CS479: Machine Learning for 3D Data

Neural Rendering

LECTURE 8
MINHYUK SUNG

Spring 2025
KAIST

Course Road Map

Representations

- Point clouds
- Implicit representation
- **Multi-view images to 3D**
- Hybrid representations
- Meshes
- ~~CAD~~
- Representation Conversion

Applications

- 3D perception (Encoding)
- **Reconstruction (Decoding)**
- Manipulation
- ~~Generation~~

Novel View Synthesis

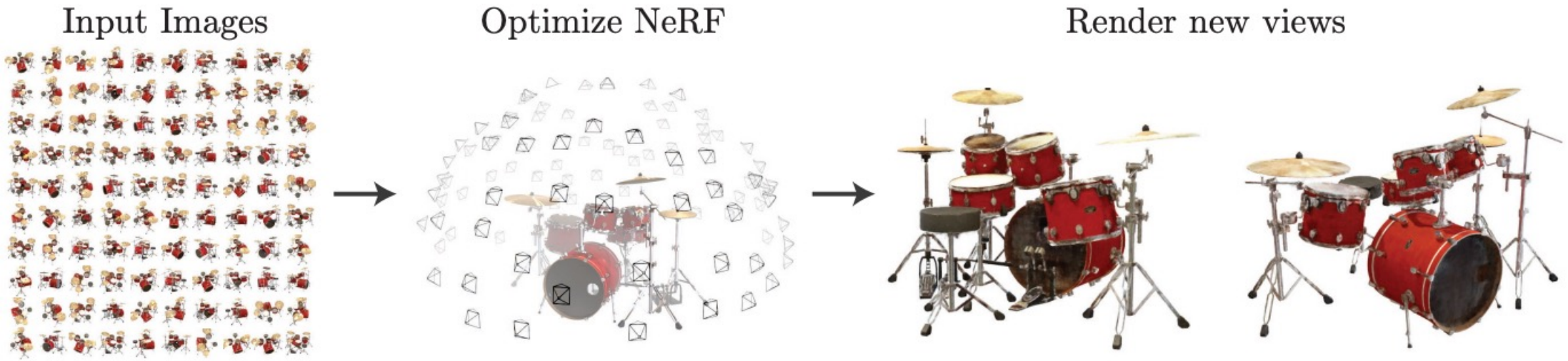
Input: Sparsely sampled images of scene.

Output: New views of the same scene.

Mildenhall et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020.

Novel View Synthesis

- 3D reconstruction + Novel view image synthesis.
- The focus is on synthesizing novel view images.
- Only images are given. No supervision for the 3D shape.



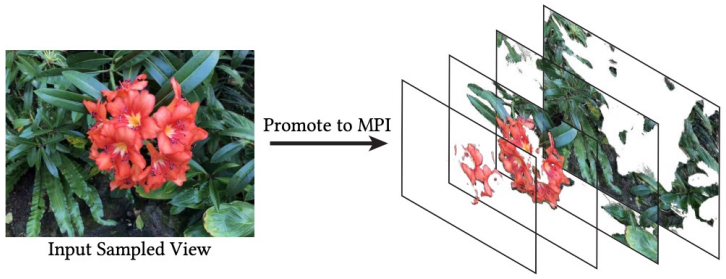
Mildenhall et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020.


https://twitter.com/nerfstudioteam
nerfstudio

Neural Novel View Synthesis – Literature

Soft 3D
Culmination of non-deep stereo matching techniques.
(Penner & Zhang 2017)

Multi-plane Image Methods
Typical deep learning pipelines - images go into a 3D CNN, big RGBA 3D volume comes out.
(Zhou et al. 2018)
(Srinivasan et al. 2019)
(Mildenhall et al. 2019)
(Flynn et al. 2019)
(Tucker & Snavely 2020)



Neural Volume
Direct gradient descent to optimize an RGBA volume, regularized by a 3D CNN (Lombardi et al. 2019)

Neural Novel View Synthesis

Training

- **Input**: A set of multi-view images of an object or scene, along with their corresponding camera matrices.
- **Output**: An implicit 3D representation of the object or scene
- **Loss Function**: L2 loss between the rendered image and the corresponding ground truth image for each view

Neural Novel View Synthesis

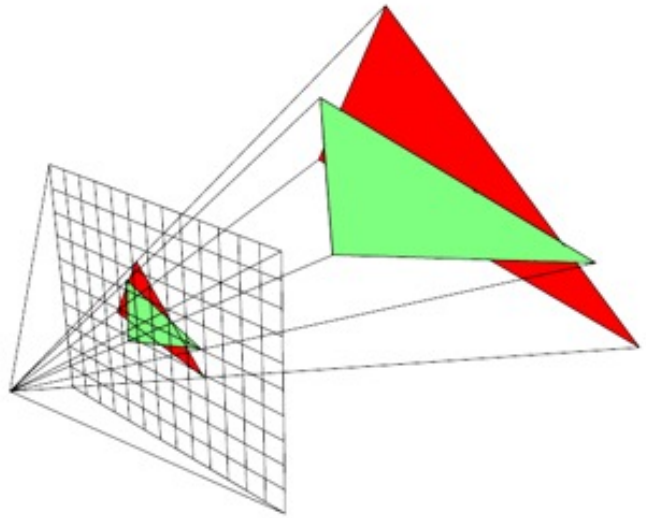
Inference

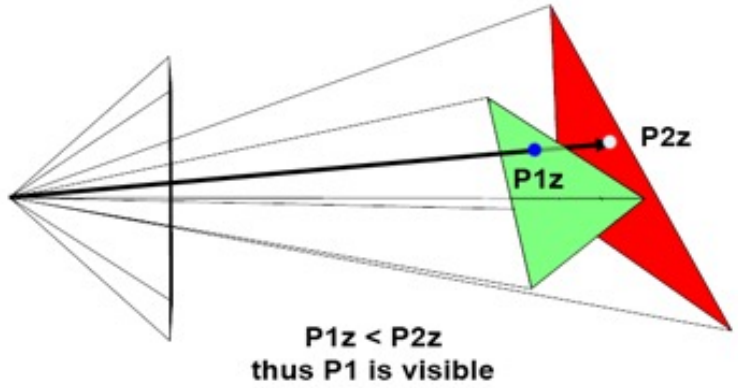
- **Input**: A camera matrix (not necessarily seen during training).
- **Output**: A 2D image rendered from the learned implicit 3D representation corresponding to the given viewpoint.

What is the rendering?

Rasterization – Object-Centric

1. Project each mesh face onto the 2D screen.
2. Interpolate vertex data across pixels within the projected face.
3. Skip pixels already processed with a closer depth; otherwise, write them.

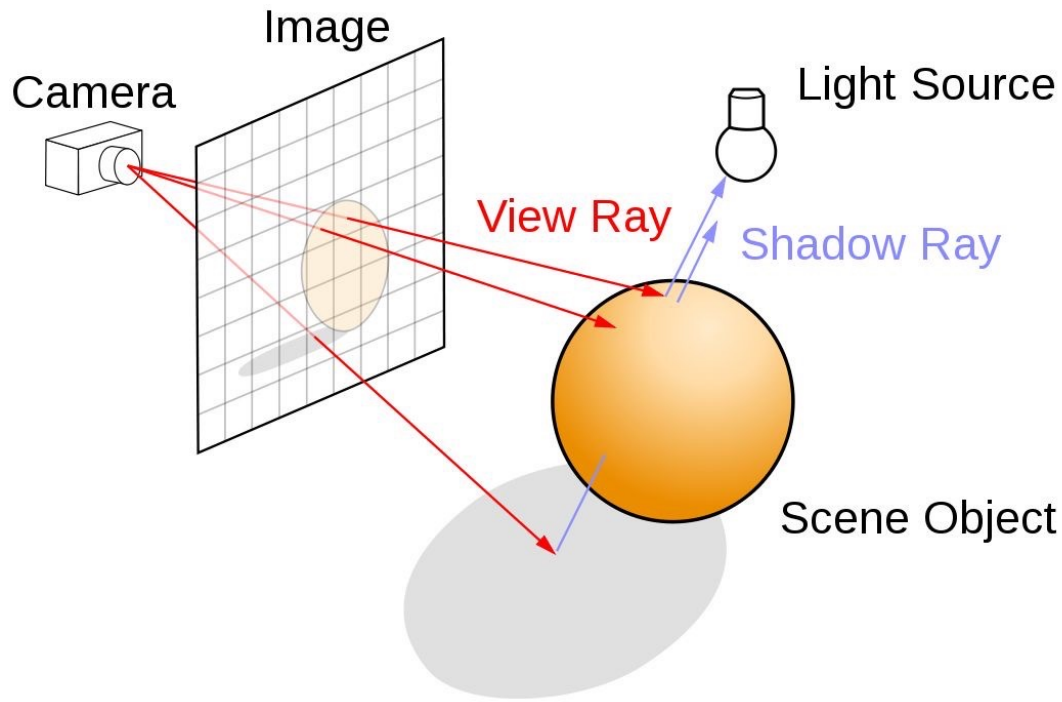




https://www.scratchapixel.com/

Ray Casting – Image-Centric

1. Shoot a ray from each pixel.
2. Find the first intersection point with the 3D object.
3. Retrieve the color information.



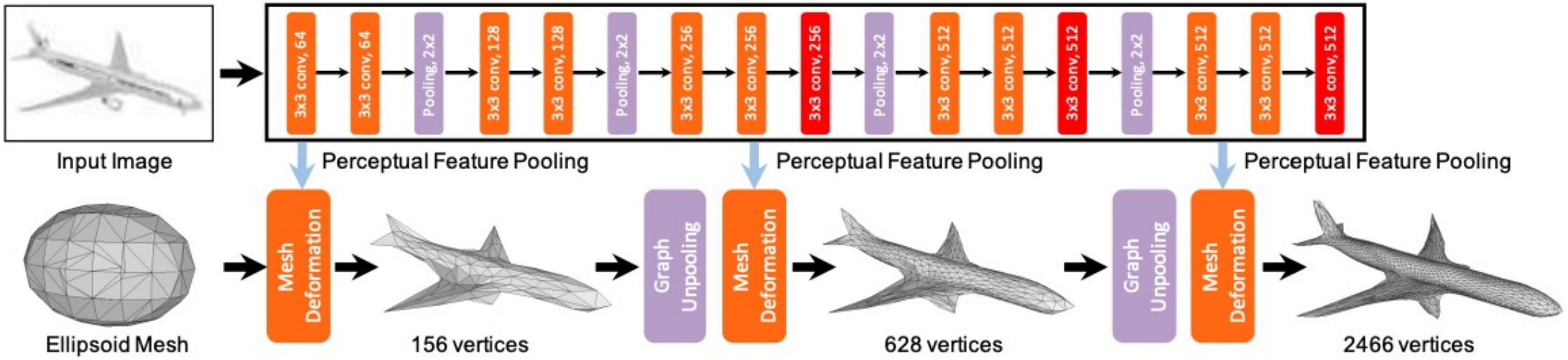
https://developer.nvidia.com/discover/ray-tracing

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

Multi-View to 3D Mesh

Can we reconstruct a 3D object/scene into a mesh?

Q. What if we deform a template mesh to align it with images from each viewpoint via rendering?



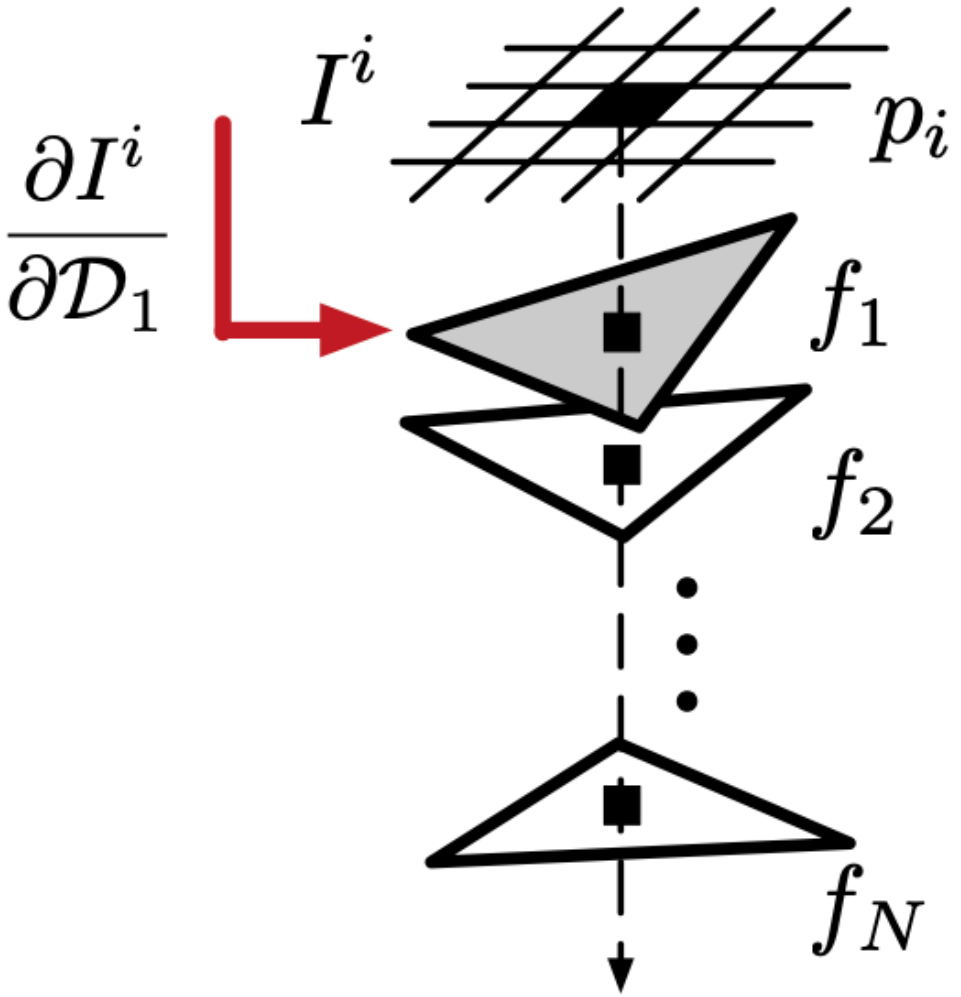
Wang et al., Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images, ECCV 2018.

Multi-View to 3D Mesh: Limitations

- Limited to generating shapes that have the same topology as the template shape.
- It is very challenging to generate a mesh with arbitrary topology directly.

Multi-View to 3D Mesh: Limitations

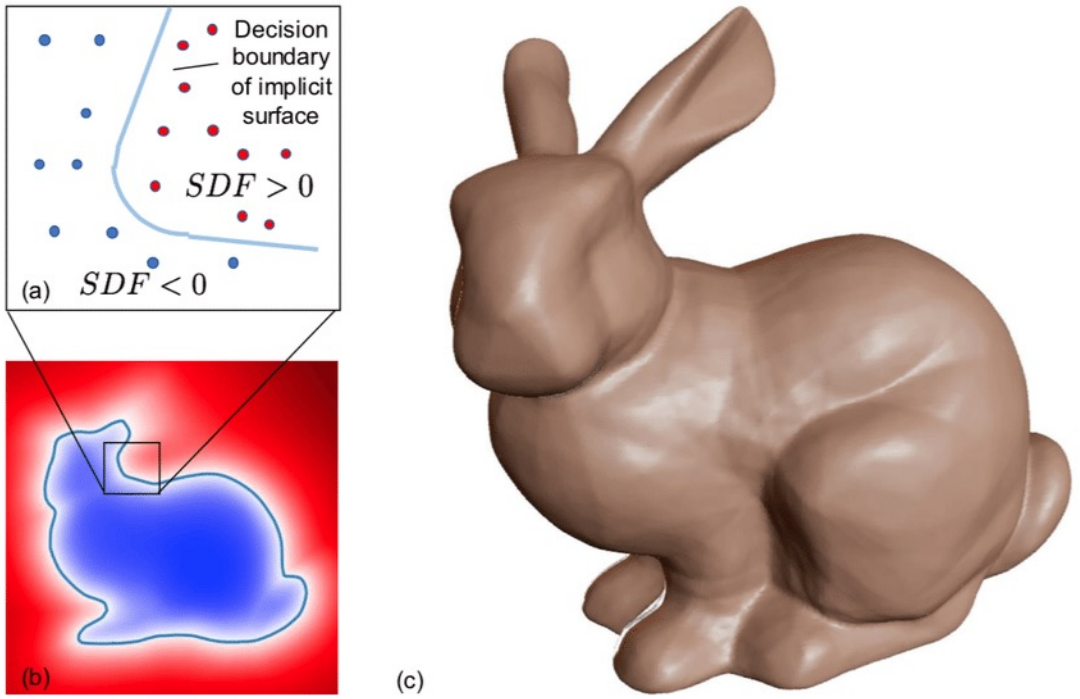
- It is also very slow to update the mesh since only the first intersection point is affected in the gradient descent.



Park et al, DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation, CVPR 2019.

Rendering with Implicit Functions

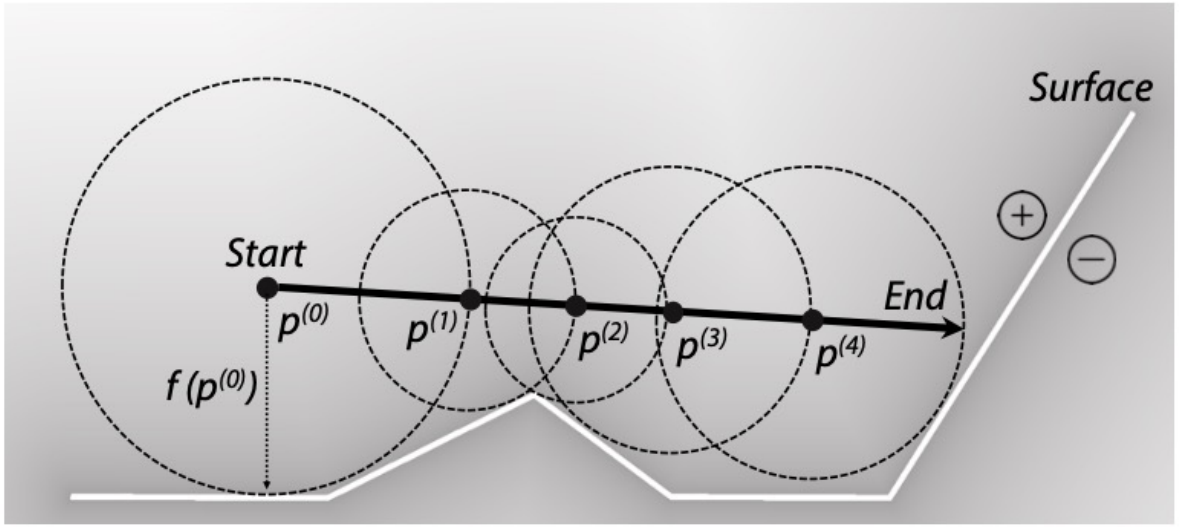
- Implicit representations (such as signed distance functions) are known to be the best for 3D reconstruction.
- But how can we render a signed distance function?



Liu et al., DIST: Rendering Deep Implicit Signed Distance Function with Differentiable Sphere Tracing, CVPR 2020.
John C. Hart, Sphere Tracing, 1995.

Sphere Tracing

- A technique to render an implicit surface with ray casting.
- Iterate advancing the ray by the current signed distance until the distance approaches (almost) zero.

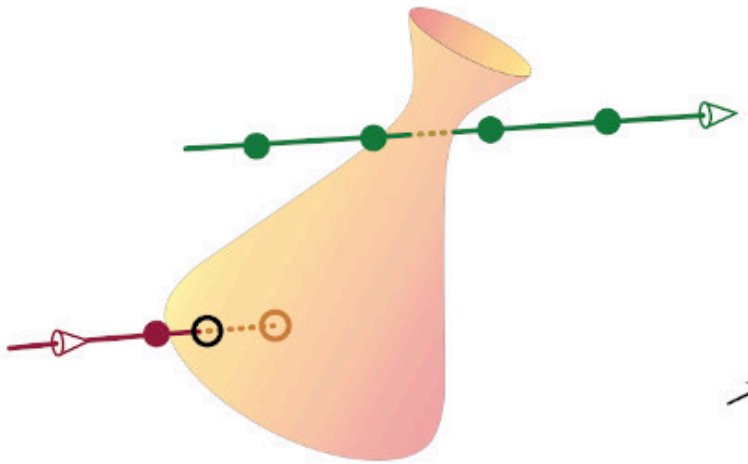


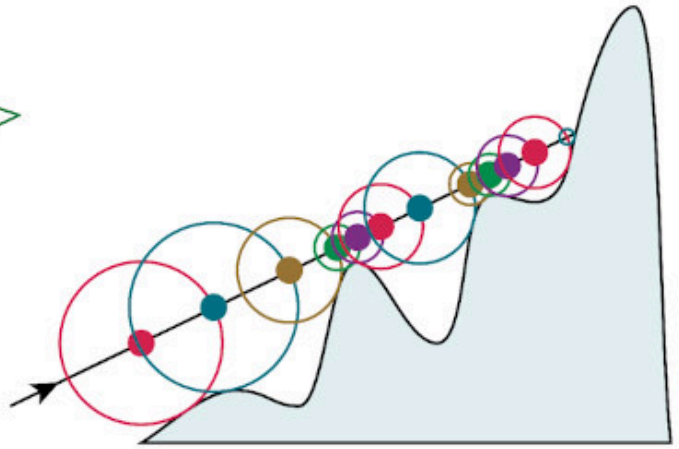
Algorithm 1 Naive sphere tracing algorithm for a camera ray $L : \mathbf{c} + d\tilde{\mathbf{v}}$ over a signed distance fields $f : \mathbb{N}^3 \rightarrow \mathbb{R}$.

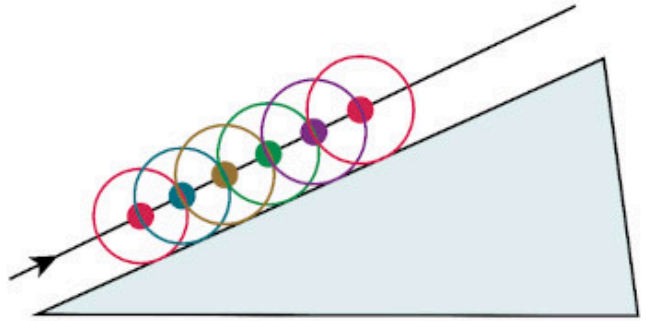
1: Initialize $n = 0$, $d^{(0)} = 0$, $\mathbf{p}^{(0)} = \mathbf{c}$.
2: **while** not converged **do**:
3: Take the corresponding SDF value $b^{(n)} = f(\mathbf{p}^{(n)})$ of the location $\mathbf{p}^{(n)}$ and make update: $d^{(n+1)} = d^{(n)} + b^{(n)}$.
4: $\mathbf{p}^{(n+1)} = \mathbf{c} + d^{(n+1)}\tilde{\mathbf{v}}$, $n = n + 1$.
5: Check convergence.
6: **end while**

Limitations of Sphere Tracing

- Not differentiable.
- Computationally expensive.
- Challenging to achieve convergence.





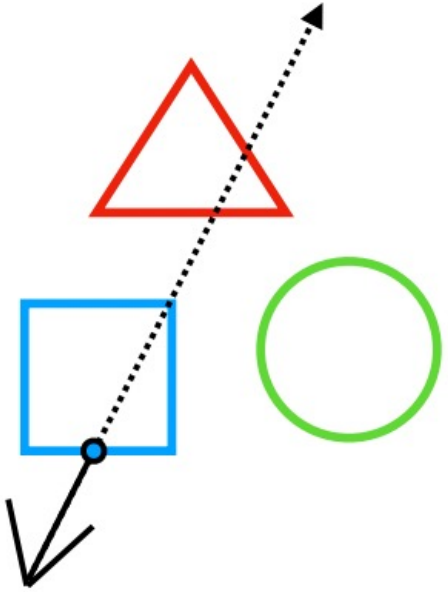


https://tovacinni.github.io/sdf-explorer/

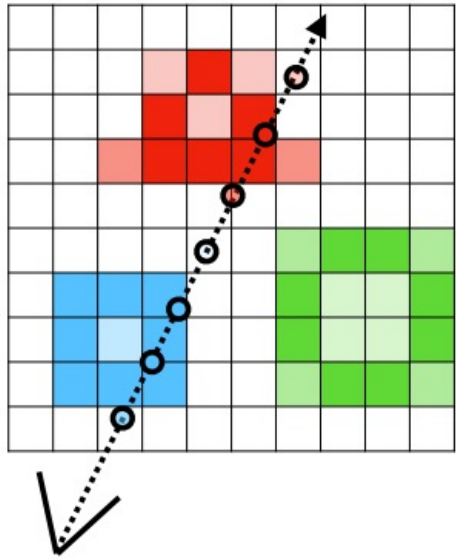
Solution – Volume Rendering

Violate the physical constraint!

Relax the condition that the shape is a solid object, and consider the shape as a volume.



Surface Rendering

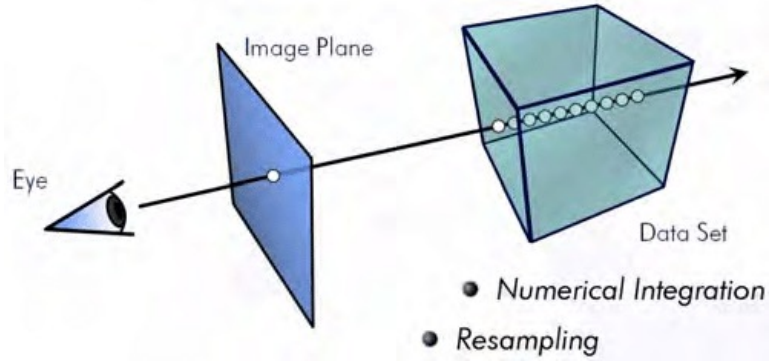


Volume Rendering

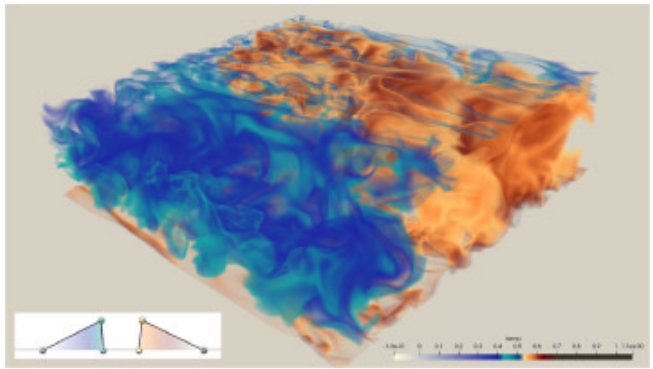
Benjamin Mildenhall, Neural Scene Representations for View Synthesis, PhD Dissertation.

Volume Rendering

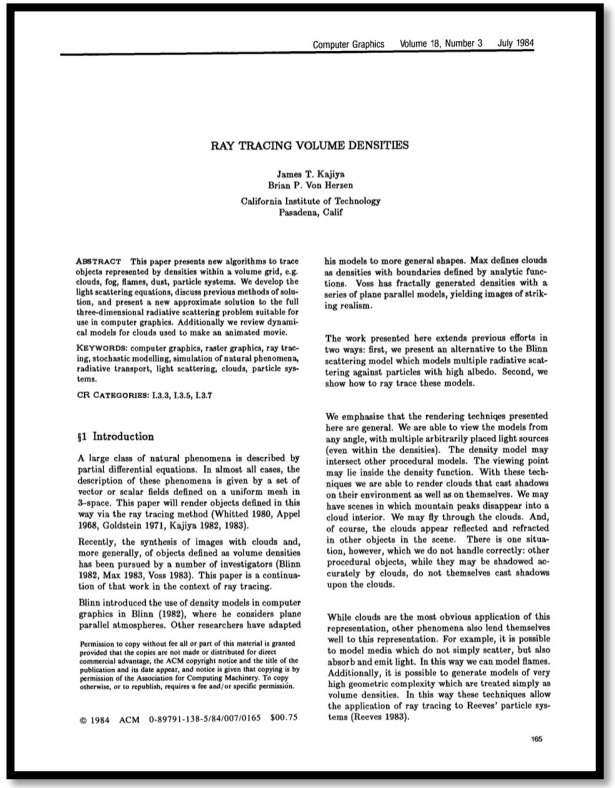
Has been widely studied and utilized for fluid and smoke simulation and rendering.



https://nuguziii.github.io/cg/CG-001/



https://www.sciencedirect.com/science/article/pii/S0167819119301280



Computer Graphics Volume 18, Number 3 July 1984

RAY TRACING VOLUME DENSITIES

James T. Kajiya
Brian P. Von Herzen
California Institute of Technology
Pasadena, Calif

ABSTRACT This paper presents new algorithms to trace objects represented by densities within a volume grid, e.g. clouds, fog, flames, dust, particle systems. We develop the light scattering equations, discuss previous methods of solution, and present a new approximate solution to the full three-dimensional radiative scattering problem suitable for use in computer graphics. Additionally we review dynamical models for clouds used to make an animated movie.

KEYWORDS: computer graphics, raster graphics, ray tracing, stochastic modelling, simulation of natural phenomena, radiative transport, light scattering, clouds, particle systems.

CR CATEGORIES: I.3.3, I.3.5, I.3.7

§1 Introduction

A large class of natural phenomena is described by partial differential equations. In almost all cases, the description of these phenomena is given by a set of vector or scalar fields defined on a uniform mesh in 3-space. This paper will render objects defined in this way via the ray tracing method (Whitted 1980, Appel 1968, Goldstein 1971, Kajiya 1982, 1983).

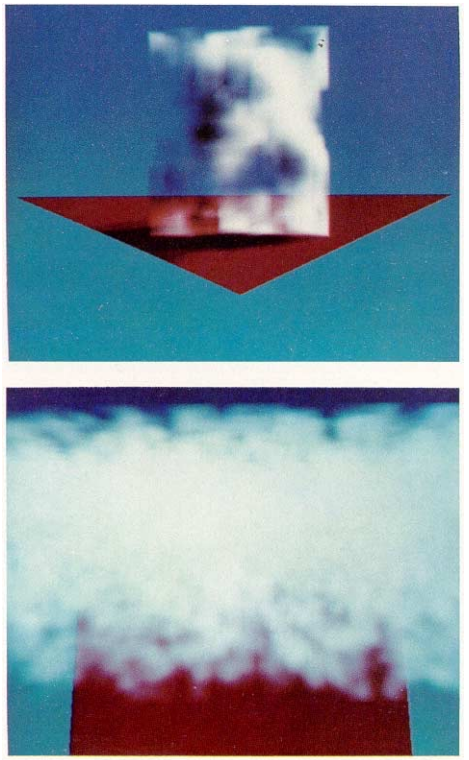
Recently, the synthesis of images with clouds and, more generally, of objects defined as volume densities has been pursued by a number of investigators (Blinn 1982, Max 1983, Voss 1983). This paper is a continuation of that work in the context of ray tracing.

Blinn introduced the use of density models in computer graphics in Blinn (1982), where he considers plane parallel atmospheres. Other researchers have adapted his models to more general shapes. Max defines clouds as densities with boundaries defined by analytic functions. Voss has fractally generated densities with a series of plane parallel models, yielding images of striking realism.

The work presented here extends previous efforts in two ways: first, we present an alternative to the Blinn scattering model which models multiple radiative scattering against particles with high albedo. Second, we show how to ray trace these models.

We emphasize that the rendering techniques presented here are general. We are able to view the models from any angle, with multiple arbitrarily placed light sources (even within the densities). The density model may intersect other procedural models. The viewing point may lie inside the density function. With these techniques we are able to render clouds that cast shadows on their environment as well as on themselves. We may have scenes in which mountain peaks disappear into a cloud interior. We may fly through the clouds. And, of course, the clouds appear reflected and refracted in other objects in the scene. There is one situation, however, which we do not handle correctly: other procedural objects, while they may be shadowed accurately by clouds, do not themselves cast shadows upon the clouds.

While clouds are the most obvious application of this representation, other phenomena also lend themselves well to this representation. For example, it is possible to model media which do not simply scatter, but also absorb and emit light. In this way we can model flames. Additionally, it is possible to generate models of very high geometric complexity which are treated simply as volume densities. In this way these techniques allow the application of ray tracing to Reeves' particle systems (Reeves 1983).



James T. Kajiya, Ray Tracing Volume Densities, SIGGRAPH 1984.

DeepVoxels

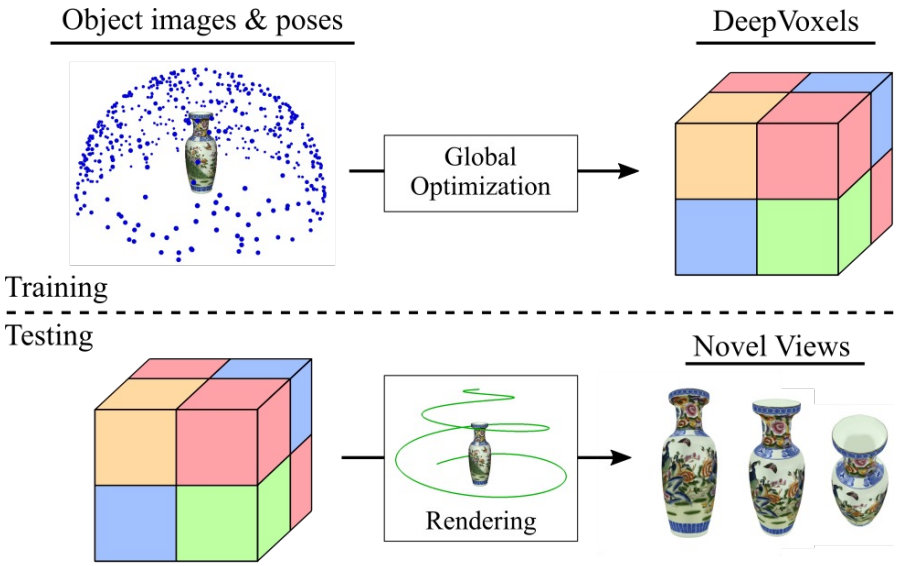
Sitzmann et al., DeepVoxels: Learning Persistent 3D Feature Embeddings, CVPR 2019.

DeepVoxels

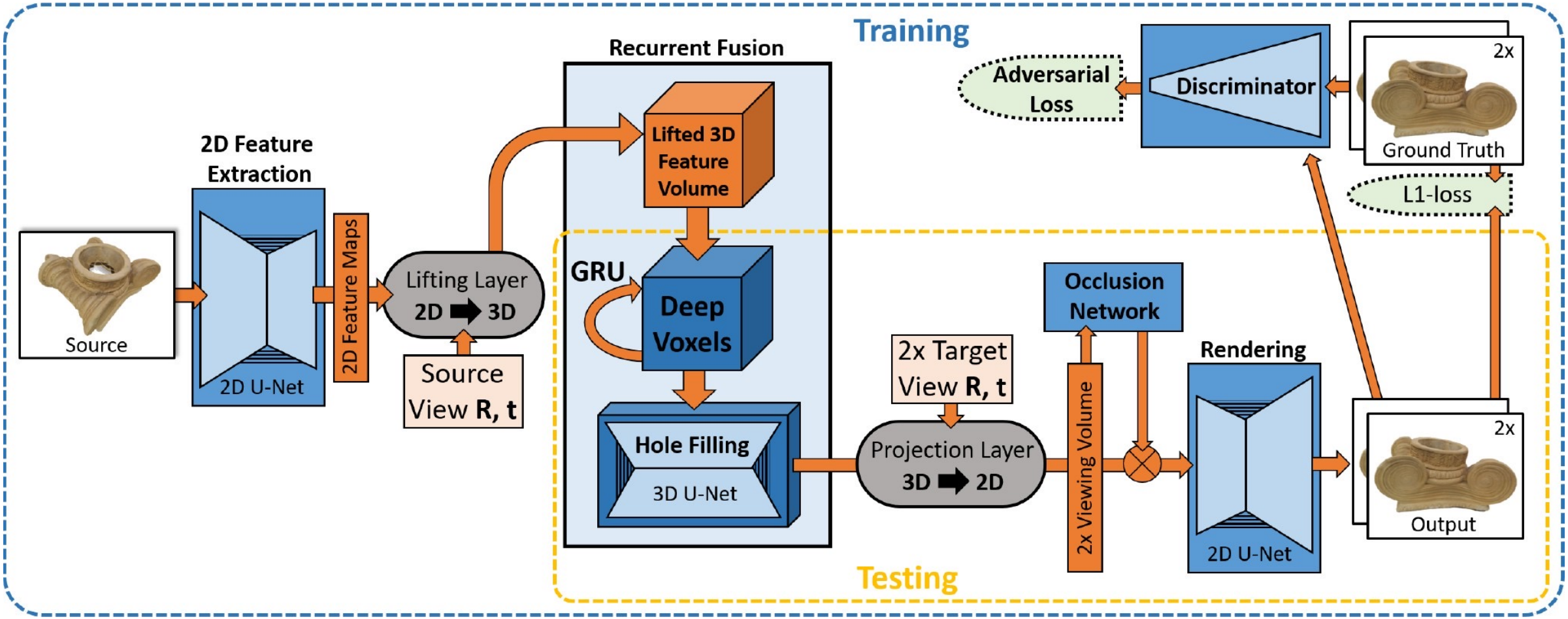
Encode the geometry information in 3D voxels.

Input: N images with camera extrinsic/intrinsic parameters.

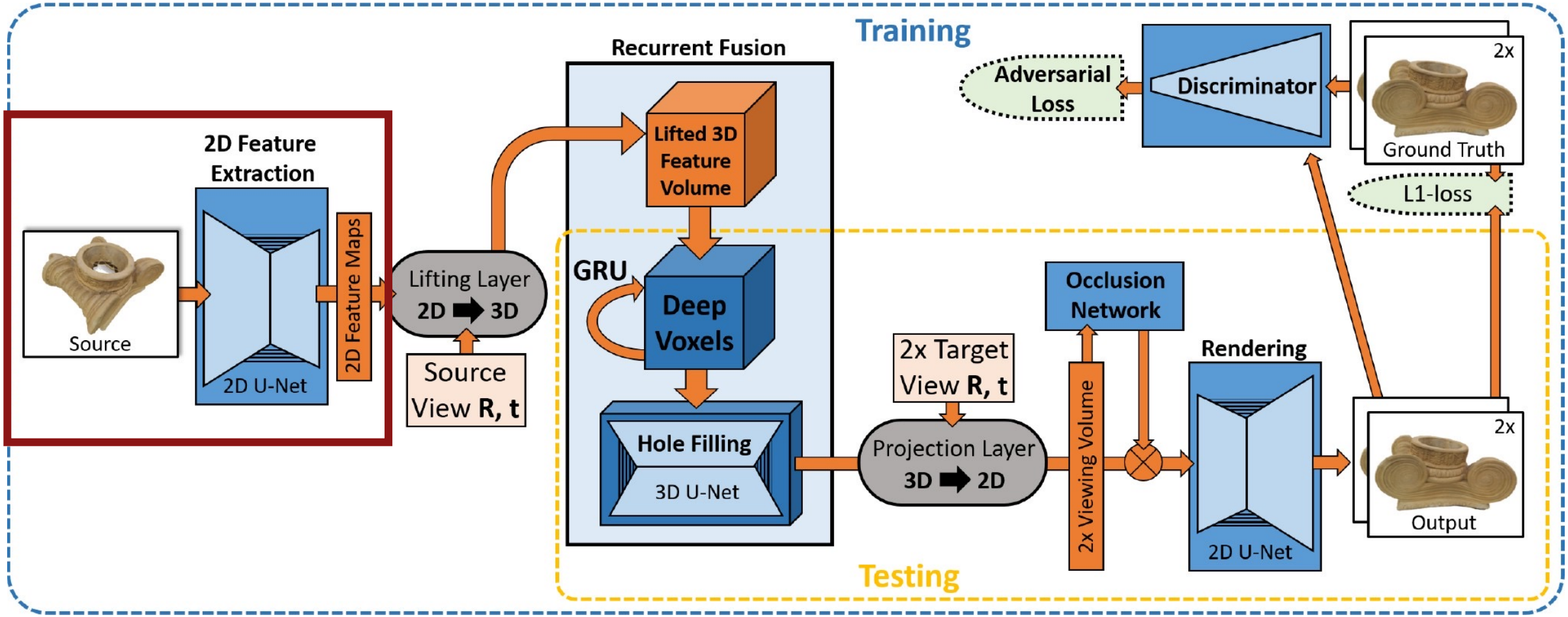
No supervision of the 3D shape.



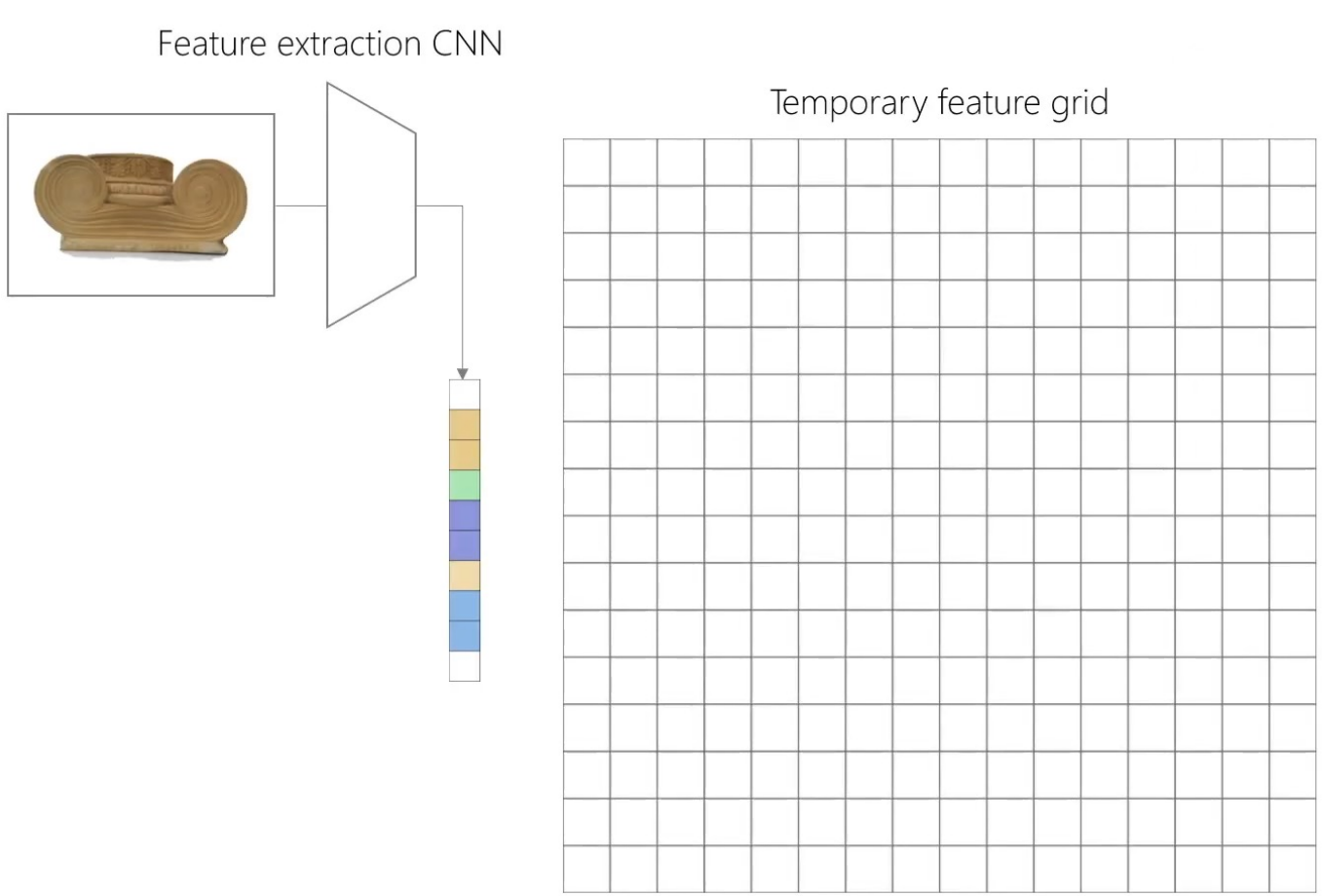
Framework


Training
Recurrent Fusion
Adversarial Loss
Discriminator
2x
Ground Truth
L1-loss
2D Feature Extraction
Lifted 3D Feature Volume
Source
2D U-Net
2D Feature Maps
Lifting Layer 2D ➡ 3D
Source View R, t
GRU
Deep Voxels
Hole Filling
3D U-Net
2x Target View R, t
Projection Layer 3D ➡ 2D
Occlusion Network
2x Viewing Volume
Rendering
2D U-Net
2x
Output
Testing

2D Feature Extraction



2D Feature Extraction

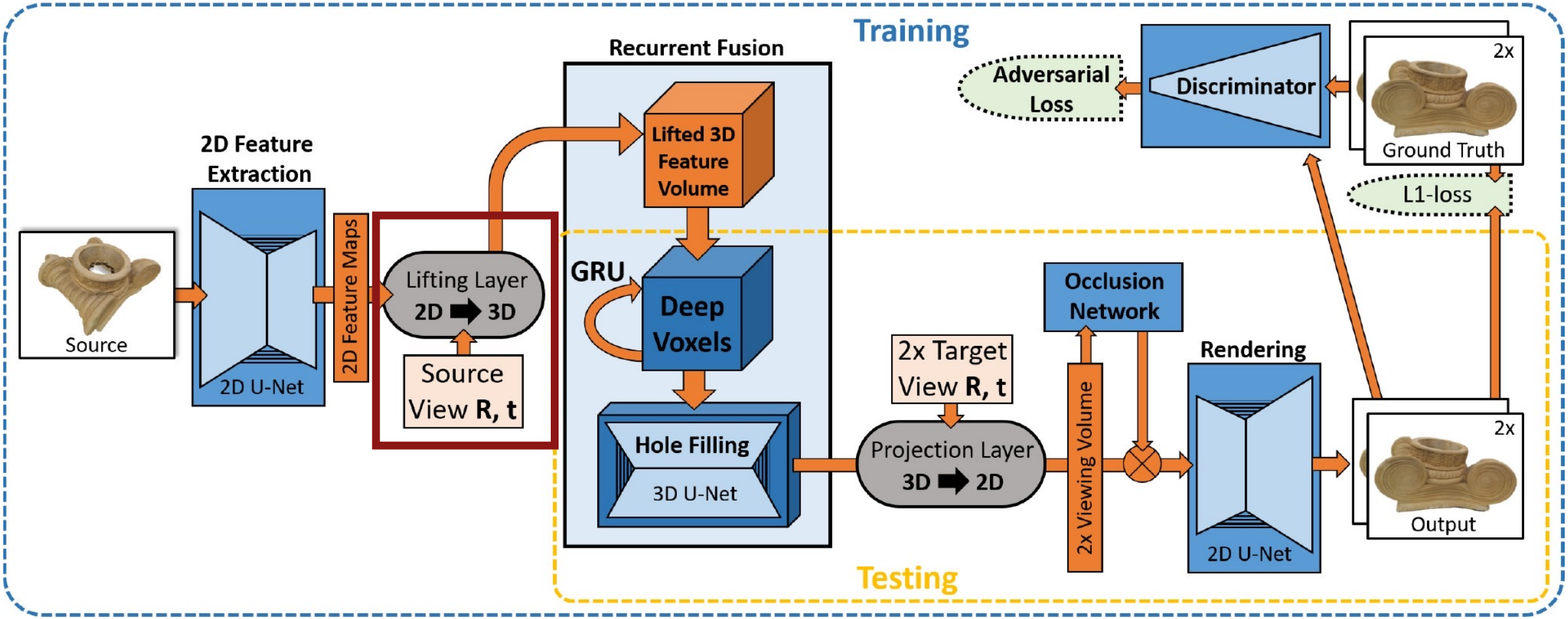


1D-to-2D Diagram

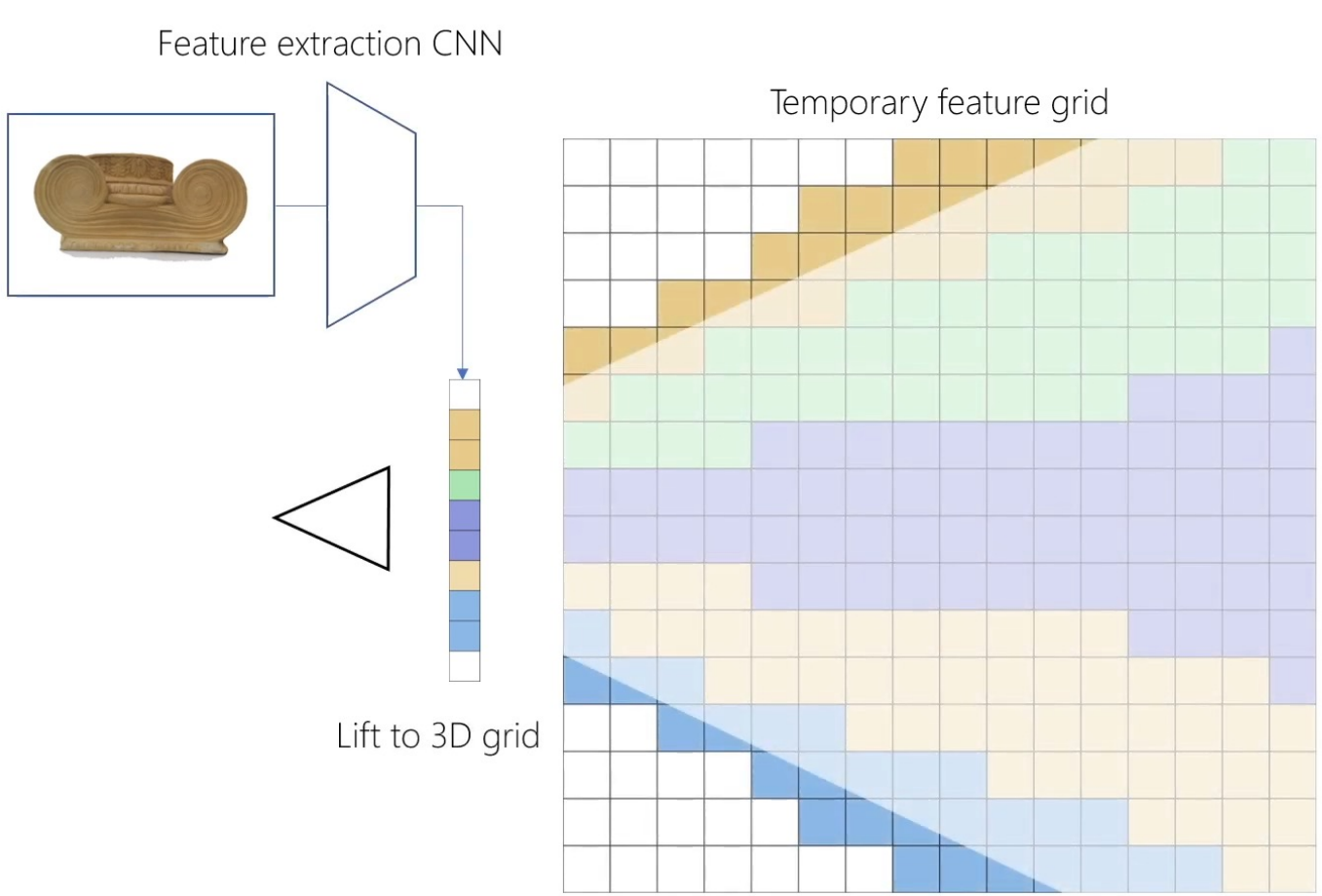
Image: 1D

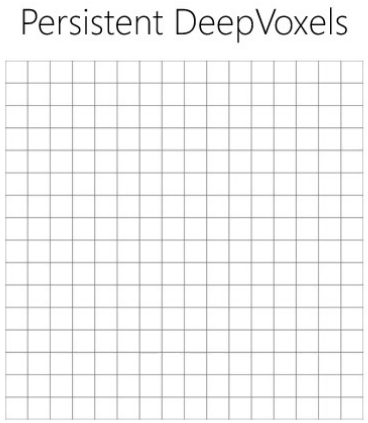
Reconstruction: 2D

2D-to-3D Lifting



2D-to-3D Lifting



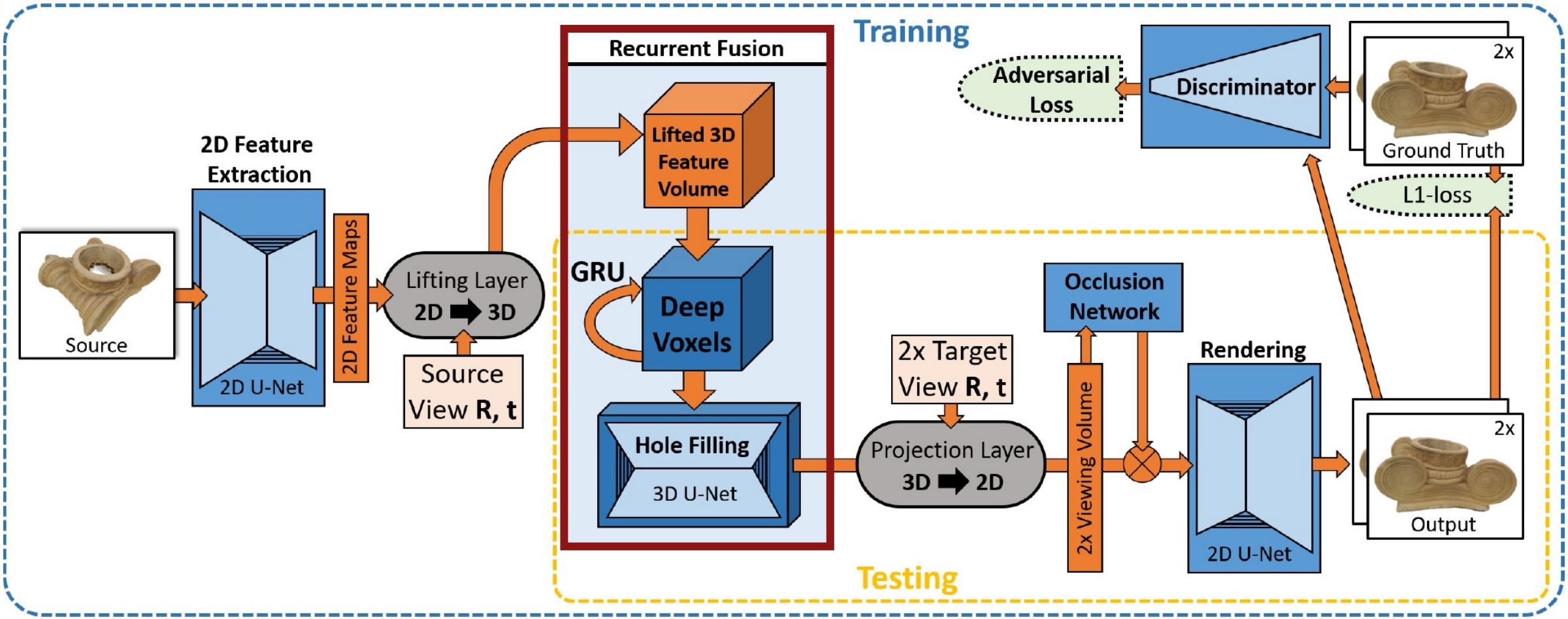


1D-to-2D Diagram

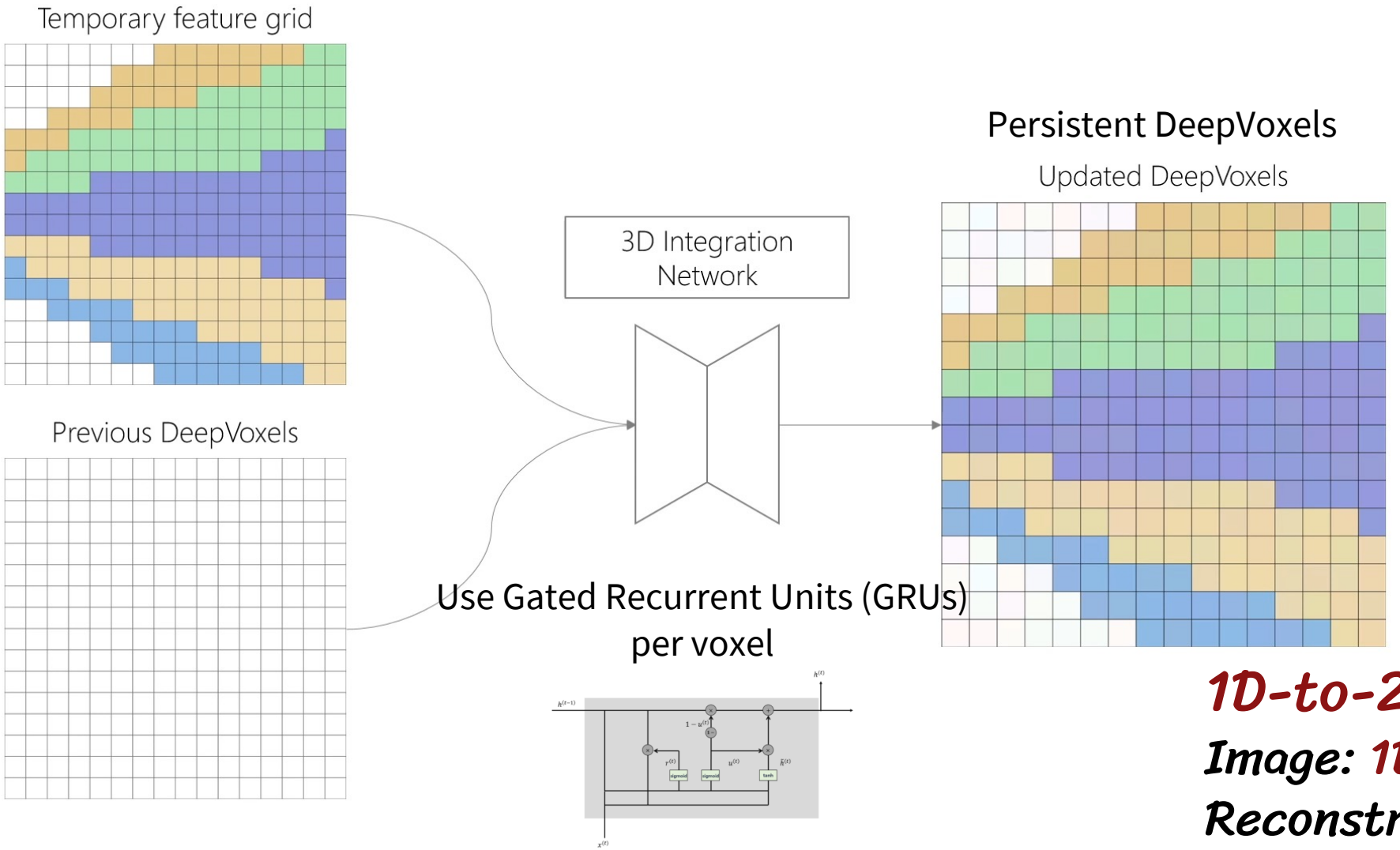
Image: 1D

Reconstruction: 2D

Deep Voxels Fusion



Deep Voxels Fusion

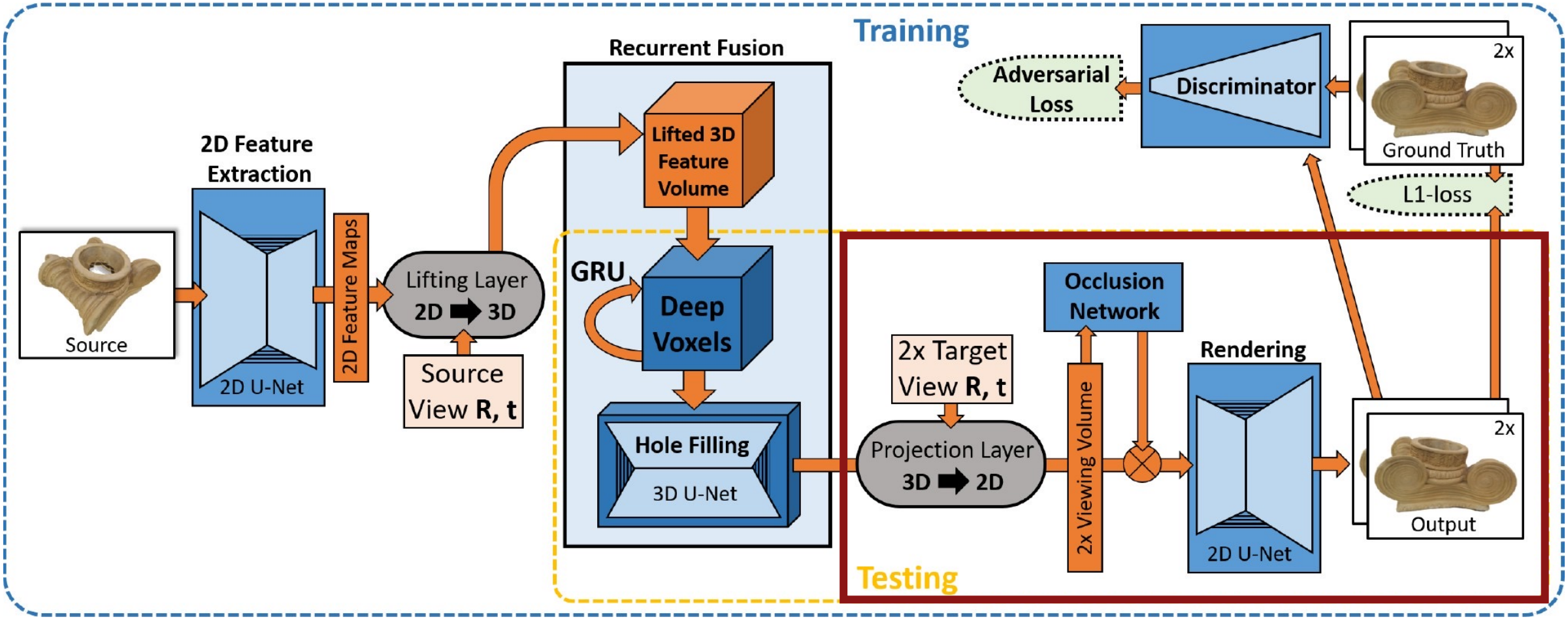


1D-to-2D Diagram

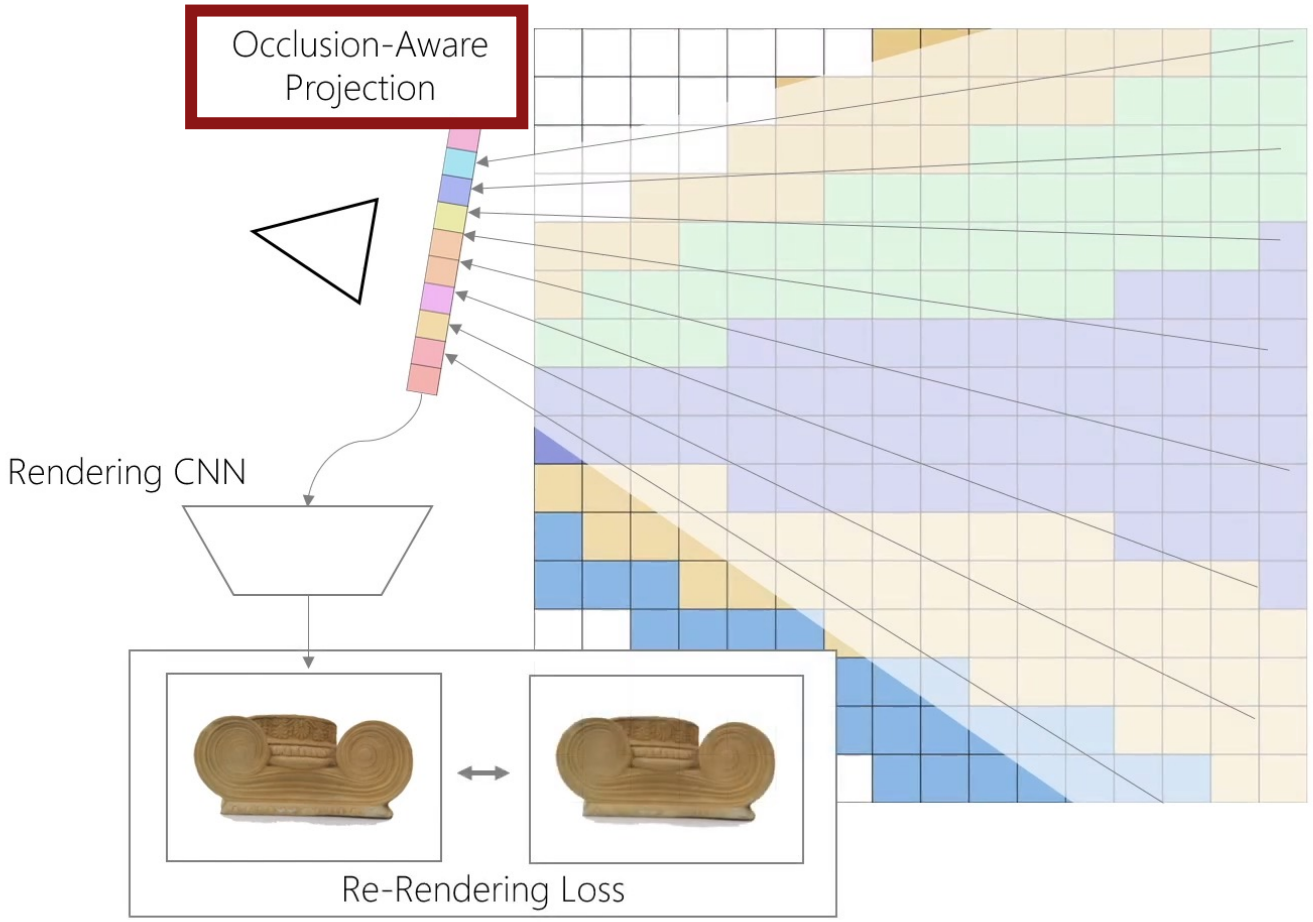
Image: 1D

Reconstruction: 2D

Rendering


Training
Recurrent Fusion
Adversarial Loss
Discriminator
2x
Ground Truth
L1-loss
2D Feature Extraction
Lifted 3D Feature Volume
Source
2D U-Net
2D Feature Maps
Lifting Layer 2D ➡ 3D
Source View R, t
GRU
Deep Voxels
Hole Filling
3D U-Net
Occlusion Network
2x Target View R, t
Projection Layer 3D ➡ 2D
2x Viewing Volume
Rendering
2D U-Net
2x
Output
Testing

Rendering

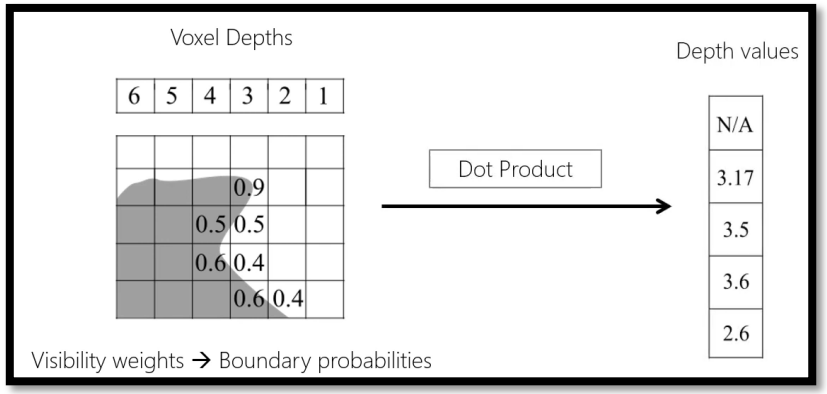


1D-to-2D Diagram

Image: 1D

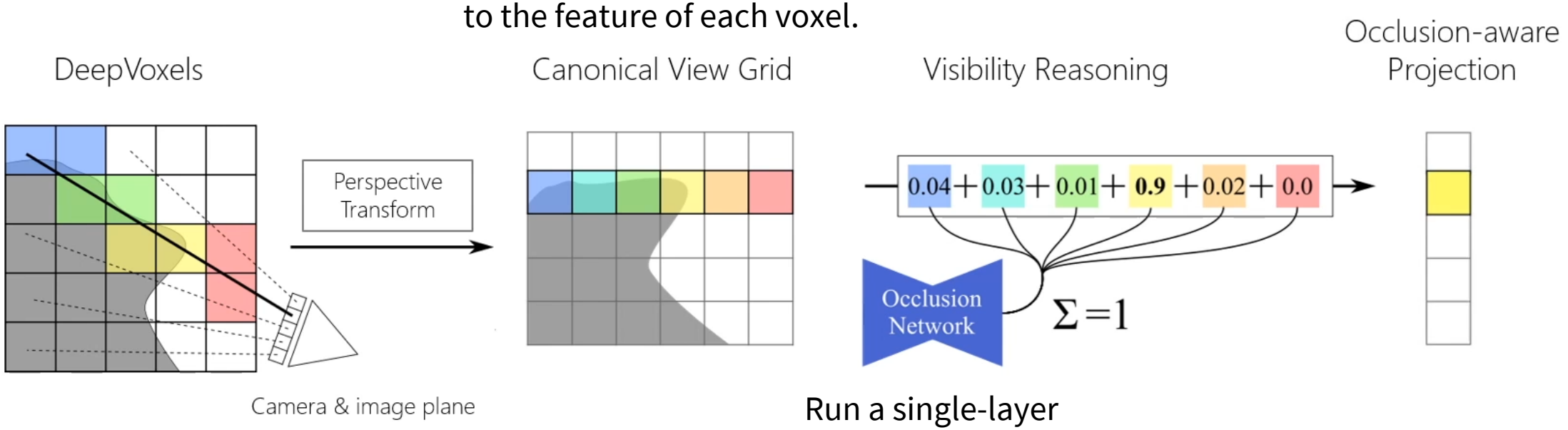
Reconstruction: 2D

Occlusion-Aware Projection



Depth prediction is possible!

Add camera-to-voxel distance
to the feature of each voxel.



Uniformly sample points
along each ray and
trilinearly interpolate features.

Run a single-layer
3D convolution network
to predict visibility per voxel

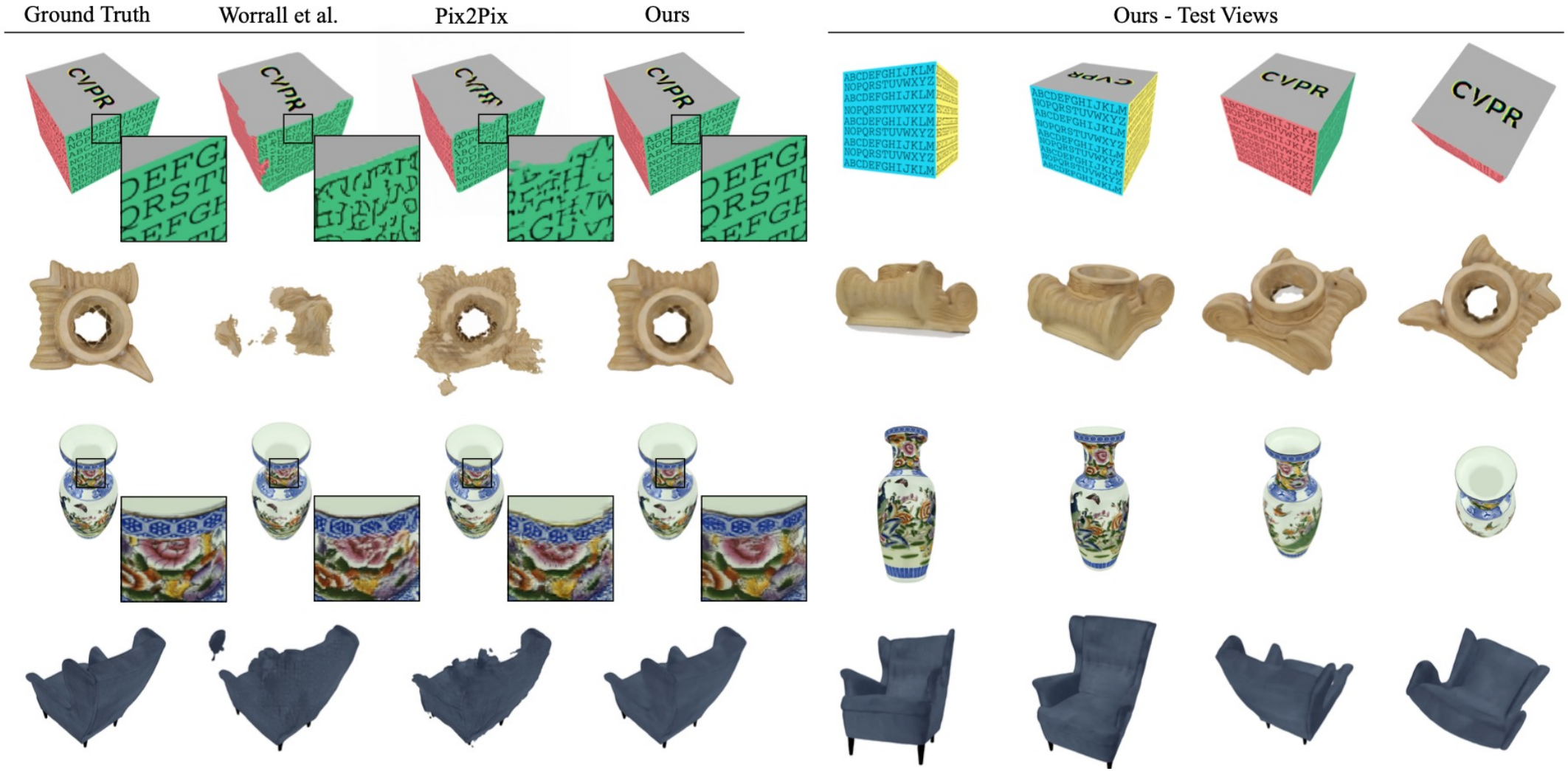
1D-to-2D Diagram

Image: 1D

Reconstruction: 2D

Novel View Synthesis Results

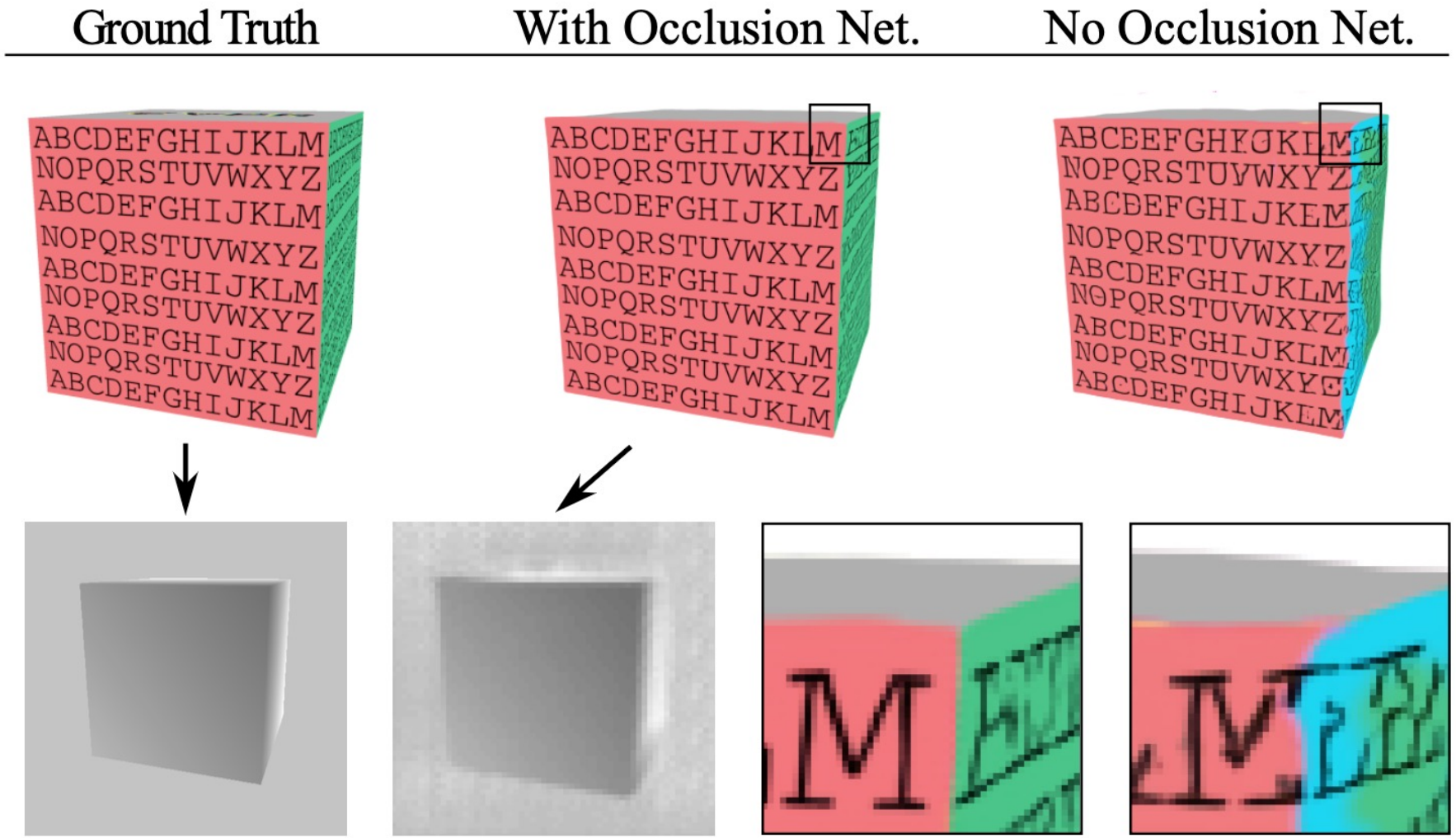
Comparison with Baselines



DeepVoxels:
Learning Persistent 3D Feature Embeddings




Ablation Study – Occlusion Reasoning



Deep Voxels – Summary

- Encoding appearance information into the latent features of 3D voxels.
- No 3D geometry supervision is needed.
- Occlusion reasoning leads to better results and generalization to novel viewpoints.

DeepVoxels – Limitations

- Inherits the limitations of using 3D voxels:
  - Memory-heavy
  - Cannot increase the resolution

- Occlusion-aware projection is not physically correct.

Q. Why? Any example?

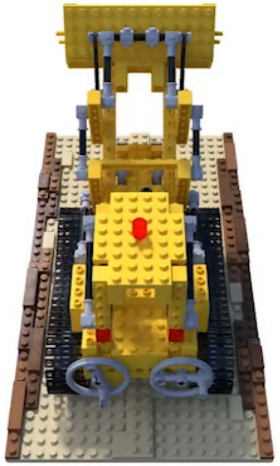
Neural Radiance Fields (NeRF)

Mildenhall et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020.

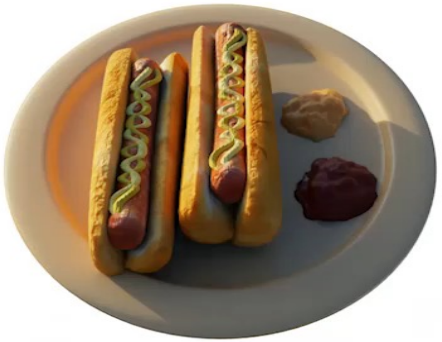
Main Differences from DeepVoxels

- Voxels → Implicit function.
- Occlusion-aware projection → Classic volume rendering.
- Latent features → View-dependent colors.

Sample Results





Radiance Field

Definition: The radiance (luminance) is the the power emitted, reflected, transmitted or received by a surface, per unit solid angle, per unit projected area.

Roughly speaking, view-dependent color information.

Q. Why?

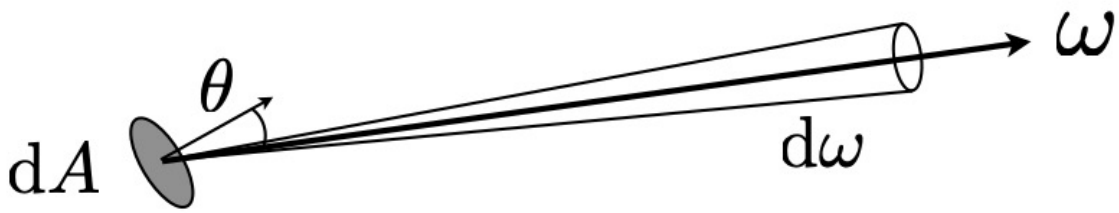
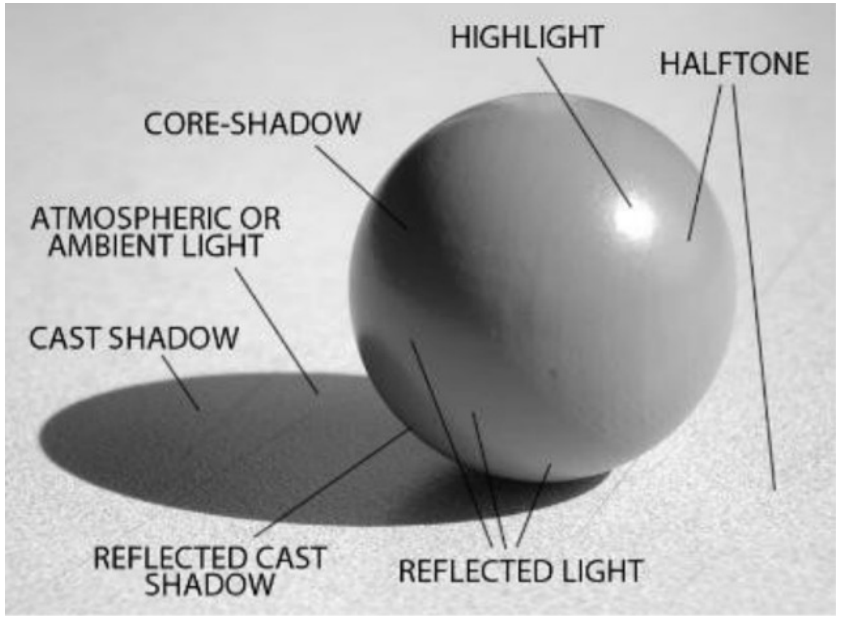


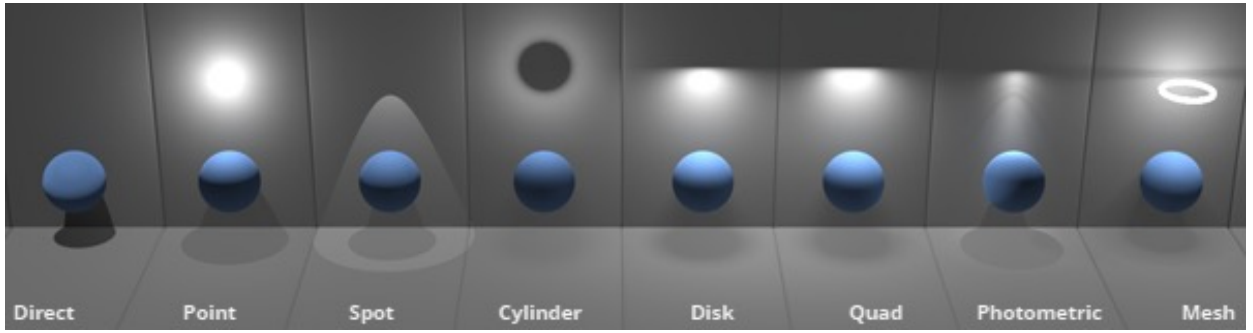
Image Credit: Ren Ng

Radiance Field

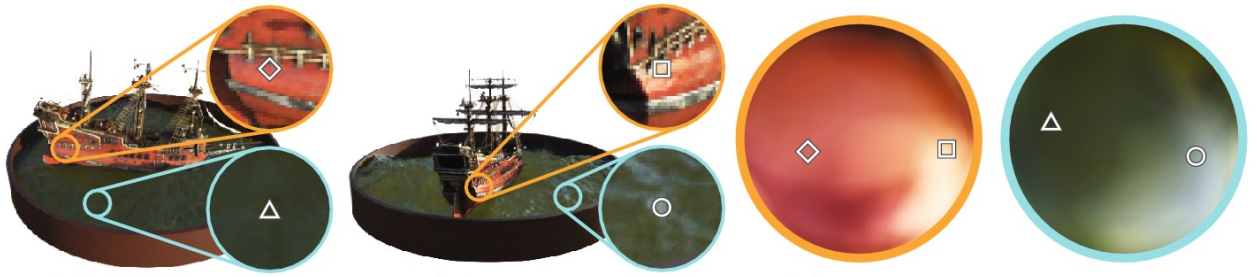
Color can be changed depending on the lighting condition, view direction, material property, and environment.



https://courses.byui.edu/art110_new/art110/glossary/Images/Light.png



https://docs.arnoldrenderer.com/display/A5AFMUG/Lights



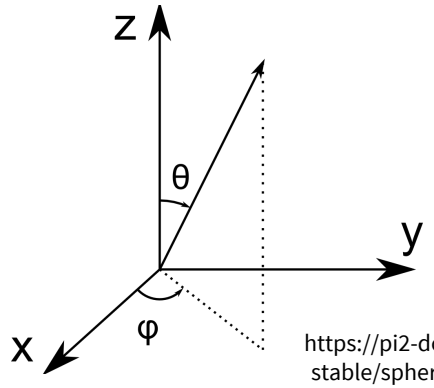
(a) View 1 (b) View 2 (c) Radiance Distributions

Implicit Functions

Learn two implicit functions using neural networks:

- **Radiance map $\mathbf{c}(\mathbf{x}, \mathbf{d})$:** $\mathbb{R}^5 \rightarrow \mathbb{R}^3$
  - Input: XYZ 3D location $\mathbf{x} \in \mathbb{R}^3$ and viewing direction $\mathbf{d} \in \mathbb{R}^2$
  - Output: RGB color

- **Opacity (Density) map $\sigma(\mathbf{x})$:** $\mathbb{R}^3 \rightarrow \mathbb{R}_{\geq 0}$
  - Input: XYZ 3D location $\mathbf{x} \in \mathbb{R}^3$
  - Output: Opacity



https://pi2-docs.readthedocs.io/en/stable/spherical_coordinates.html

Volume Rendering

The color of the ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is computed as:

$$C(\mathbf{r}) = \int_0^{t_{max}} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,,$$

Networks

$$\text{where } T(t) = \exp\left(-\int_0^t \sigma(\mathbf{r}(s))ds\right)$$

Transparency:
How likely the ray is to NOT be terminated until time t.

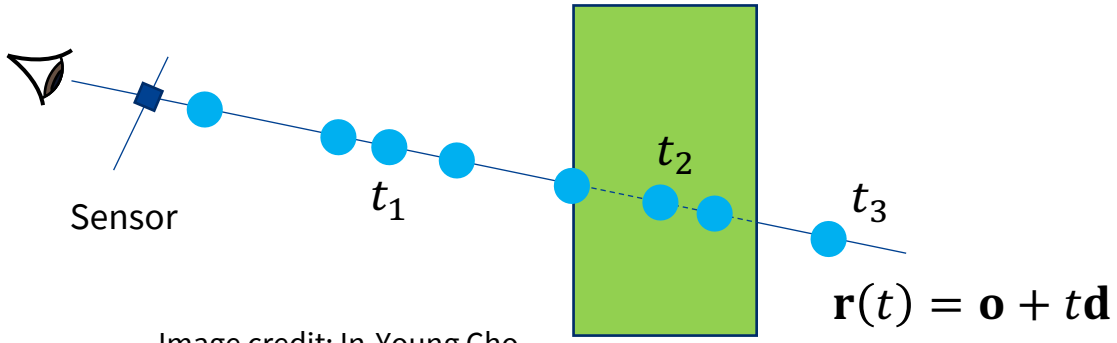


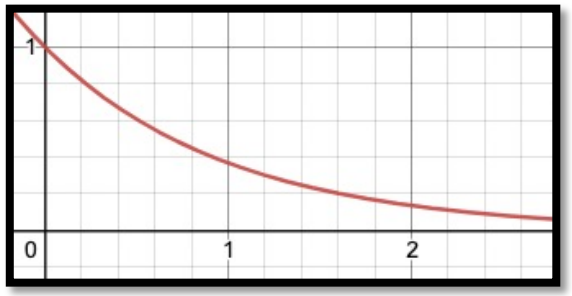
Image credit: In-Young Cho

Volume Rendering

- The transparency function $T(t)$ is defined as a non-increasing function.
- It aligns with the physical concept that transparency cannot be increased over time along a ray.

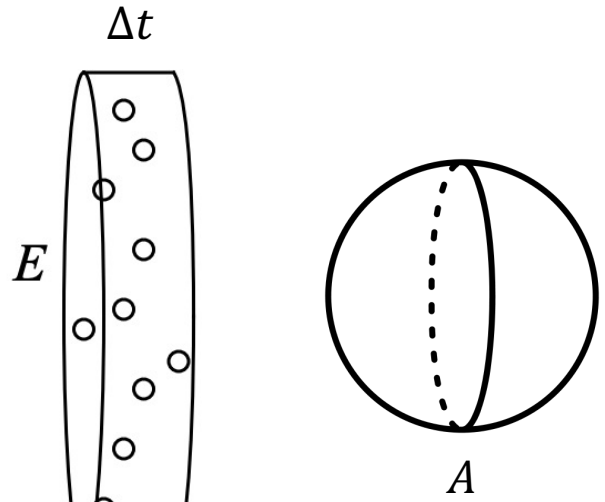
$$T(t) = \exp\left(-\int_0^t \sigma(\mathbf{r}(s))ds\right)$$

$y = \exp(-x)$



Volume Rendering

- $\rho(t)$: Particle density.
- $\rho(t)\mathrm{E}\Delta t$: Number of particles.
- A: Projected particle area.
- $\frac{A\rho(t)\mathrm{E}\Delta t}{\mathrm{E}} = A\rho(t)\Delta t$: Ratio of the occluded area.
- **$\boldsymbol{\sigma(\mathrm{t}) \coloneqq A\rho(\mathrm{t})}$: Opacity (extinction coefficient).**
- $T(t)$: Transparency



$$T(t+\Delta t) = T(t)(1-\sigma(t)\Delta t) = T(t) - T(t)\sigma(t)\Delta t$$
$$\Delta T(t) = T(t+\Delta t) - T(t) = -T(t)\sigma(t)\Delta t$$

Solve ODE with the initial condition $T(0) = 1$:

$$T(t) = \exp\left(-\int_0^t \sigma(t)dt\right)$$

Max and Chen, Local and Global Illumination in the Volume Rendering Integral, 2010.

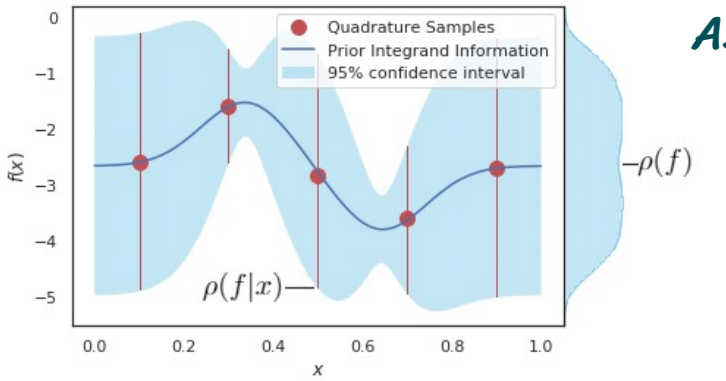
Volume Rendering – Discretization

Use a stratified sampling.

Partition the ray range into evenly-spaced bins and draw one sample uniformly at random from within each bin.

$$T_i = \exp\left(-\int_0^{t_i} \sigma(s)ds\right) = \exp\left(-\sum_{j=1}^{i} \sigma_j \Delta t\right)$$

Transparency until the end of the i-th bin.



Vanslette et al., 2020.

Assume that σ_j and $\mathbf{c}_i$ are constant within each bin.

Volume Rendering – Discretization

The color of the ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is computed as:

$$C(\mathbf{r}) = \int_0^{t_{max}} T(t)\sigma\big(\mathbf{r}(t)\big)\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

$$\approx \sum_{i=1}^{N} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i dt = \sum_{i=1}^{N} \mathbf{c}_i \underbrace{\int_{t_i}^{t_{i+1}} T(t)\sigma_i dt}_{w_i}$$

The probability of the ray stopping at the point.

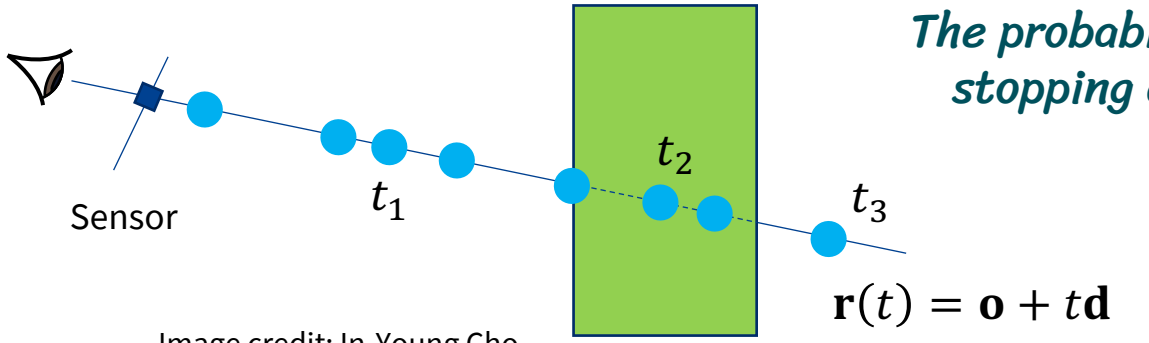


Image credit: In-Young Cho

Volume Rendering – Discretization

Approximation via Quadrature

$$C_i = \mathbf{c}_i \int_{t_i}^{t_{i+1}} \sigma_i T(t) dt = \mathbf{c}_i \int_{t_i}^{t_{i+1}} \sigma_i \exp\left(-\int_0^t \sigma(s) ds\right) dt$$

Integral for each bin

$$= \mathbf{c}_i \int_{t_i}^{t_{i+1}} \sigma_i \exp\left(-\int_0^{t_i} \sigma(s) ds\right) \exp\left(-\int_{t_i}^t \sigma(s) ds\right) dt$$

$$= \mathbf{c}_i T_i \int_{t_i}^{t_{i+1}} \sigma_i \exp\left(-\sigma_i \int_{t_i}^t ds\right) dt$$

$$= \mathbf{c}_i T_i \int_{t_i}^{t_{i+1}} \sigma_i \exp\left(-\sigma_i (t - t_i)\right) dt$$

$$= \mathbf{c}_i \underbrace{T_i (1 - \exp(-\sigma_i \Delta t))}_{w_i}$$

The probability of the ray stopping at the point.

For a more precise approximation, compute $T(t)$ based on σ_j and $\mathbf{c}_i$ instead of assuming that $T(t)$ is constant within each bin.

Volume Rendering – Discretization

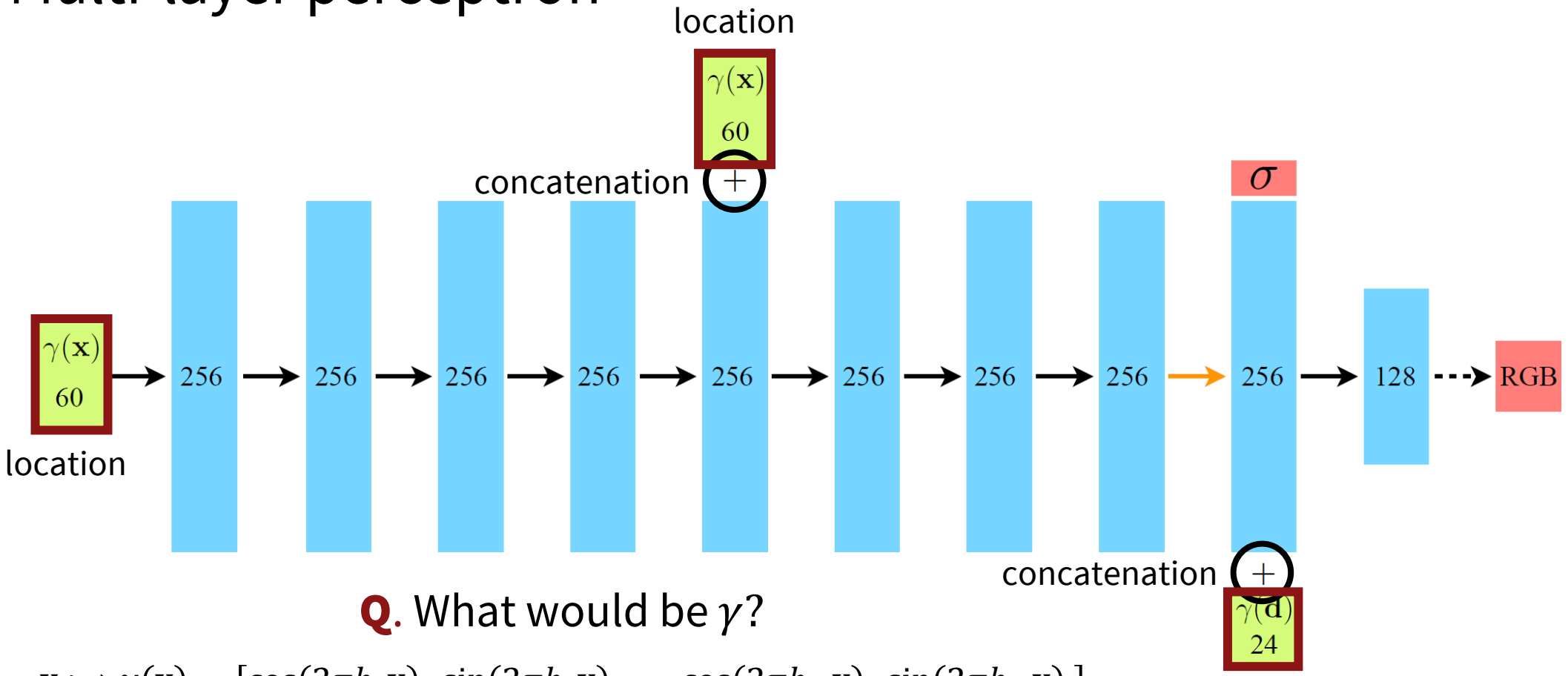
Approximation via Quadrature

$$C(\mathbf{r}) = \sum_{i=1}^{N} \mathbf{c}_i w_i = \sum_{i=1}^{N} \mathbf{c}_i T_i (1 - \exp(-\sigma_i \Delta t)) \,,$$

$$\text{where } T_\text{i} = \exp\left(- \sum_{\text{j}=1}^{\text{i}-1} \sigma_\text{j} \Delta t \right)$$

Network Structure

Multi-layer perceptron

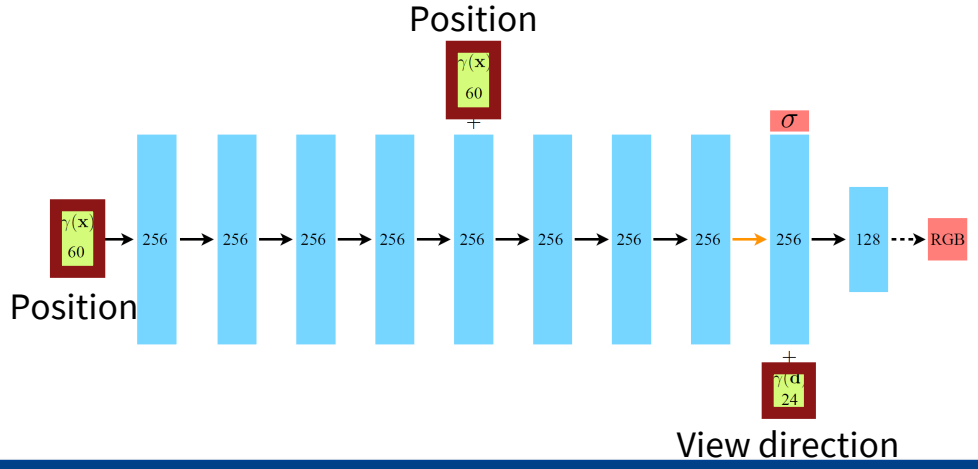


Q. What would be γ?

$$\mathbf{x} \mapsto \gamma(\mathbf{x}) = [\cos(2\pi b_1 \mathbf{x}),\ \sin(2\pi b_1 \mathbf{x}),\ \cdots, \cos(2\pi b_m \mathbf{x}),\ \sin(2\pi b_m \mathbf{x})\ ]$$

Network Structure

- **Position**
  - Mapped to a higher-dimensional space through positional encoding and processed by a deeper network.
  - Want to learn a high-frequency (rapidly fluctuating) opacity function.
- **View Direction**
  - Mapped to a lower-dimensional space through positional encoding and processed by a shallower network.
  - Want to learn a low-frequency (smooth) radiance function w.r.t. the view direction.

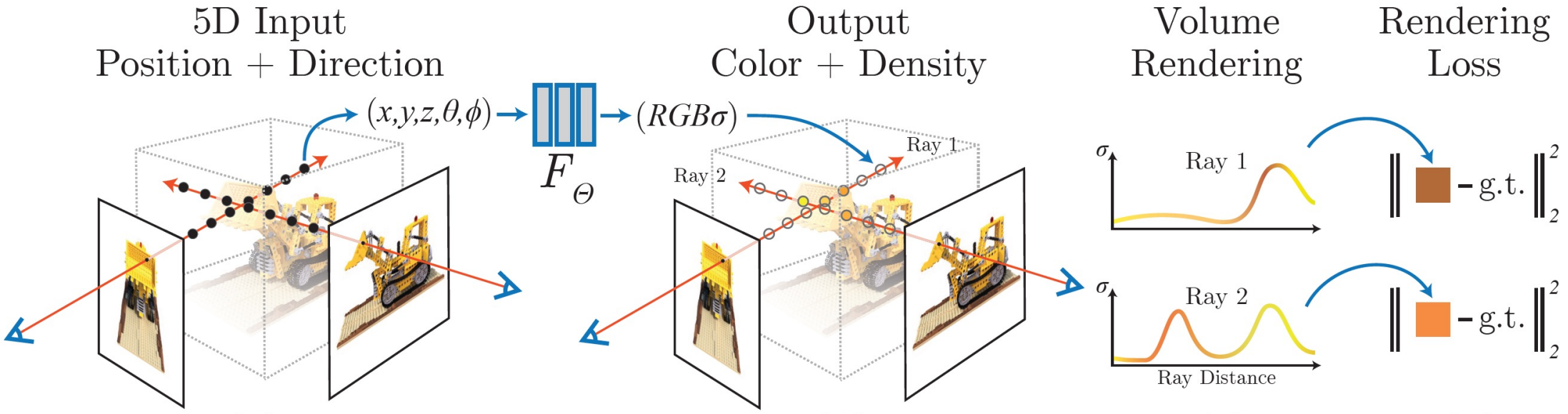


Hierarchical Sampling

- **Two networks**: one for **coarse** sampling, and the other for **fine** sampling.
- First, sample N_c points with the stratified sampling and compute the probability w_i at each point using the **coarse** network.
- Second, sample N_f more points **adaptively** with the probability w_i and compute the final pixel color using the **fine** network and all the $N_c + N_f$ points.

Training

1. Batch pixels.
2. Sample points along rays.
3. Predict color and density at each point.
4. Interate the outputs w/ volume rendering and compute loss.



Comparisons

Can reconstruct high-frequency details while using much smaller memory (< 5MB MLP parameters)!

SRN [Sitzmann 2019]

Voxel-based!

Nearest Input

NeRF

Geometry Visualization

NeRF – Limitations (1/2)

- Slow speed.
  - Training: 1-2 days for a NVIDIA V100 GPU.
  - Inference: A few seconds for a FHD image.

⟶ ***KiloNeRF, FastNeRF, Plenoxels, HashNeRF…***

- Not scalable to large scenes.

⟶ ***Neural Scene Graph Rendering, NeRF-OSR, CitiNeRF, Block-NeRF…***

NeRF – Limitations (2/2)

- Requires accurate camera information

⟶ ***NeRF in the Wild, BARF, SCNeRF, GNeRF, iNeRF, NeRF--...***

- Cannot handle dynamic scenes / moving objects.

⟶ ***Nerfies, HyperNeRF, NeRFlow, D-NeRF, AD-NeRF, NR-NeRF...***

- Cannot change the lighting.

⟶ ***NeRF-W, NeRD, NeRV...***

- Often fails to precisely reconstruct the shape.

⟶ ***VolSDF...***

- Cannot generalize to unseen scenes – one scene, one model.

⟶ ***GIRAFFE, Pi-GAN, StyleNeRF, EG3D...***

NeRF – More

- Improving output quality

⟶ ***MIP-NeRF, MVSNeRF, DietNeRF, UNISURF, NerfingMFS…***

- Conditional

⟶ ***CRF, GSN, GANcraft, CodeNeRF…***

- Composition

⟶ ***EditNeRF…***

- Articulated

⟶ ***NARF, AnimatableNeRF…***

References:
https://dellaert.github.io/NeRF/
https://dellaert.github.io/NeRF21/