ROB 498/599: Deep Learning for Robot Perception (DeepRob)

Lecture 20: 3D Vision Techniques; Nerf

03/31/2025

https://deeprob.org/w25/

ROBOTICS

Today

- Feedback and Recap (5min)
- 3D Shape Representations (40min)
- NeRF (25min)
- Summary and Takeaways (5min)

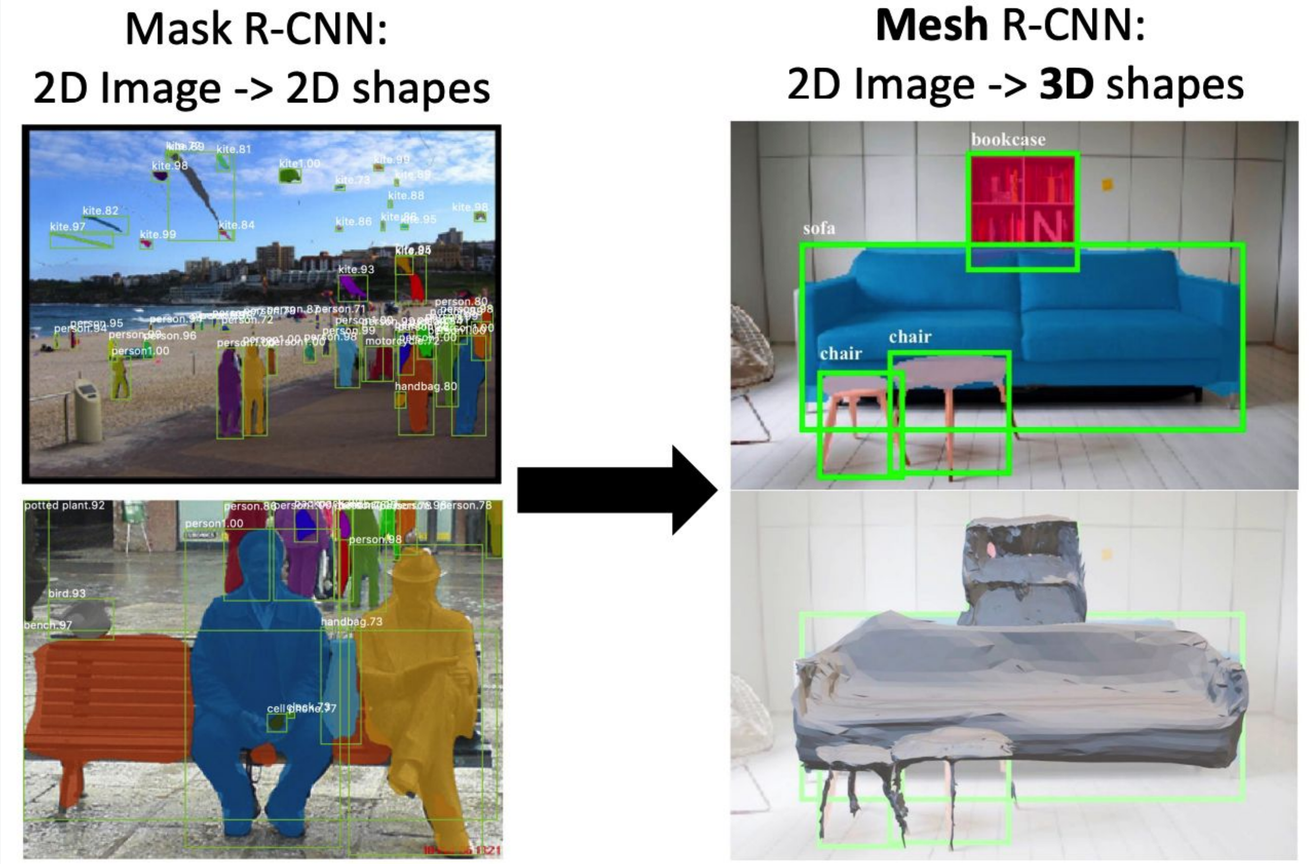
Tomorrow

Tuesday April 1, 2025
Starts 3:30pm EST @CSRB 2246
5min “lightning talk” - final project proposal presentations

- All group members must speak
- Upload presentation files to Final_Project_LightningTalk folder on Google Drive by April 1st 3:30pm EST
- We will go by group number order
- 5% grade

3D Shape Representations

Previously… Predicting 3D Shapes of Objects

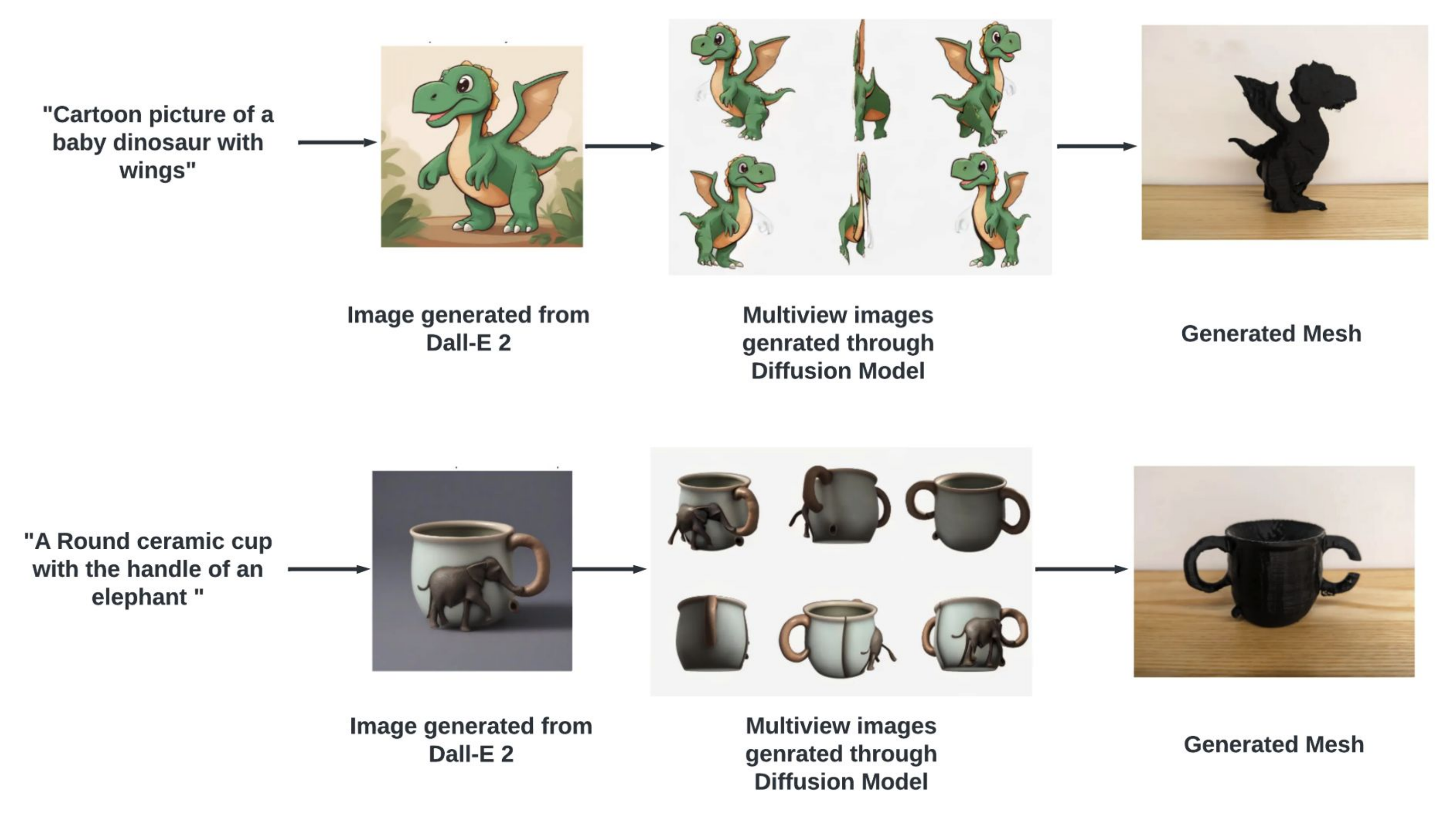


He, Gkioxari, Dollár, and Girshick, “Mask R-CNN”, ICCV 2017
https://arxiv.org/abs/1703.06870

Gkioxari, Malik, and Johnson, “Mesh R-CNN”, ICCV 2019
https://openaccess.thecvf.com/content_ICCV_2019/papers/Gkioxari_Mesh_R-CNN_ICCV_2019_paper.pdf

ROBOTICS

Also recall… “Speech2Mesh”



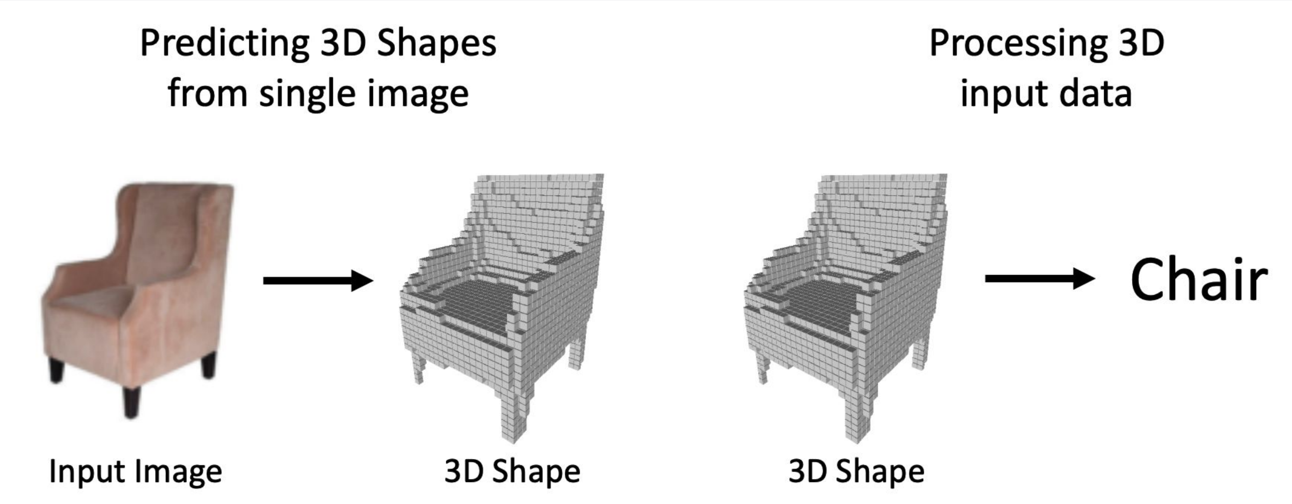
https://deeprob.org/w24/reports/speech2mesh/

DeepRob W24 Project

(Adeshara et al.)

ROBOTICS

Today



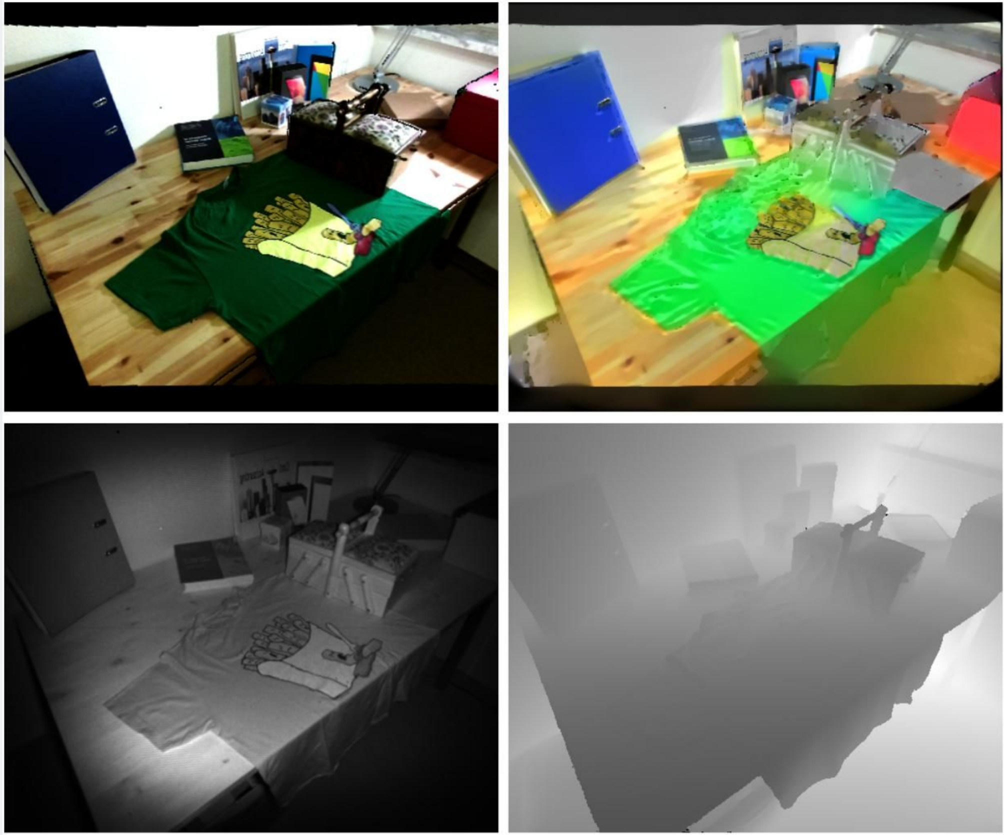
More topics in 3DV:

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

https://3dvconf.github.io/2025/ (3DV)
https://s2025.siggraph.org/ (SIGGRAPH)

M | ROBOTICS

3D data modalities



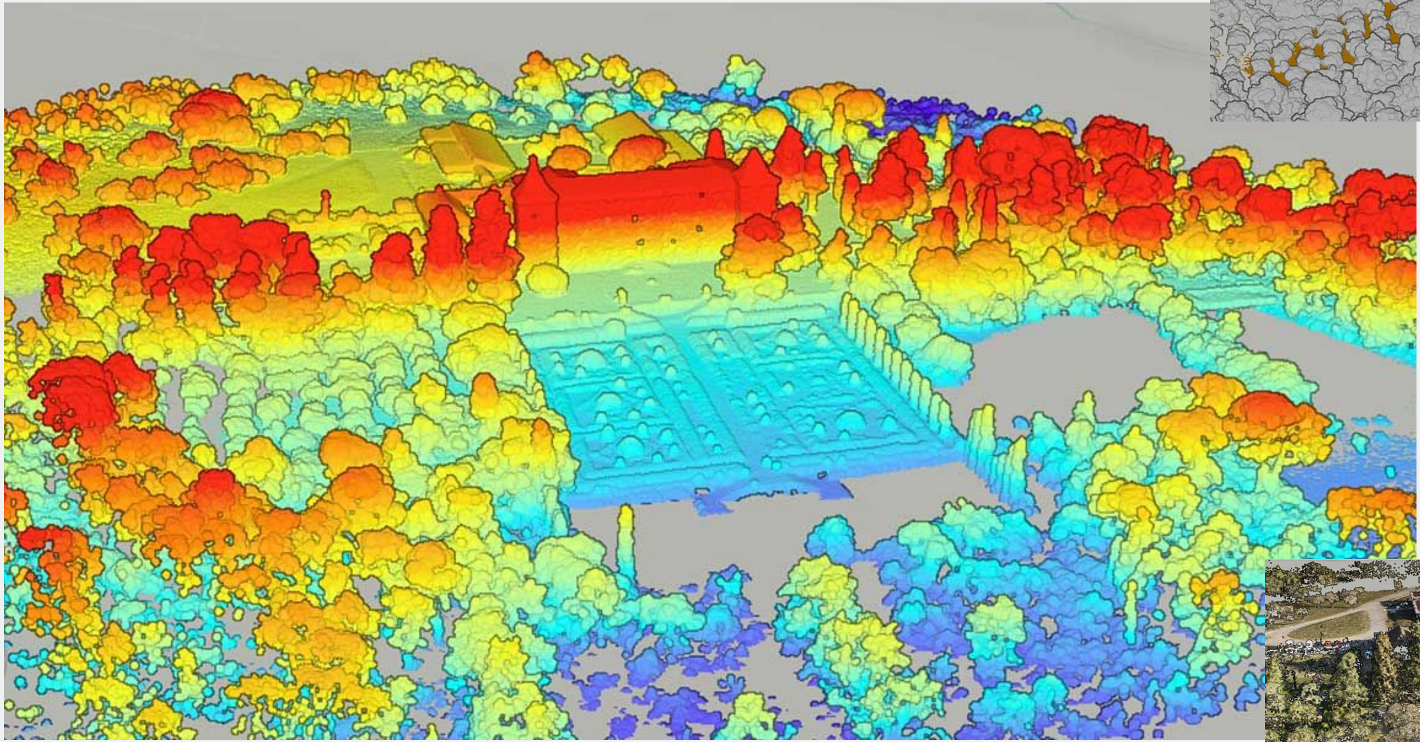
https://paperswithcode.com/dataset/tum-rgb-d

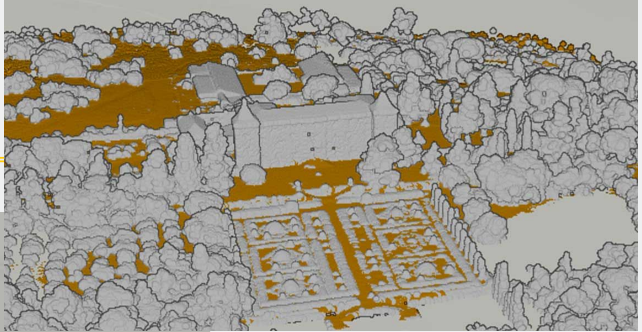
https://www.nist.gov/image/hyperspectralcube4501png

https://www.flir.com/oem/adas/adas-dataset-form/?srsltid=AfmBOoovTeVL1G2vPoiodz5G9ciTWsL_18DTOSWGiuV5qXcCWXr2g39S

RGB-T
(thermal)

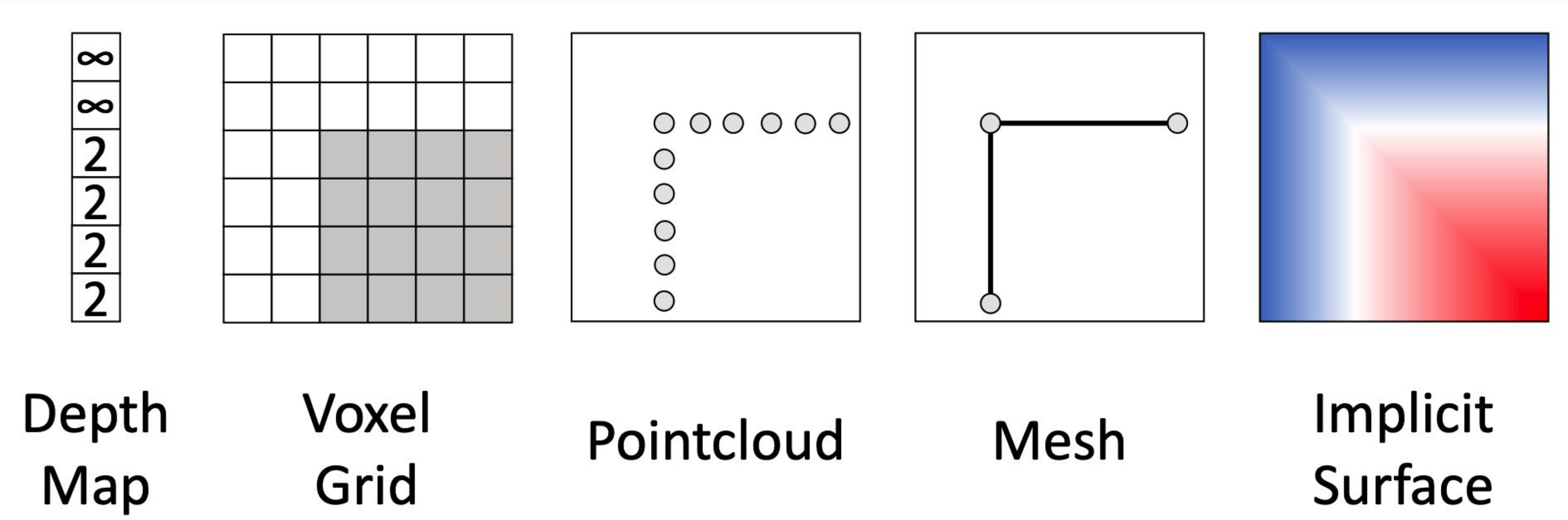
3D data modalities





https://www.yellowscan.com/knowledge/lidar-point-cloud-basics/

3D Shape Representations



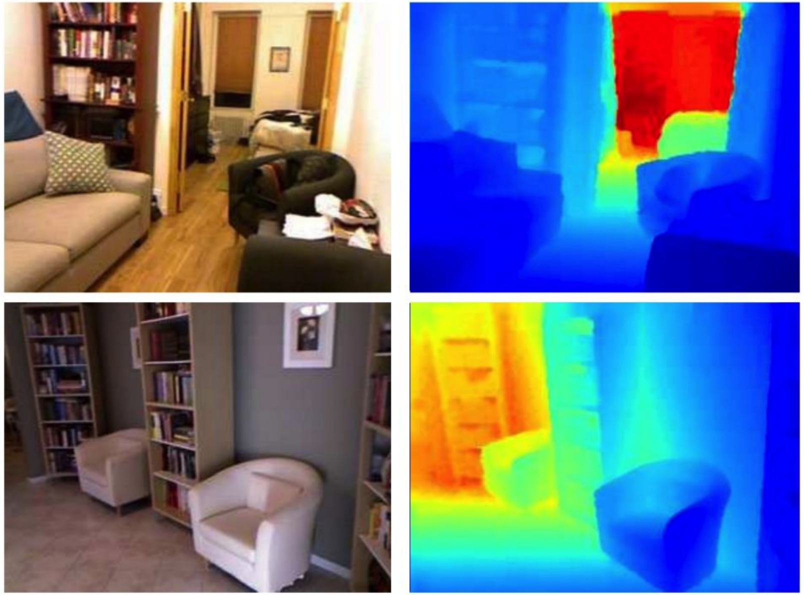
3D Shape Representations

Q: Why 2.5D?

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

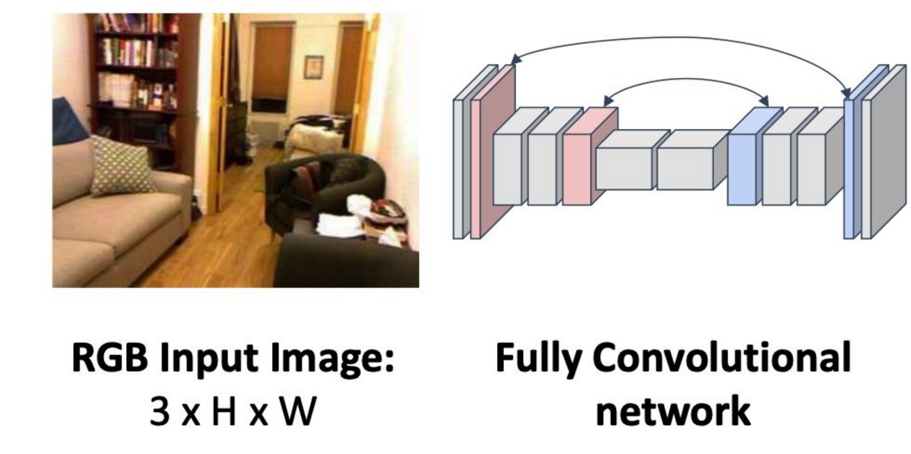
https://openaccess.thecvf.com/content_iccv_2015/papers/Eigen_Predicting_Depth_Surface_ICCV_2015_paper.pdf

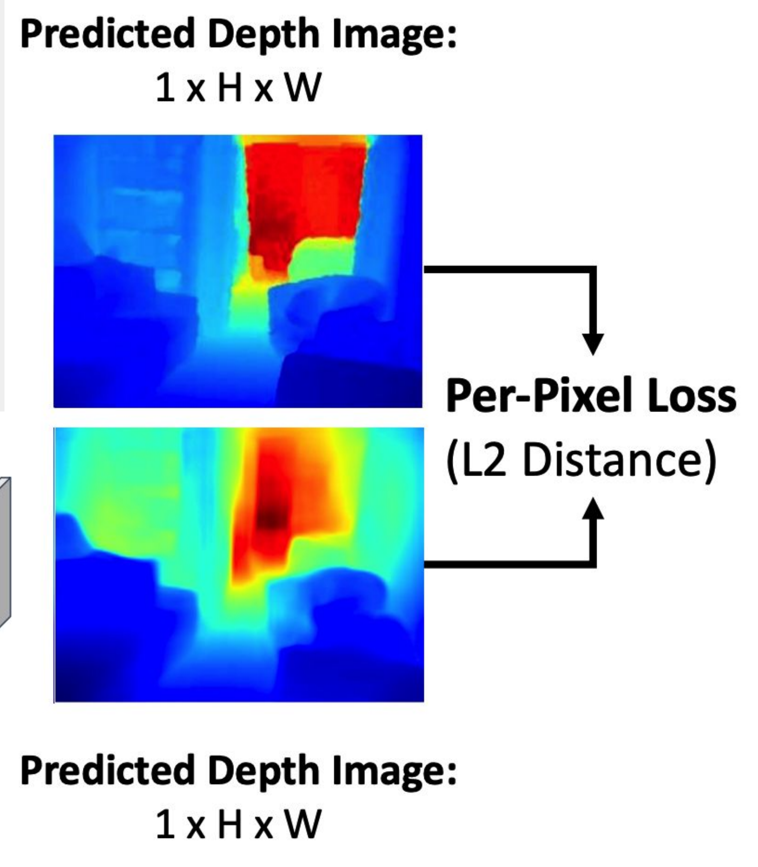
Predicting Depth Maps

https://papers.nips.cc/paper_files/paper/2014/hash/91c56ce4a249fae5419b90cba831e303-Abstract.html

https://www.cvlibs.net/datasets/kitti/

(KITTI dataset)

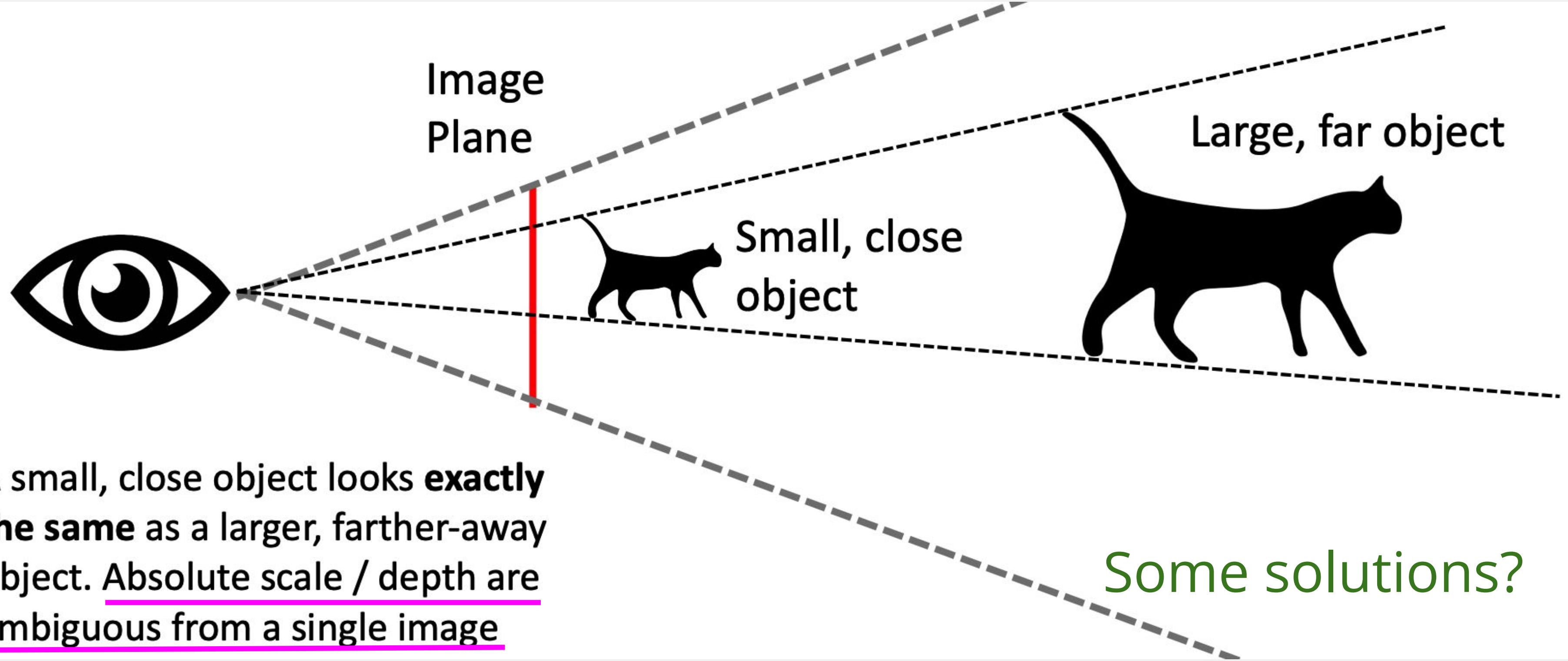




Aha Slides
(In-class participation)

https://ahaslides.com/YSRO4

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

Some solutions?

Predicting Depth Maps

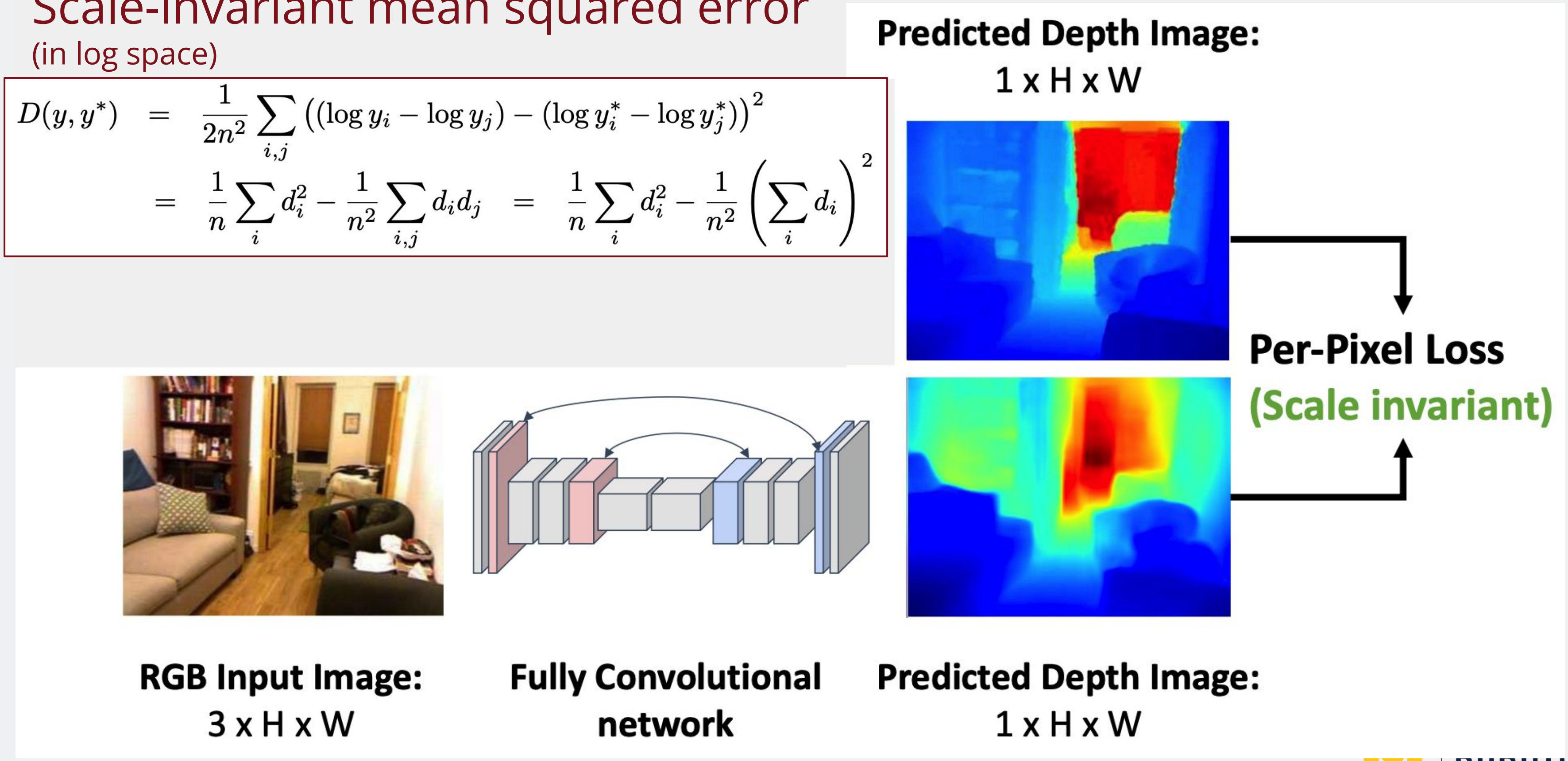
https://papers.nips.cc/paper_files/paper/2014/hash/91c56ce4a249fae5419b90cba831e303-Abstract.html

Scale-invariant mean squared error

(in log space)

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

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

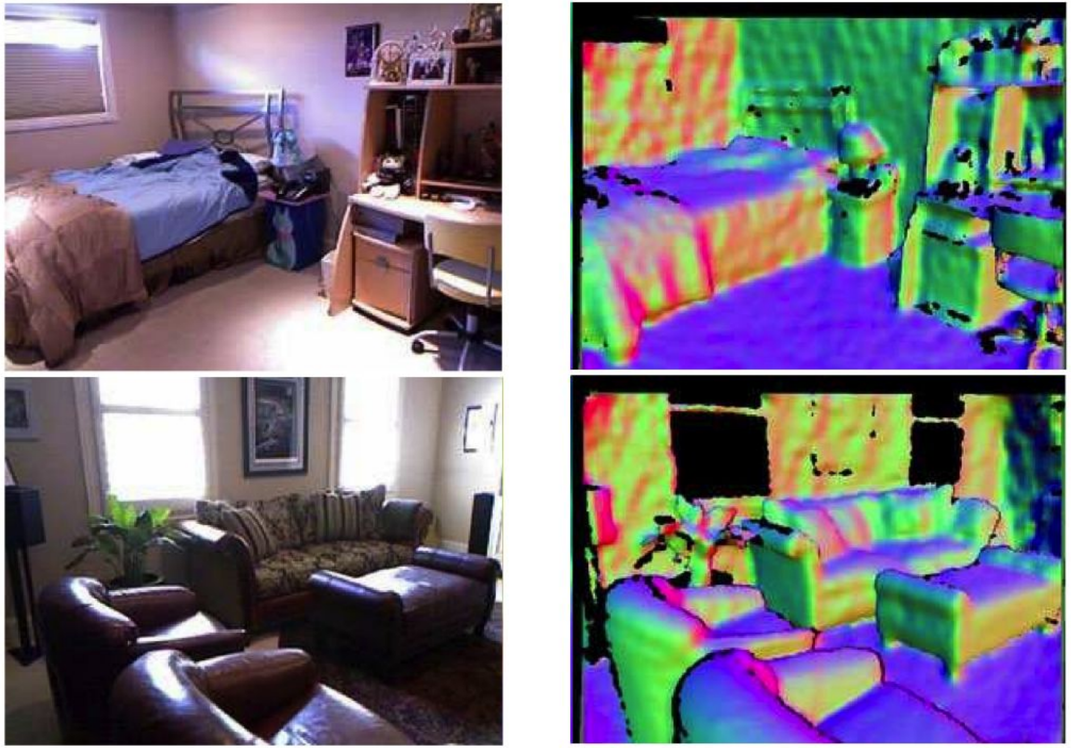


3D Shape Representations: Surface Normals

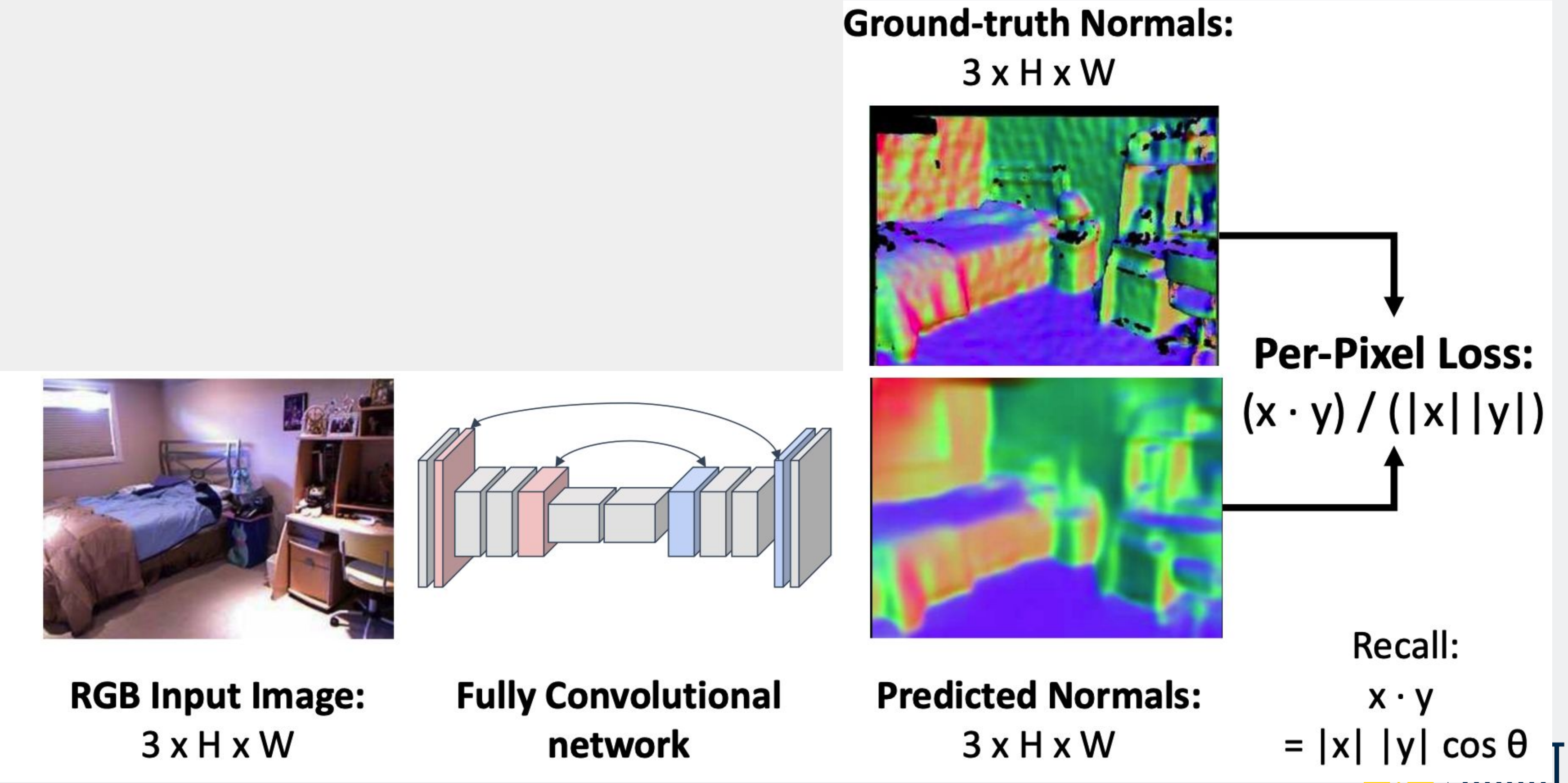
For each pixel, **surface normals** give a vector giving the normal vector to the object in the world for that pixel

Q: How to obtain surface normals?

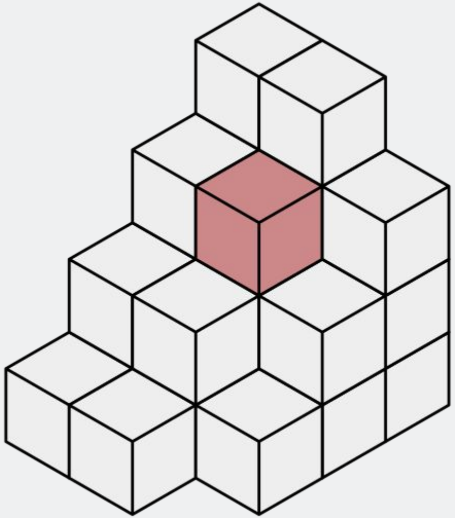
https://arxiv.org/pdf/1411.4734



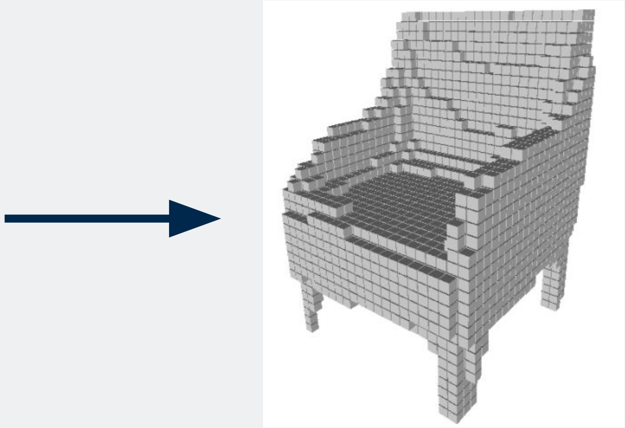
Predicting Surface Normals



3D Shape Representations: Voxels

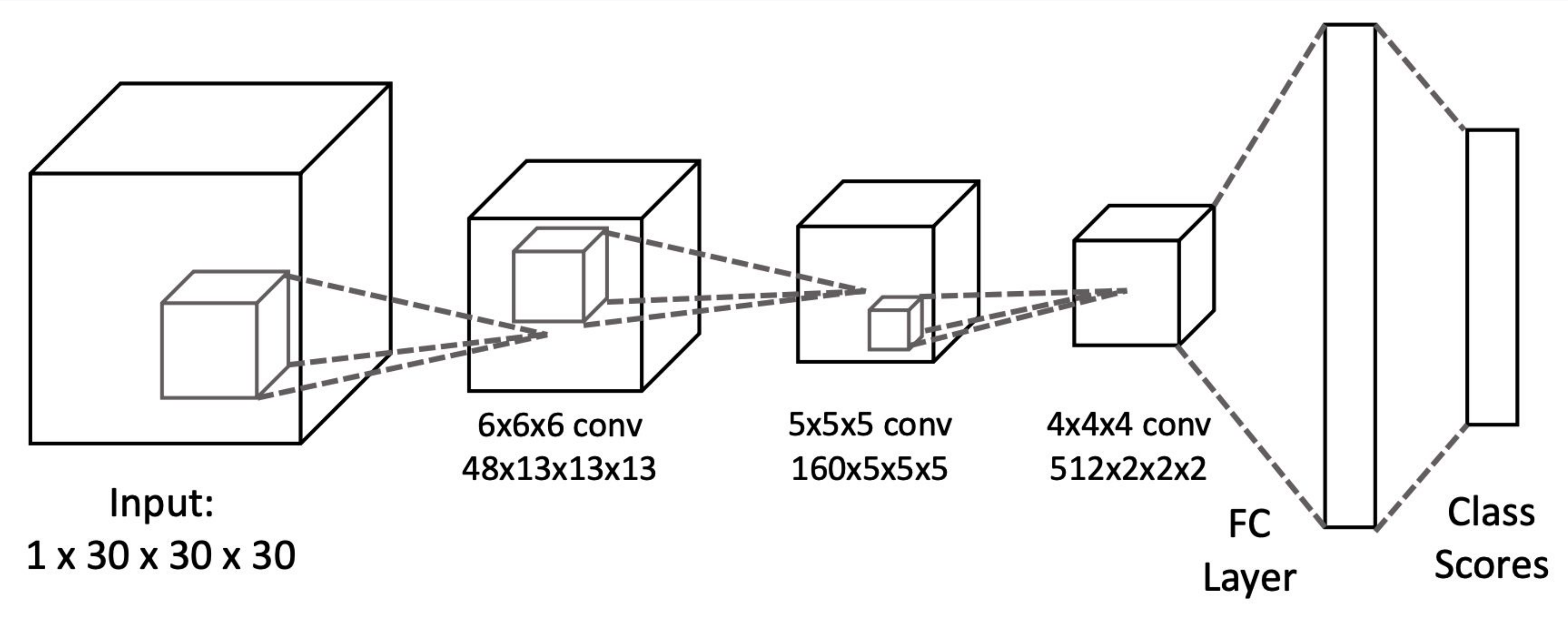


- Represent a shape with a V x V x V grid of occupancies
- Just like segmentation masks in Mask R-CNN, but in 3D!
- (+) Conceptually simple: just a 3D grid!
- (-) Need high spatial resolution to capture fine structures
- (-) Scaling to high resolutions is nontrivial!



https://arxiv.org/abs/1604.00449

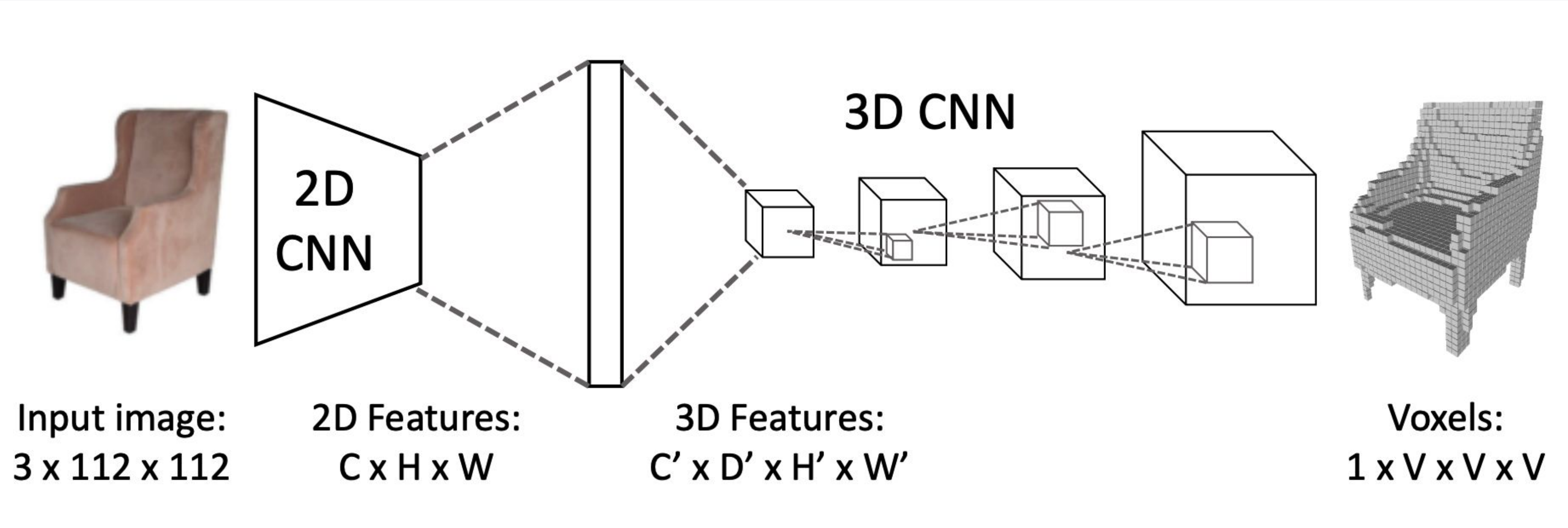
Processing Voxel Inputs: 3D Convolution



Train with classification loss

https://arxiv.org/pdf/1406.5670
https://3dshapenets.cs.princeton.edu/

Generating Voxel Shapes: 3D Convolution



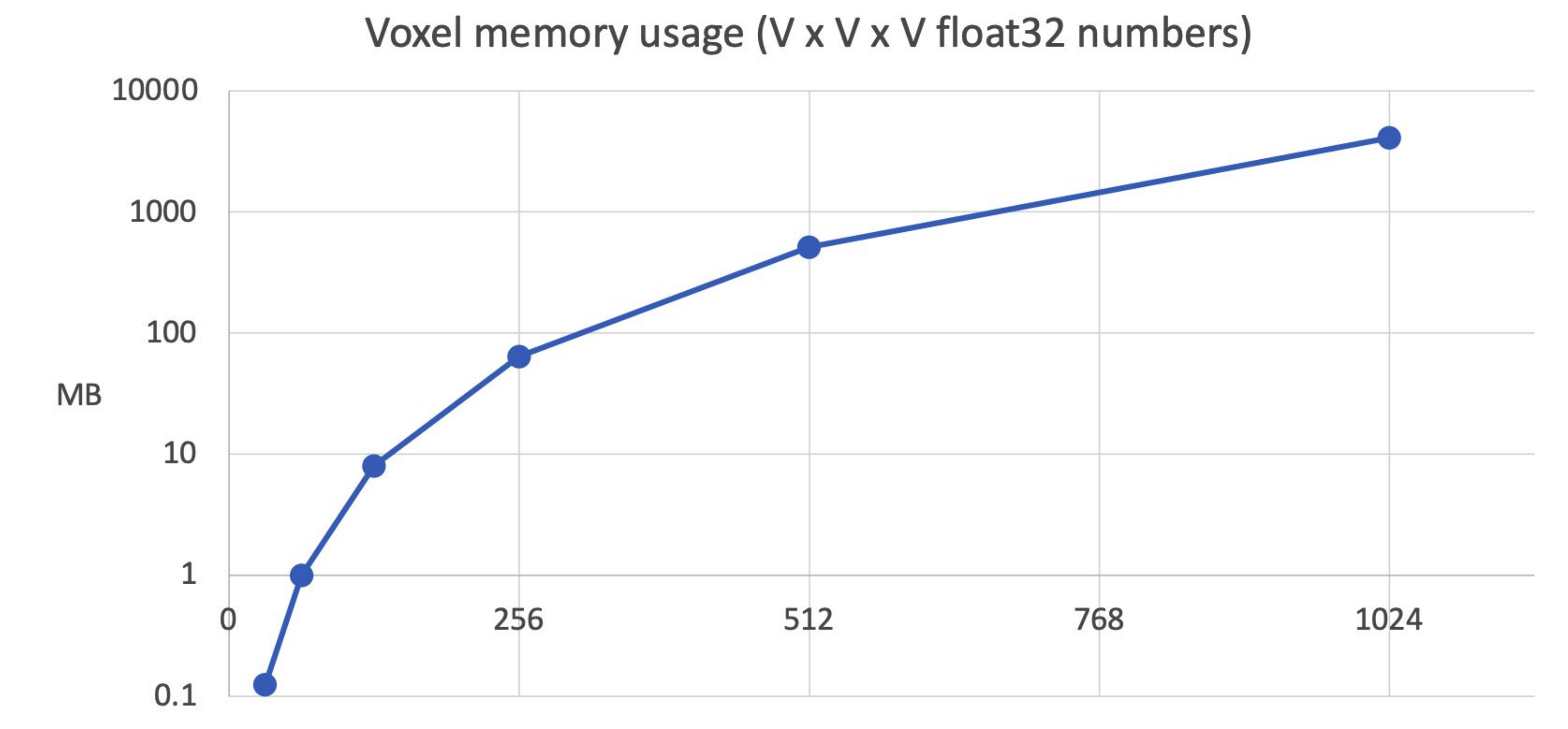
Train with per-voxel cross-entropy loss

https://arxiv.org/pdf/1604.00449

Q: voxels with occluded parts?

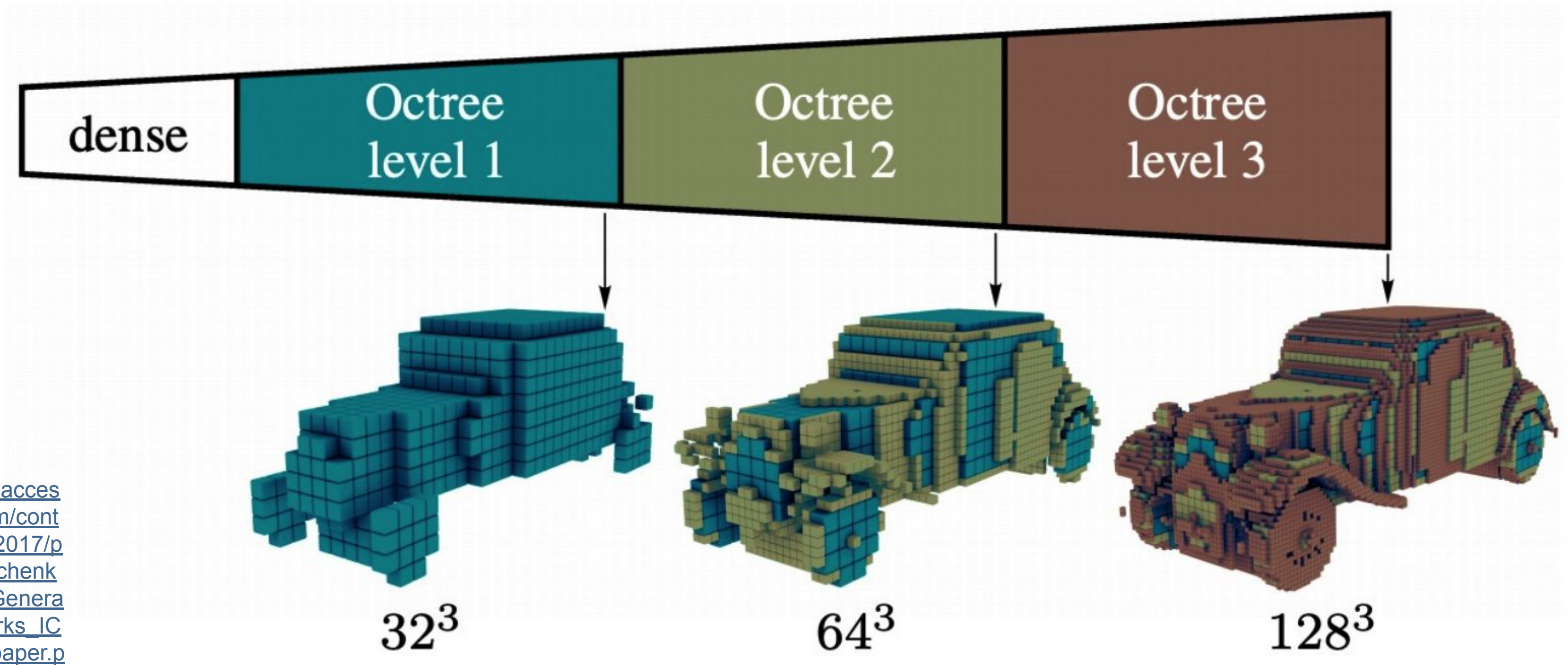
Voxel: Memory Usage

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



Scaling Voxels: Oct-Trees

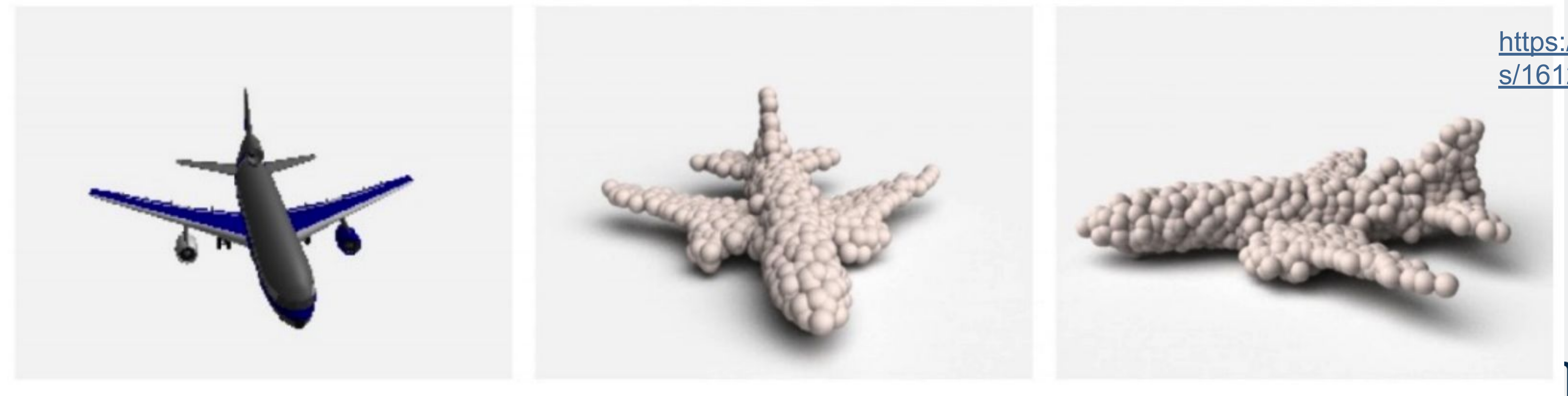
Use voxel grids with heterogenous resolution!



https://openaccess.thecvf.com/content_ICCV_2017/papers/Tatarchenko_Octree_Generating_Networks_ICCV_2017_paper.pdf

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



https://arxiv.org/abs/1612.00603

PointNet

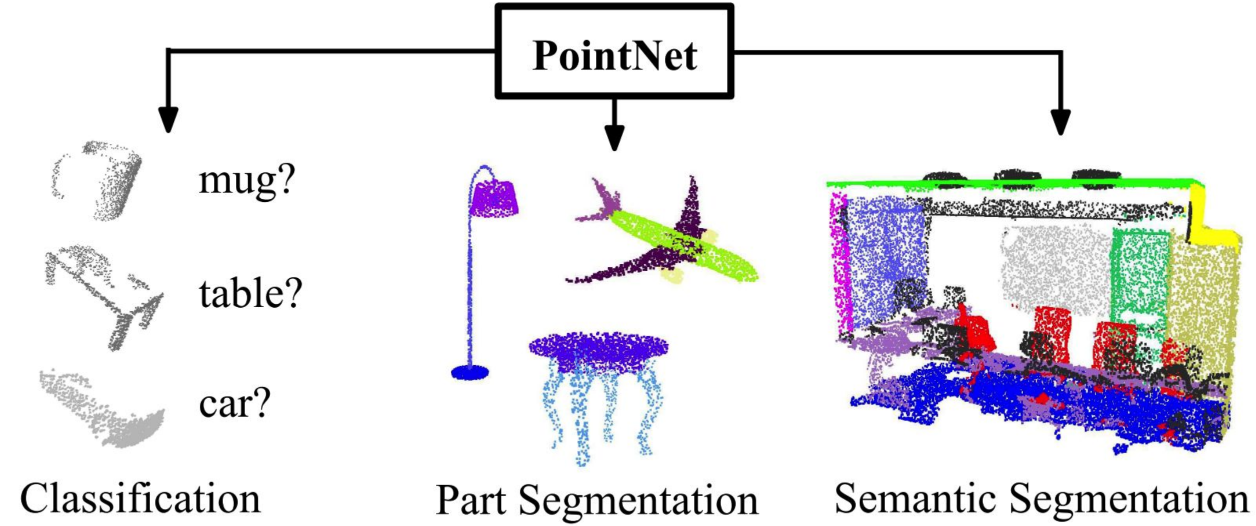
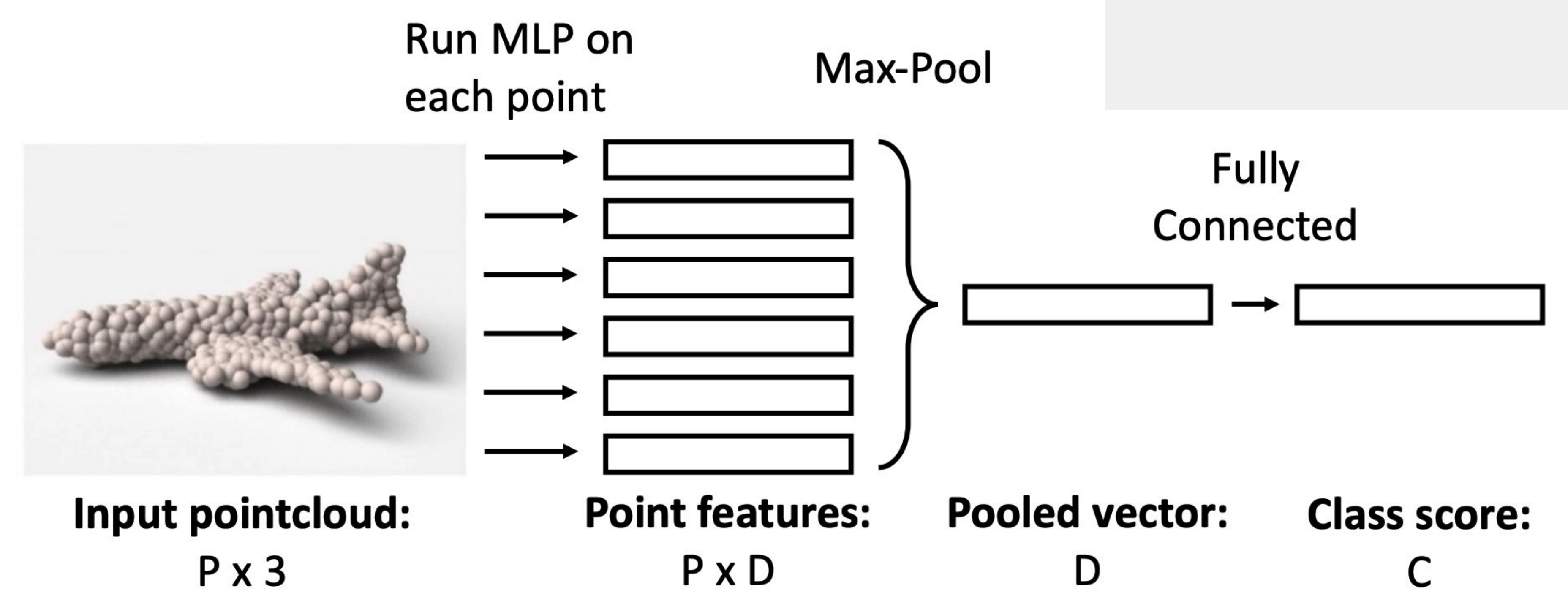


Figure 1. **Applications of PointNet.** We propose a novel deep net architecture that consumes raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks.

https://openaccess.thecvf.com/content_cvpr_2017/papers/Qi_PointNet_Deep_Learning_CVPR_2017_paper.pdf

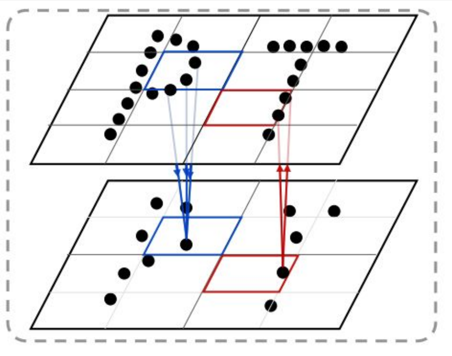
PointNet

Q: does order matter?



PointNet and more

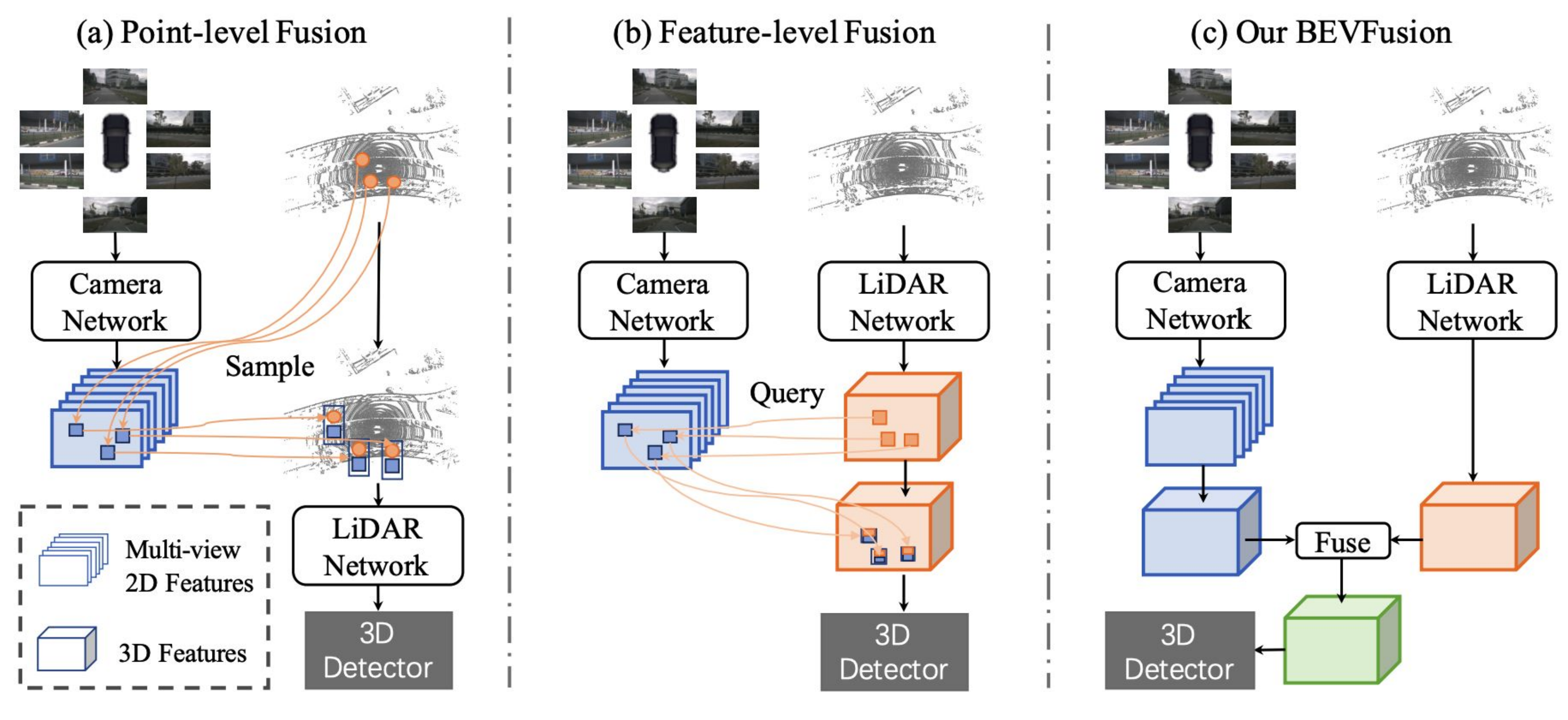
PointNet (3D point cloud processing, no voxel or rendering):
https://openaccess.thecvf.com/content_cvpr_2017/papers/Qi_PointNet_Deep_Learning_CVPR_2017_paper.pdf
PointNet++ (adaptively combine features from multiple scales):
https://arxiv.org/pdf/1706.02413
PointNeXt (Improved Training and Scaling Strategies):
https://proceedings.neurips.cc/paper_files/paper/2022/file/9318763d049edf9a1f2779b2a59911d3-Paper-Conference.pdf
Point Transformer (v2) (partition-based group vector attention):
https://proceedings.neurips.cc/paper_files/paper/2022/file/d78ece6613953f46501b958b7bb4582f-Paper-Conference.pdf



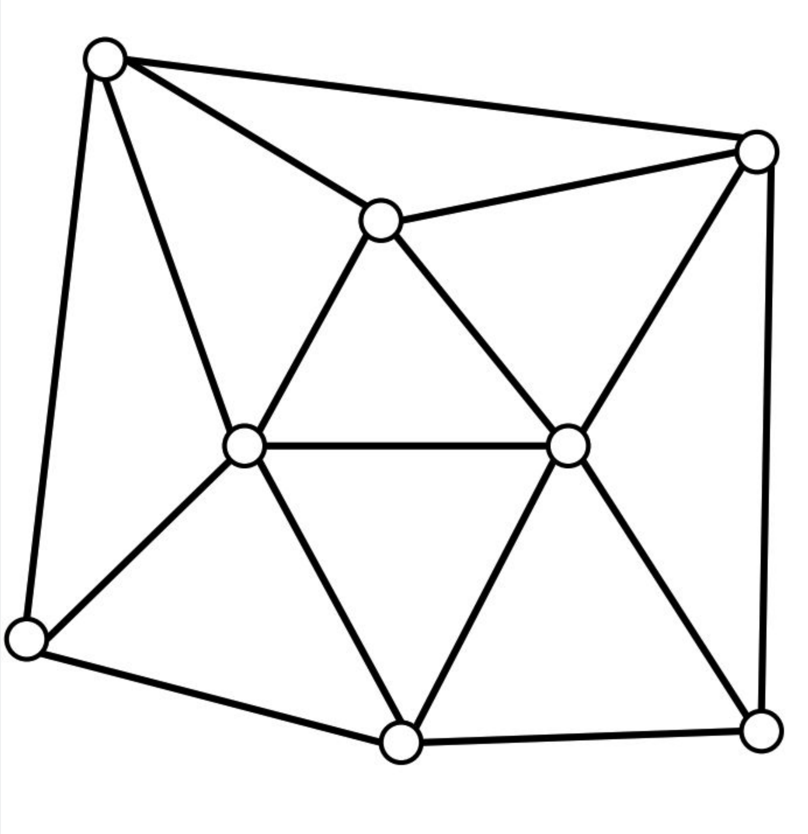
PointNet and more

BEVFusion
https://proceedings.neurips.cc/paper_files/paper/2022/file/43d2b7fbee8431f7cef0d0afed51c691-Paper-Conference.pdf

Applications (e.g., LiDAR+Camera Fusion)



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

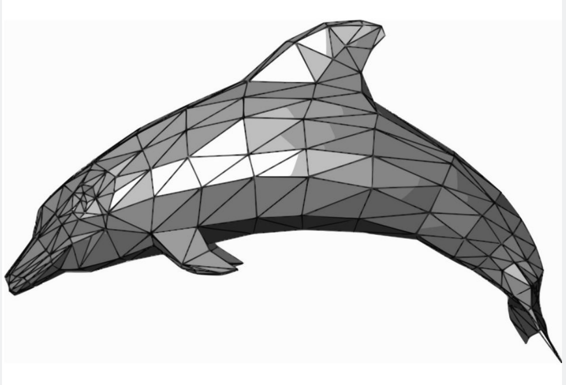
(+) Adaptive: Can represent flat surfaces very efficiently, can allocate more faces to areas with fine detail

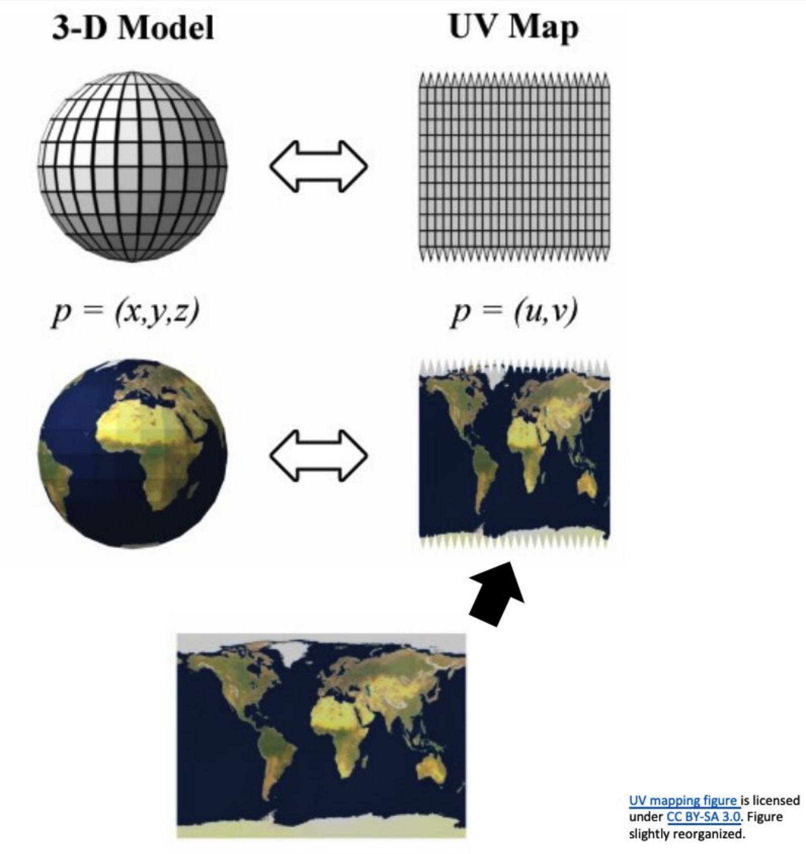
(+) Can attach data on verts and interpolate over the whole surface: RGB colors, texture coordinates, normal vectors, etc.

Examples

https://graphics.stanford.edu/data/3Dscanrep/

"Stanford bunny"





SMPL

https://smpl.is.tue.mpg.de/

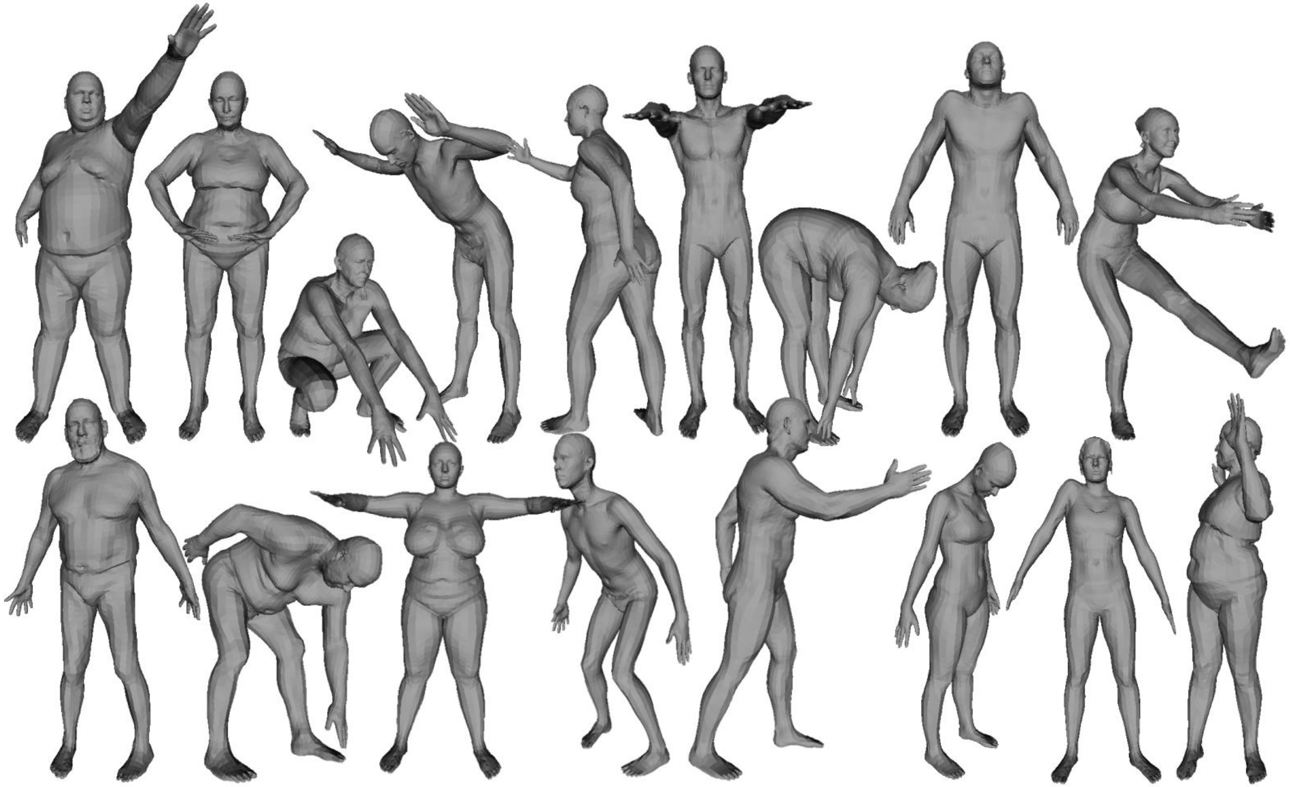
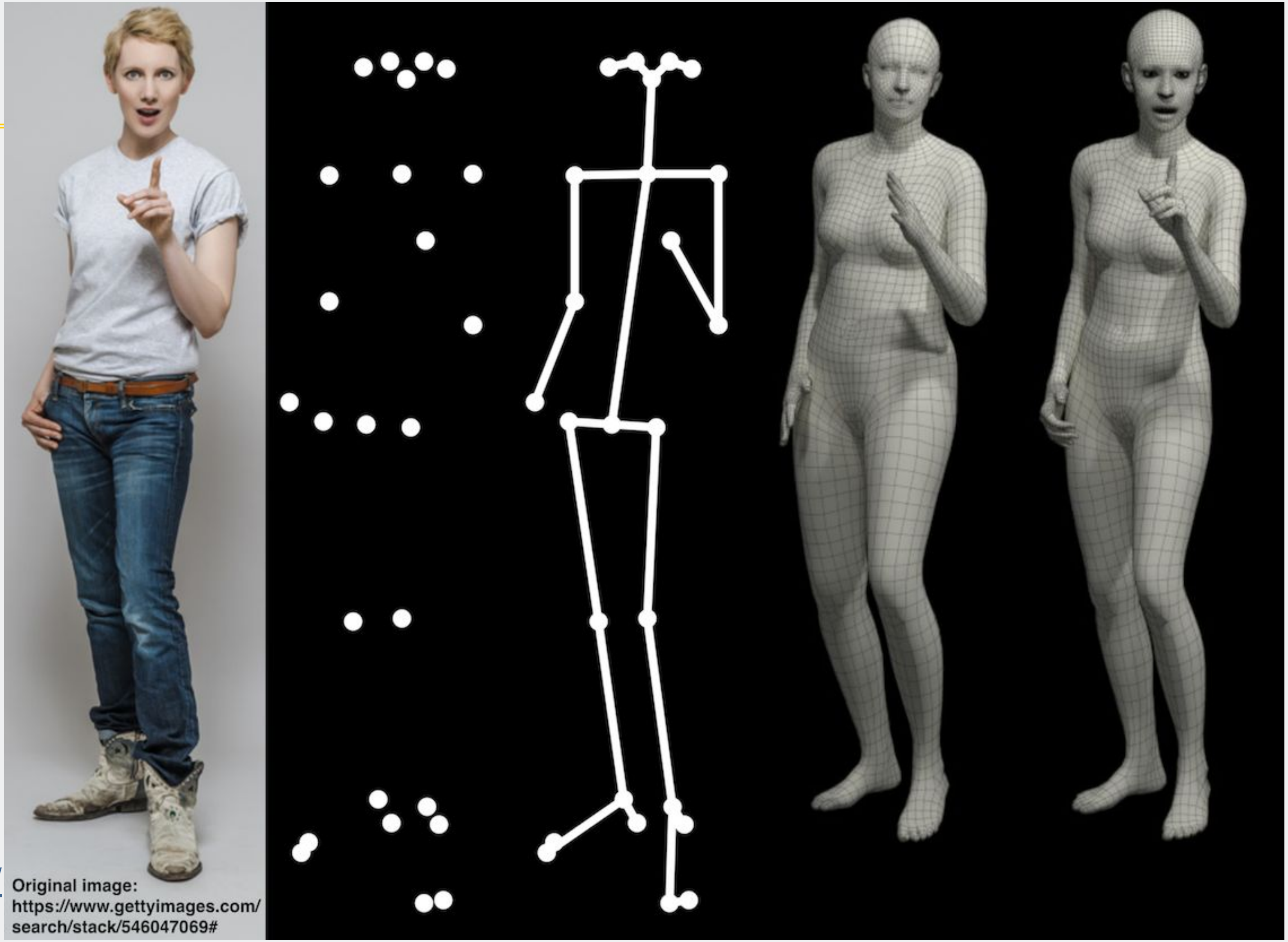


Figure 4: *Sample registrations from the multipose dataset.*

SMPL-X



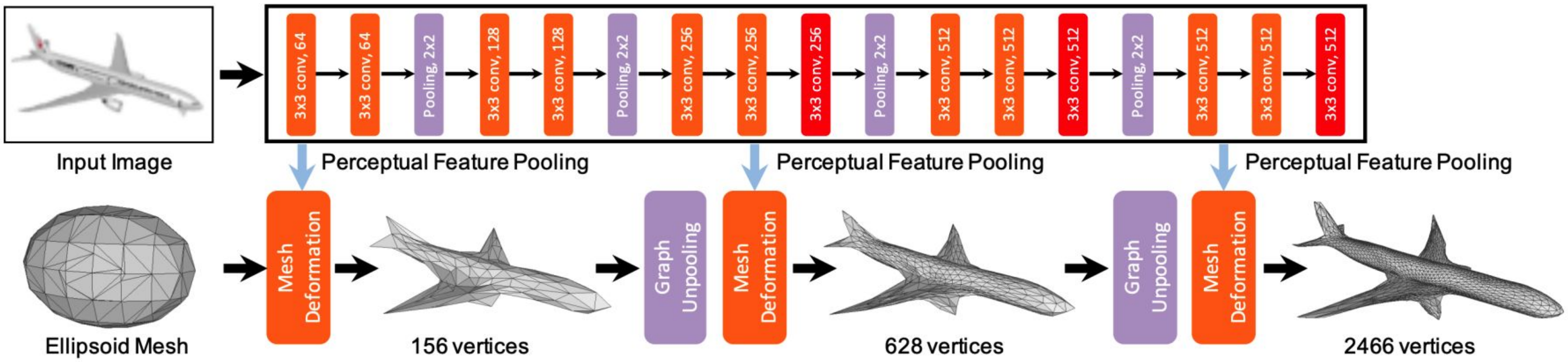
https://smpl-x.is.tue.mpg.de/

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



https://arxiv.org/pdf/1804.01654

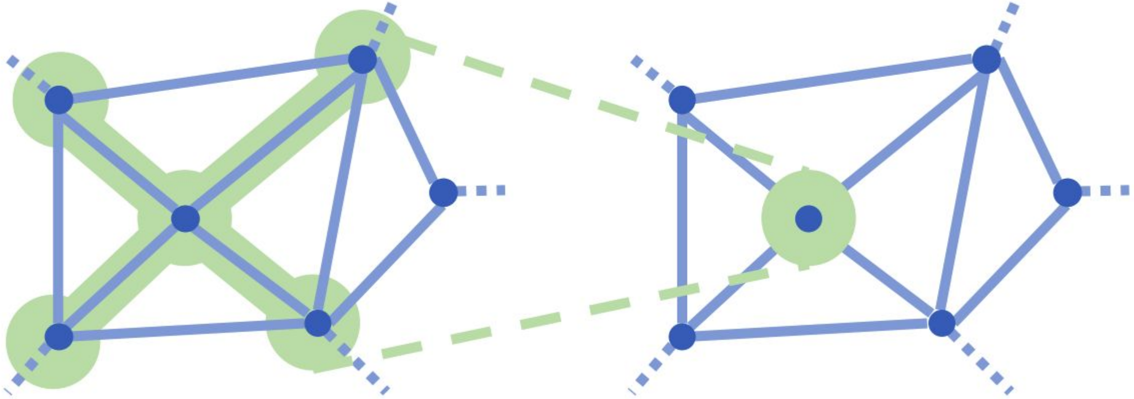
Graph Convolution

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

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

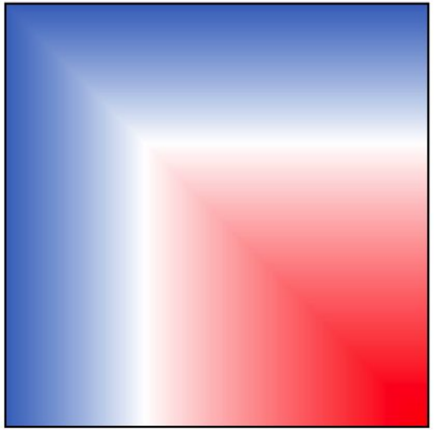
Output: New feature vector for each vertex

3D Shape Representations: Implicit Functions

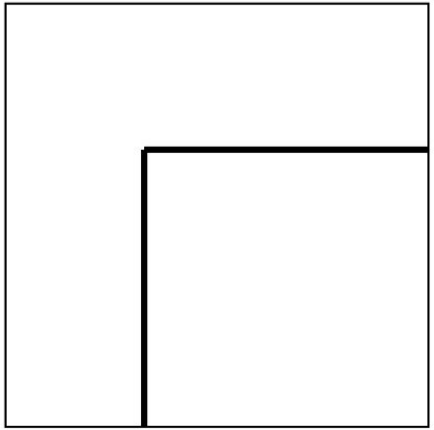
Learn a function to classify arbitrary 3D points as inside / outside the shape

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

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



Implicit function



Explicit Shape

Binary classification problem

- Occupancy
- SDF (signed distance function)

ROBOTICS

Neural Radiance Field (NeRF) For View Synthesis

https://arxiv.org/pdf/2003.08934

View Synthesis

https://www.matthewtancik.com/nerf

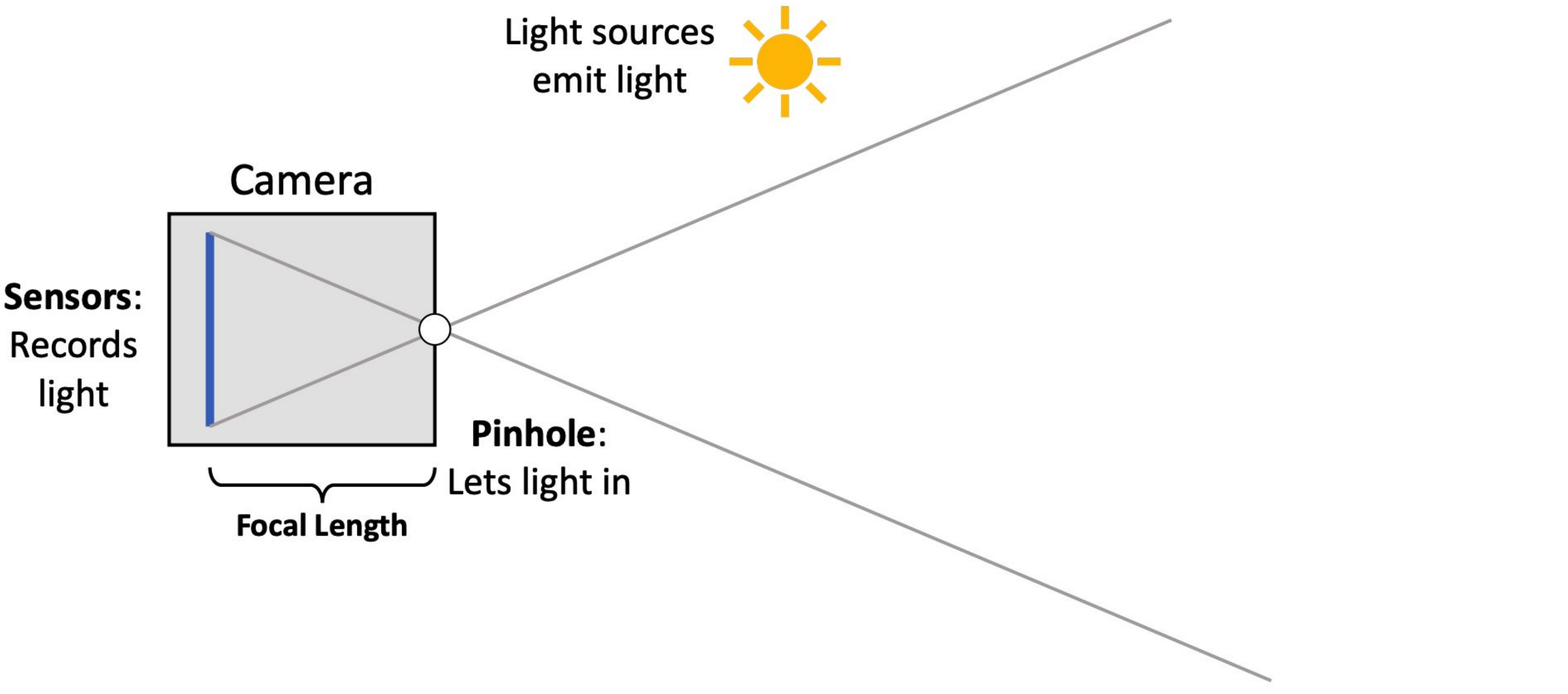
Input: Many images of the same scene (with known camera parameters)

Output: Images showing the scene from novel viewpoints

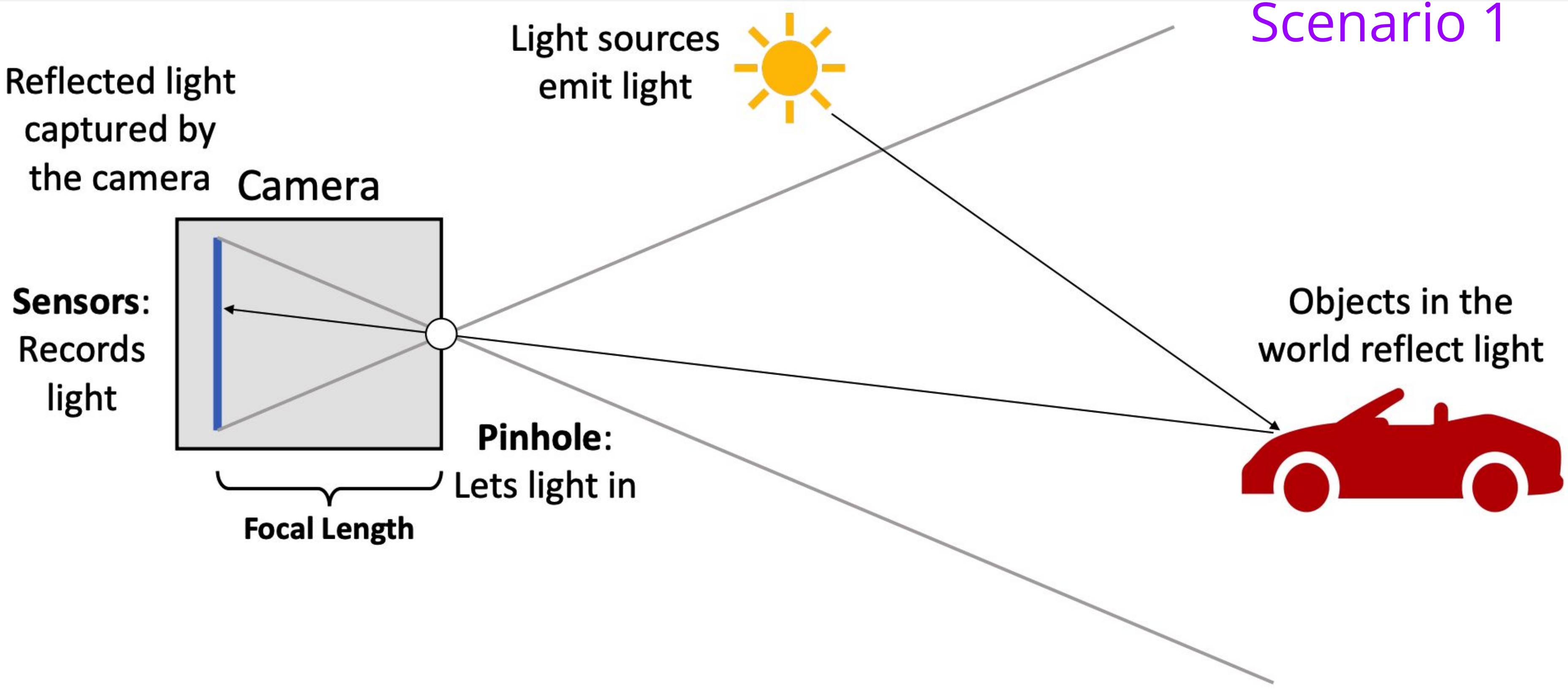
Scene rendering

ROBOTICS

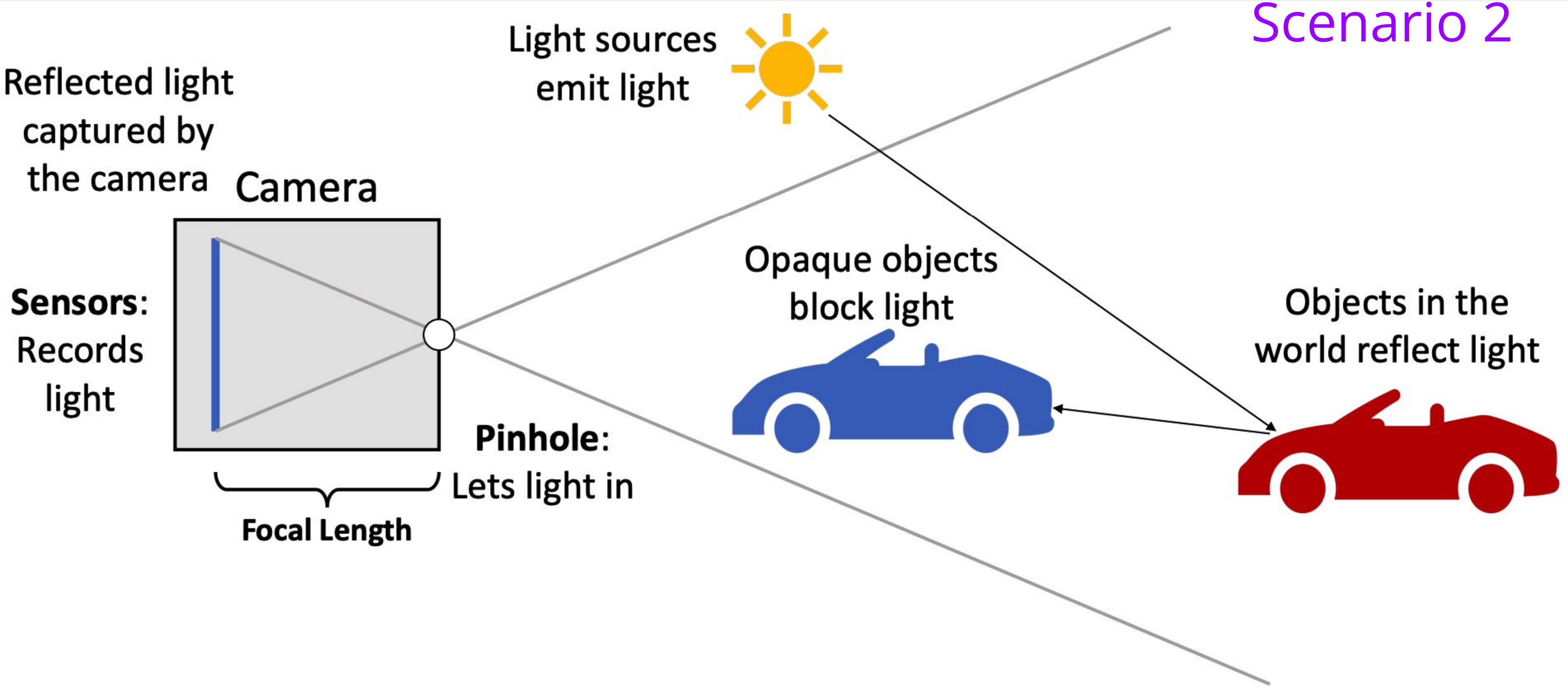
Stepping Back: Pinhole Camera Model



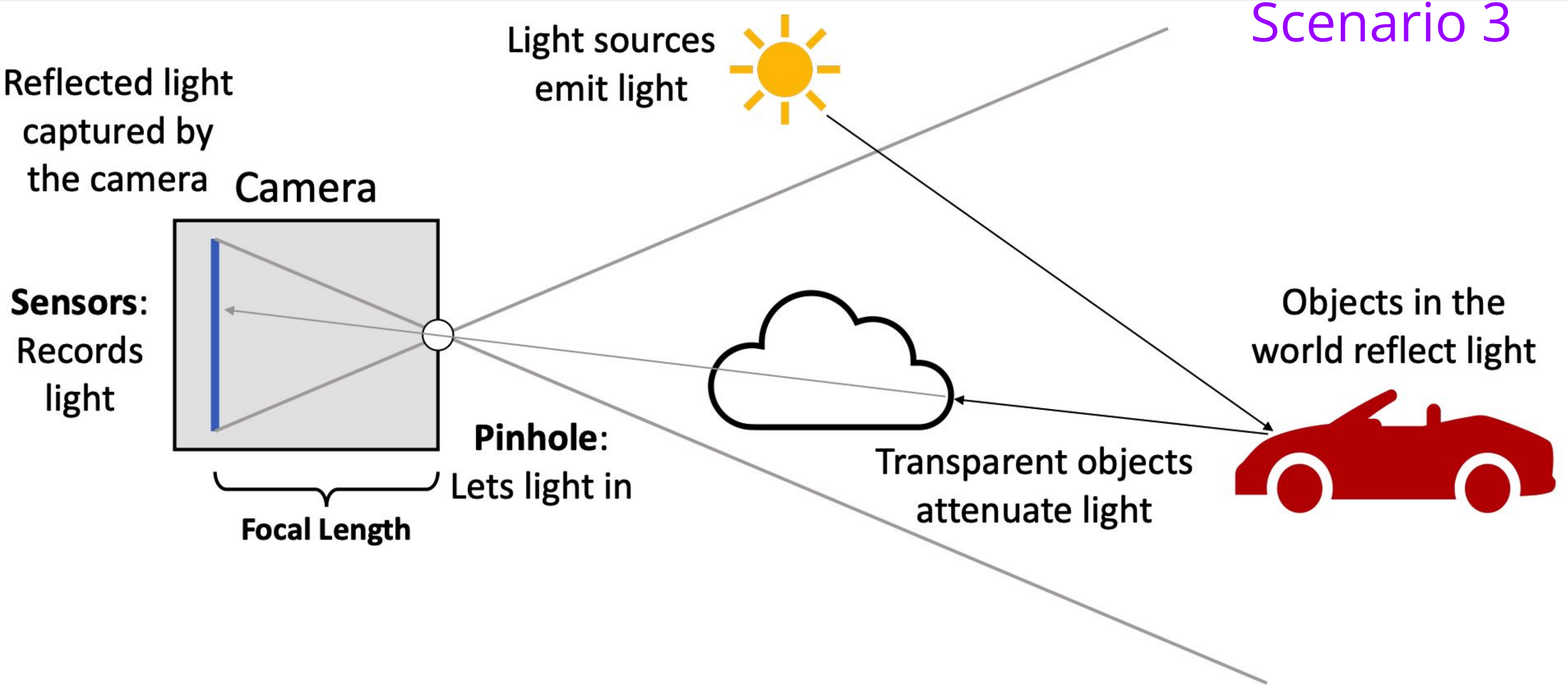
Stepping Back: Pinhole Camera Model



Stepping Back: Pinhole Camera Model



Stepping Back: Pinhole Camera Model



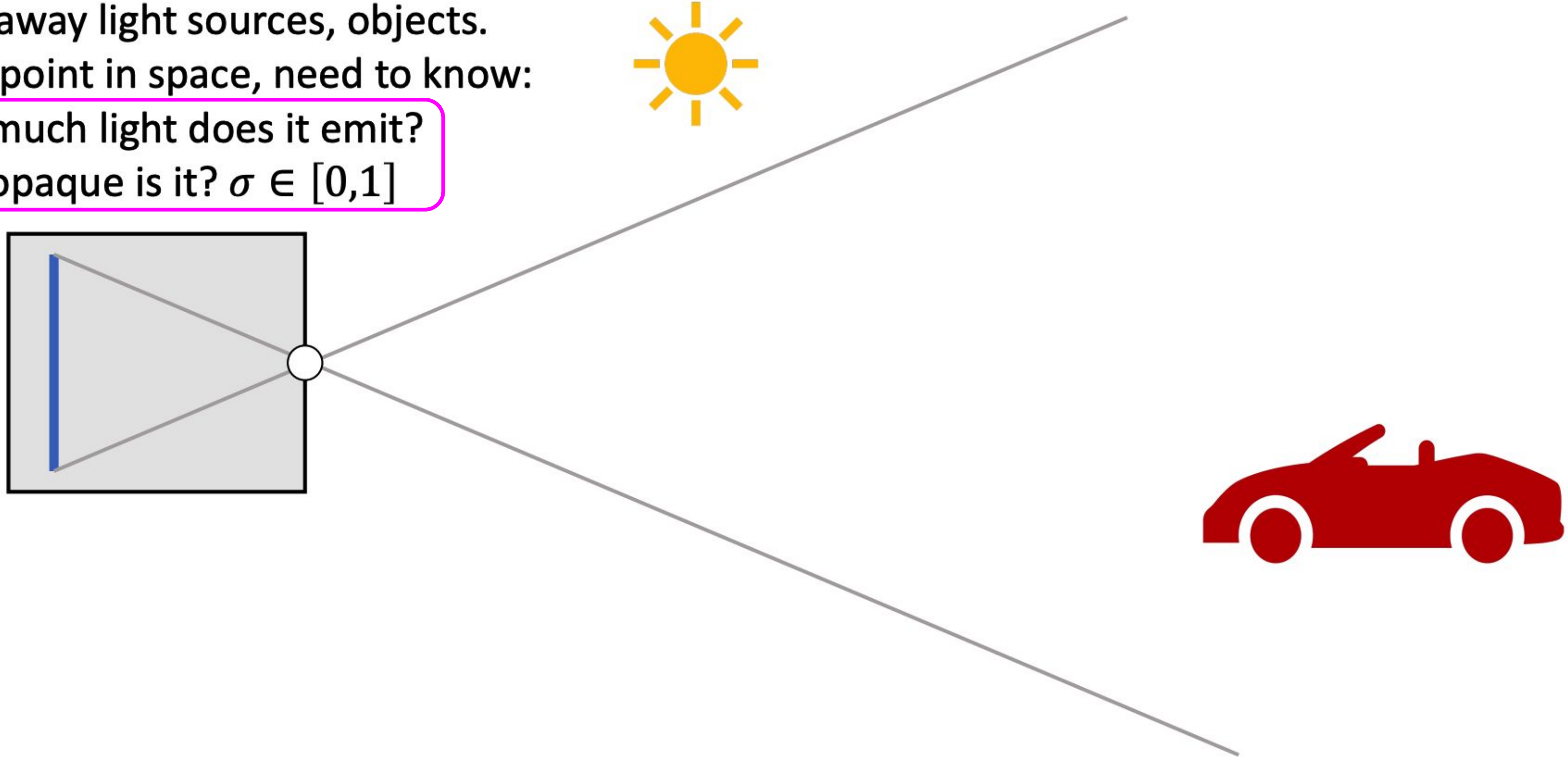
Volume Rendering

Abstract away light sources, objects.

For each point in space, need to know:

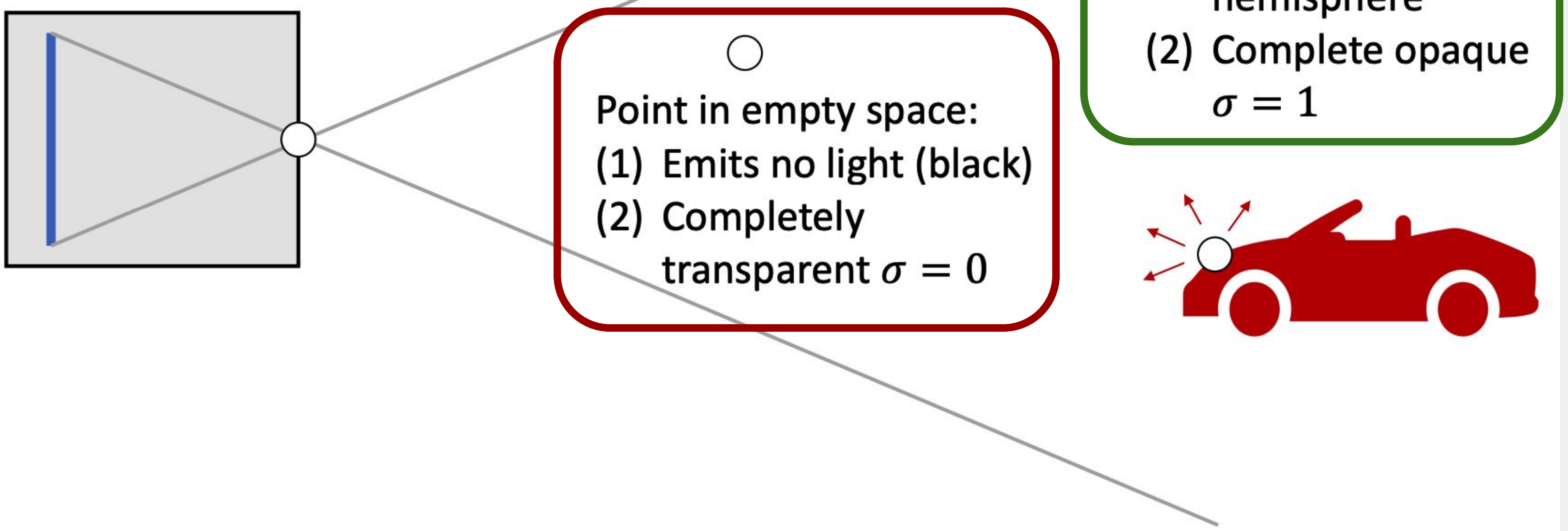
(1) How much light does it emit?

(2) How opaque is it? $\sigma \in [0,1]$

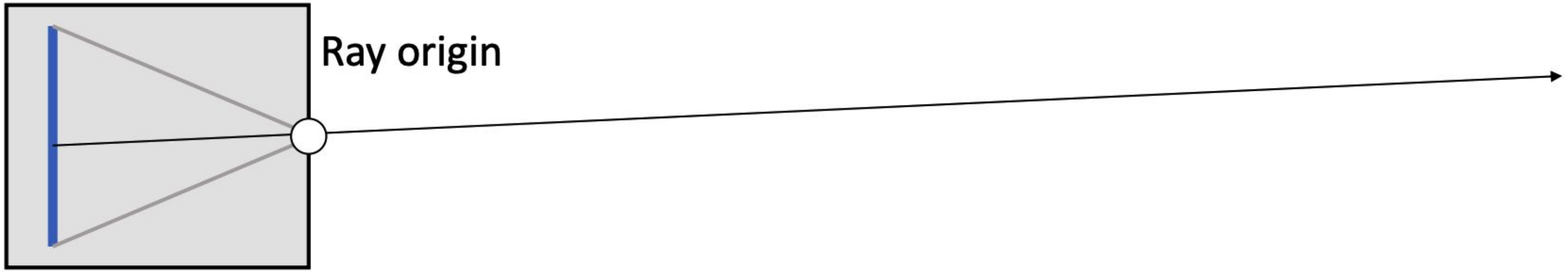


Stepping Back: Pinhole Camera Model

Abstract away light sources, objects.
For each point in space, need to know:
(1) How much light does it emit?
(2) How opaque is it? $\sigma \in [0,1]$



Volume Rendering



Parameterize each ray as origin plus direction: $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$

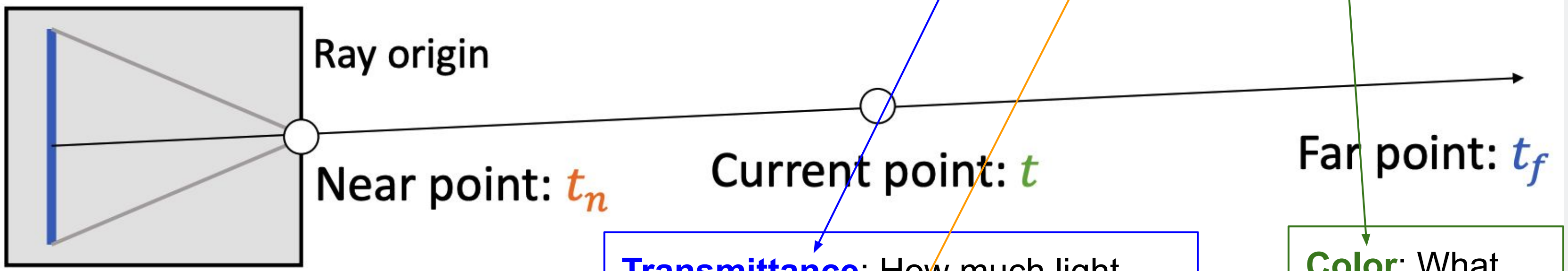
Volume Density is $\sigma(\boldsymbol{p}) \in [0,1]$

Color that a point **p** emits in direction **d** is $c(\boldsymbol{p}, \boldsymbol{d}) \in [0,1]^3$

Volume Rendering

Color observed by the camera given by **volume rendering equation:**

$$C(\boldsymbol{r}) = \int_{t_n}^{t_f} \underline{T(t)}\,\underline{\sigma(\boldsymbol{r}(t))}\,\underline{\boldsymbol{c}(\boldsymbol{r}(t), \boldsymbol{d})}\,dt$$



Parameterize each ray as origin plus direction: $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$

Volume Density is $\sigma(\boldsymbol{p}) \in [0,1]$

Color that a point **p** emits in direction **d** is $c(\boldsymbol{p}, \boldsymbol{d}) \in [0,1]^3$

Transmittance: How much light from the current point will reach the camera?

Opacity: How opaque is the current point?

Color: What color does the current point emit along the direction toward the camera?

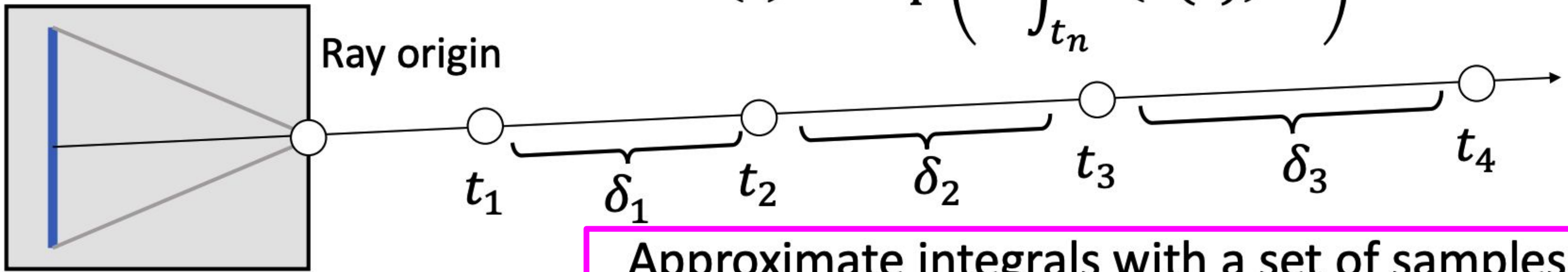
Transmittance

Compute transmittance by **accumulating volume density** up to current point

Color observed by the camera given by **volume rendering equation**:

$$C(\boldsymbol{r}) = \int_{t_n}^{t_f} T(t)\sigma(\boldsymbol{r}(t))\boldsymbol{c}(\boldsymbol{r}(t),\boldsymbol{d})dt$$

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\boldsymbol{r}(s))ds\right)$$



Parameterize each ray as origin plus direction: $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$

Volume Density is $\sigma(\boldsymbol{p}) \in [0,1]$

Color that a point **p** emits in direction **d** is $c(\boldsymbol{p},\boldsymbol{d}) \in [0,1]^3$

Approximate integrals with a set of samples:

$$C(\boldsymbol{r}) \approx \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\,\boldsymbol{c}_i$$

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right)$$

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

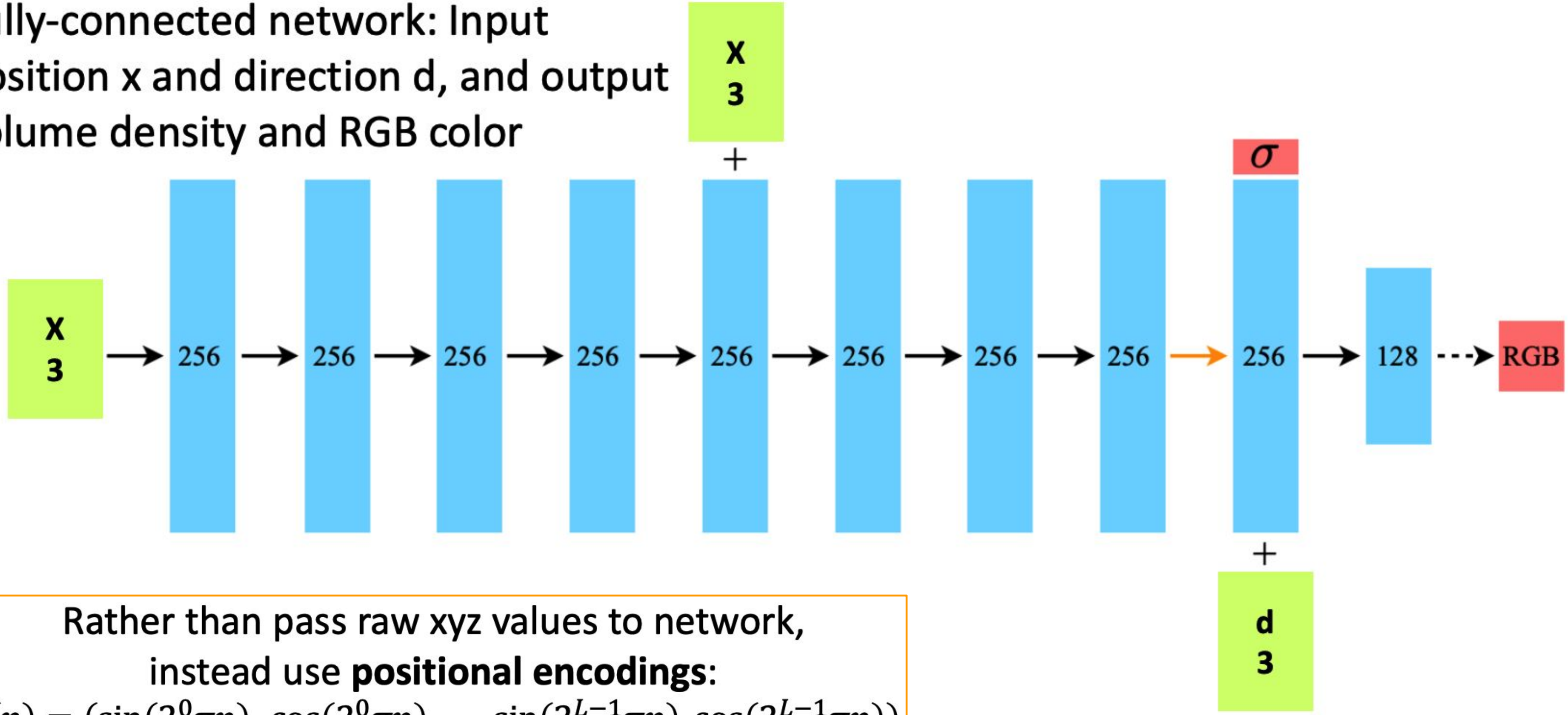
NeRF

Train a neural network to input position **p** and direction **d**, output $\sigma(\boldsymbol{p})$ and $c(\boldsymbol{p}, \boldsymbol{d})$

Training loss: Estimated pixel colors $C(\boldsymbol{r})$ should match actual pixel colors from images

NeRF: Network Architecture

Fully-connected network: Input position x and direction d, and output volume density and RGB color



Rather than pass raw xyz values to network, instead use **positional encodings**:

$$\gamma(p) = (\sin(2^0\pi p), \cos(2^0\pi p), \ldots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p))$$

NeRF: Network Architecture

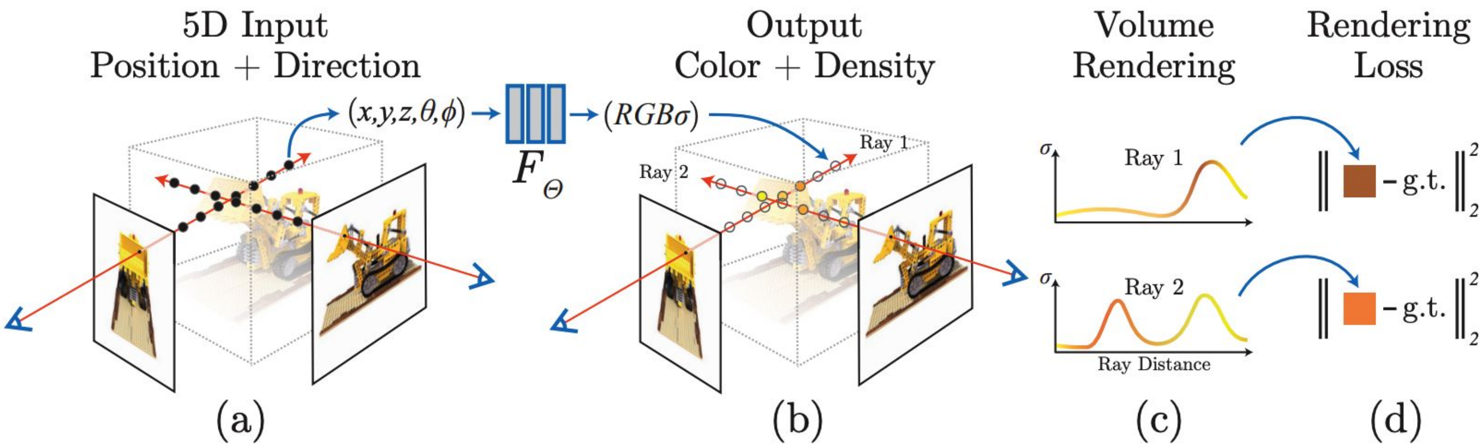
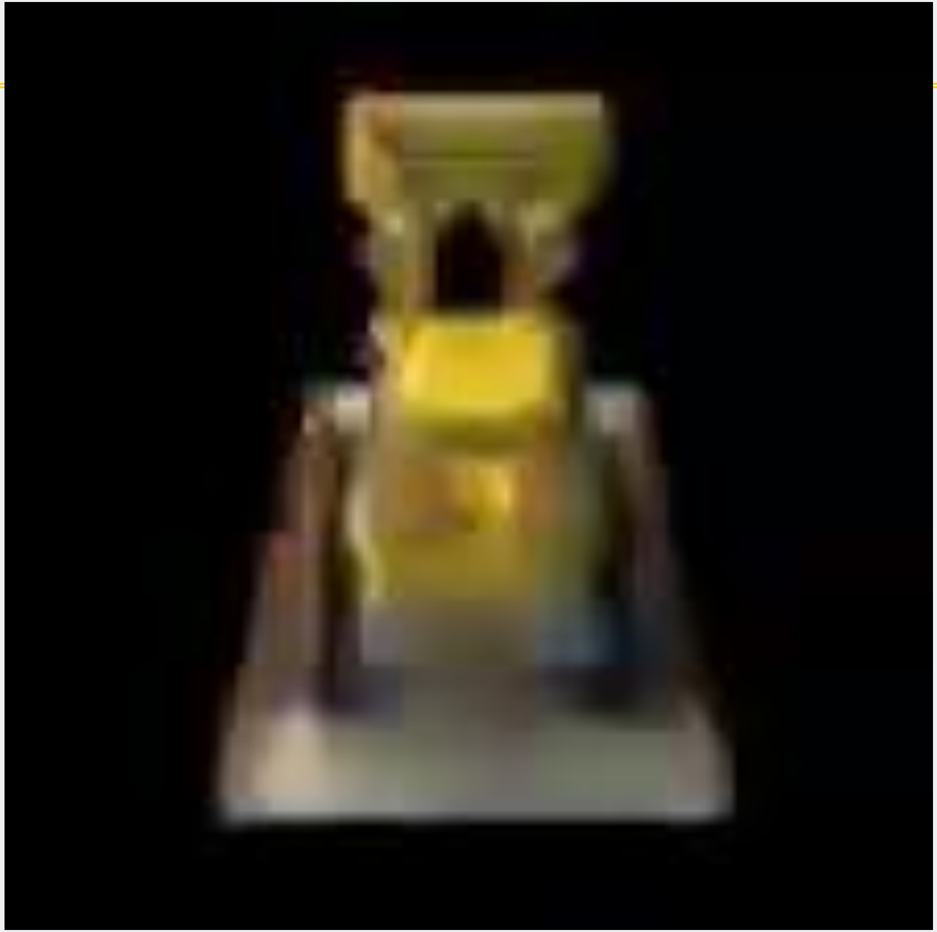


Fig. 2: An overview of our neural radiance field scene representation and differentiable rendering procedure. We synthesize images by sampling 5D coordinates (location and viewing direction) along camera rays (a), feeding those locations into an MLP to produce a color and volume density (b), and using volume rendering techniques to composite these values into an image (c). This rendering function is differentiable, so we can optimize our scene representation by minimizing the residual between synthesized and ground truth observed images (d).

NeRF



NeRF (2020)

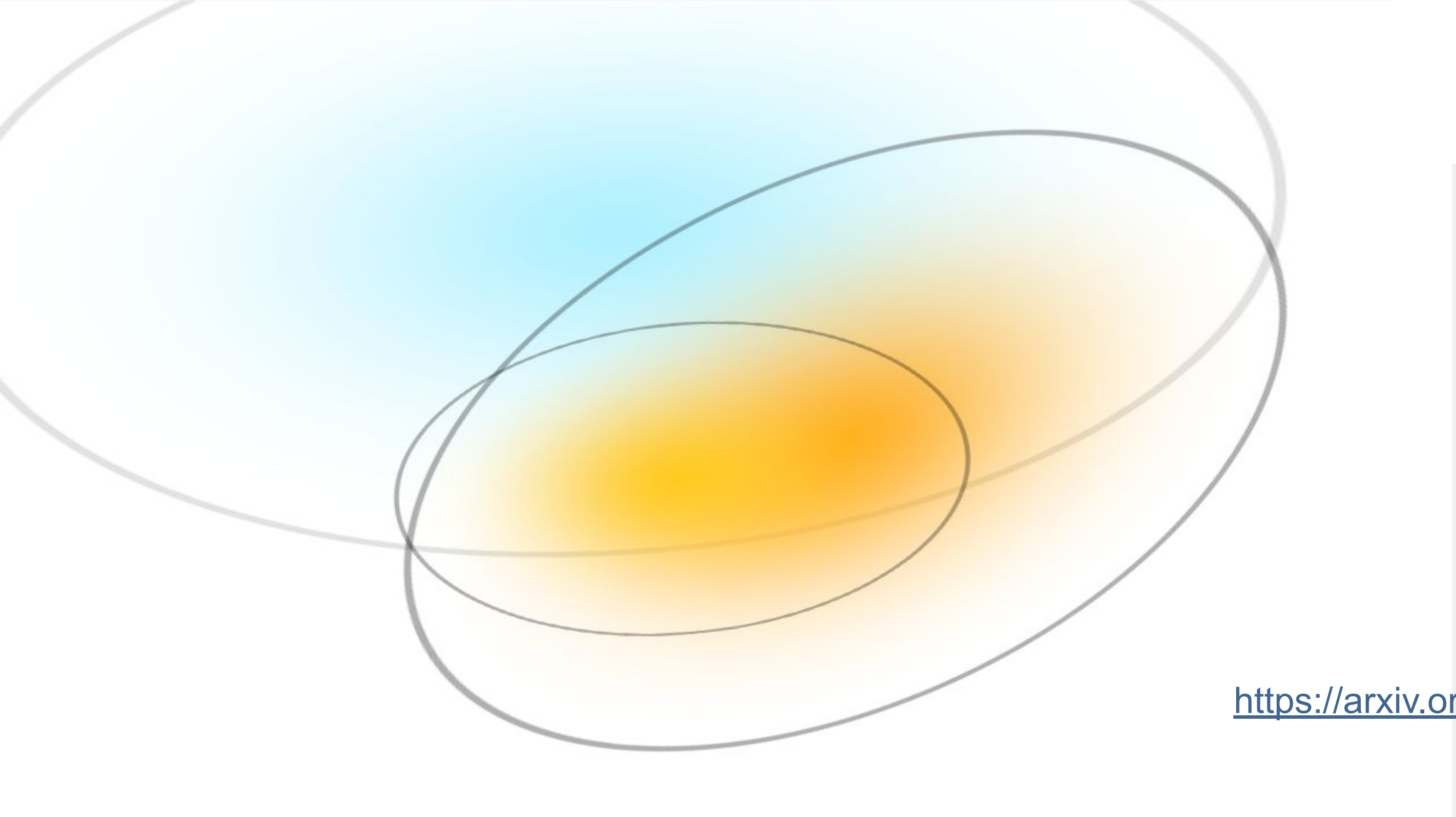
Main Problem: Very slow!

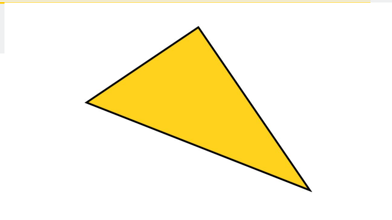
Training: 1-2 days on a V100 GPU, for just a single scene!

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! Cited by 11521

Gaussian Splatting (2023)



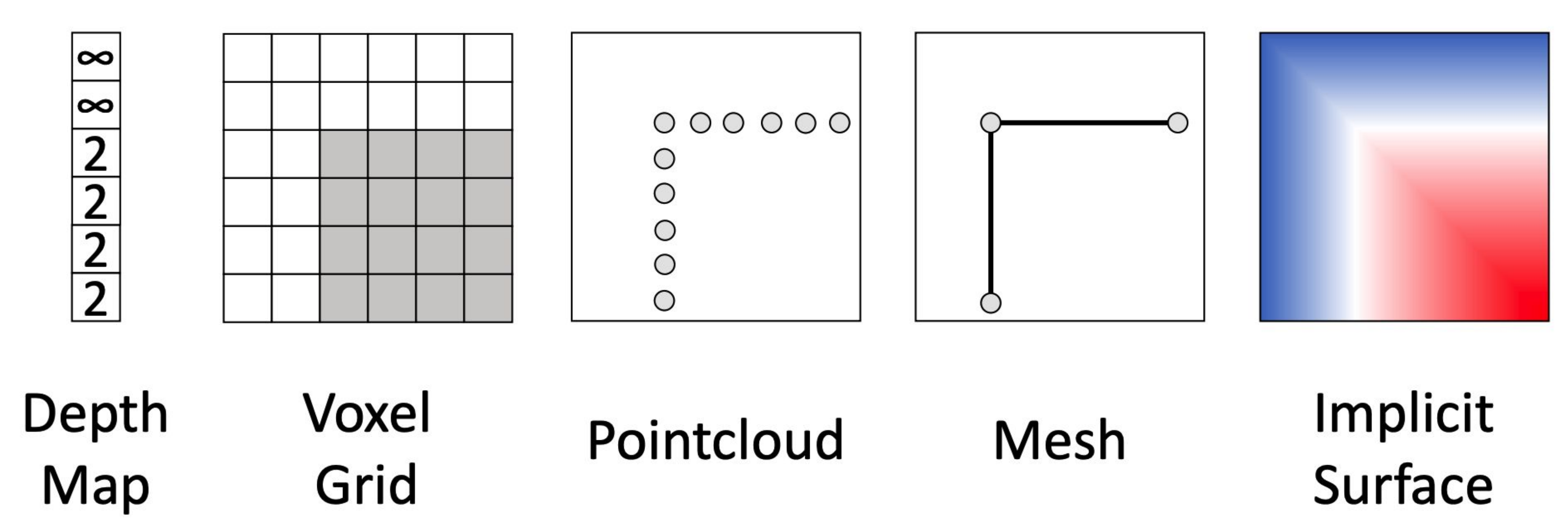


https://arxiv.org/pdf/2308.04079

Gaussian Splatting

(More on this)

Summary



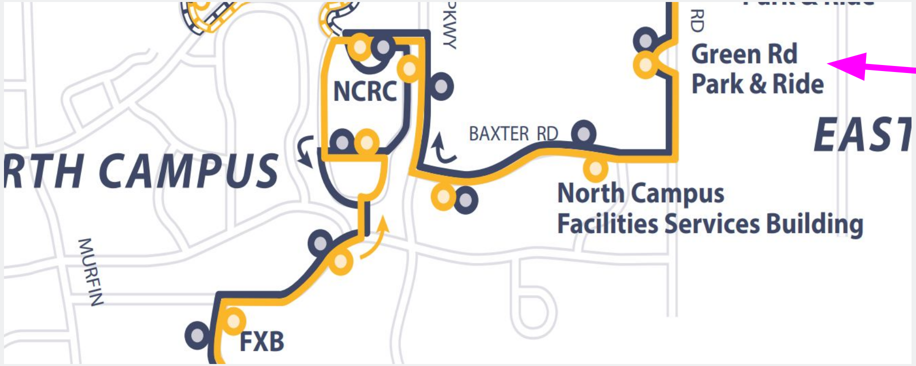
Reminder:

- **Final Project** “lightning talk” April 1, 2025 (tomorrow during discussion)
- **Canvas Quiz**

MCity Data Collection

Tomorrow (April 1st), 8am EST
Reach out to Cale Colony (GSI) ccolony@umich.edu
Dress warmly (26-44F)

https://piazza.com/class/m4pgejar4ua2qf/post/228



MCity