Computer Vision

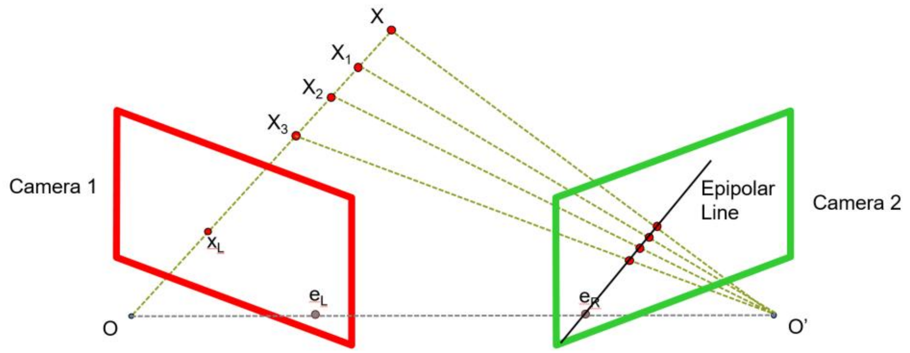
Lecture 10: 3D Vision

Oliver Bimber

JOHANNES KEPLER
UNIVERSITY LINZ

Last Week: Multi-View Geometry

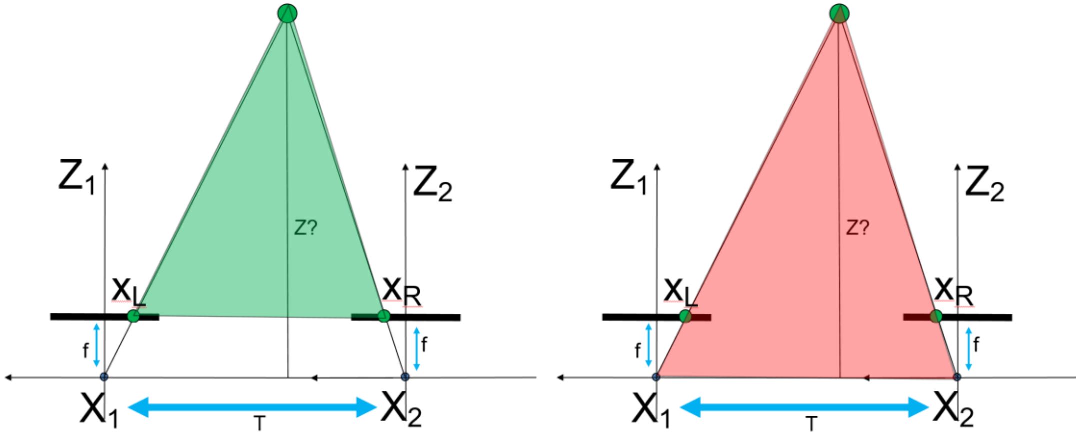
Epipolar Constraints



It's a 1D Search Problem!
But how do we get the Epipolar Line for a given Point?

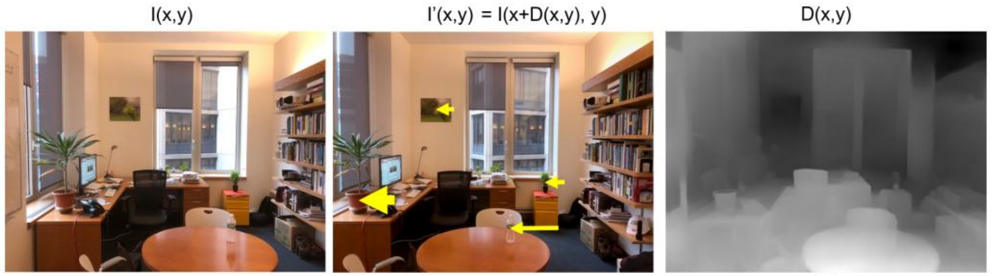
JOHANNES KEPLER UNIVERSITY LINZ

Stereoscopic Depth-Reconstruction



JOHANNES KEPLER UNIVERSITY LINZ

Disparity Maps



$$Z(x,y) \propto \frac{1}{D(x,y)}$$

JOHANNES KEPLER UNIVERSITY LINZ

Structure-from-Motion

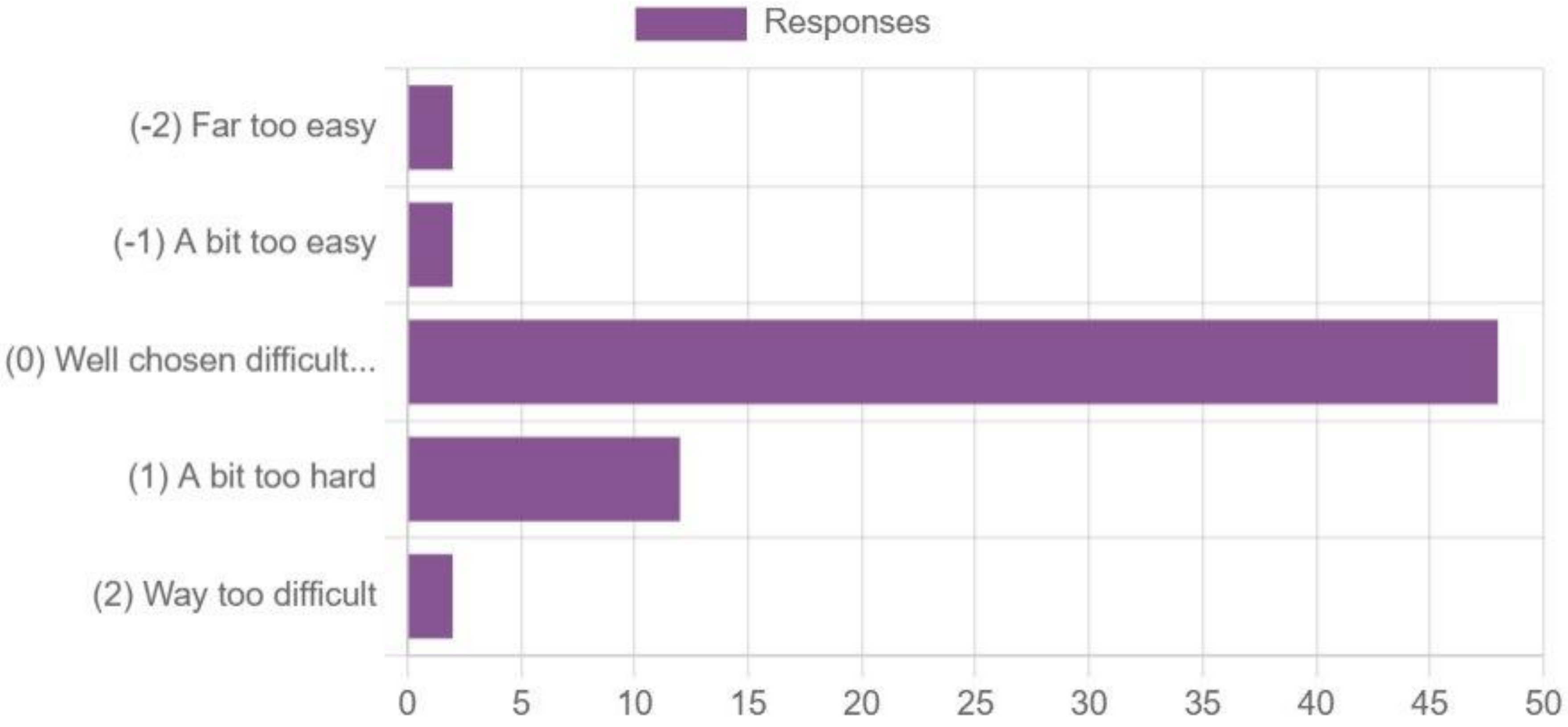
JOHANNES KEPLER UNIVERSITY LINZ

Course Overview

CW	Topic	Date	Place	Lab
41	Introduction and Course Overview	07.10.2025	Zoom	Lab 1
42	Capturing Digital Images	14.10.2025	Zoom	Lab 2
43	Digital Image Processing	21.10.2025	Zoom	Assignment 1
44	Machine Learning	28.10.2025	Zoom	
45	Feature Extraction	04.11.2025	Zoom	Open Lab 1
46	Segmentation	11.11.2025	Zoom	Assignment 2
47	Optical Flow	18.11.2025	Zoom	Open Lab 2
48	Object Detection	25.11.2025	Zoom	Assignment 3
49	Multi-View Geometry	02.12.2025	Zoom	Open Lab 3
50	3D Vision	10.12.2025	Zoom	Assignment 4
3	Trends in Computer Vision	13.01.2026	Zoom	
4	Q&A	20.01.2026	Zoom	Open Lab 4
5	Exam	27.01.2026	HS1 (Linz), S1/S3 (Vienna), S5 (Bregenz)	
9	Retry Exam	24.02.2026	tba	

Assignment 3

How do you rate the Difficulty of the Assignment?

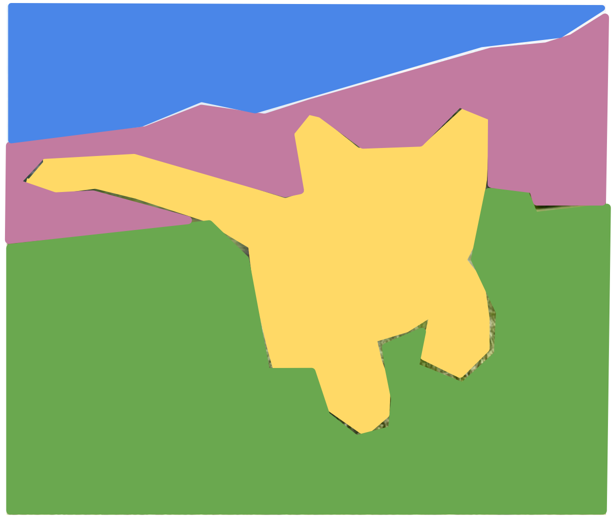
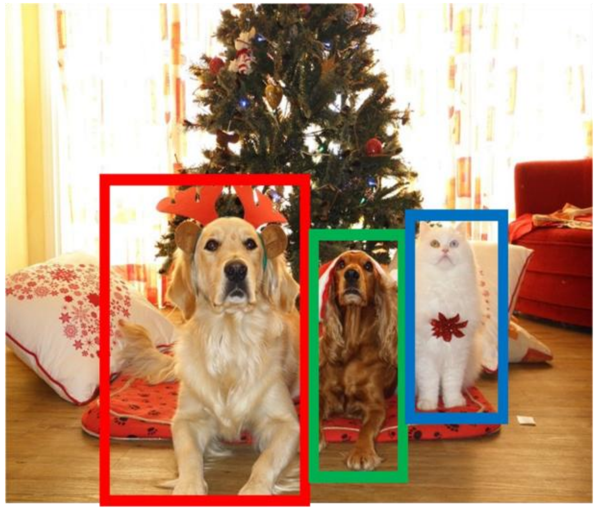
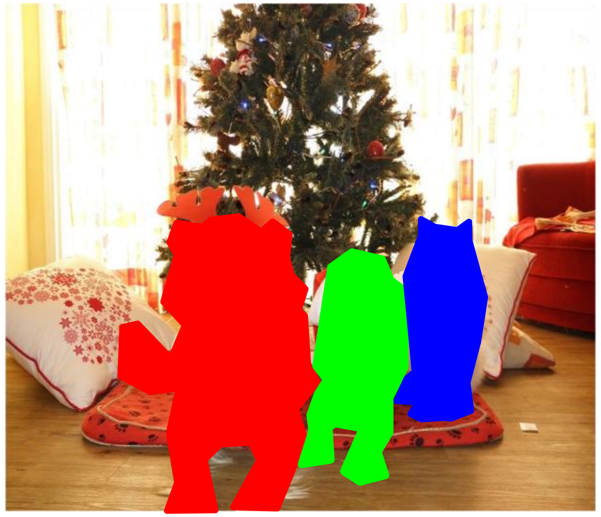


Average Time spent: 9.40 h

275 total submitted Assignments

66 gave Feedback – thank you!

We have seen how to predict 2D Shapes of Objects

Classification	Semantic Segmentation	Object Detection	Instance Segmentation
			
CAT	GRASS, CAT, TREE, SKY	DOG, DOG, CAT	DOG, DOG, CAT

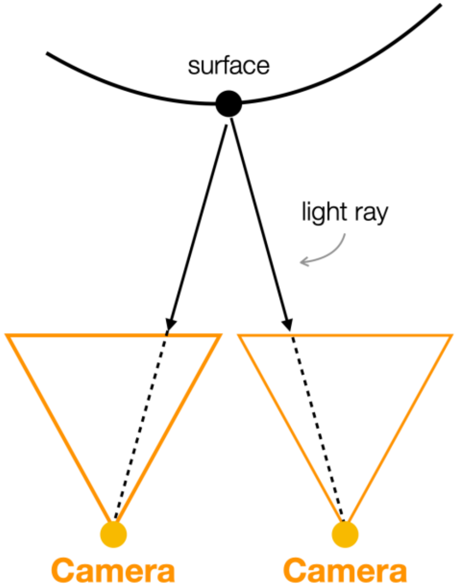
No spatial extent

No objects, just pixels

Multiple Objects

But we interpret the World in 3D

Recap: Depth from Stereo (and multiple Views)



Key Idea: use difference in Perspective to estimate Depth

Solution: develop Models for Feature Extraction, Matching, Depth Computation, Camera Calibration, Pose estimation, etc.

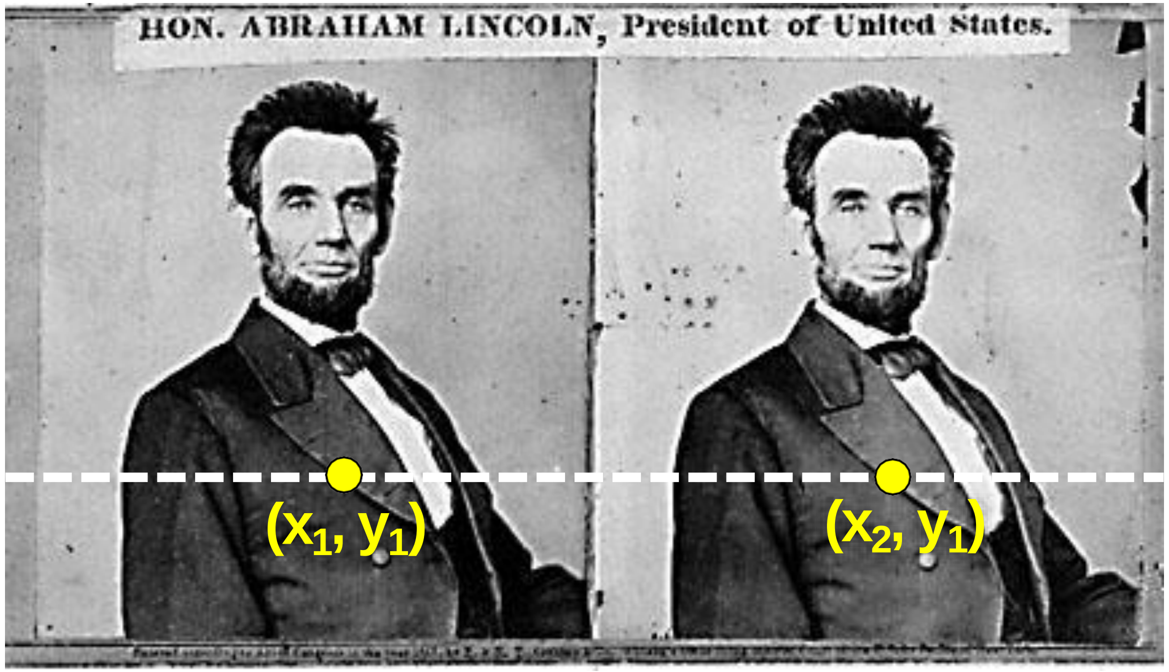
Depth from X

X = Shading
Key Idea: use difference in Shading to estimate Depth
Solution: develop Models for Inverse Shading

X = Defocus
Key Idea: use difference in Focus to estimate Depth
Solution: develop Models for Focus Estimation

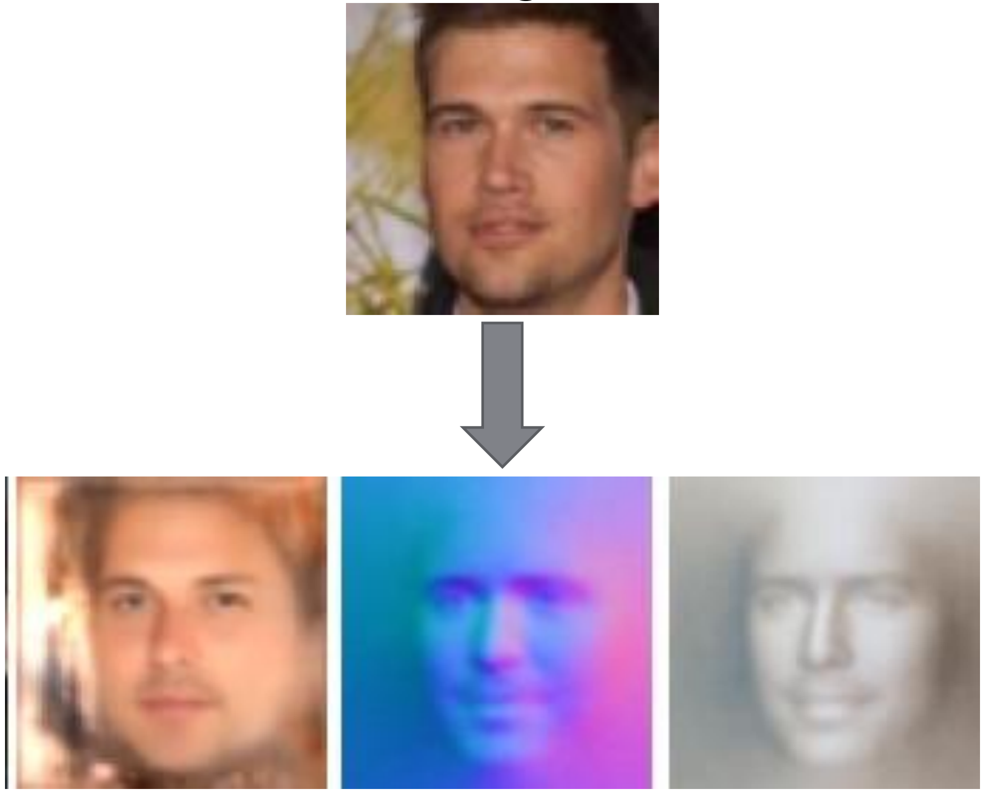
Why Learning how to estimate Depth?

epipolar lines



Matching is hard!

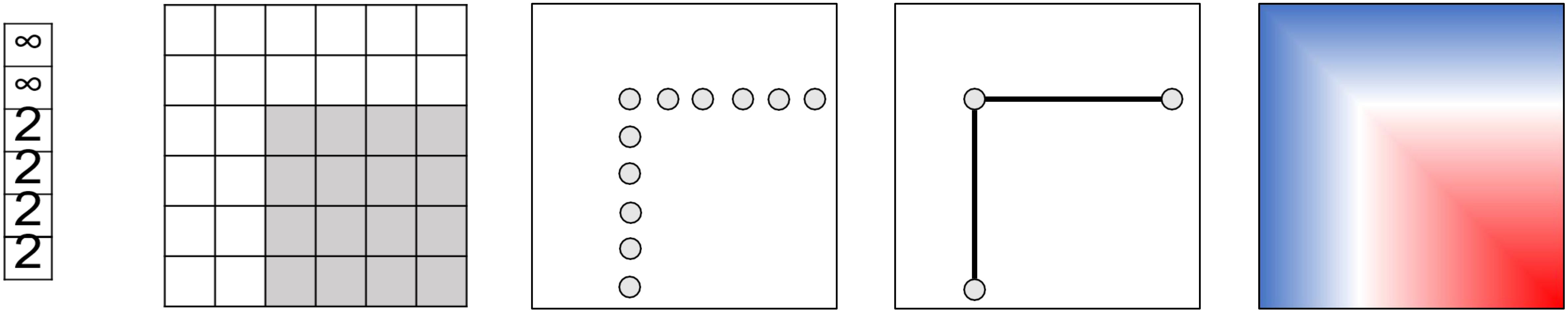
Image



Reflectance Shape Shading

Ill-posed Problem!

How to represent Depth



Depth Map

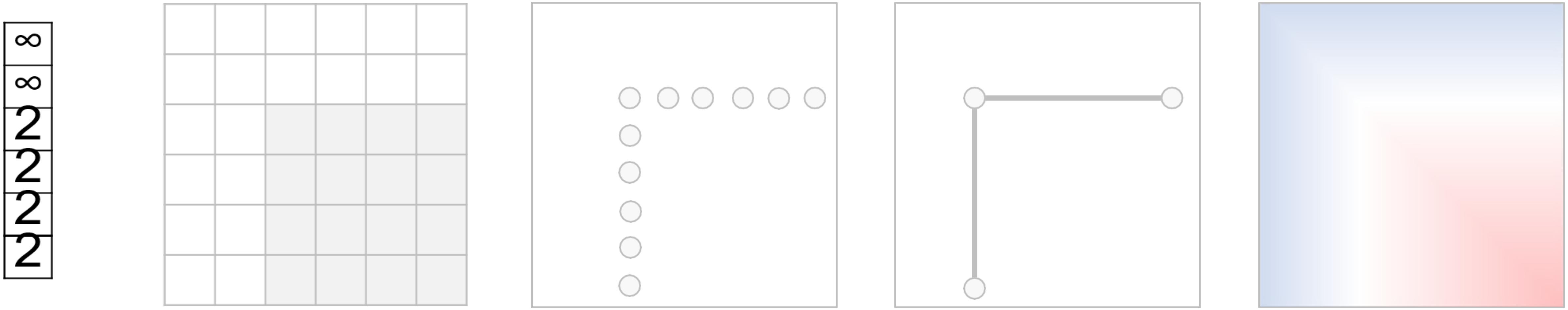
Voxel Grid

Point Cloud

Mesh

Implicit Surface

Depth Maps



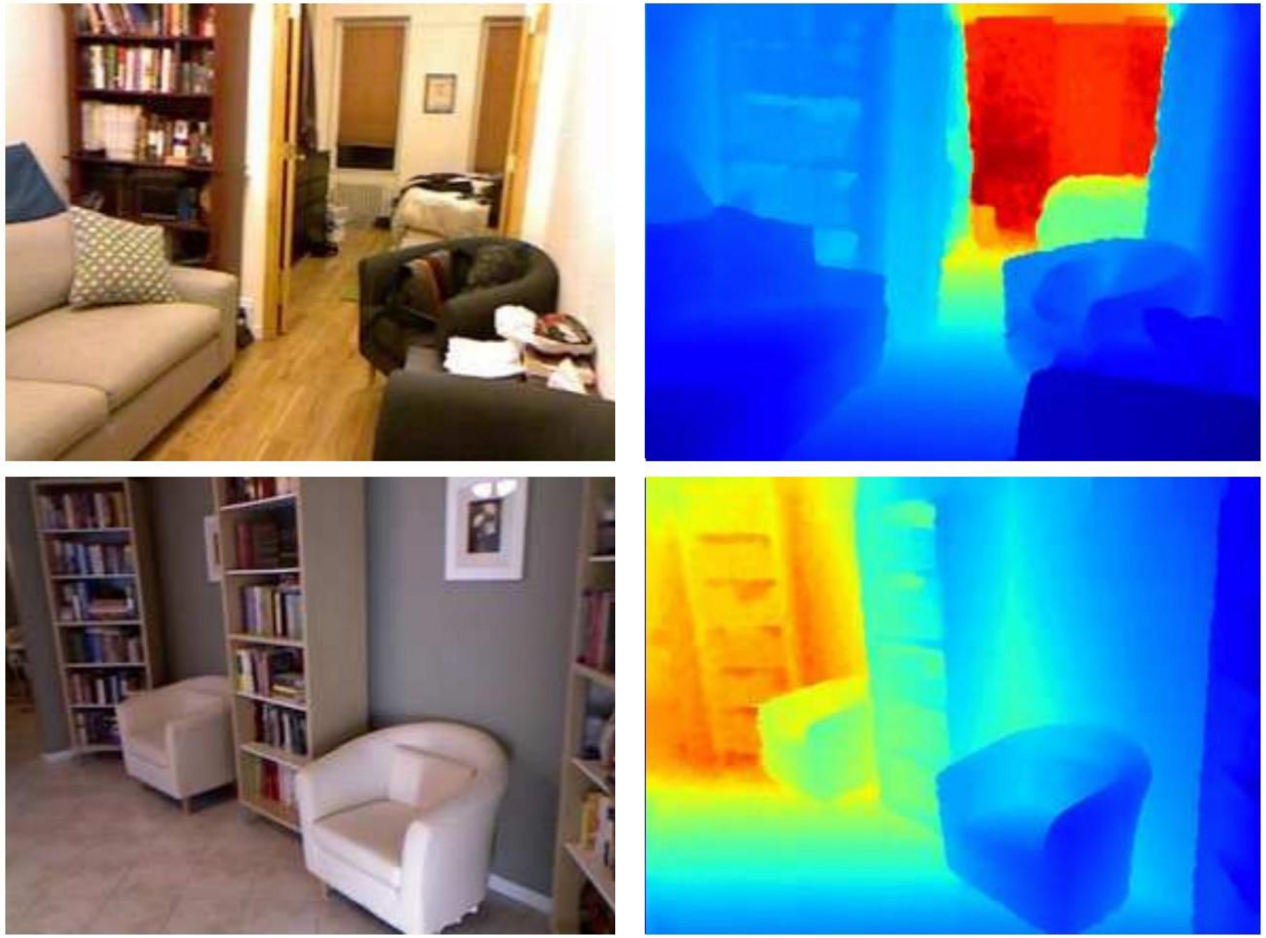
Depth Map | Voxel Grid | Point Cloud | Mesh | Implicit Surface

How to measure Depth Maps

A Depth Map gives the Distance from the Camera to the Object in the World at each Pixel

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

This Type of Data can be recorded directly for some Types of 3D Sensors (e.g. Microsoft Kinect)



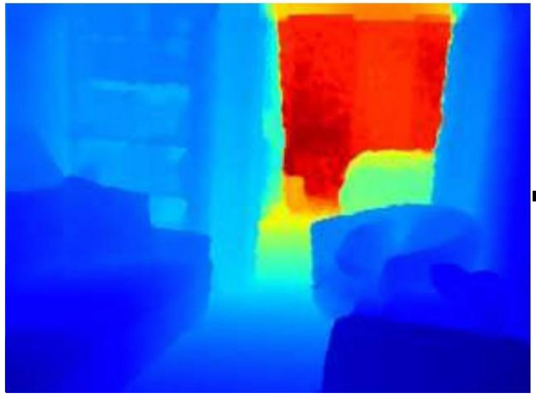
RGB Image: 3 x H x W Depth Map: H x W

Predicting Depth Maps

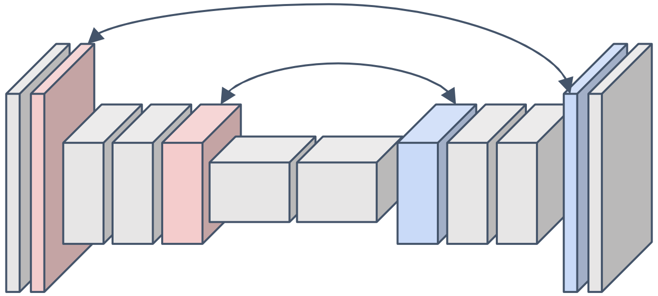
Estimate log Depth instead of Depth. Defining y_i as the Ground Truth Depth on Pixel i, and y^*_i its estimated Depth:

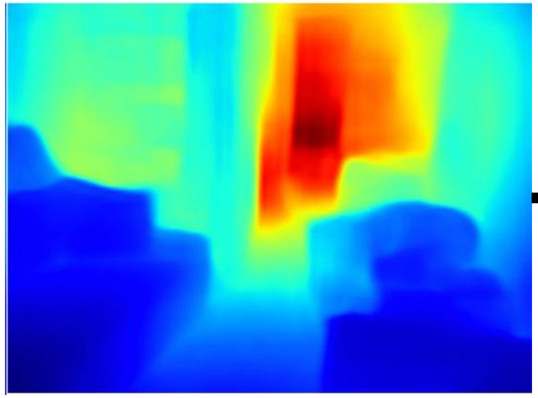
$$D_{L2}(y, y^*) = \frac{1}{n} \sum_{i=1}^{n} (\log y_i - \log y_i^*)^2$$

Measured Depth Image:
1 x H x W



Per-Pixel Loss
(L2 Distance)



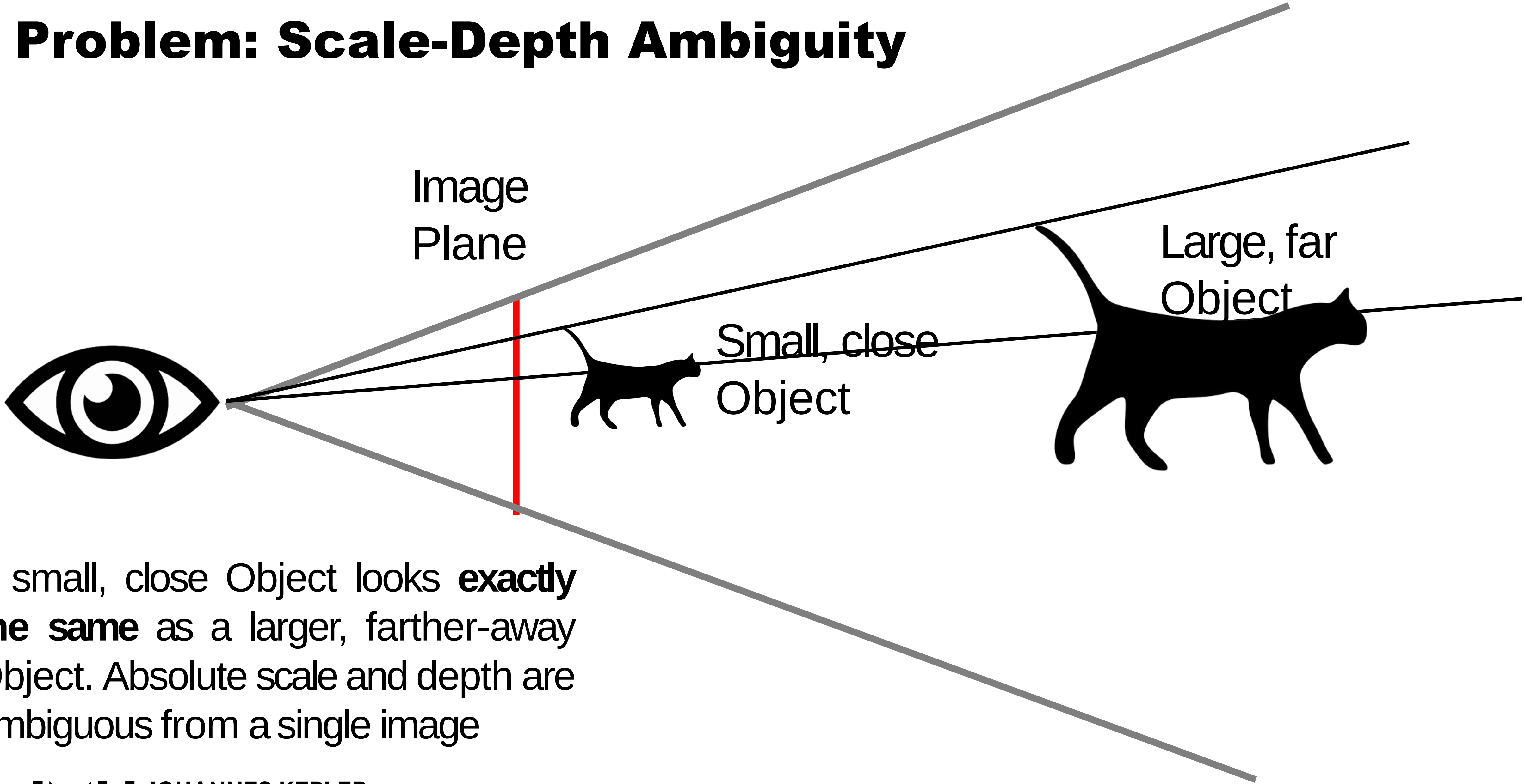


RGB Input Image:
3 x H x W

CNN

Predicted Depth Image:
1 x H x W

Problem: Scale-Depth Ambiguity



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

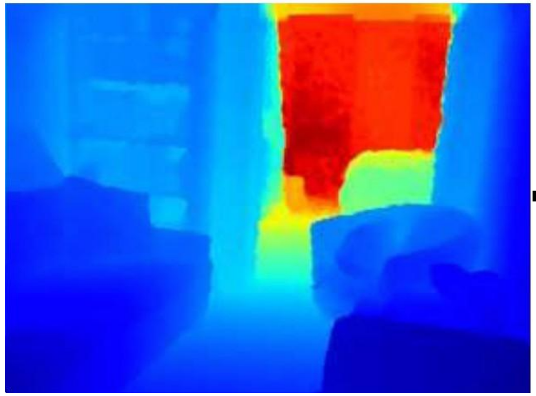
Scale Invariant Error

The global scale of a scene is a fundamental Ambiguity in Depth Prediction! So it is considered in the Loss Function.

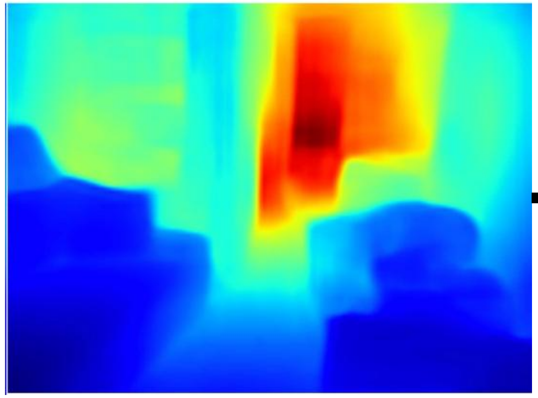
$$D_{SI}(y, y^*) = \frac{1}{n}\sum_{i=1}^{n}\left(\log y_i - \log y_i^* + \alpha(y, y^*)\right)^2$$

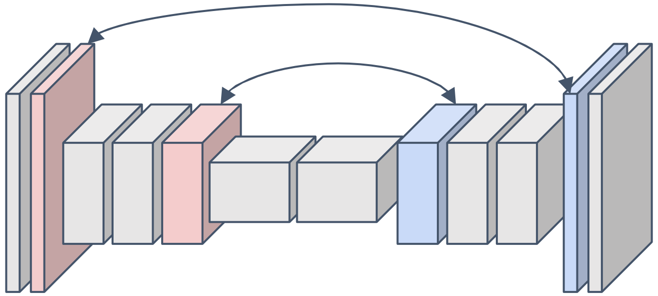
$$\alpha(y, y^*) = \frac{1}{n}\sum_{j=1}^{n}\left(\log y_j - \log y_j^*\right)$$

Measured Depth Image:
1 x H x W



Per-Pixel Loss
(Scale Invariance)





RGB Input Image:
3 x H x W

CNN

Predicted Depth Image:
1 x H x W

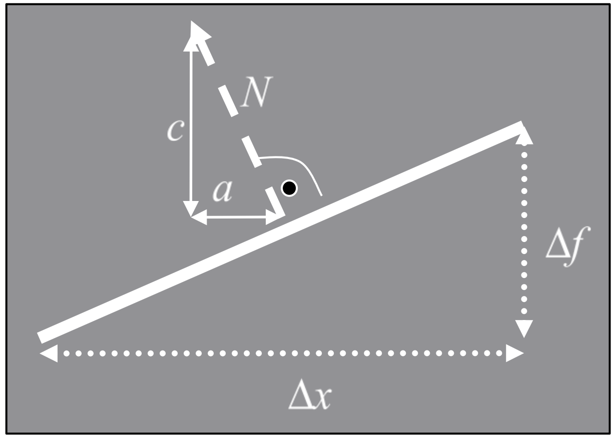
http://arxiv.org/pdf/1406.2283

Surface Normals

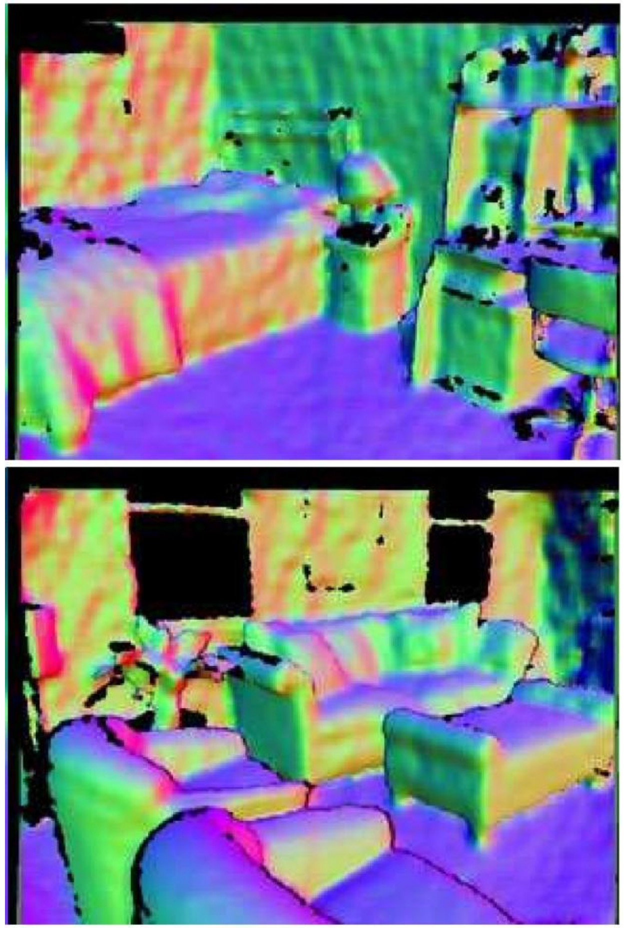
For each Pixel, its Surface Normal is the Normal Vector (i.e., Unit Vectors perpendicular to the tangential Surface) of the Pixel

We can compute Surface Normals from given Depth (Gradients), and we can compute Depth from given Surface Normals (Integration)

$$\frac{\partial f}{\partial x} = \frac{a}{c}$$



RGB Image: 3 x H x W

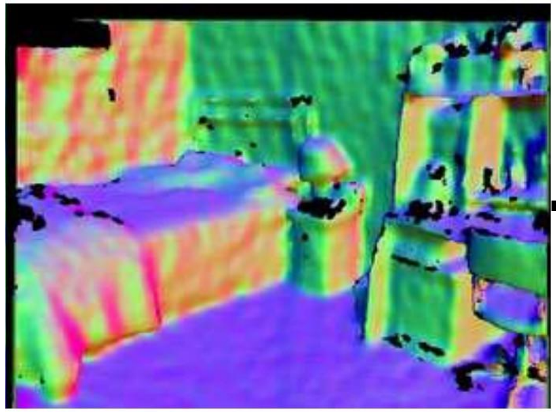


Normals: 3 x H x W

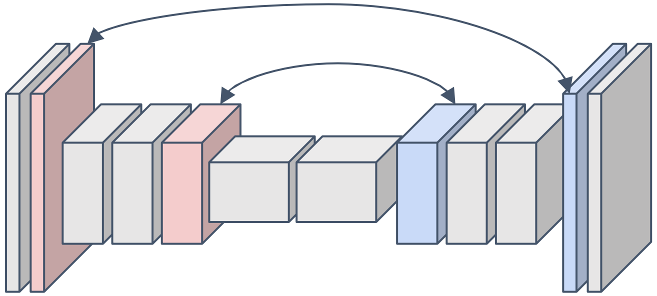
Predicting Surface Normals

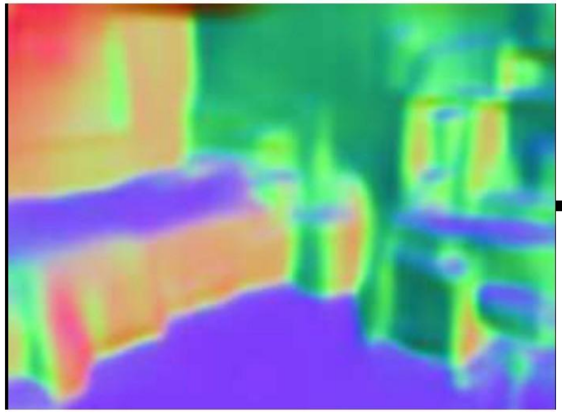
Here, x is the estimated and y the ground truth surface normal. The loss function considers the solid angle difference.

Ground-truth Normals:
3 x H x W



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





RGB Input Image:
3 x H x W

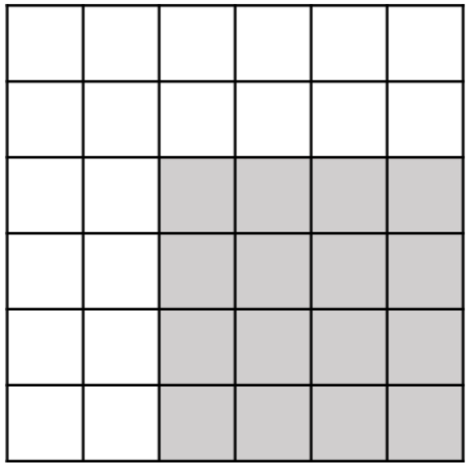
Fully Convolutional network

Predicted Normals:
3 x H x W

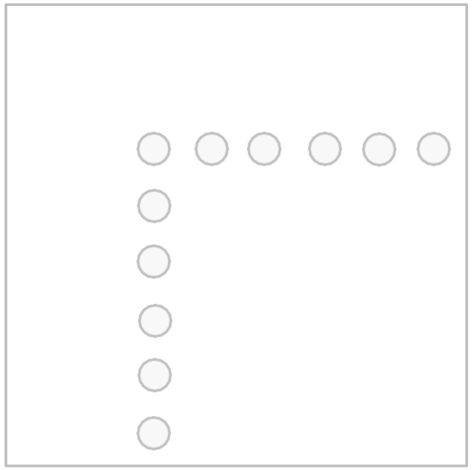
Recall:
$x \cdot y$
$= |x|\,|y| \cos\theta$

Voxel Grids

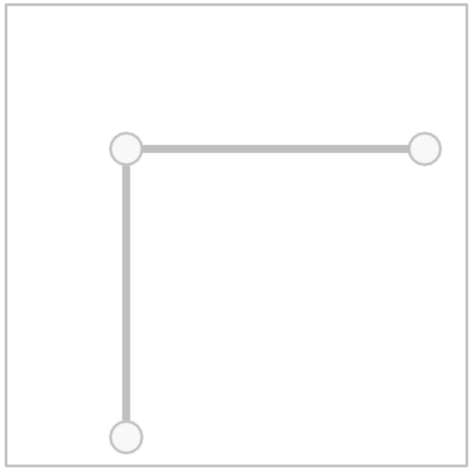
Depth
Map



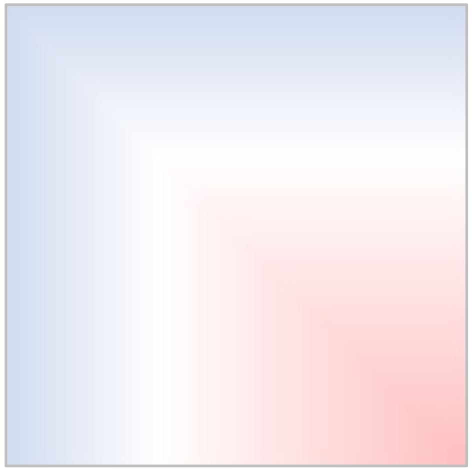
Voxel
Grid



Point
Cloud



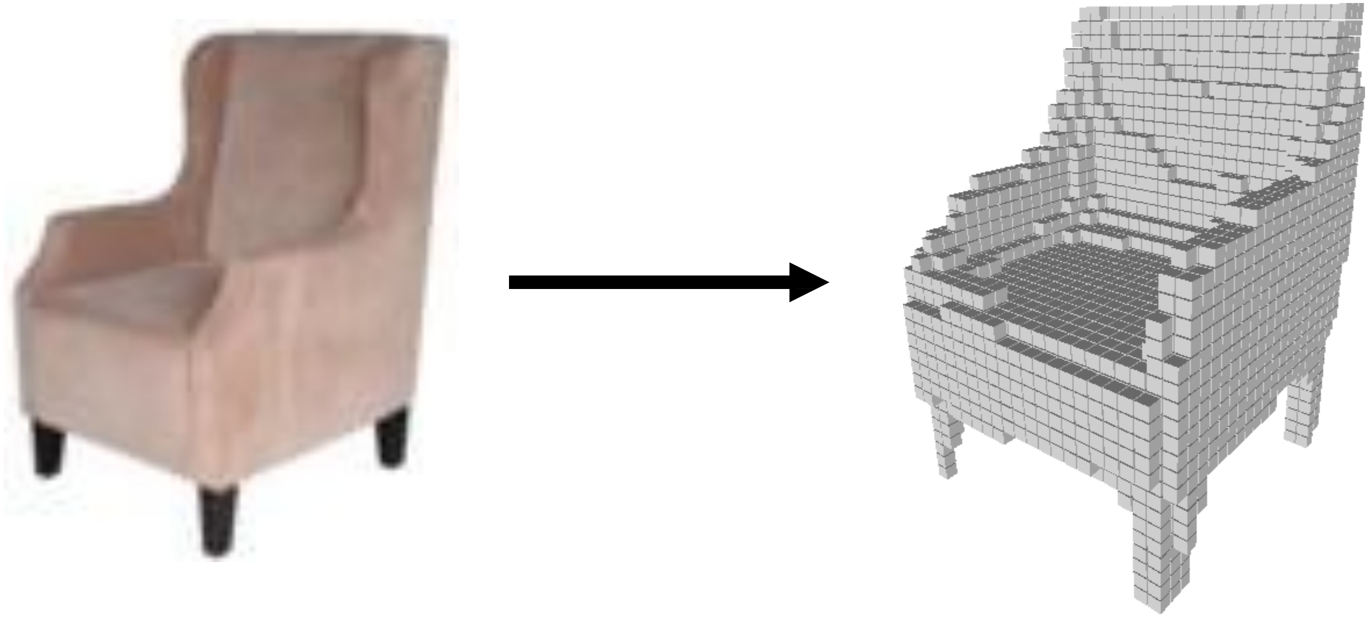
Mesh



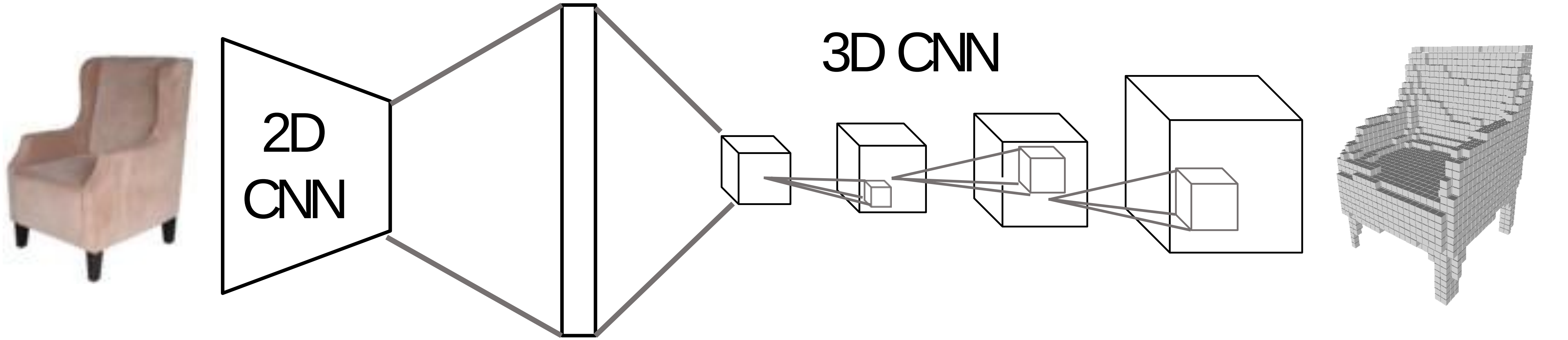
Implicit
Surface

Voxel Grids

- Represent a Shape with a V x V x V Grid of Occupancies
- Basically just like Segmentation, but in 3D!
- (+) Conceptually simple: just a 3D grid!
- (-) Need high spatial Resolution to capture fine Structures
- (-) Scaling to high Resolutions is nontrivial!



Example for Generating Voxel Shapes



Input image:
3 x 112 x 112

2D Features:
C x H x W

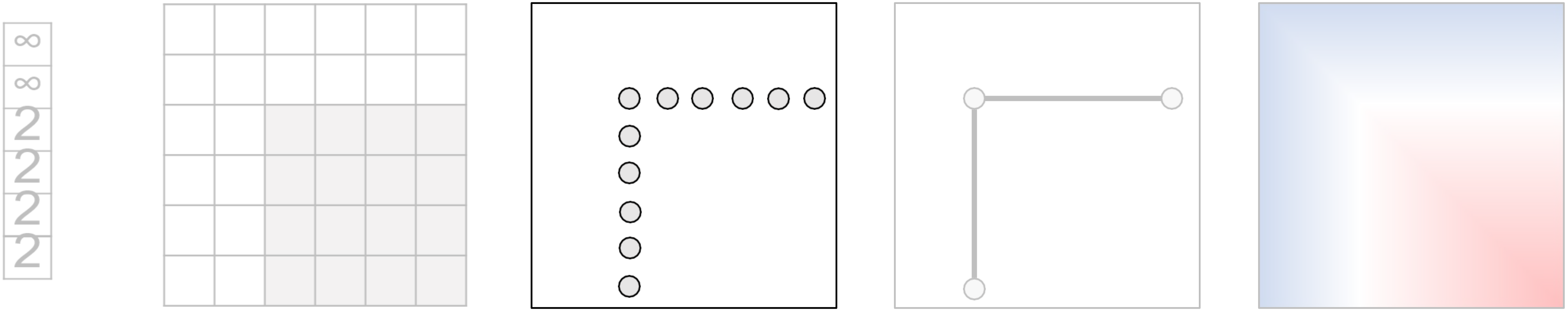
3D Features:
C' x D' x H' x W'

Voxels:
1 x V x V x V

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

Point Clouds



Depth Map

Voxel Grid

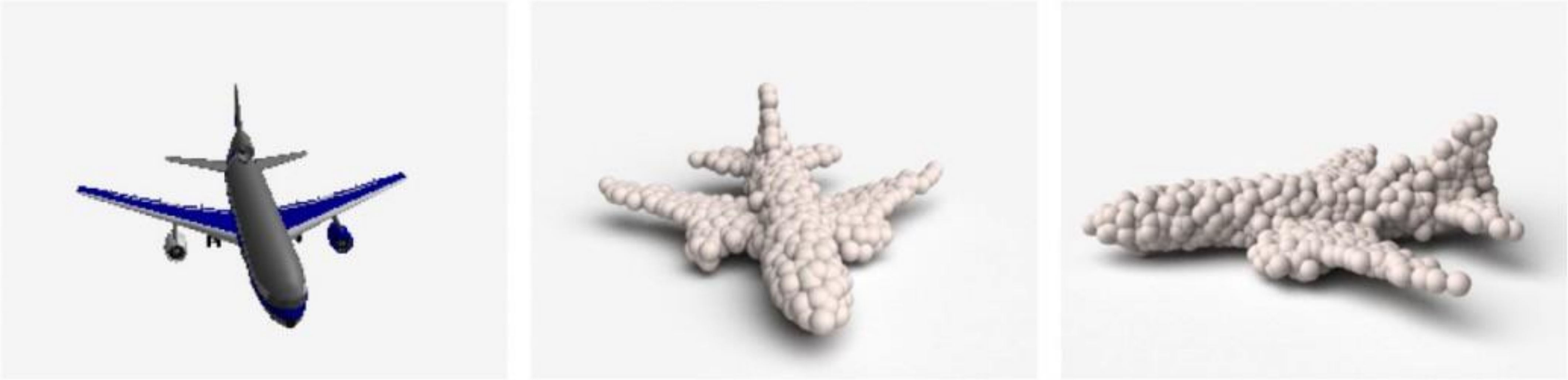
Point Cloud

Mesh

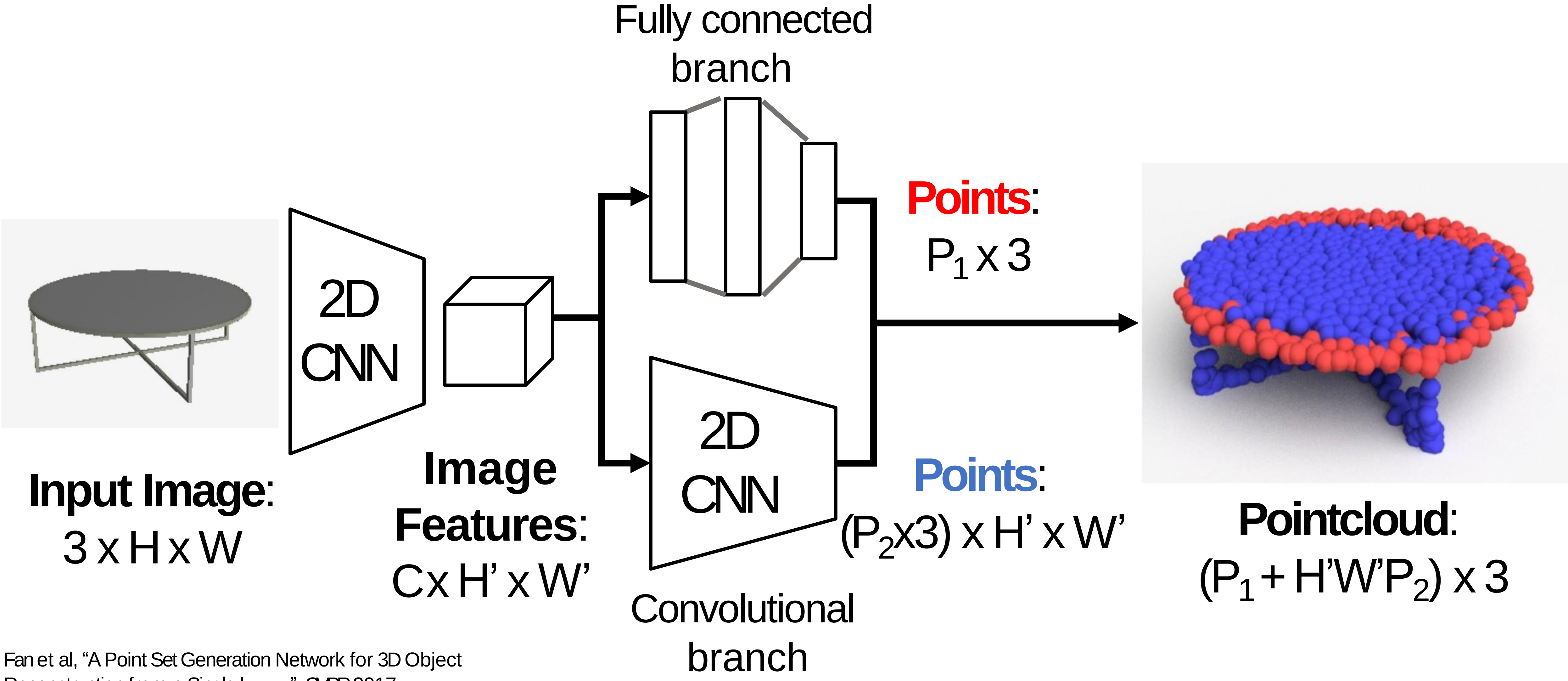
Implicit Surface

Point Clouds

- 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

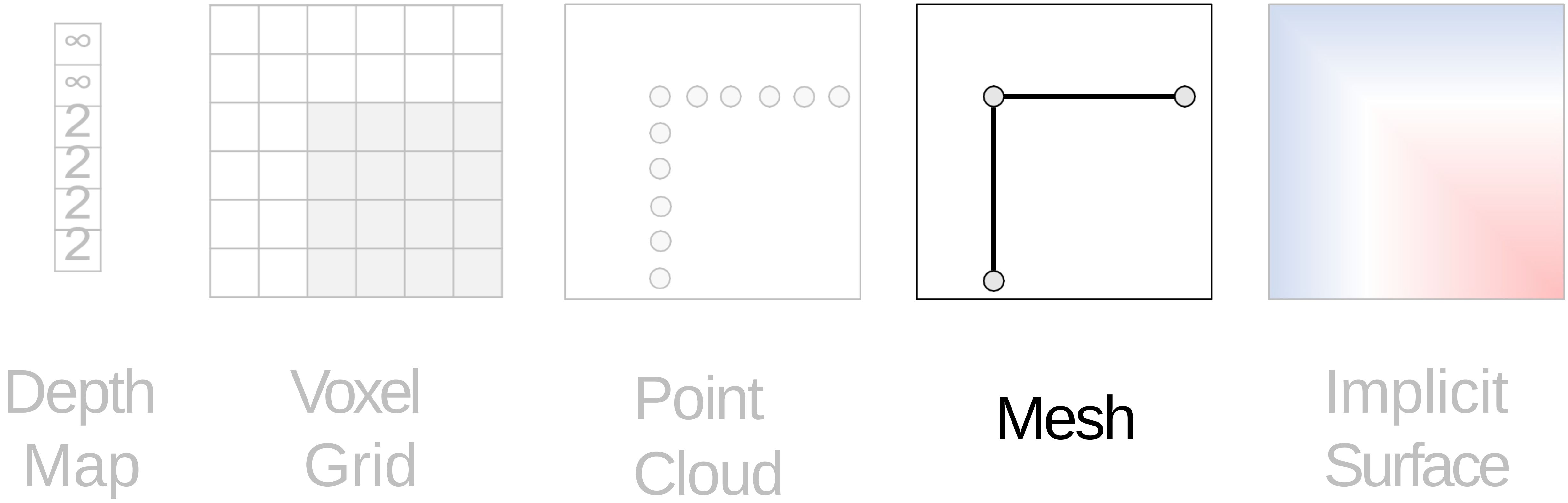


Example for Generating Point Clouds



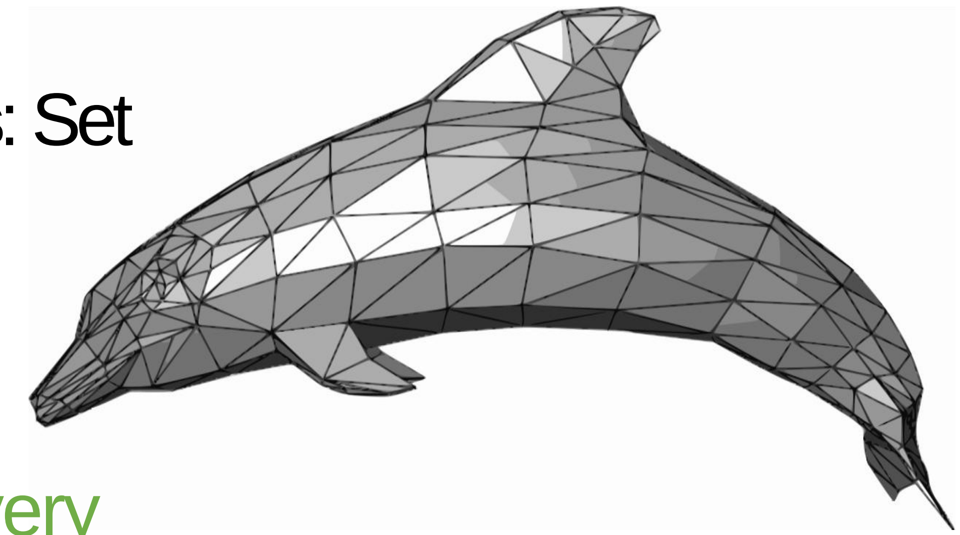
Fan et al, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image", CVPR 2017

Meshes


8
8
2
2
2
2
Depth
Map
Voxel
Grid
Point
Cloud
Mesh
Implicit
Surface

Meshes

- 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.

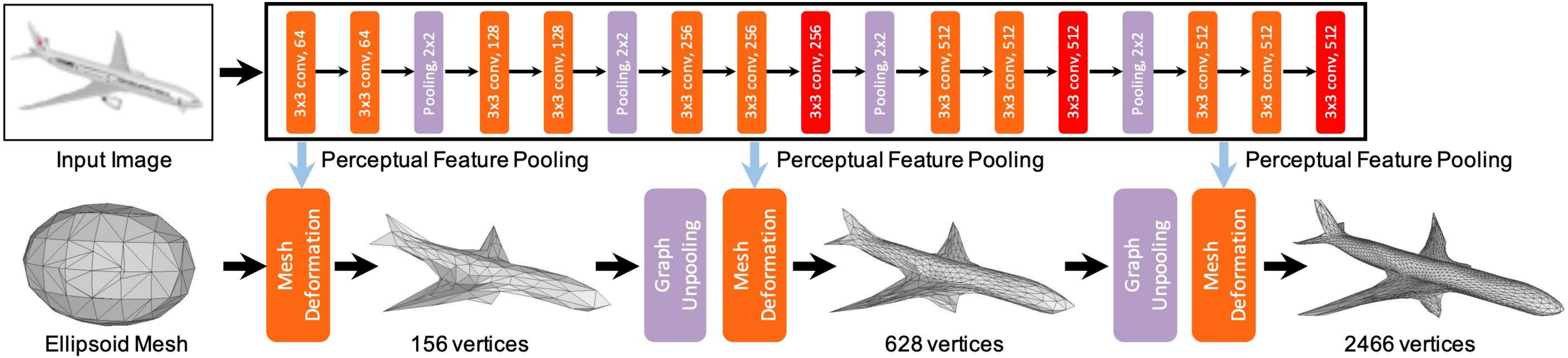


Example for Predicting Meshes

Input: Single RGB Image of an object

Key ideas:
Iterative Refinement
Graph Convolution Vertex
Aligned-Features
(Projected to Image Space)

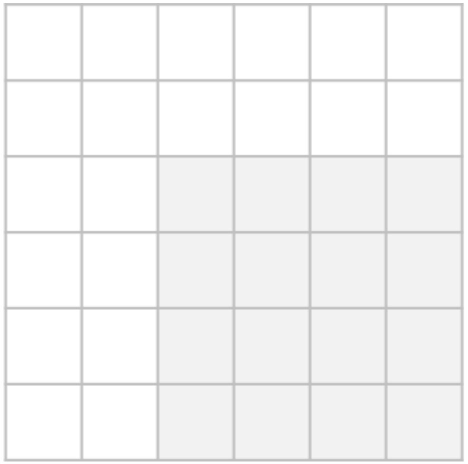
Output: Triangle mesh for the object



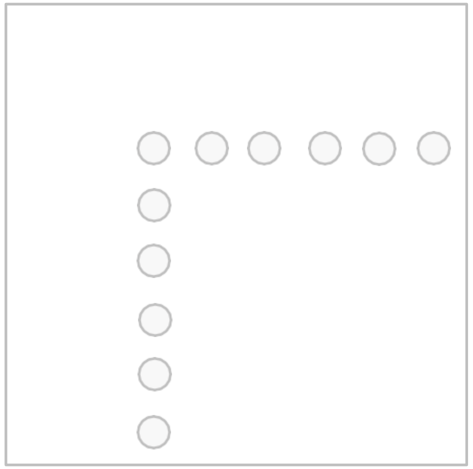
Wang et al, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images", ECCV 2018

Implicit Surfaces

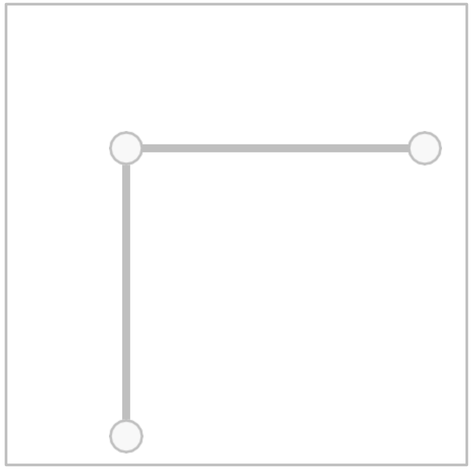
Depth
Map



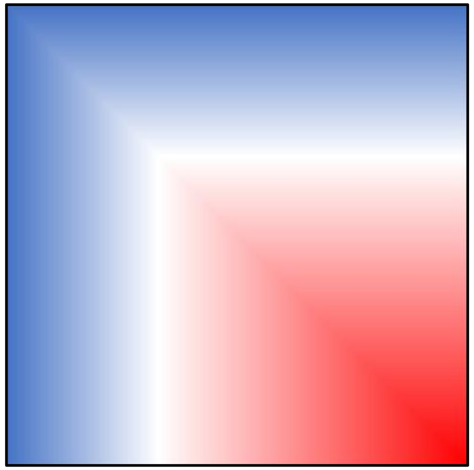
Voxel
Grid



Point
Cloud



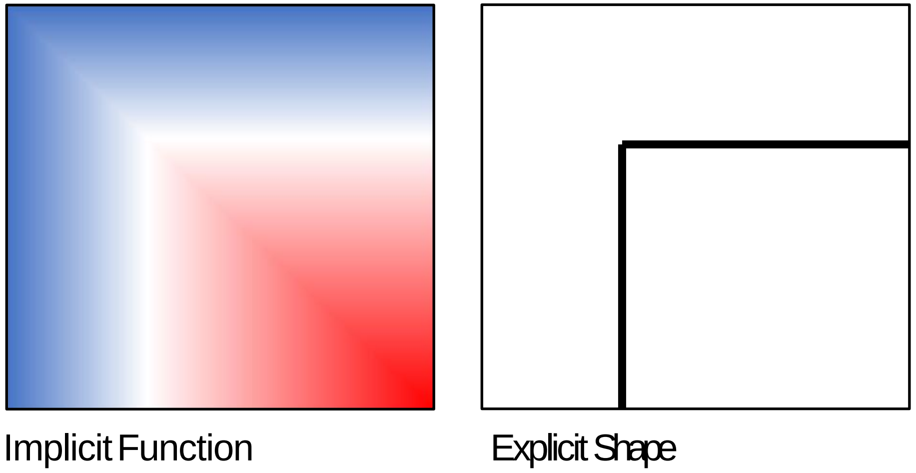
Mesh



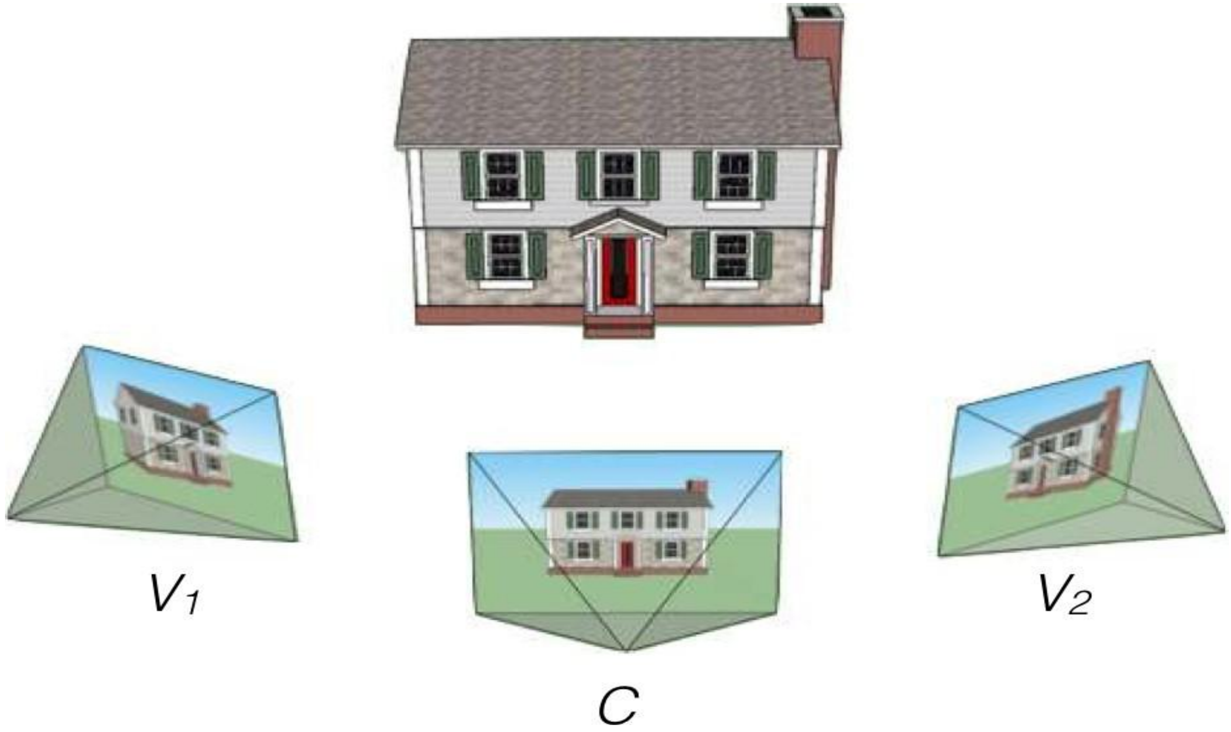
Implicit
Surface

Implicit Surfaces

- Learn a Function to classify arbitrary 3D Points as inside / outside the shape
- Application: Novel View Synthesis

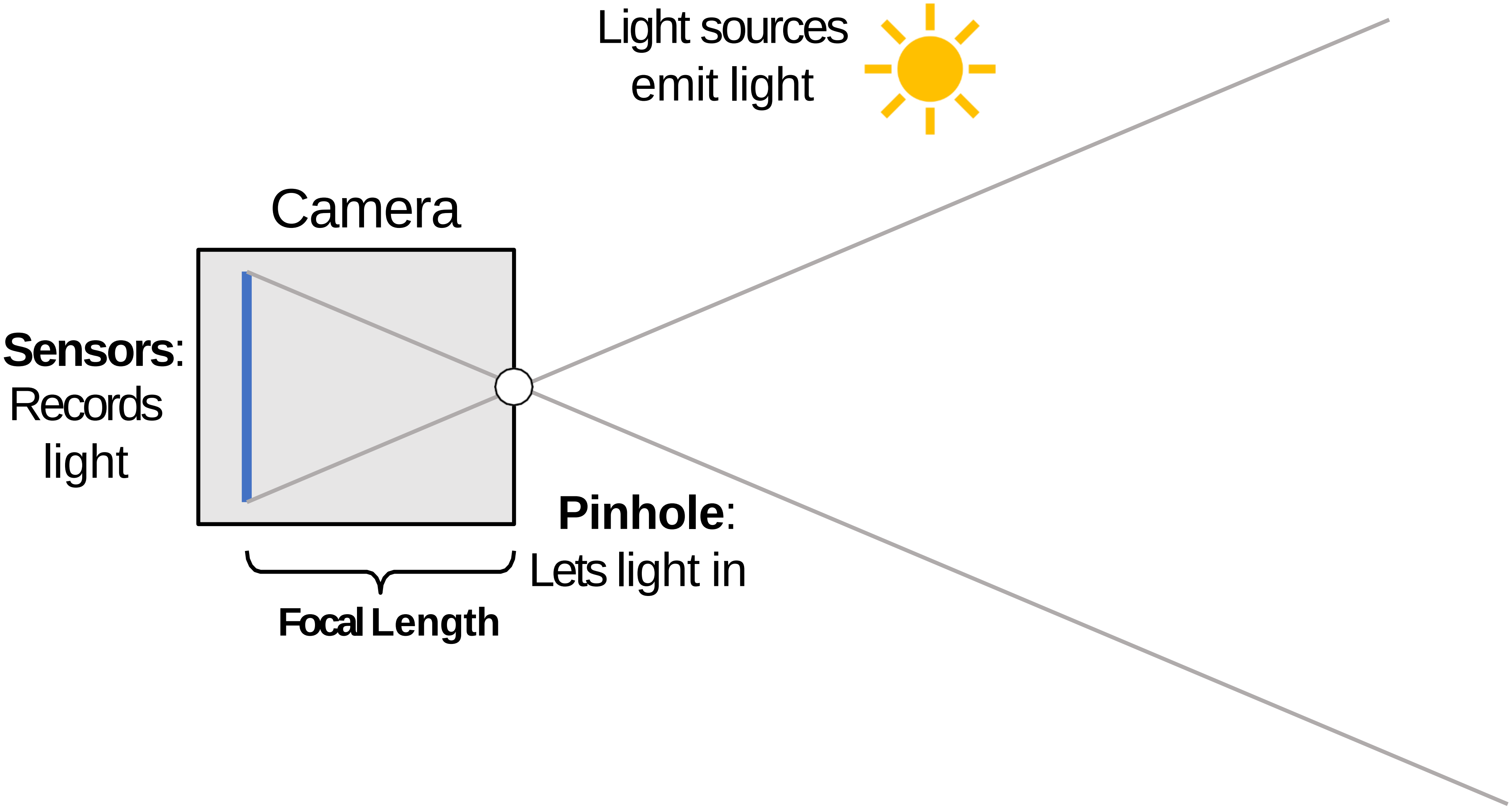


Same idea: **Signed Distance Function (SDF)** gives the Euclidean Distance to the Surface of the Shape; Sign gives Inside / Outside

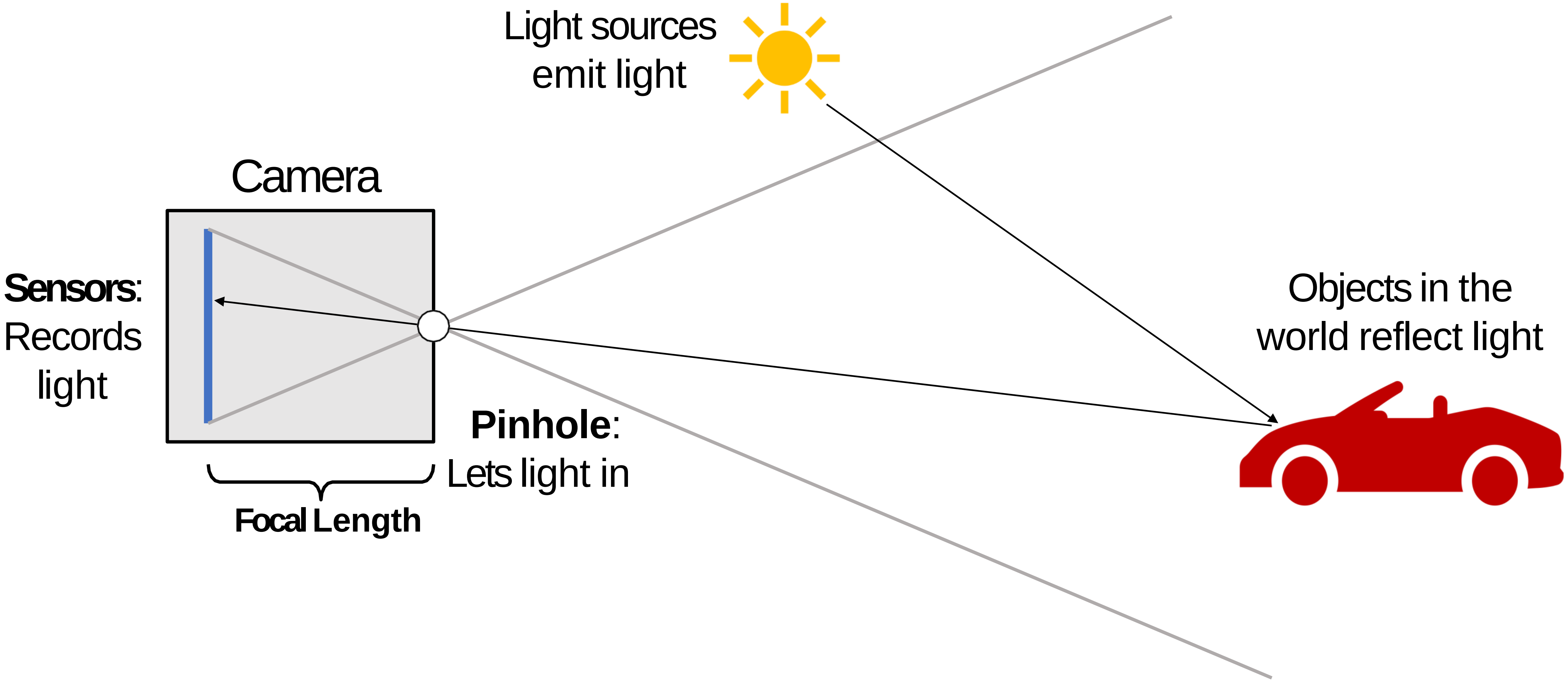


Given v1 and v2, reconstruct the new View C

Recap: Pinhole Cameras

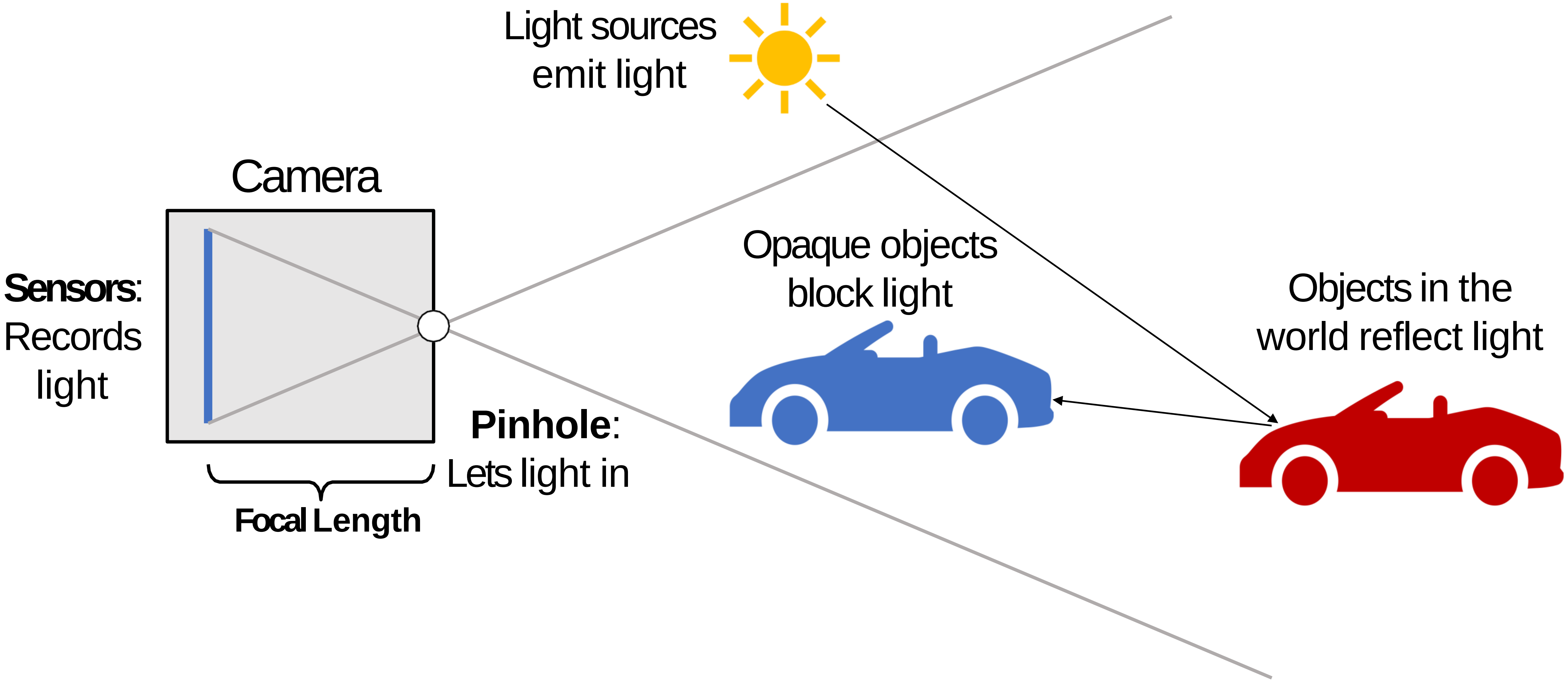


Recap: Pinhole Cameras

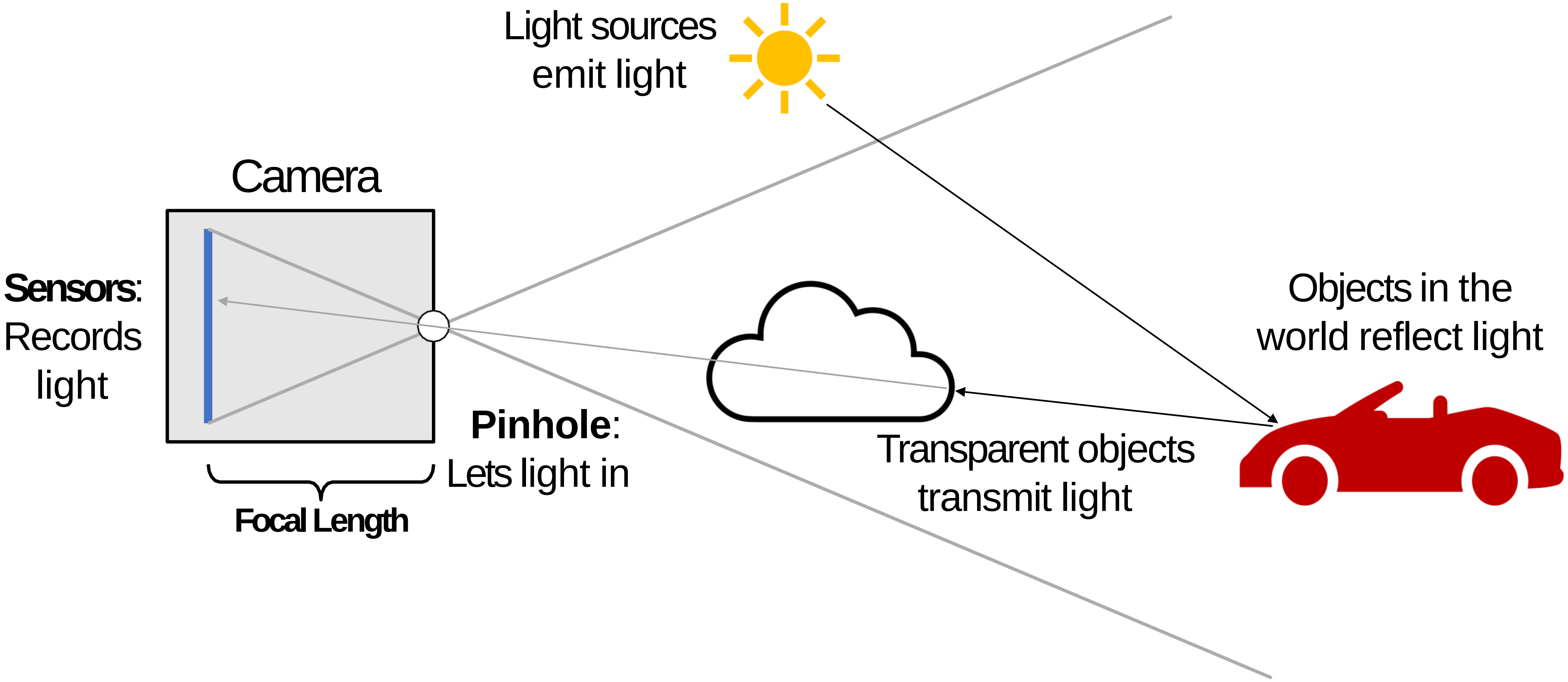


JOHANNES KEPLER
UNIVERSITY LINZ

Recap: Pinhole Cameras



Recap: Pinhole Cameras



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? $\sigma \in [0,1]$

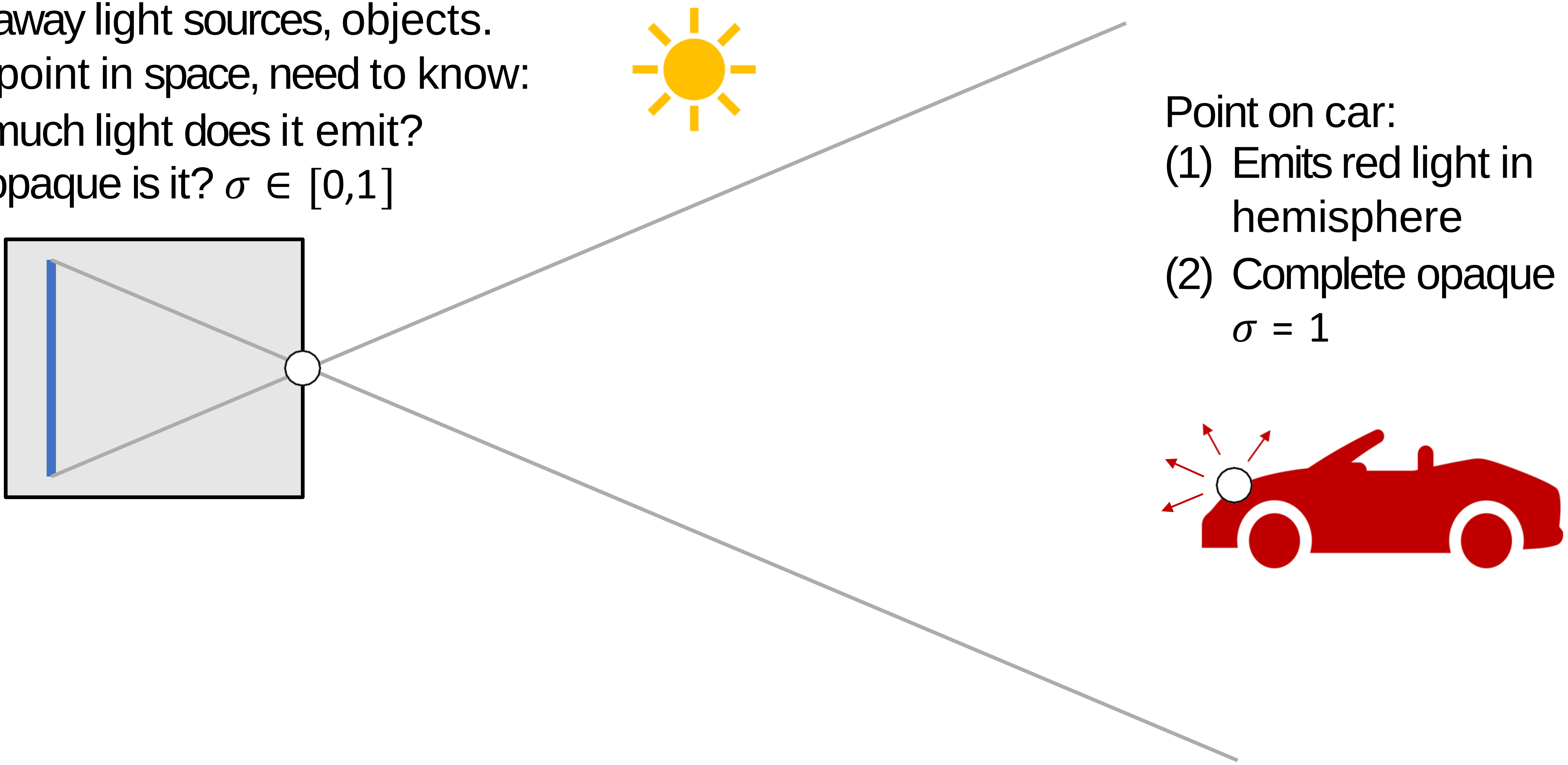
Volume Rendering

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? $\sigma \in [0,1]$



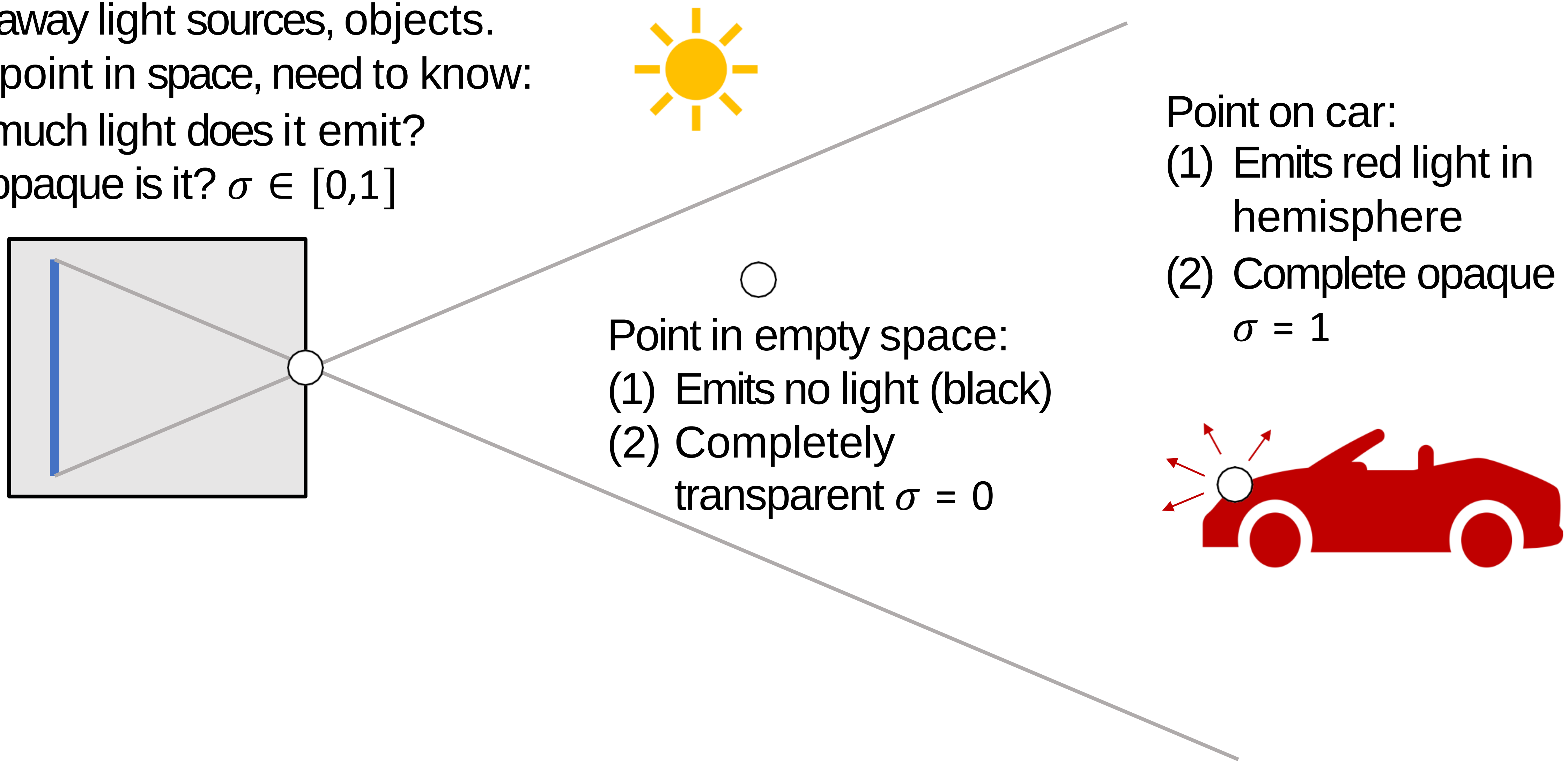
Point on car:

(1) Emits red light in hemisphere

(2) Complete opaque $\sigma = 1$

Volume Rendering

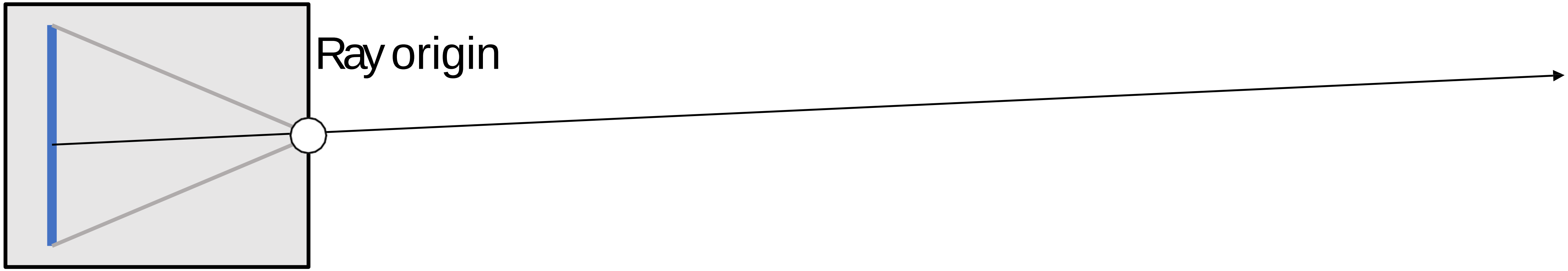
Abstract away light sources, objects.
For each point in space, need to know:
(1) How much light does it emit?
(2) How opaque is it? $\sigma \in [0,1]$



Point in empty space:
(1) Emits no light (black)
(2) Completely transparent $\sigma = 0$

Point on car:
(1) Emits red light in hemisphere
(2) Complete opaque $\sigma = 1$

Volume Rendering



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

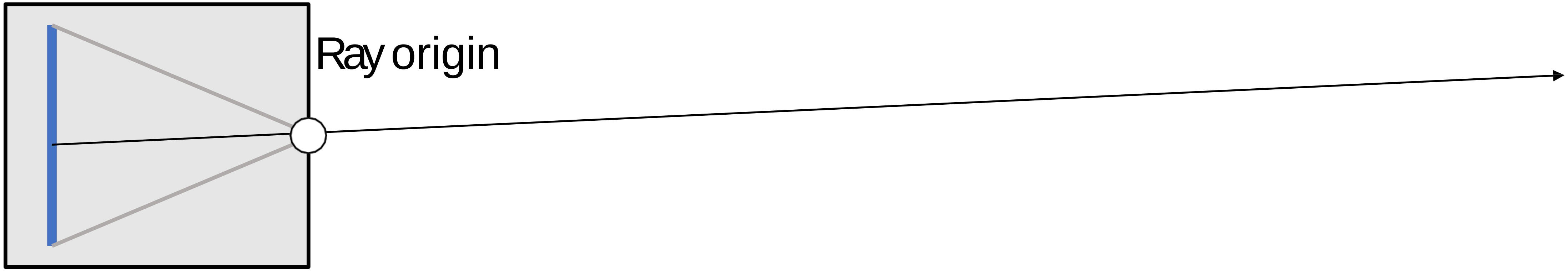
Volume Density is $\sigma(\boldsymbol{p}) \in [0,1]$

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:

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



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

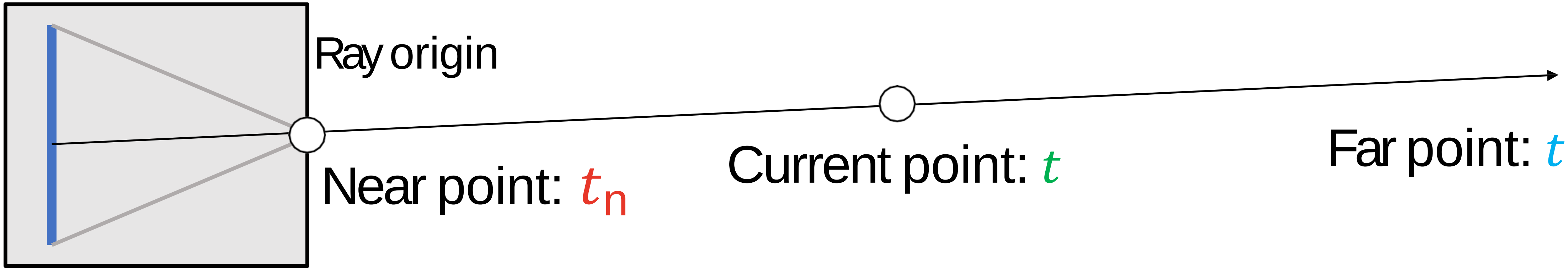
Volume Density is $\sigma(\boldsymbol{p}) \in [0,1]$

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

Volume Rendering

Color observed by the camera given by volume rendering equation:

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



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

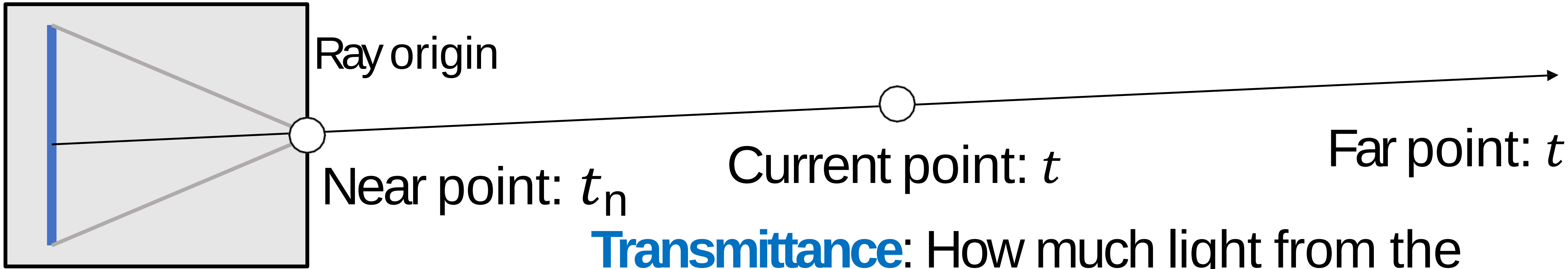
Volume Density is $\sigma(\mathbf{p}) \in [0,1]$

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

Volume Rendering

Color observed by the camera given by volume rendering equation:

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



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

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

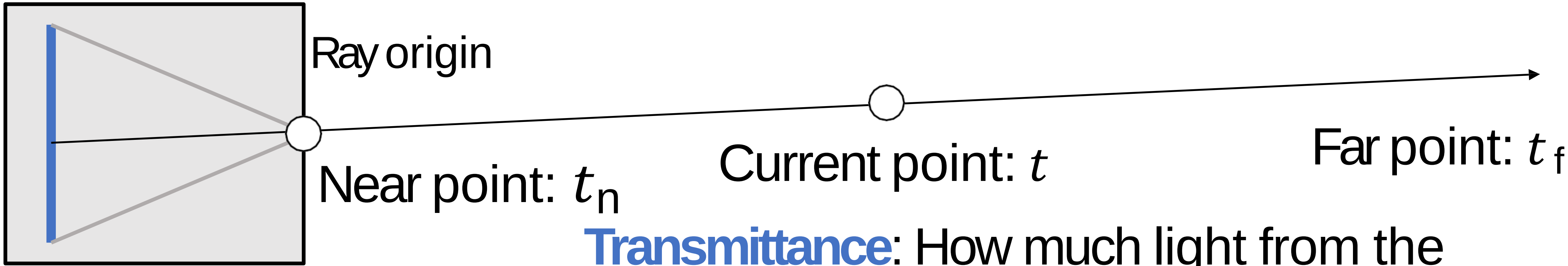
Volume Density is $\sigma(\boldsymbol{p}) \in [0,1]$

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:

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



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

Opacity: How opaque is the current point?

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

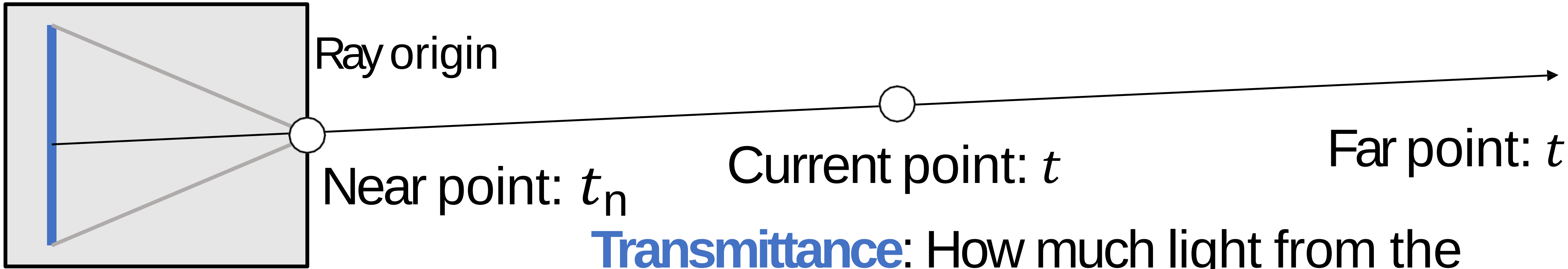
Volume Density is $\sigma(\boldsymbol{p}) \in [0,1]$

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:

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



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?

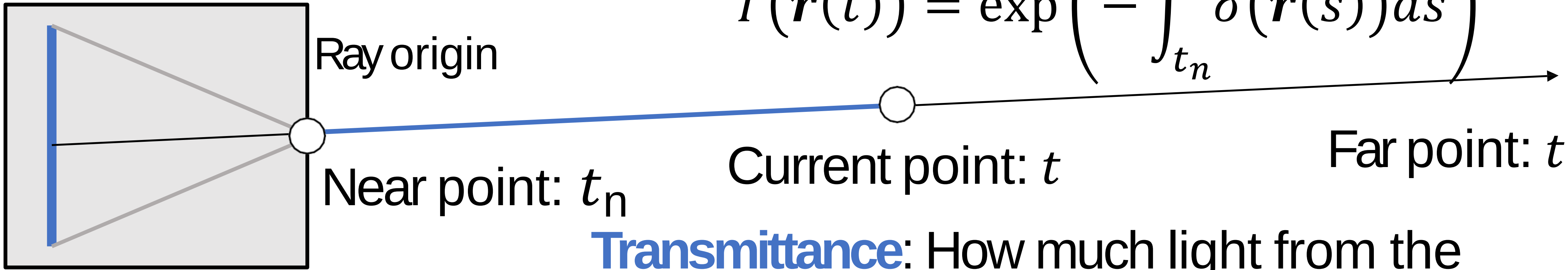
JOHANNES KEPLER UNIVERSITY LINZ

Volume Rendering

Color observed by the camera given by volume rendering equation:

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

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



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 $c(\boldsymbol{p}, \boldsymbol{d}) \in [0,1]^3$

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

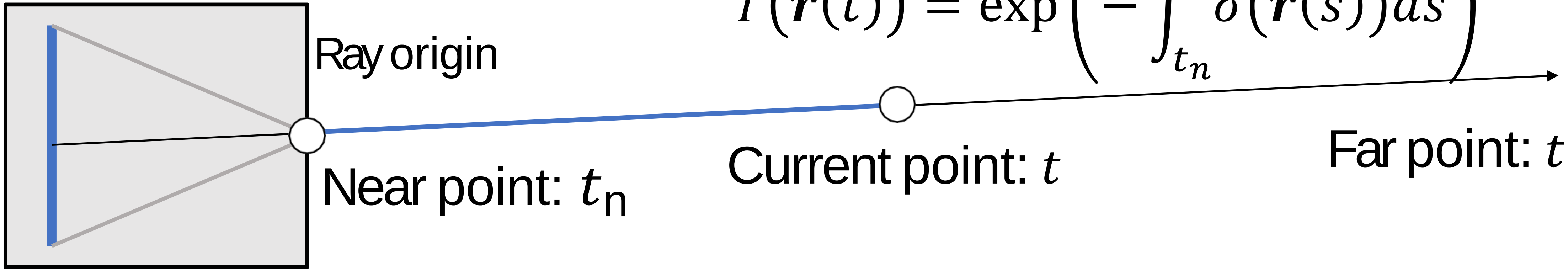
Compute transmittance by accumulating volume density up to current point

Neural Radiance Fields (NeRFs)

Color observed by the camera given by volume rendering equation:

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

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



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

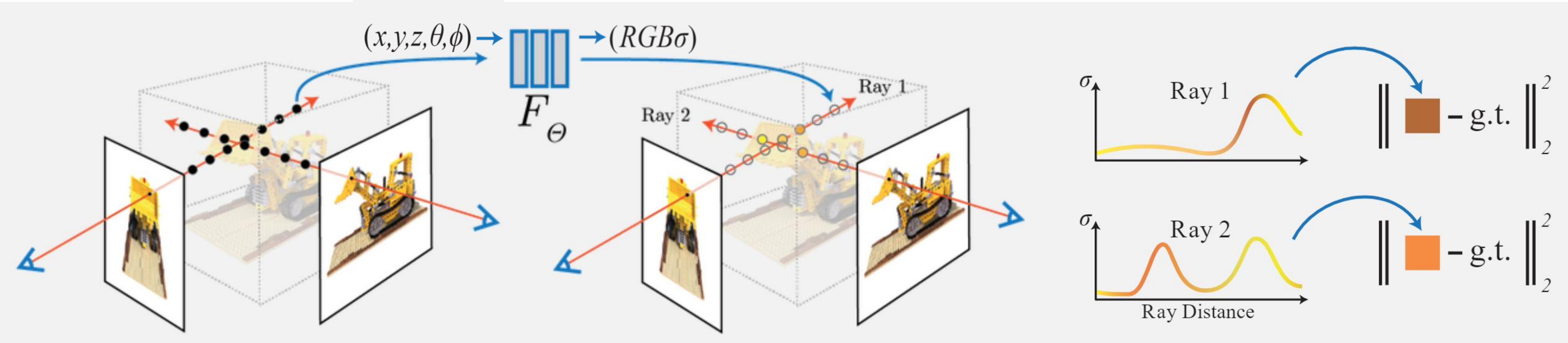
Volume Density is $\sigma(\boldsymbol{p}) \in [0,1]$

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

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

Neural Radiance Fields (NeRFs)

Fully-connected Network: Input Position p=x,y,z and Direction d=θ,Φ, and output Volume Density (σ) and RGB color



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

Neural Radiance Fields (NeRFs)

Neural Radiance Fields (NeRFs)

Some Pre-Trained Models (try out)

https://github.com/ByteDance-Seed/Depth-Anything-3

https://github.com/gangweix/pixel-perfect-depth

https://github.com/DepthAnything/Video-Depth-Anything

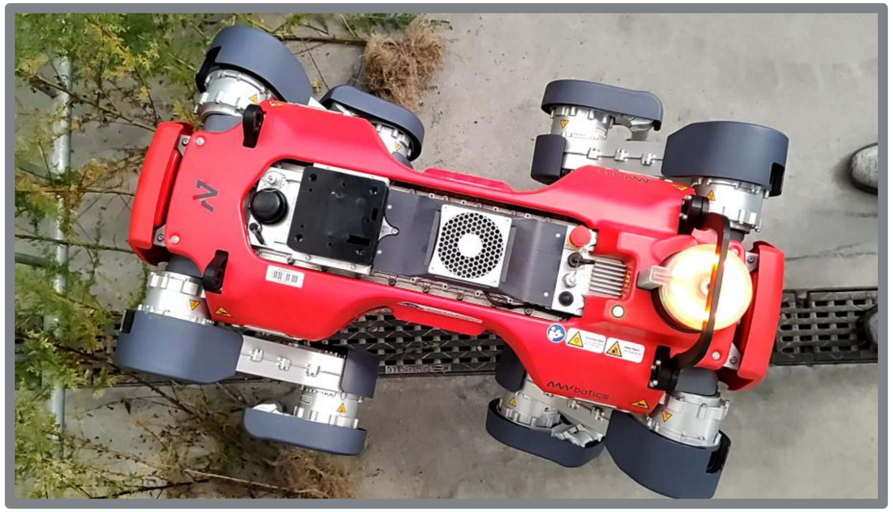
https://github.com/DepthAnything/Depth-Anything-V2

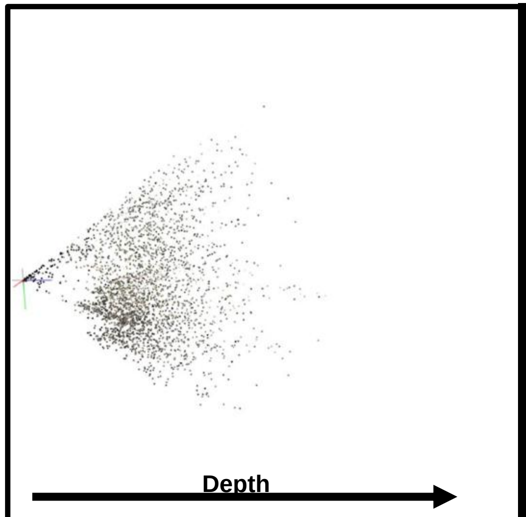
https://github.com/facebookresearch/vggt

https://github.com/NVlabs/instant-ngp

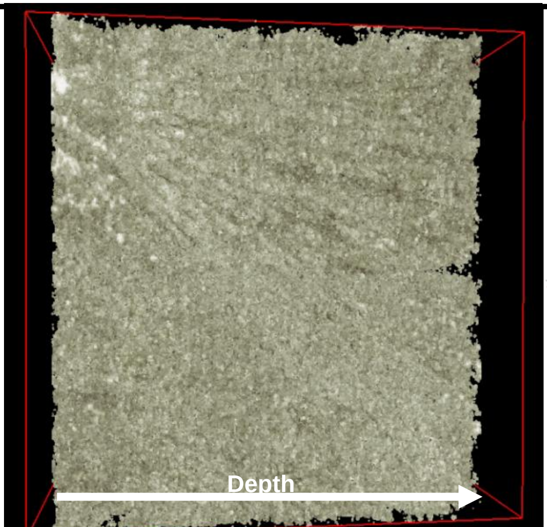
VGGT

Research Example: Occlusion, however...

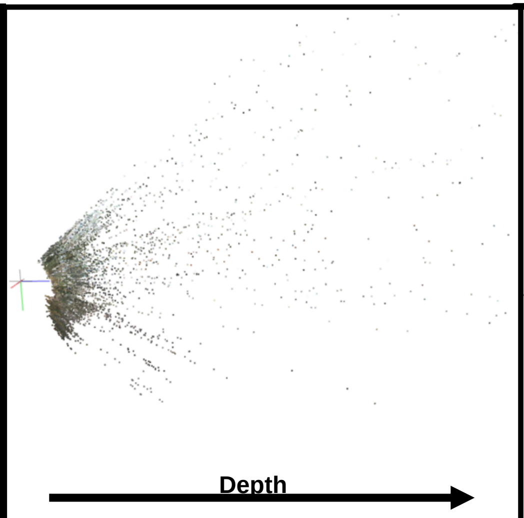




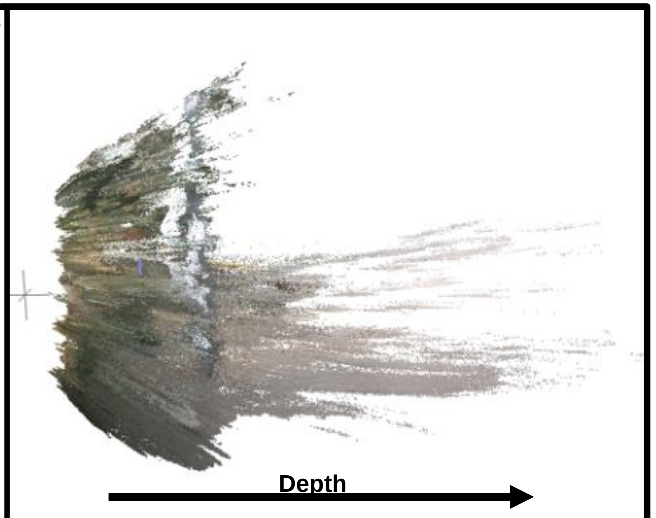
Structure-from-Motion



Neural Radiance Fields



Visual Geometry Group Transformer



Depth Anything 3

Course Overview

CW	Topic	Date	Place	Lab
41	Introduction and Course Overview	07.10.2025	Zoom	Lab 1
42	Capturing Digital Images	14.10.2025	Zoom	Lab 2
43	Digital Image Processing	21.10.2025	Zoom	Assignment 1
44	Machine Learning	28.10.2025	Zoom	
45	Feature Extraction	04.11.2025	Zoom	Open Lab 1
46	Segmentation	11.11.2025	Zoom	Assignment 2
47	Optical Flow	18.11.2025	Zoom	Open Lab 2
48	Object Detection	25.11.2025	Zoom	Assignment 3
49	Multi-View Geometry	02.12.2025	Zoom	Open Lab 3
50	3D Vision	10.12.2025	Zoom	Assignment 4
3	Trends in Computer Vision	13.01.2026	Zoom	
4	Q&A	20.01.2026	Zoom	Open Lab 4
5	Exam	27.01.2026	HS1 (Linz), S1/S3 (Vienna), S5 (Bregenz)	
9	Retry Exam	24.02.2026	tba	

Thank You

Happy Holidays