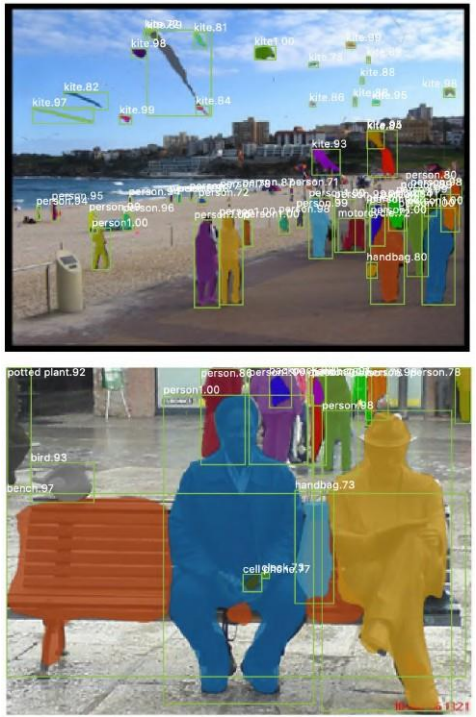
**Output**: Triangle mesh for the object



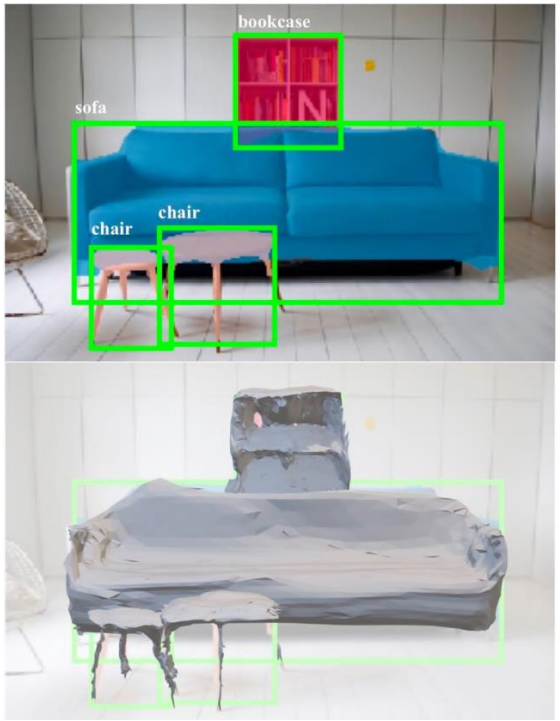

Wang et al, “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images”, ECCV 2018

# 3D Shape 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**:

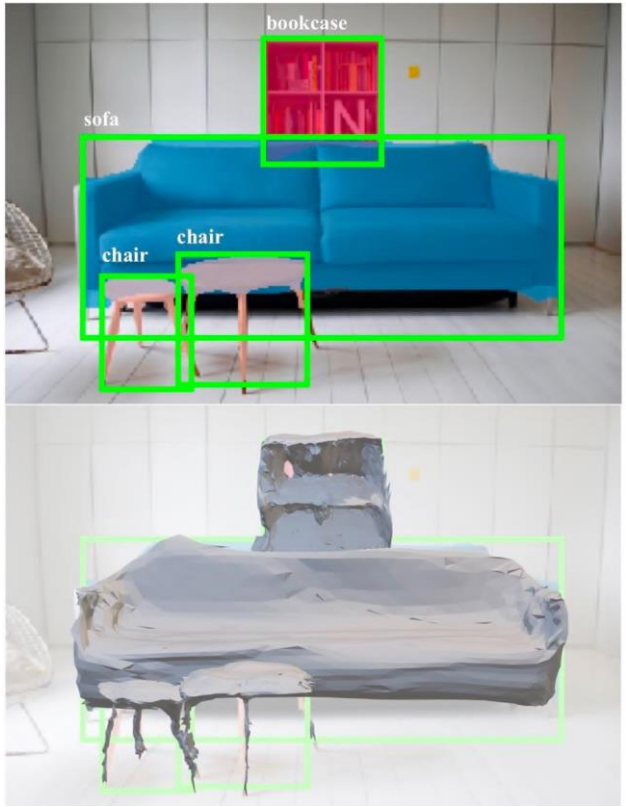
A set of detected objects

For each object:

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

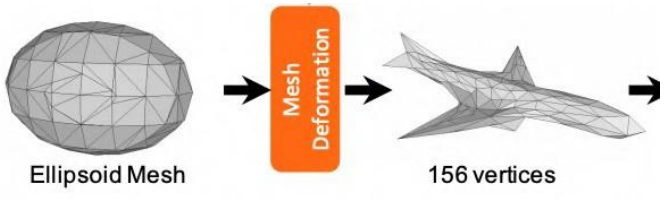
Mask R-CNN

Mesh head



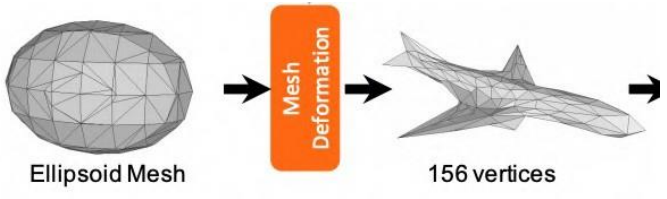
# Mesh R-CNN: Hybrid 3D shape representation

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

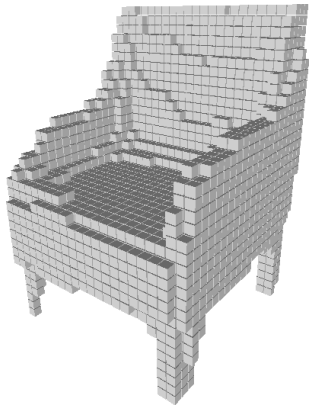


# Mesh R-CNN: Hybrid 3D shape representation

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




**Mesh R-CNN**: 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

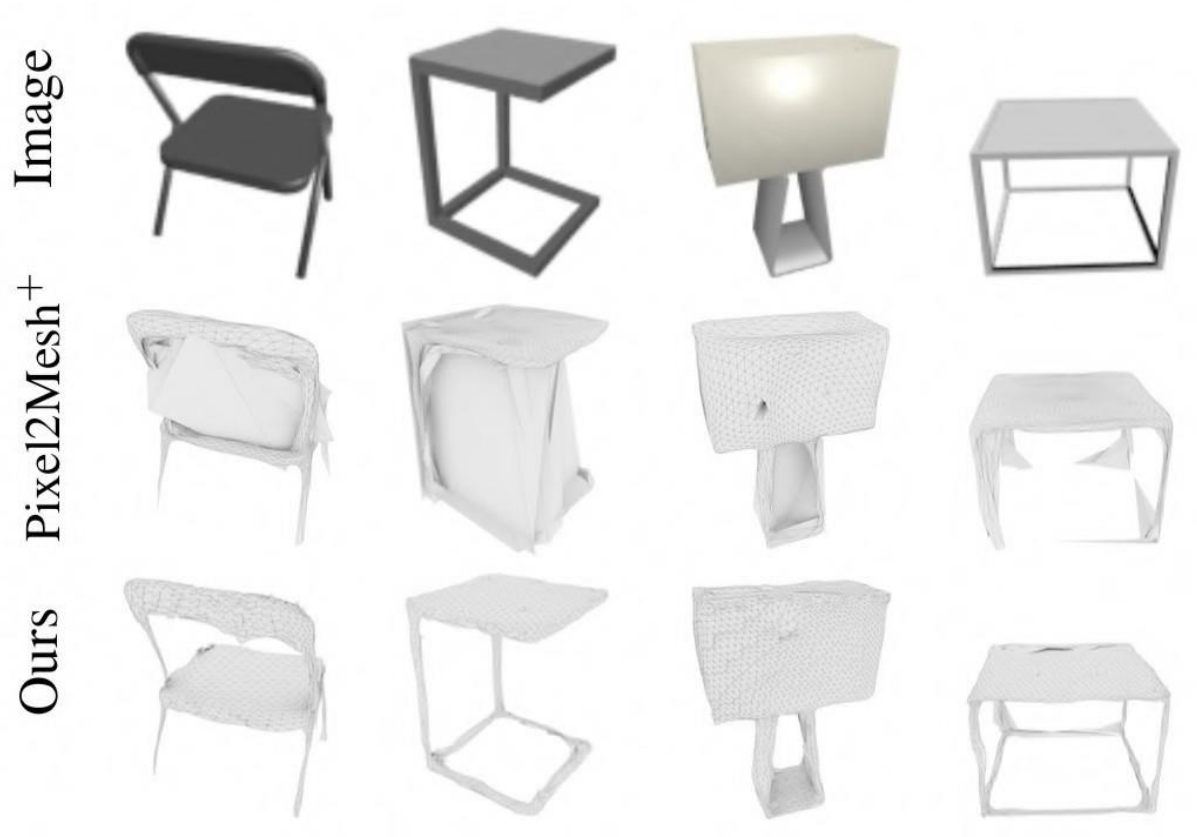
Input image

2D object recognition

3D object meshes

3D object voxels

# Mesh R-CNN: ShapeNet Results



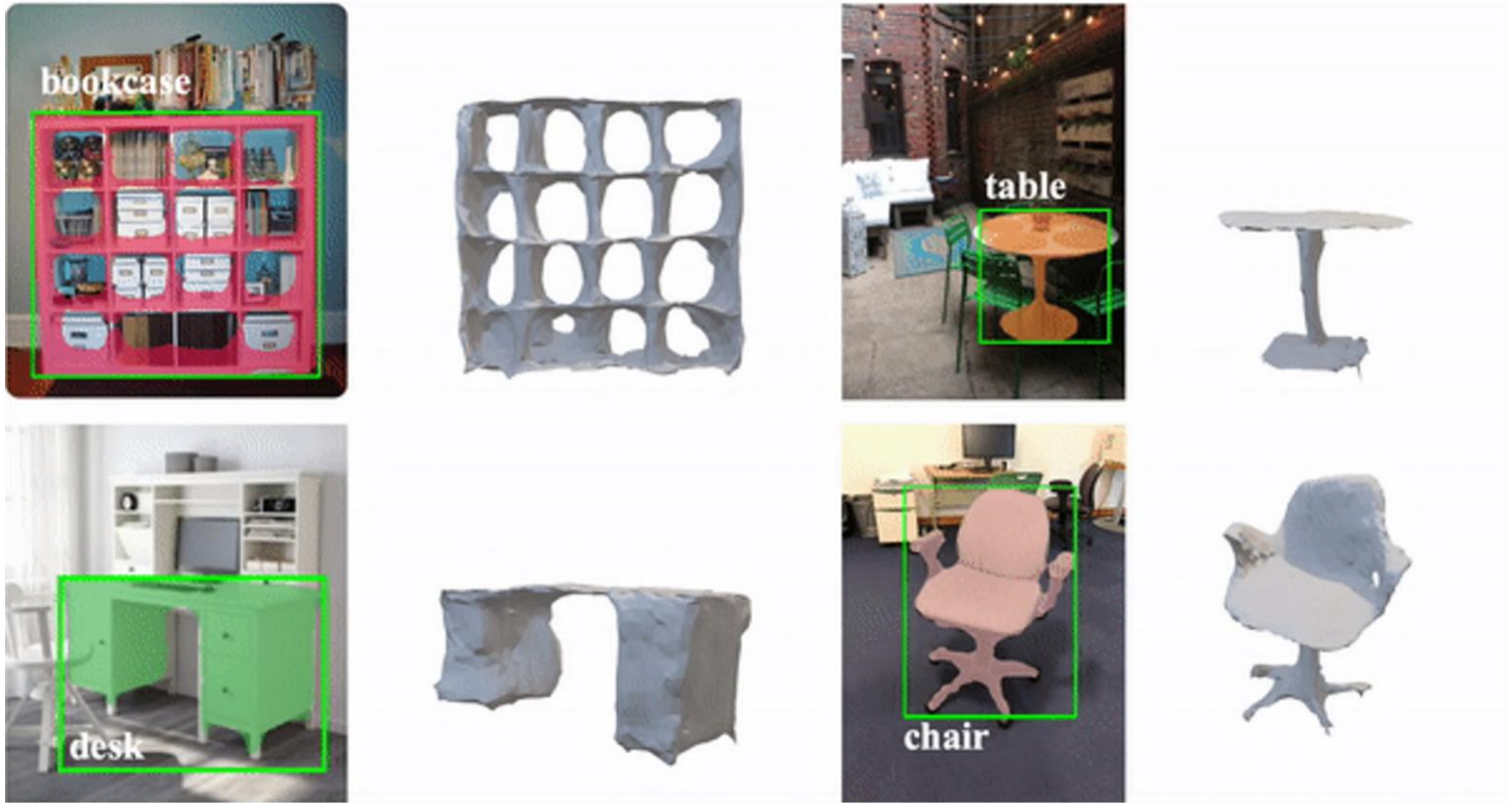
# Datasets for 3D Objects

- Large-scale Synthetic Objects: ShapeNet, 3M models
- ModelNet: absorbed by ShapeNet
- ShapeNetCore: 51.3K models in 55 categories

Chang et al. ShapeNet. arXiv 2015

Wu et al. 3D ShapeNets. CVPR 2015

# Mesh R-CNN: Pix3D Results



# Pix3D

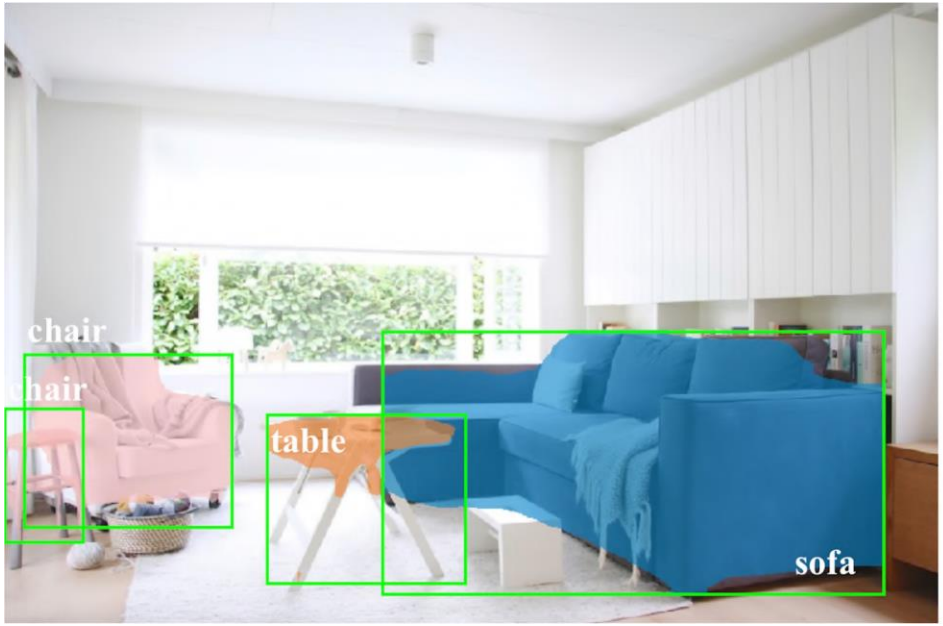
- 10,069 images
- 395 shapes (IKEA furniture + 3D scan)



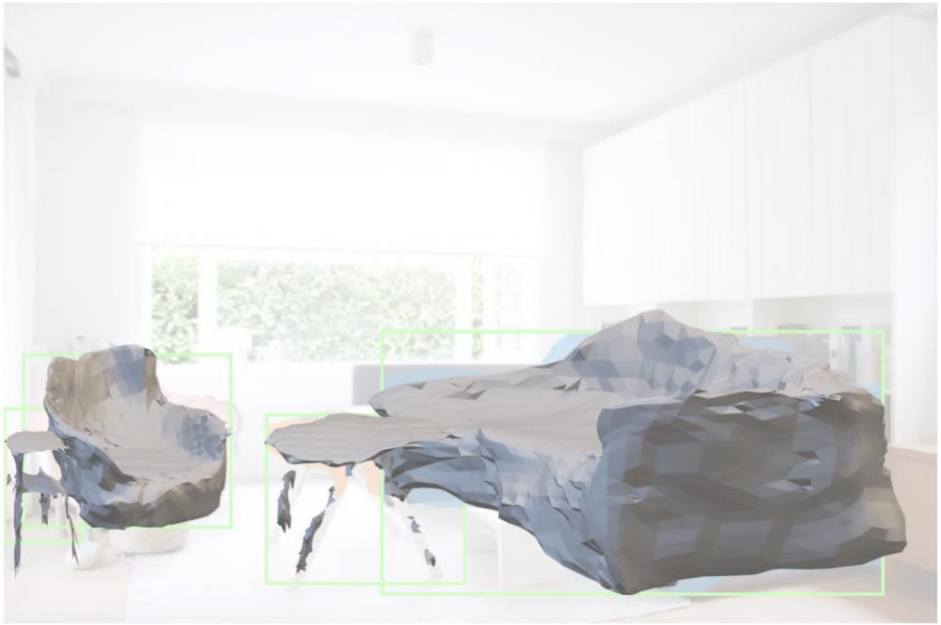

Sun et al. CVPR 2018, building upon Lim et al. ICCV 2013

# Mesh R-CNN: Pix3D Results

Predicting many objects per scene



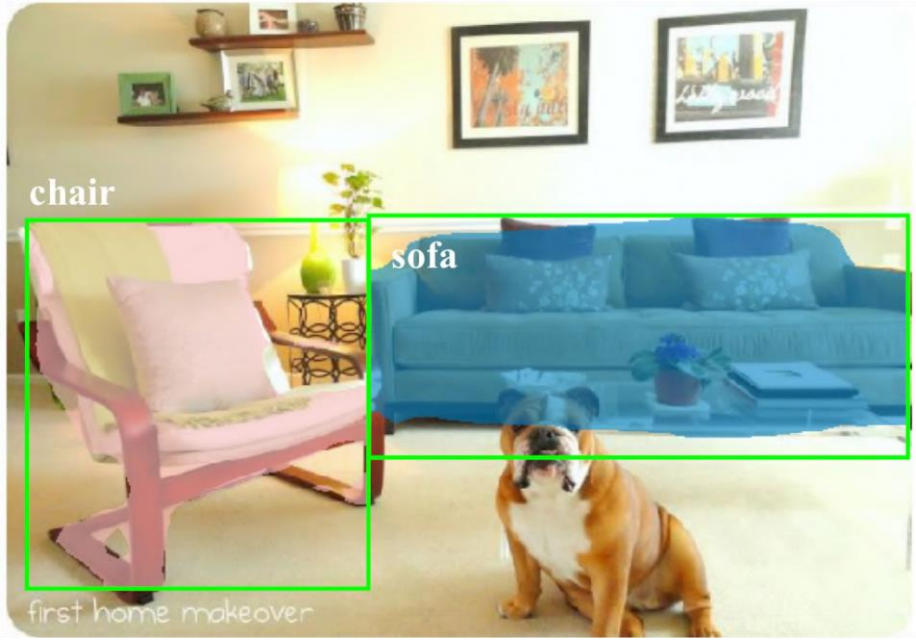

Box & Mask Predictions



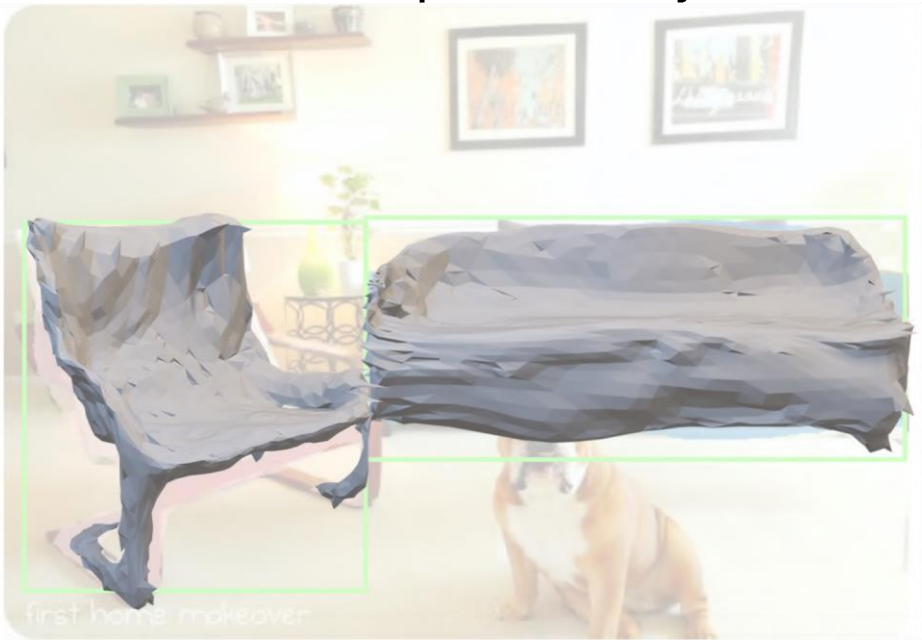
Mesh Predictions

# 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

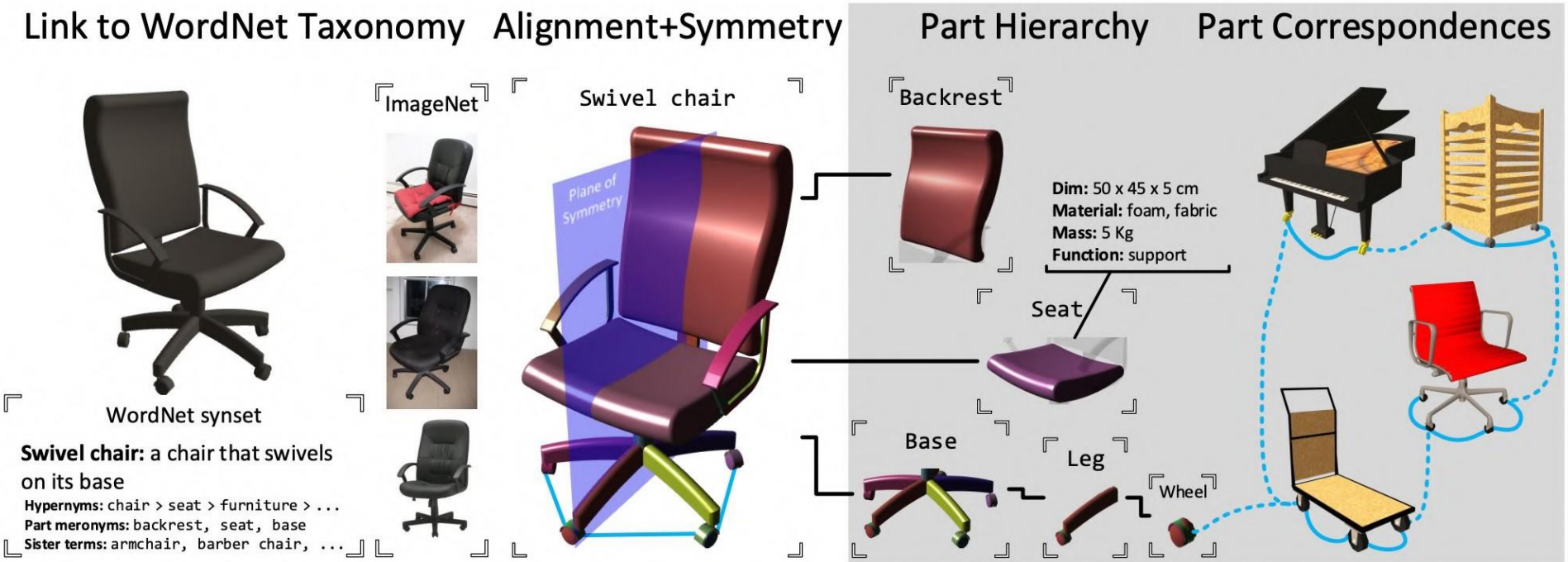
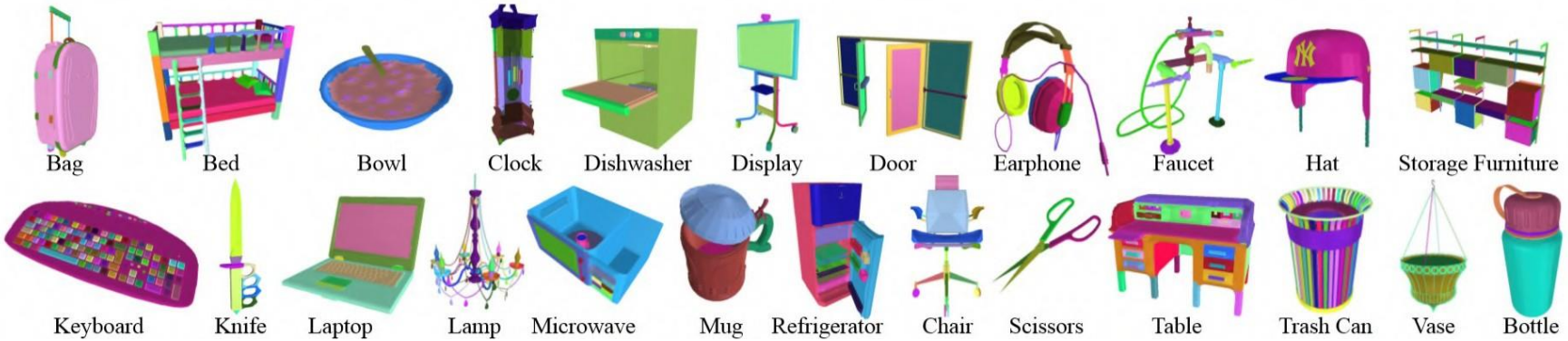



Figure from the ShapeNet paper, Chang et al. arXiv 2015

# Datasets for 3D Object Parts

**Fine-grained Parts: PartNet**

- Fine-grained (+mobility)
- Instance-level
- Hierarchical




Mo et al. CVPR 2019
Slide credit: Hao Su

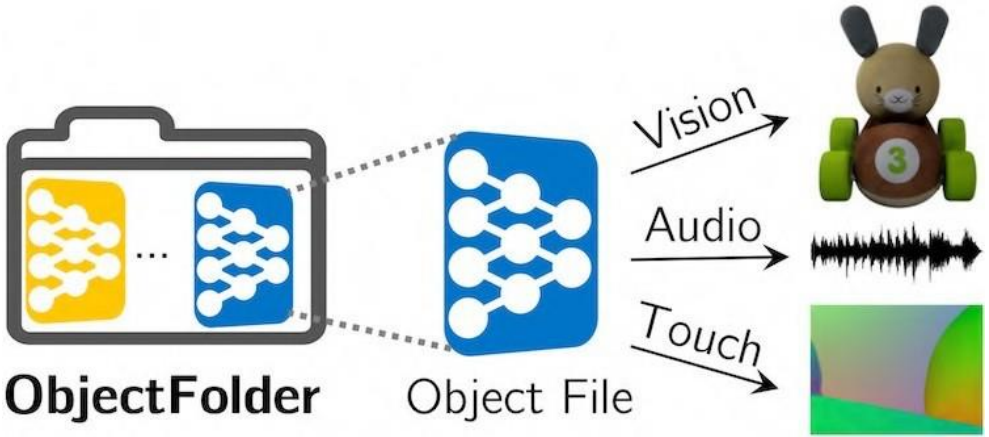
# Physical Interaction with Articulated Objects

**300+ door annotations**

**support articulated objects**
(cabinets, doors, fridge, oven, window etc.)

http://svl.stanford.edu/igibson/

# ObjectFolder




Multisensory neural objects

Multisensory real objects

Gao et al. CVPR 2023. https://objectfolder.stanford.edu/

# Visual Data in ObjectFolder Real

Gao et al. CVPR 2023. https://objectfolder.stanford.edu/

# Acoustic Data in ObjectFolder Real

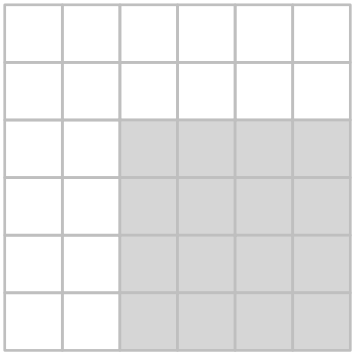
Gao et al. CVPR 2023. https://objectfolder.stanford.edu/

# Tactile Data in ObjectFolder Real

Gao et al. CVPR 2023. https://objectfolder.stanford.edu/

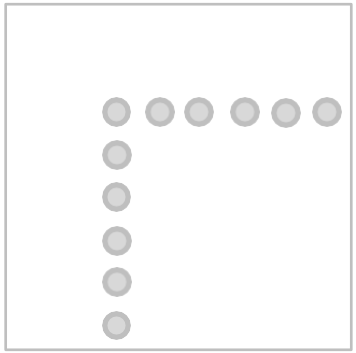
# 3D Shape Representations


Depth Map

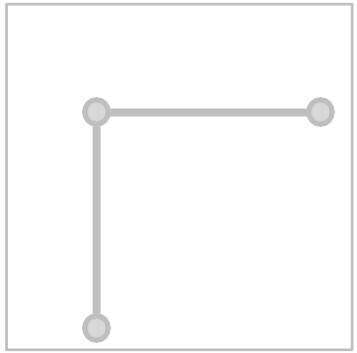


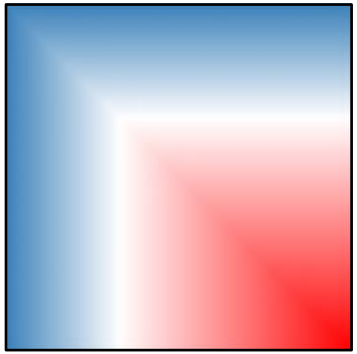
Voxel Grid

Mesh



Pointcloud





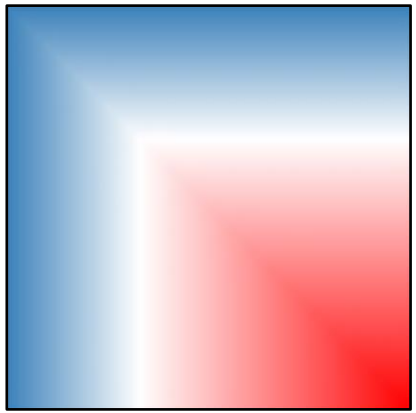
Implicit Surface

# 3D Shape Representations: Implicit Functions

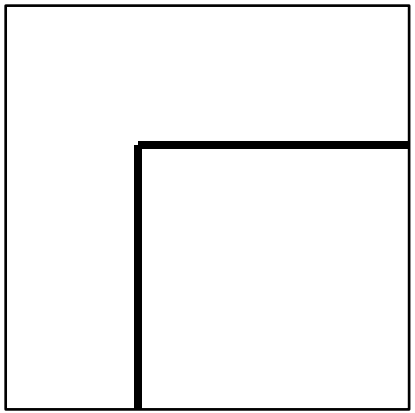
Learn a function to classify arbitrary 3D points as inside / outside the shape

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

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



Implicit function



Explicit Shape

# Algebraic Surfaces (Implicit)

**Surface is zero set of a polynomial in x, y, z**

$x^2+y^2+z^2=1$

$(R-\sqrt{x^2+y^2})^2+z^2=r^2$

$(x^2+\frac{9y^2}{4}+z^2-1)^3=x^2z^3+\frac{9y^2z^3}{80}$

Slide credit: Ren Ng

# Algebraic Surfaces (Implicit)

**Surface is zero set of a polynomial in x, y, z**

$$x^2+y^2+z^2=1$$

$$(R-\sqrt{x^2+y^2})^2+z^2=r^2$$

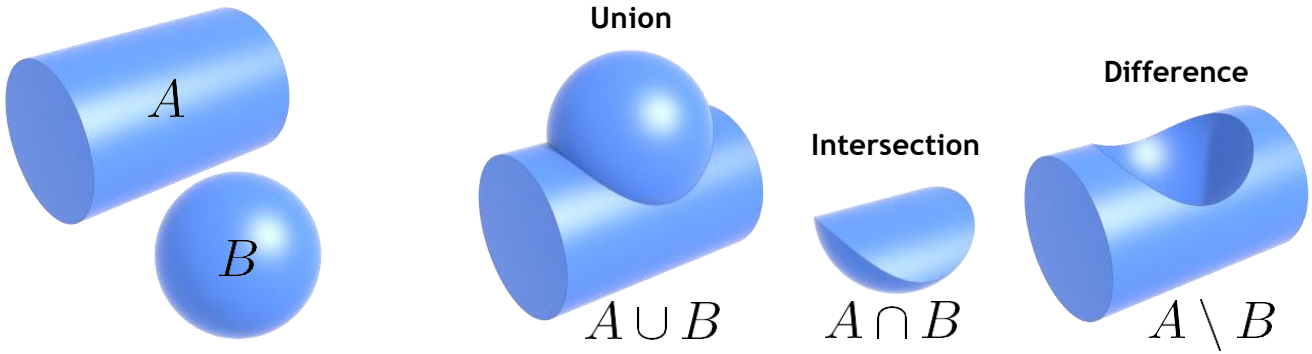
$$(x^2+\frac{9y^2}{4}+z^2-1)^3=x^2z^3+\frac{9y^2z^3}{80}$$

**More complex shapes?**

Slide credit: Ren Ng

# Constructive Solid Geometry (Implicit)

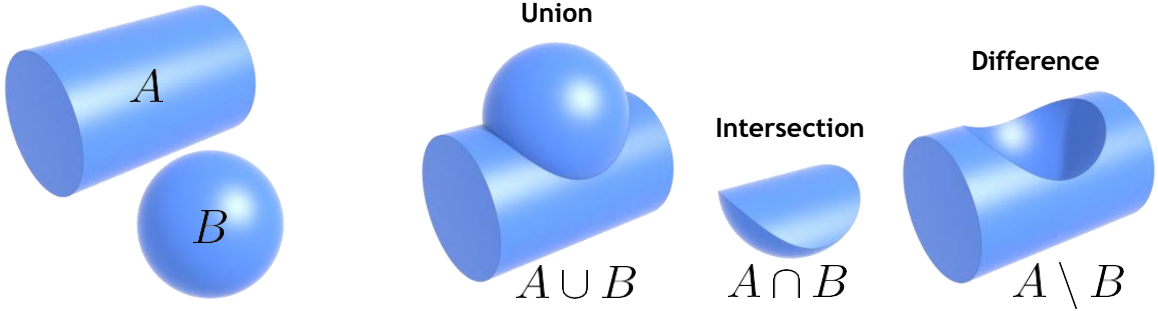
## Combine implicit geometry via Boolean operations



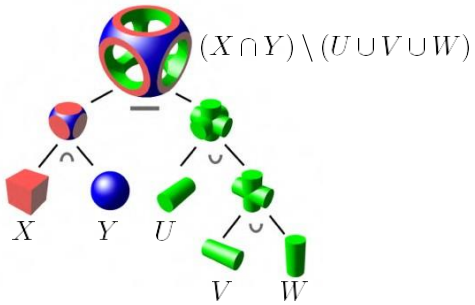

Slide credit: Ren Ng

# Constructive Solid Geometry (Implicit)

## Combine implicit geometry via Boolean operations




## Boolean expressions:





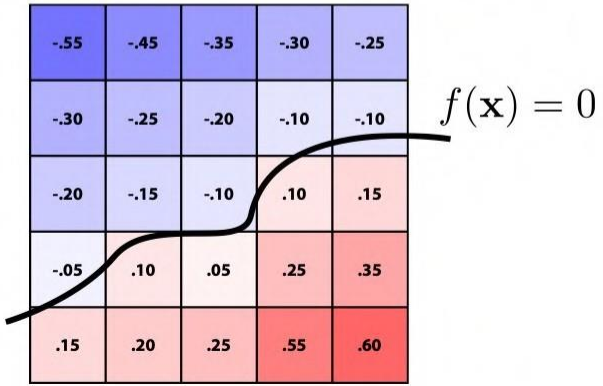

Slide credit: Ren Ng

# Level Set Methods (Implicit)

Implicit surfaces have some nice features (e.g., merging/splitting)

But, hard to describe complex shapes in closed form

Alternative: store a grid of values approximating function



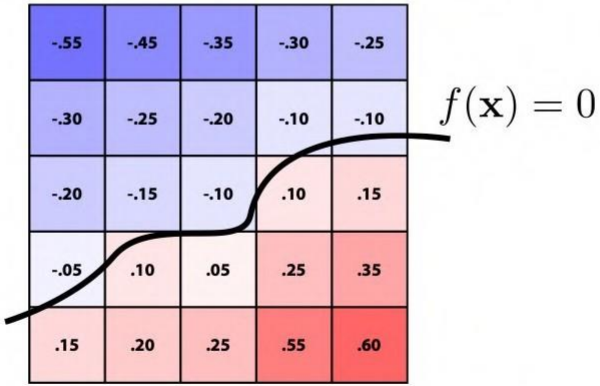

Slide credit: Ren Ng

# Level Set Methods (Implicit)

Implicit surfaces have some nice features (e.g., merging/splitting)

But, hard to describe complex shapes in closed form

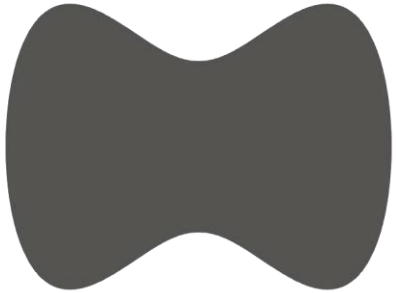
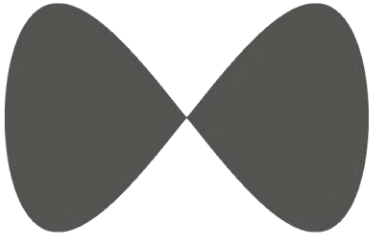
Alternative: store a grid of values approximating function

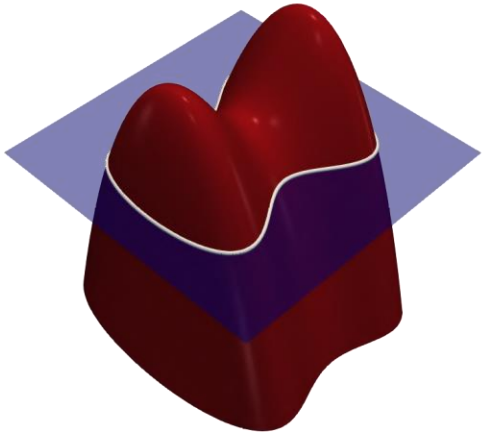
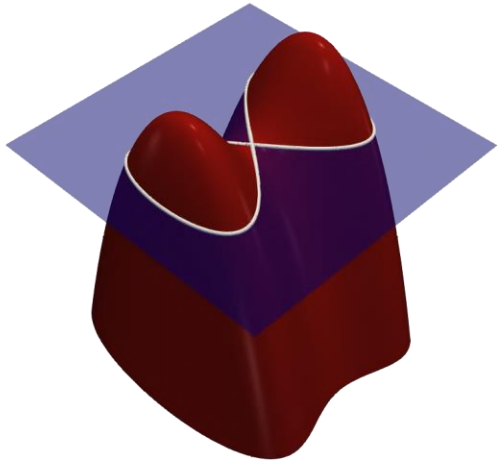
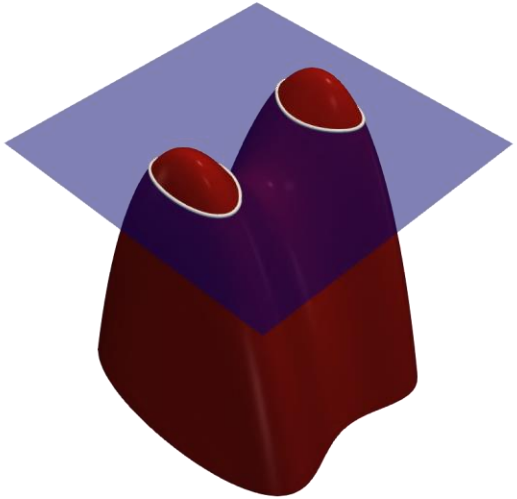



Surface is found where interpolated values equal zero

Provides much more explicit control over shape (like a texture)

Slide credit: Ren Ng

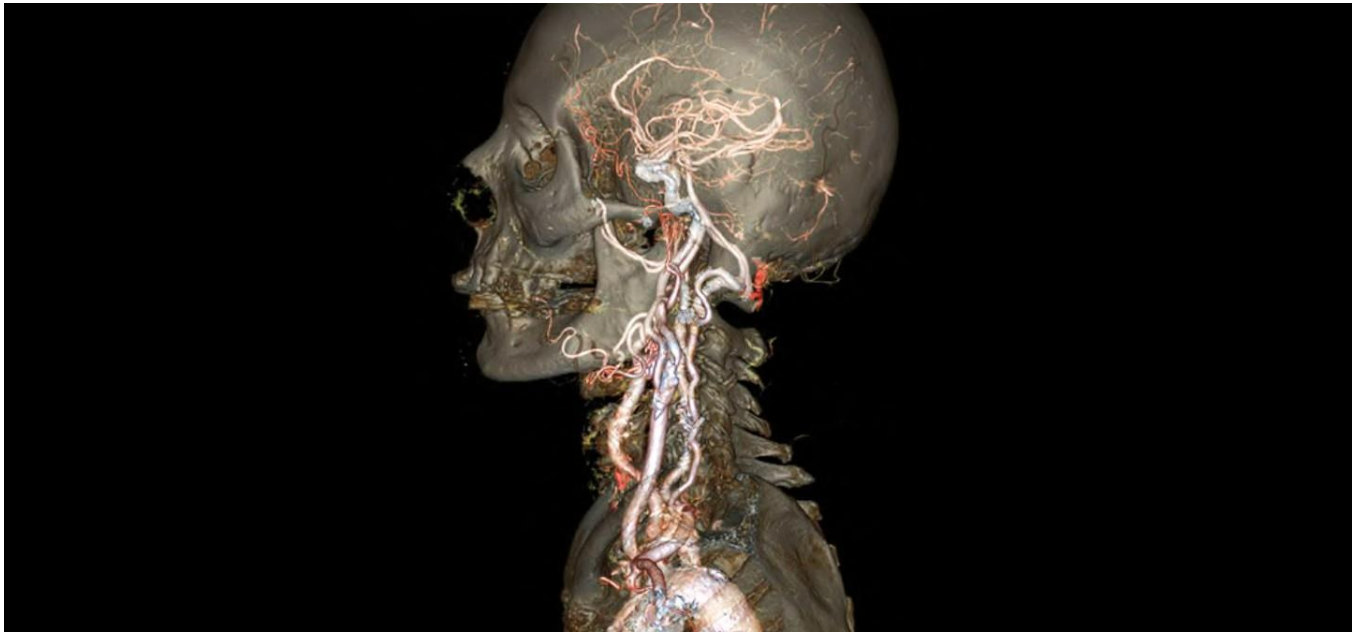



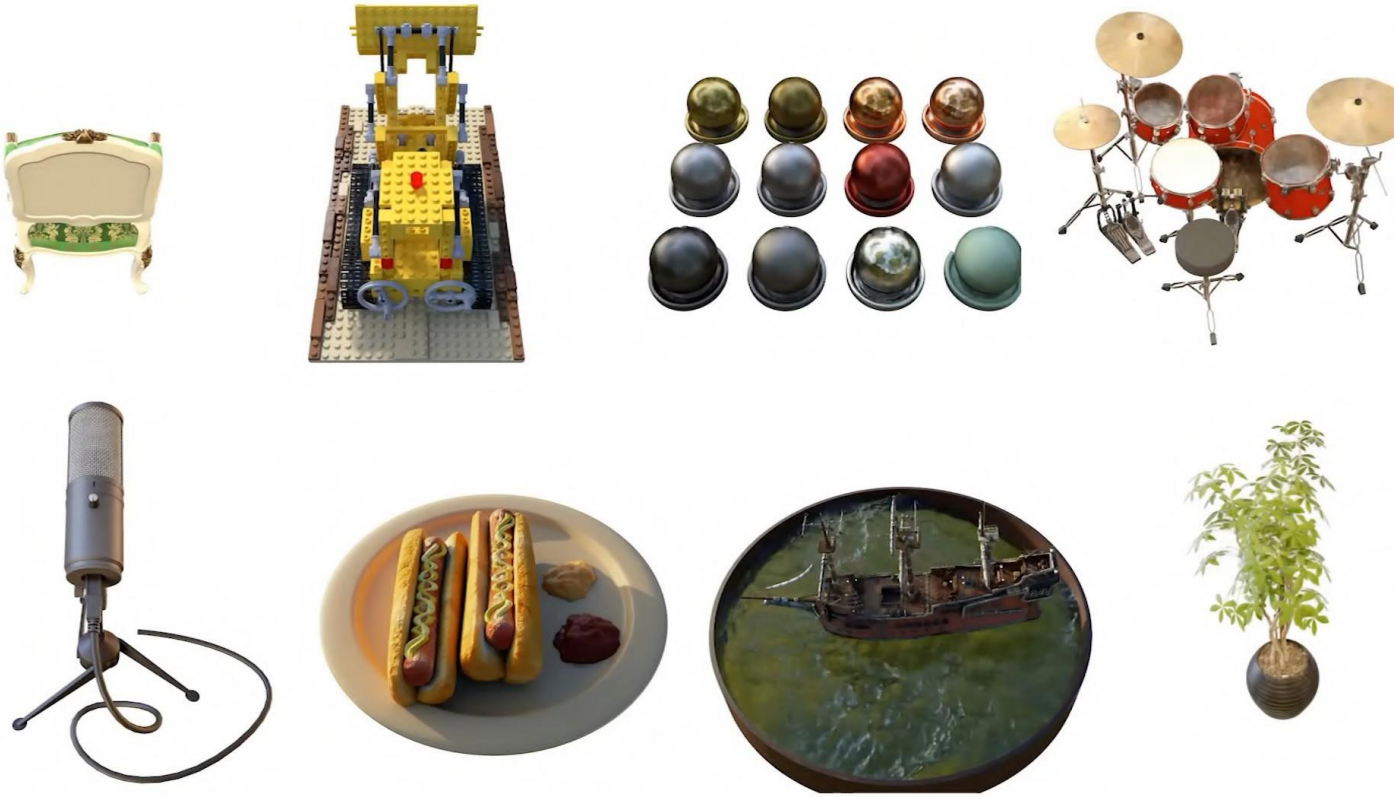
Slide credit: Ren Ng

# Level Sets from Medical Data (CT, MRI, etc.)

**Level sets encode, e.g., constant tissue density**

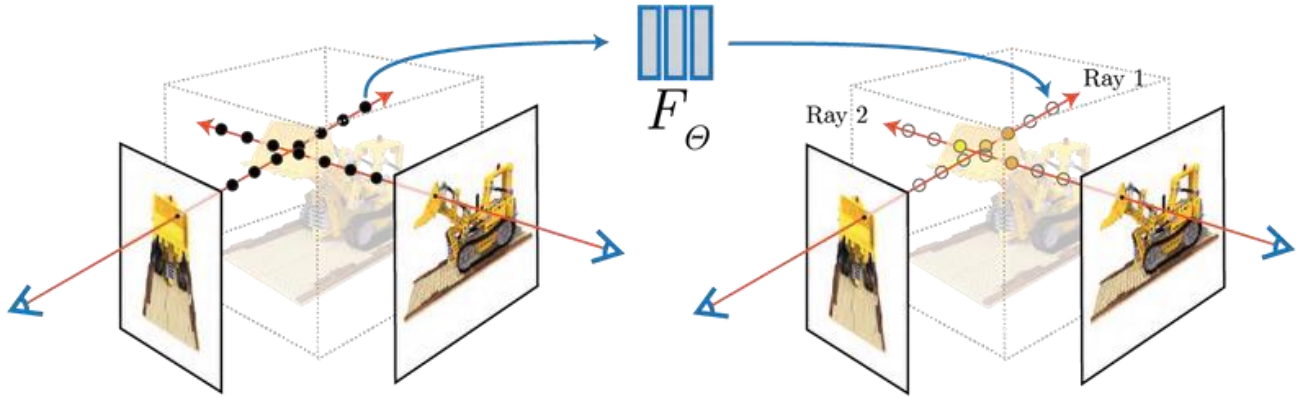


Slide credit: Ren Ng



Mildenhall et al, “Representing Scenes as Neural Radiance Fields for View Synthesis”, ECCV 2020

# NeRF: Representing Scenes as Neural Radiance Fields




Novel view synthesis

Scene location $(x, y, z)$ Viewing direction $(\theta, \phi)$

$\rightarrow$ $F_\theta$ $\rightarrow$

Output color $(r, g, b)$ Output density $(\sigma)$

Mildenhall et al, “Representing Scenes as Neural Radiance Fields for View Synthesis”, ECCV 2020

Mildenhall et al, “Representing Scenes as Neural Radiance Fields for View Synthesis”, ECCV 2020

Mildenhall et al, “Representing Scenes as Neural Radiance Fields for View Synthesis”, ECCV 2020

Main Problem: Very slow!

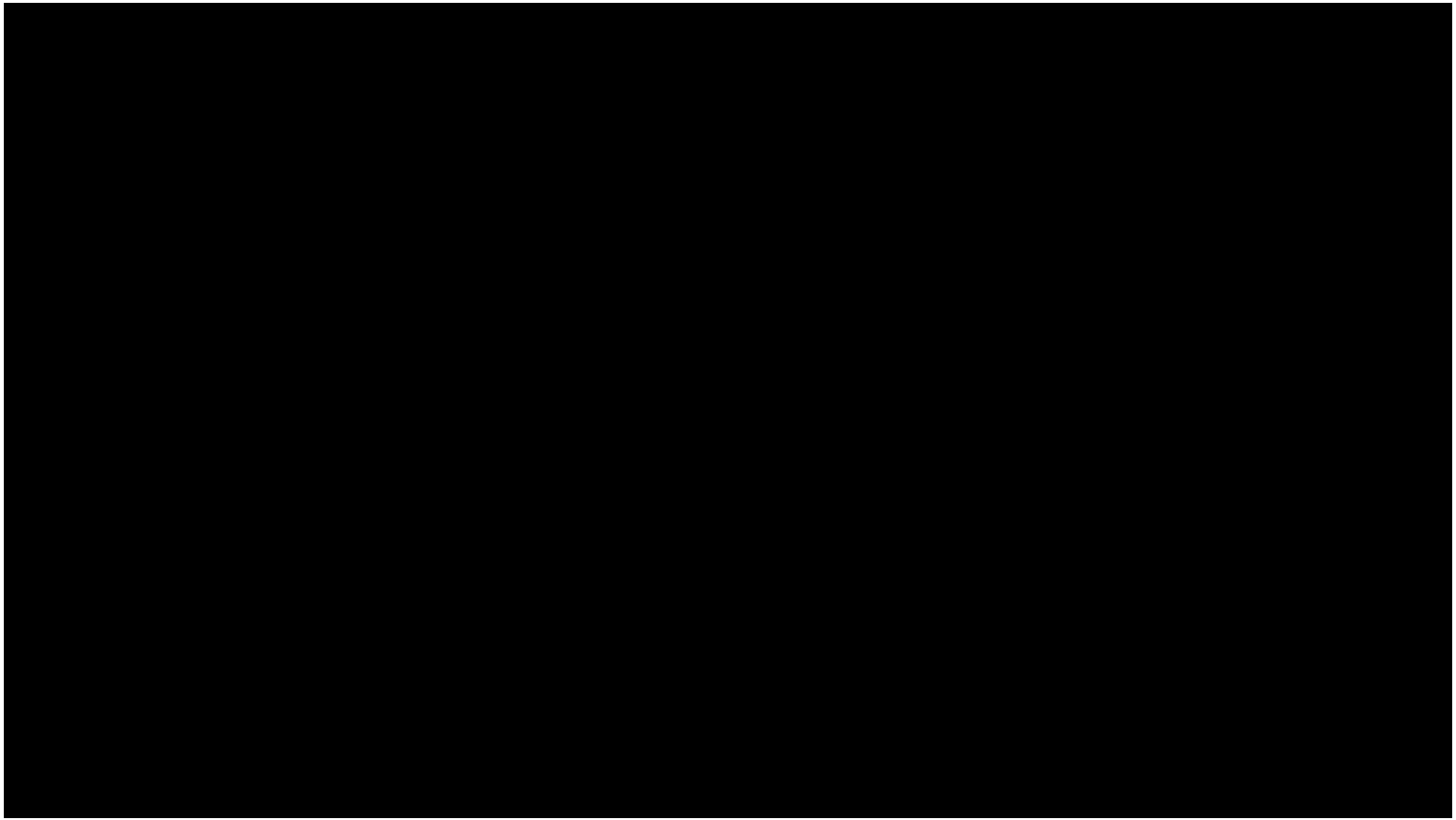
**Training**: 1-2 days on a V100 GPU, for just a single scene!

**Inference**: Sampling an image from a trained model:
(256 x 256 pixels) x (224 samples per pixel)
= 14.6M forward passes through MLP

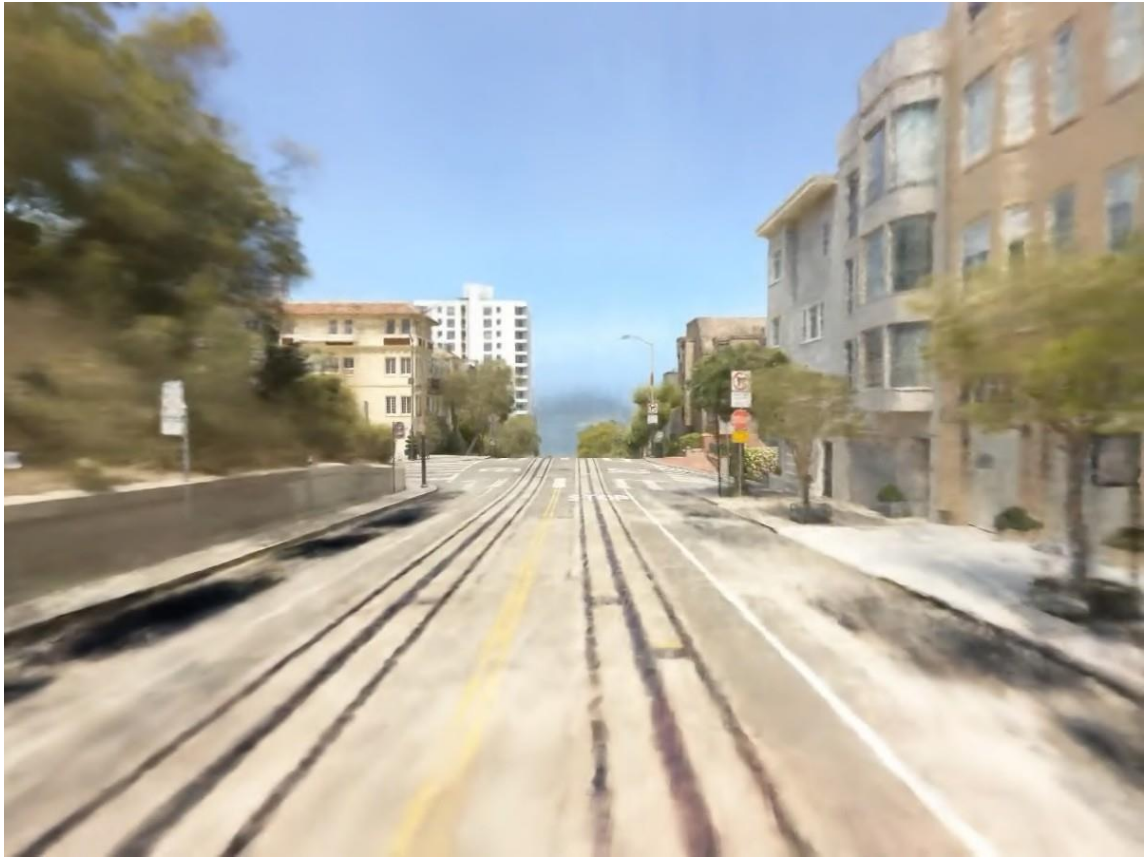
Tons of follow-up work!

Mildenhall et al, “Representing Scenes as Neural Radiance Fields for View Synthesis”, ECCV 2020


Park et al, “Nerfies: Deformable Neural Radiance Fields”, ICCV 2021
https://nerfies.github.io/



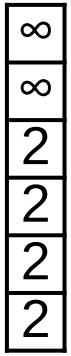
Mildenhall et al, “NeRF in the Dark: High Dynamic Range View Synthesis from Noisy Raw Images”, CVPR 2022
https://bmild.github.io/rawnerf/



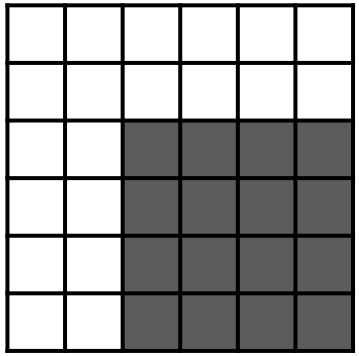
Tancik et al, “Block-NeRF: Scalable Large Scene Neural View Synthesis”, CVPR 2022
https://waymo.com/intl/zh-cn/research/block-nerf/

DreamFusion: Text-to-3D using 2D Diffusion, Ben et al., arXiv 2022

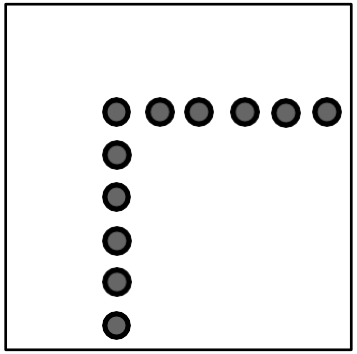
# Summary: 3D Shape Representations



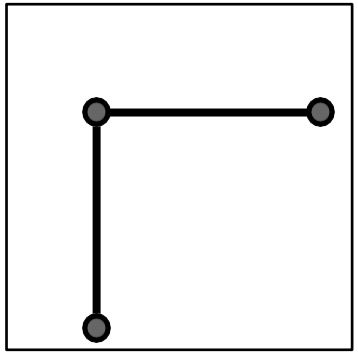

Depth Map



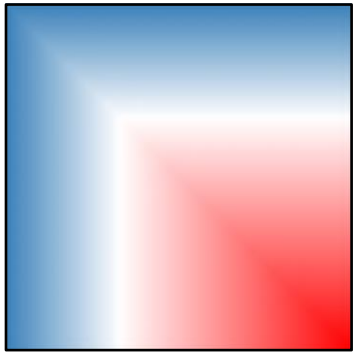
Voxel Grid



Pointcloud



Mesh



Implicit Surface