TODOs

1. ***Paper Selection and Registration***: **[Important! Deadline was yesterday. If you haven’t done it, please do it now!]**
Please select and register for the two papers you would like to present using the following Excel link:
https://docs.google.com/spreadsheets/d/1_FJueXqWnKWoYOGZTNiwp2qRmSP0u1H6ayEYE5j3Ib0/edit?gid=0#gid=0

2. ***Presentation Preparation***:
- Ensure you are fully prepared **one class before your scheduled class for presentation**.
- Upload your slides to the Google folder (https://drive.google.com/drive/folders/1NO-JdWIRtKiLGZOMQxCOUso0AjdtrypY) **at least one hour before the class prior to your assigned class for presentation**. This is important in case of an emergency requiring us to reschedule your talk.
- For example, if you’re presenting on Monday, upload your slides by the previous Wednesday at 2:30 PM. If presenting on Wednesday, upload by Monday at 2:30 PM.

3. ***Class Participation***:
•Before each class, please read the papers that will be discussed and submit two questions **at least one hour before the class** using the following link:
https://docs.google.com/forms/d/e/1FAIpQLSfSxryv_JO9Ffbd7iKCIqnczqPWJUqv3OGFI6K-2sAKOJmBYQ/viewform

Course Link:
https://neural-representation-2024.github.io/topics.html

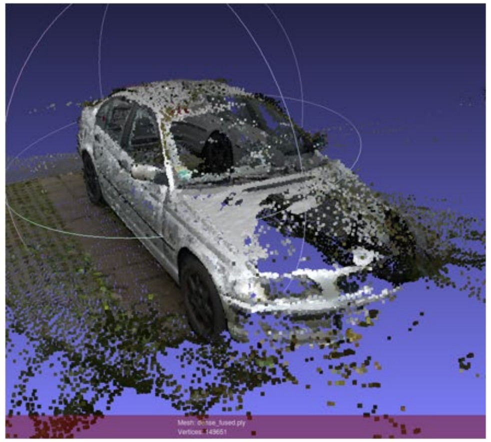
Review of Last Class

1. Challenges in using classical computer graphics pipeline for 3D reconstruction and photorealistic rendering

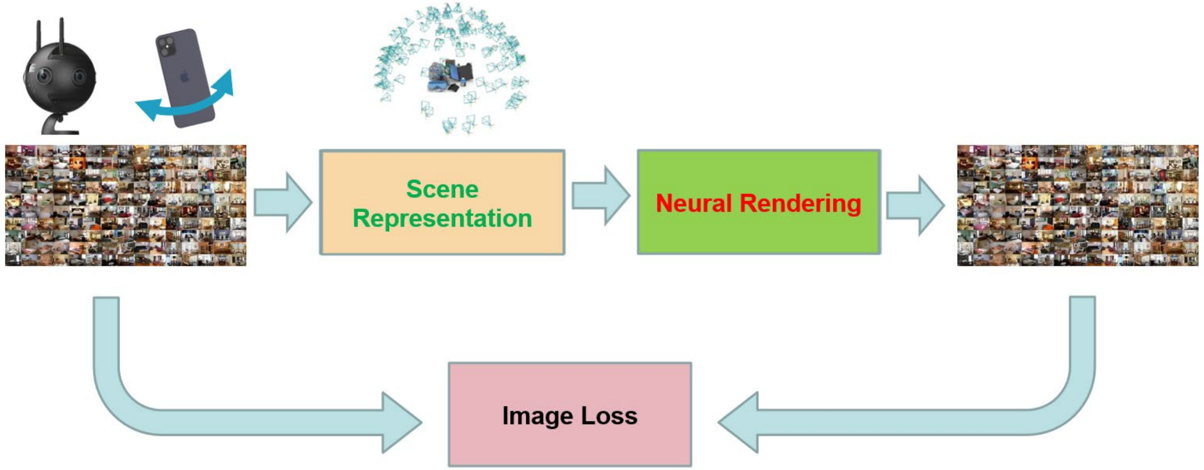
2. Neural scene representation and neural rendering is the rescue

3. Neural Rendering:

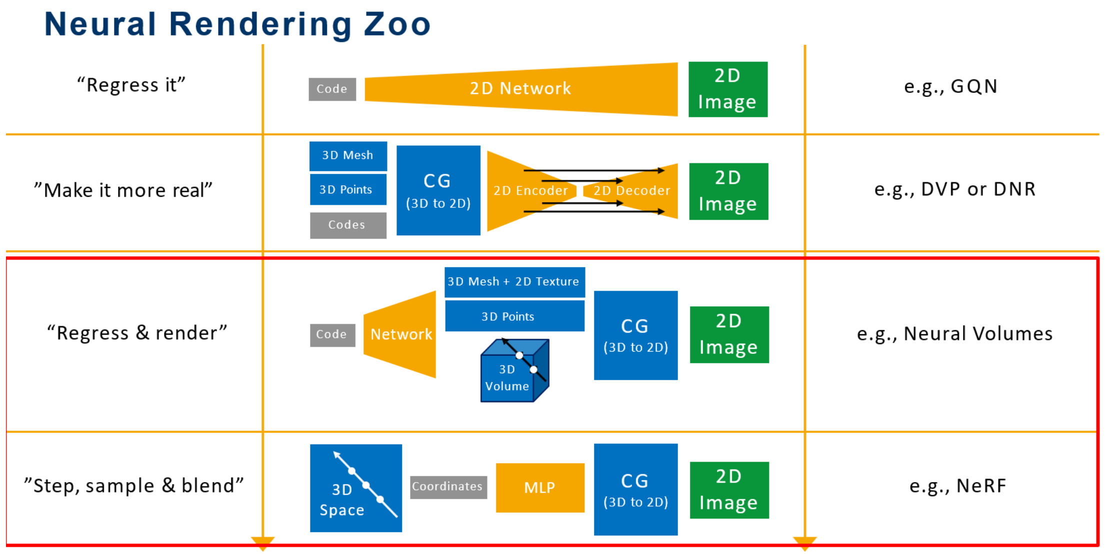
Deep neural networks for **image or video generation** that enable **explicit or implicit control** of **scene properties**

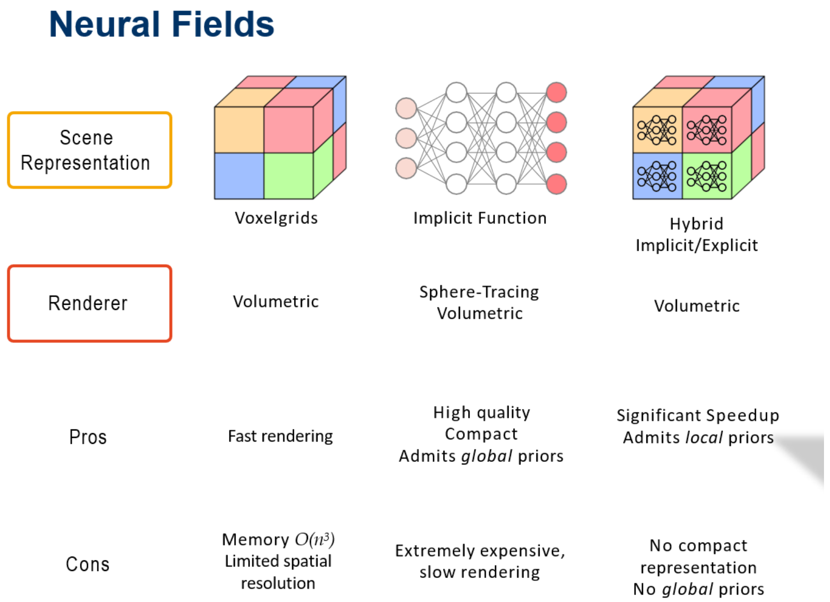


VS



Review of Last Class

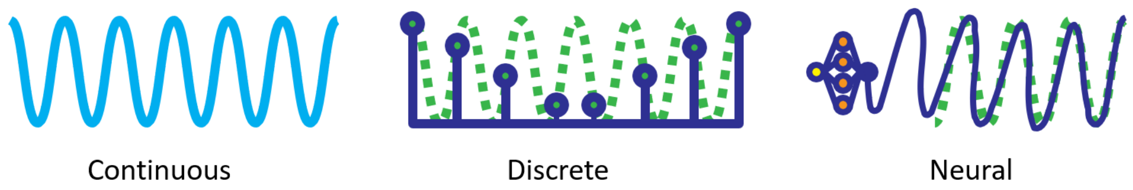




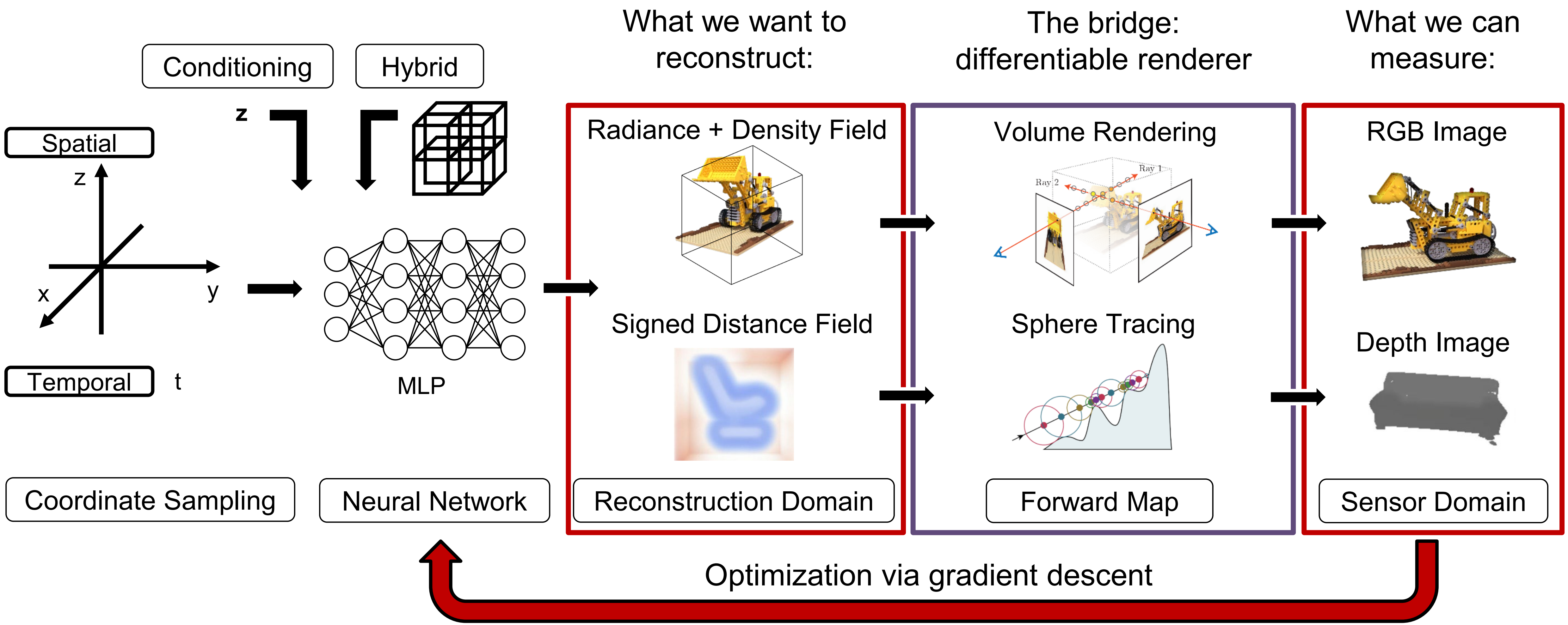
4. Different neural rendering methods
- Using neural rendering to learn neural fields.

5. Neural Fields:
A field is a quantity defined for all spatial and/or temporal coordinates;
A neural field is a field that is parameterized fully or in part by a neural network.

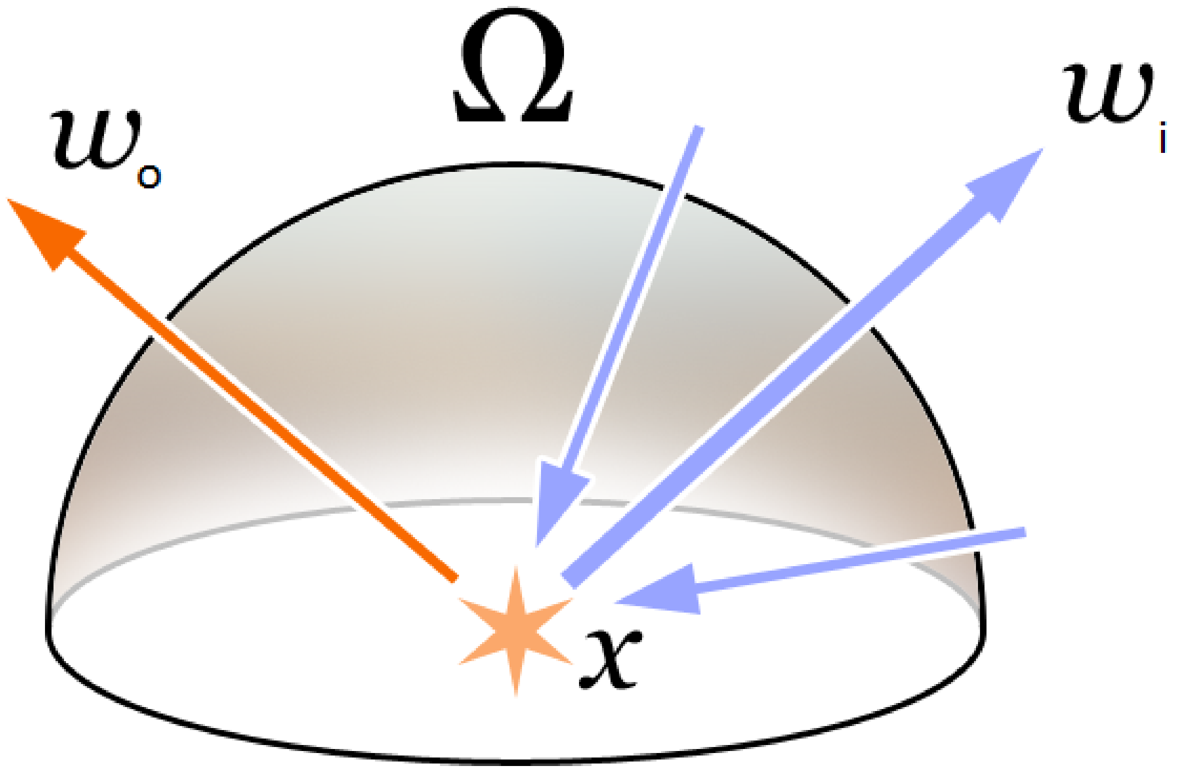
Fields / signals can be represented in many ways.



Neural Fields General Framework



BRDF Shading



$$L(\mathbf{x},\vec{\omega}_o) = L_e(\mathbf{x},\vec{\omega}_o) + \int_S f_r(\mathbf{x},\vec{\omega}_i \rightarrow \vec{\omega}_o) L(\mathbf{x}',\vec{\omega}_i) G(\mathbf{x},\mathbf{x}') V(\mathbf{x},\mathbf{x}') d\omega_i$$

Overview of This Class

0. Fundamentals of Classical Rendering Techniques in Computer Graphics

1. Three pioneering works in Neural Scene Representations and Neural Rendering
- Scene Representation Networks (SRN)
- Neural Volumes (before that: Deep Appearance Models)
- Neural Radiance Fields (NeRF)

2. Different Neural Scene Representations (Next Class)
- Uniform Grids -> Sparse Grids -> Multiresolution Grids -> Hash Grids
- Point Clouds
- Surface Mesh / Volumetric Mesh (Tetrahedron)
- Multiplane Images

Computer Graphics

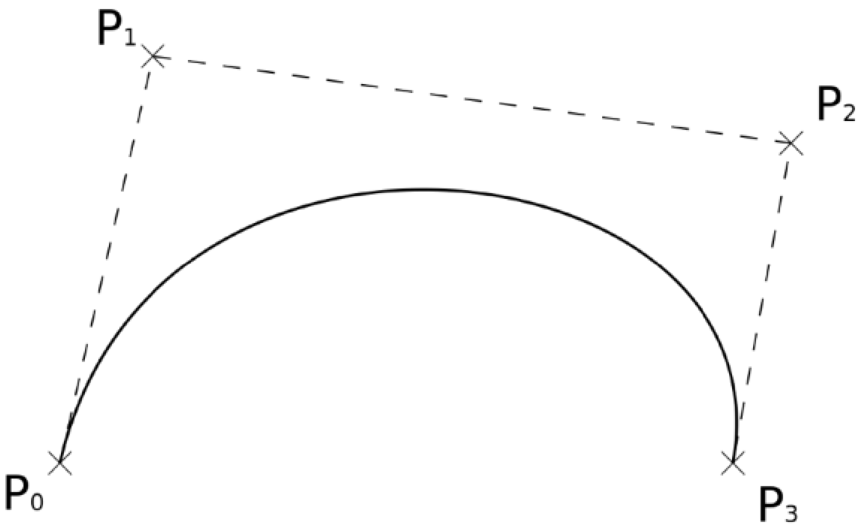
Geometry Processing

Rendering

Animation / Simulation

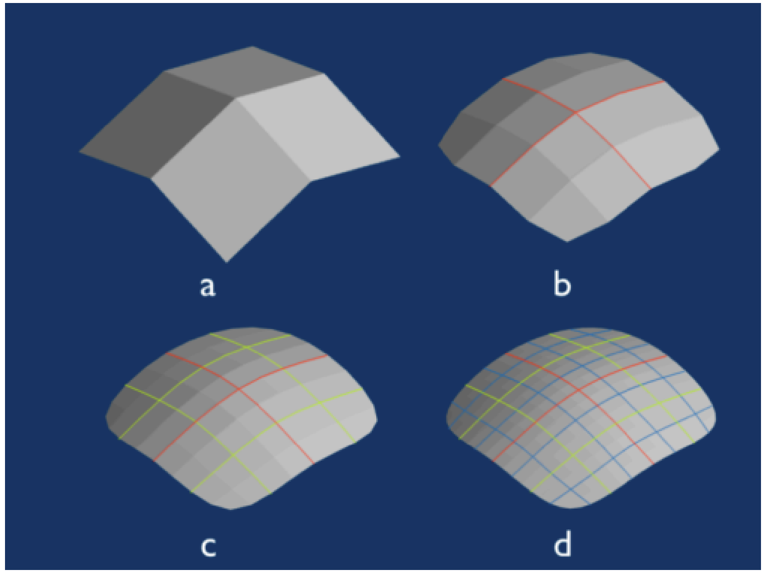
Curves and Meshes

- How to represent geometry in Computer Graphics



Bezier Curve

https://en.wikipedia.org/wiki/B%C3%A9zier_curve

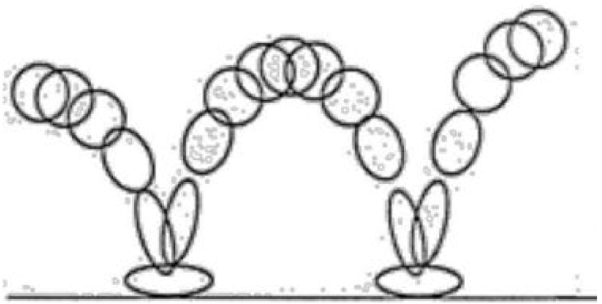


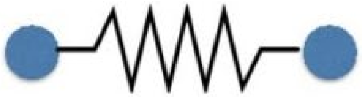
Catmull-Clark subdivision

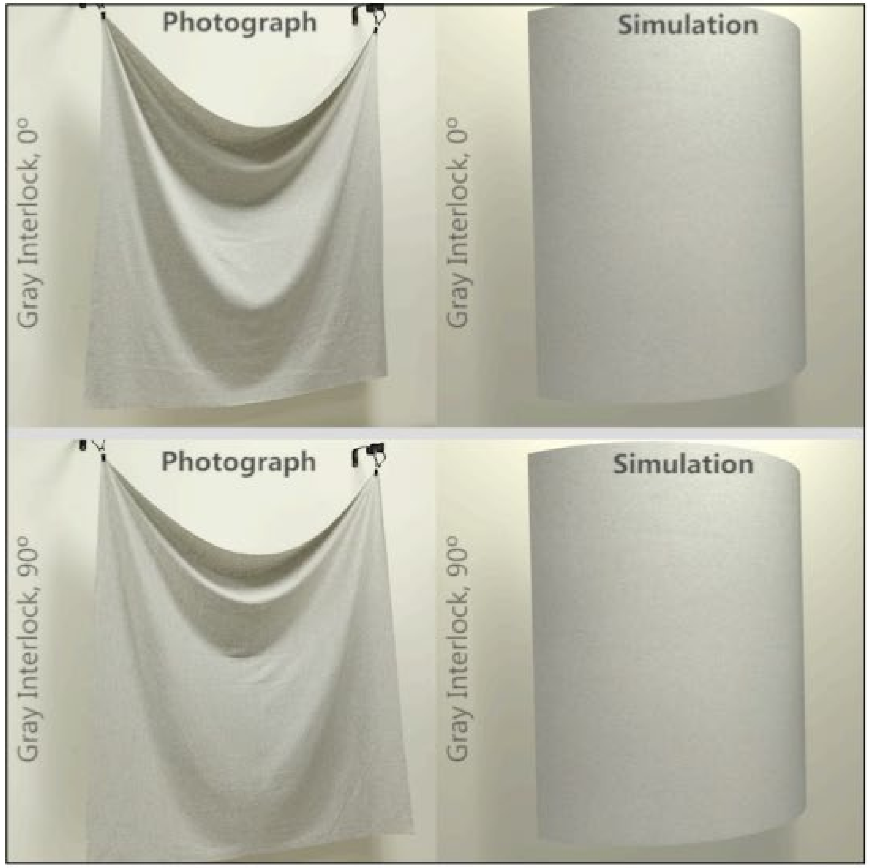
https://commons.wikimedia.org/wiki/File:Catmull-Clark_subdivision_of_4_planes.png

Animation / Simulation

- Key frame Animation
- Mass-spring System

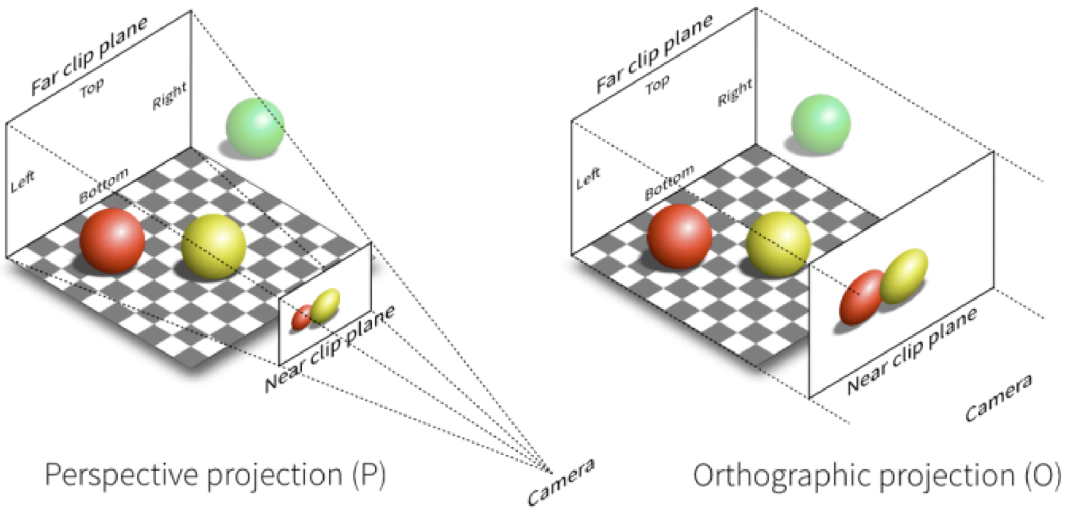




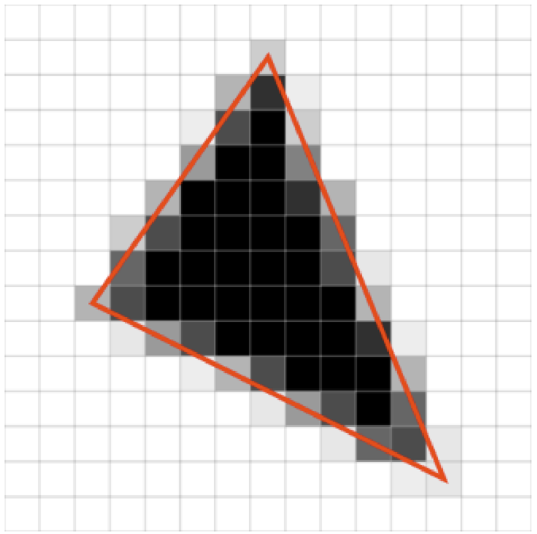


Rasterization

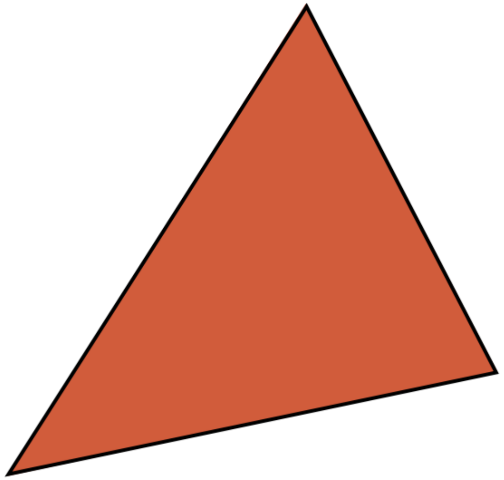
- Project geometry primitives (3D triangles / polygons) onto the screen
- Break projected primitives into fragments (pixels)
- Gold standard in Video Games (Real-time Applications)



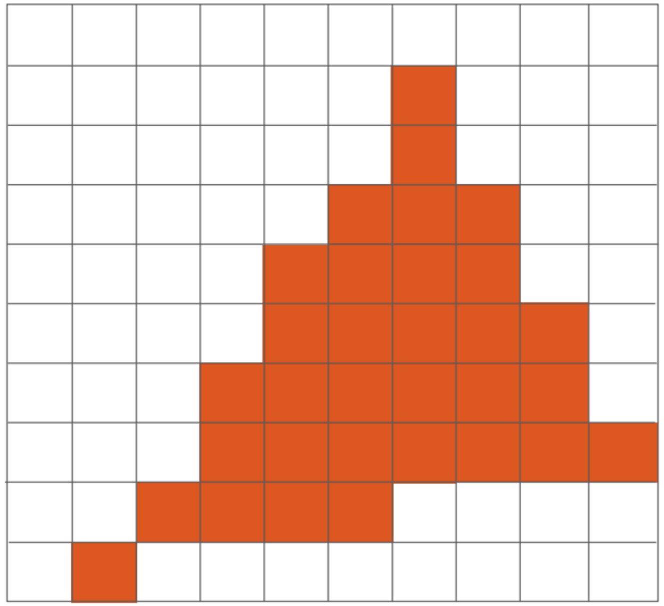
http://vispy.org/modern-gl.html



https://commons.wikimedia.org/wiki/File:Rasterisation-triangle_example.svg

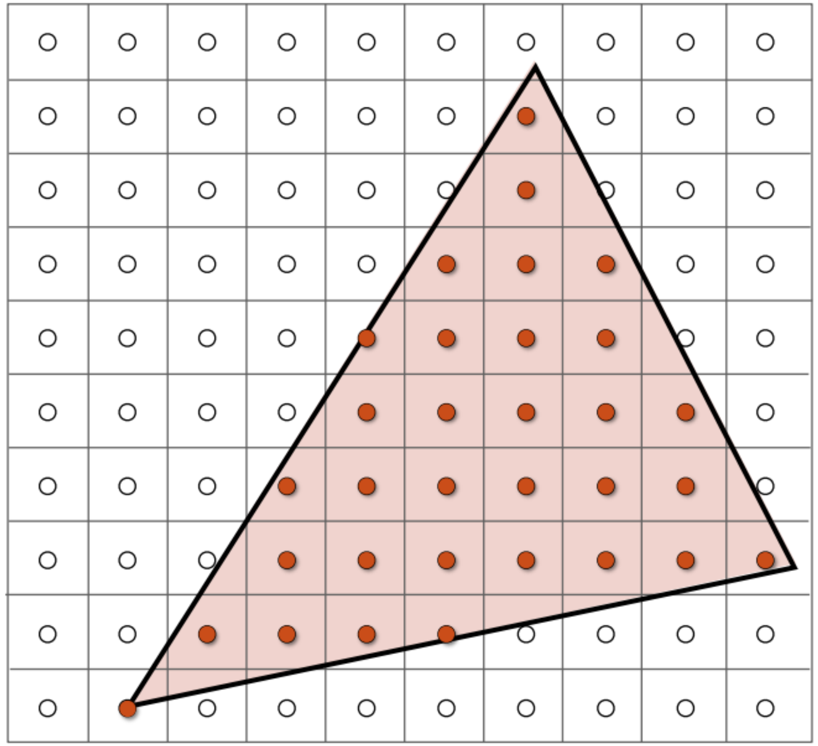


Continuous Triangle Function



After Rasterization

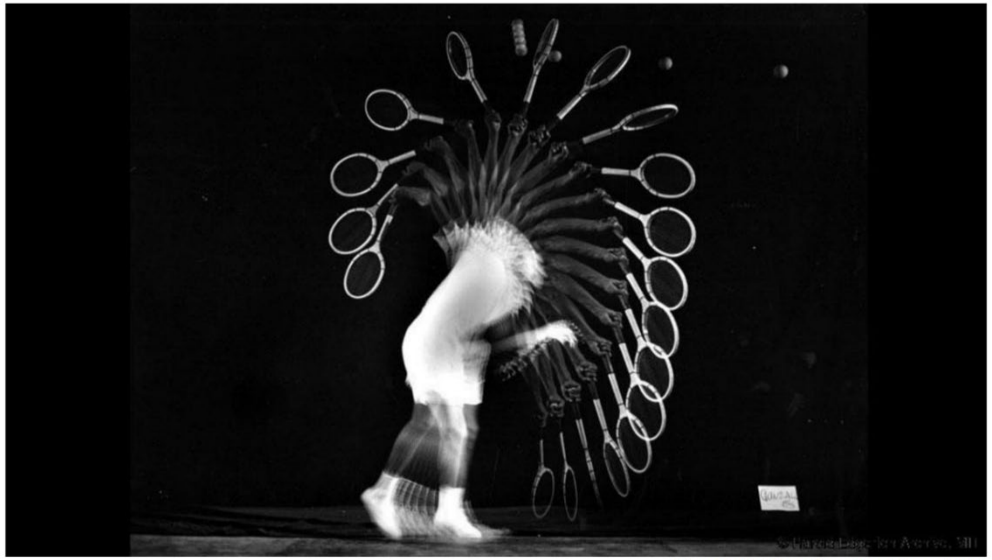
Jaggies! (Aliasing)



Rasterization = Sample 2D Positions

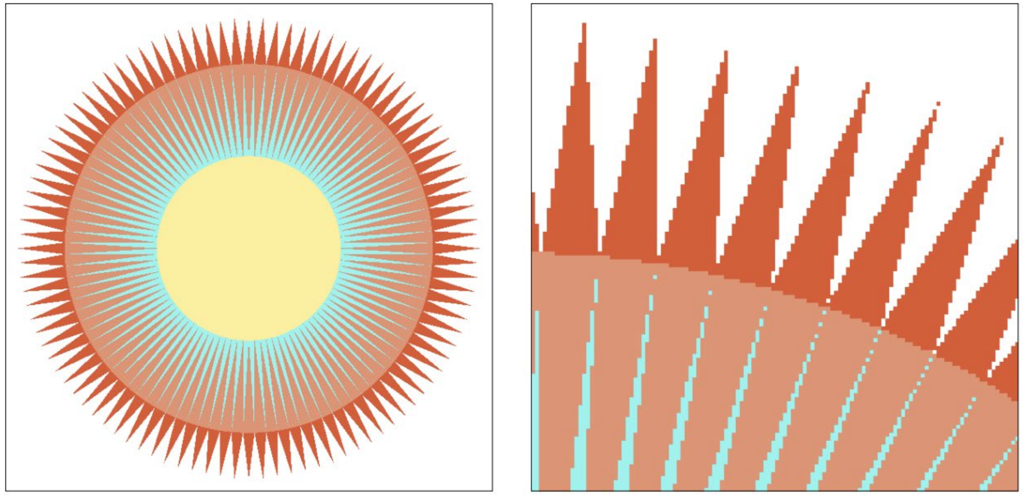
Photograph = Sample Image Sensor Plane

Video = Sample Time



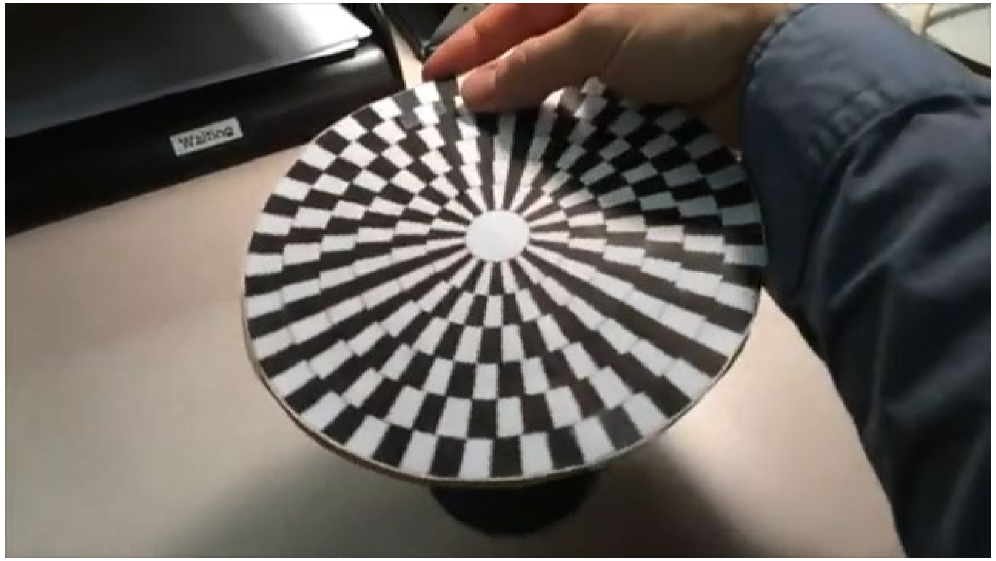
Harold Edgerton Archive, MIT

Sampling Artifacts (Errors / Mistakes / Inaccuracies) in Computer Graphics



Jaggies (Aliasing)

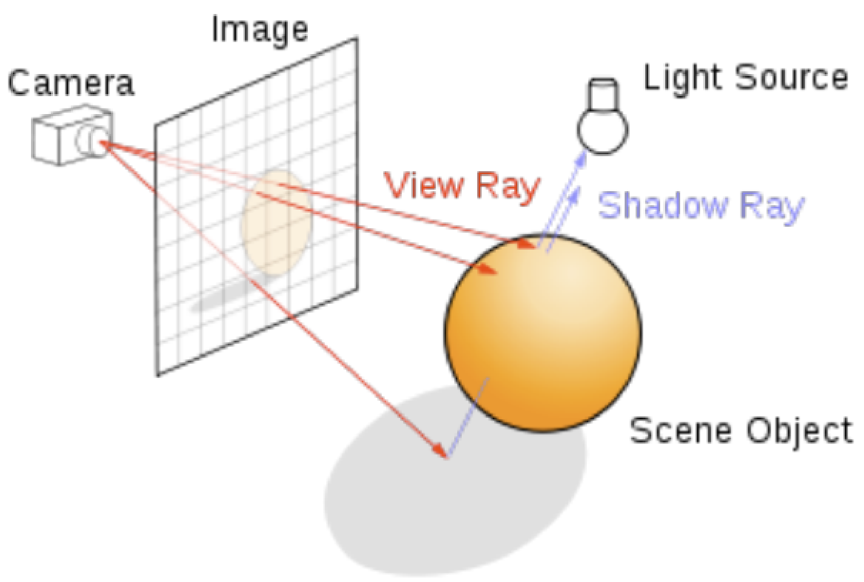
Moire Patterns

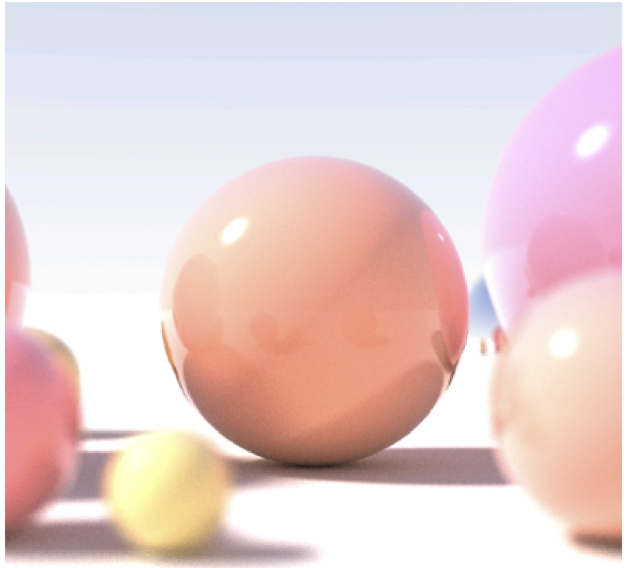


Wagon Wheel Illusion (False Motion)

Ray Tracing

- Shoot rays from the camera though each pixel
  - Calculate intersection and shading
  - Continue to bounce the rays till they hit light sources
- Gold standard in Animations / Movies (Offline Applications)





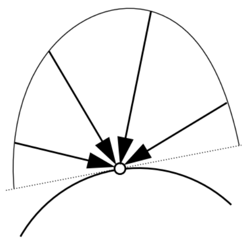
https://en.wikipedia.org/wiki/Ray_tracing_(graphics)

How to do photo-realistic rendering?

Radiometry

- Radiant flux (power):
  - the energy emitted, reflected, transmitted or received, per unit time

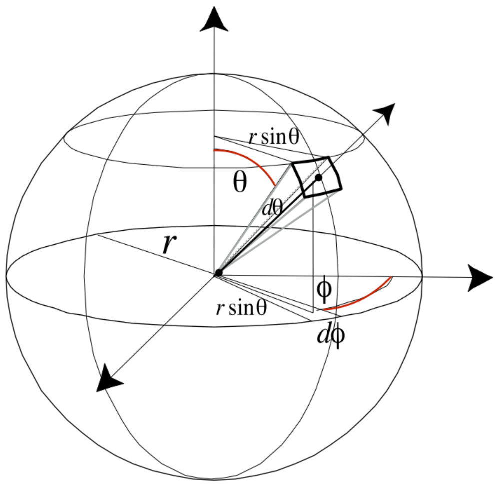
$$\Phi$$



- Irradiance:
  - How much light received by a “surface”
  - **Definition:** power per unit area on a surface point
  - **Lambert’s Law:** irradiance at surface is proportional to **cosine** of angle between light direction and surface normal.

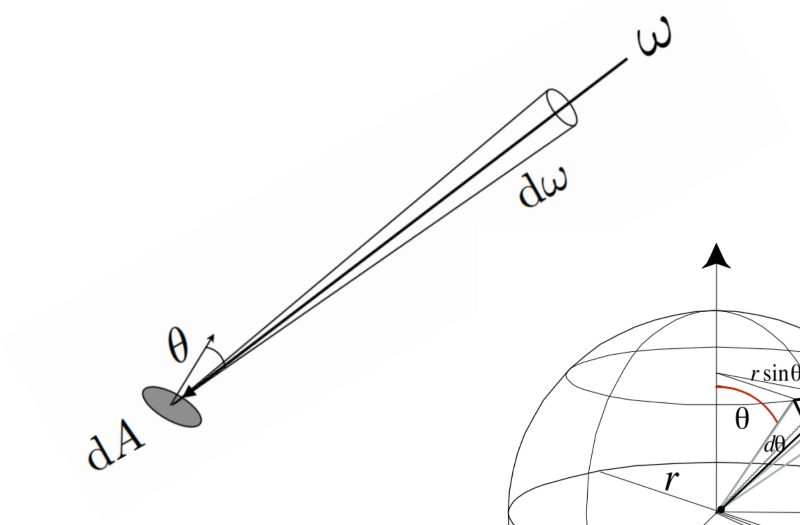
$$E(x) = \frac{\mathrm{d}\Phi}{\mathrm{d}A}$$

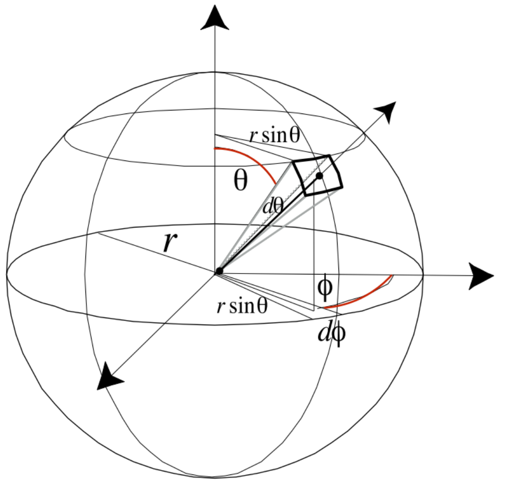
where $dA = (rd\theta)(r\sin\theta d\phi) = r^2 \sin\theta d\theta d\phi$



Radiometry

- Radiance:
  - How much light travelling along a “ray” (light received by an area from a direction)
  - **Definition:** power emitted, reflected, transmitted or received by a surface, per unit solid angle, per projected unit area





$$L(x,\omega) = \frac{d^2\Phi}{d\omega dA\cos\theta}$$

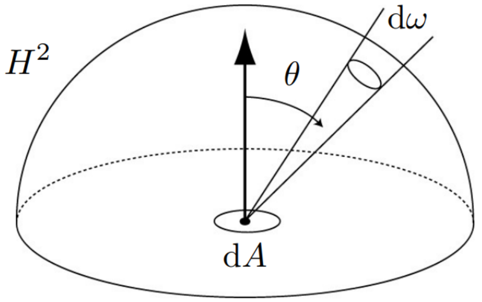
where $d\omega = \frac{dA}{r^2} = \sin\theta d\theta d\phi$

Radiance = Irradiance per solid angle

$$L(x,\omega) = \frac{dE(x)}{d\omega\cos\theta}$$

$$E(x) = \int_{H^2} L(x,\omega)\cos\theta dw$$

Unit Hemisphere: H^2



BRDF

Bidirectional Reflectance Distribution Function (BRDF)

- How much light is reflected into each outgoing direction ω_r from each incoming direction ω_i
- BRDF can be simply regarded as diffusion + specular (we will discuss this in detail in later classes).

$$f_{BRDF}(\omega_i \to \omega_r) = \frac{\mathrm{d}L_r(\omega_r)}{\mathrm{d}E_i(\omega_i)} = \frac{dL_r(\omega_r)}{L_i(\omega_i)\cos\theta_i dw_i}$$

irradiance

The Reflection Equation:

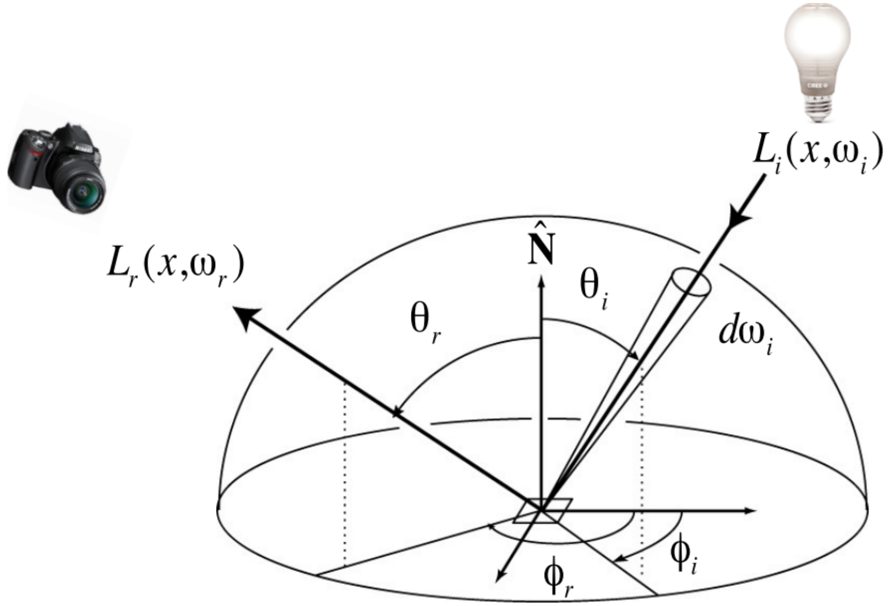
- Total light reflected from the outgoing direction ω_r

$$L_r(x, \omega_r) = \int_{H^2} f_{BRDF}(x, \omega_i \to \omega_r) dE_i(x, w_i)$$

Outgoing radiance

$$= \int_{H^2} f_{BRDF}(x, \omega_i \to \omega_r) L_i(x, \omega_i) \cos\theta_i dw_i$$

Incoming radiance



The Rendering Equation

- The rendering equation can be derived by adding an **emission term** to the reflection equation.

$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_{H^2} L_i(x,\omega_i) f_{BRDF}(x,\omega_i \to \omega_r)(n\cdot\omega_i)d\omega_i$$

Reflected light (output image) | Emission | Incident light | BRDF | Cosine of incident angle

- The rendering equation is a Fredholm Integral Equation of second kind:

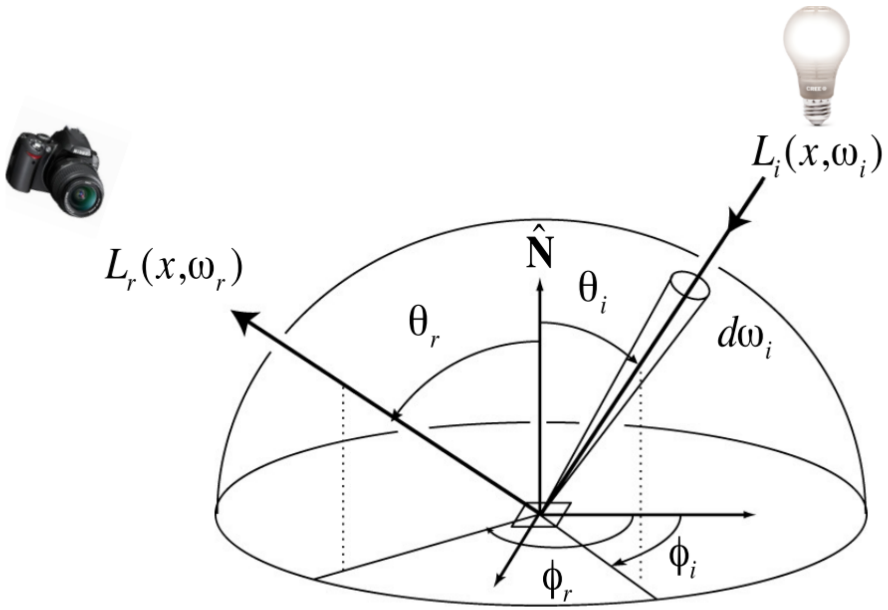
$$l(u) = l_e(u) + \int l(v)k(u,v)dv$$

- Use linear operators:

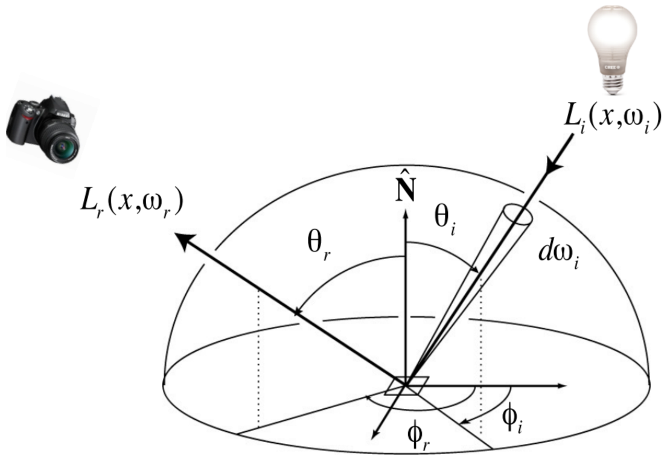
$$L = L_e + KL$$

Light transport matrix

$$(K \circ f)(x) = \int k(x,x')f(x')dx$$



Ray Tracing



Solve the rendering equation using linear operators

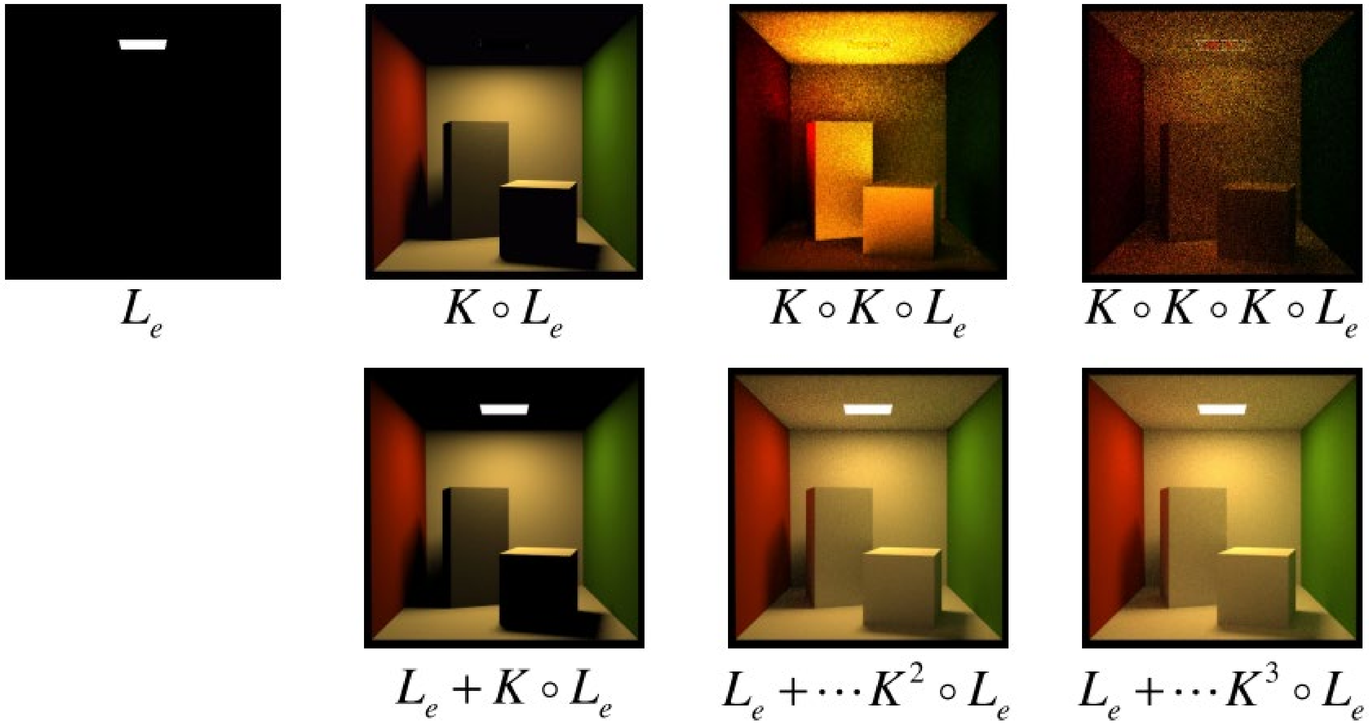
$$L = L_e + KL$$

$$
\begin{aligned}
L &= (I - K)^{-1} L_e \\
&= \left(I + K + K^2 + K^3 + \cdots\right) L_e \\
&= \boxed{L_e} + \boxed{K L_e} + \boxed{K^2 L_e + K^3 L_e + \cdots}
\end{aligned}
$$

Emission directly from light sources

Direct illumination

Indirect illumination (one bounce, two bounce, …)



It involves:
(1) solving the integral over the hemisphere
(2) ***recursive execution****.*

Monte Carlo Integration

It is intractable to solve the integrals from the rendering equation directly. Instead, we go for Monte Carlo integration (emission is omitted):

$$
\begin{aligned}
L_r(x,\omega_r) &= \int_{H^2} L_i(x,\omega_i) f_{BRDF}(x,\omega_i \to \omega_r)(n\cdot\omega_i)d\omega_i \\
&\approx \frac{1}{N}\sum_{k=1}^{N} L_i(x,\omega_i) f_{BRDF}(x,\omega_i \to \omega_r)(n\cdot\omega_i)/p(\omega_i)
\end{aligned}
$$

where we sample N incoming directions from a given Probability Density Function

(pdf) $\omega_i \sim p(\omega_i)$

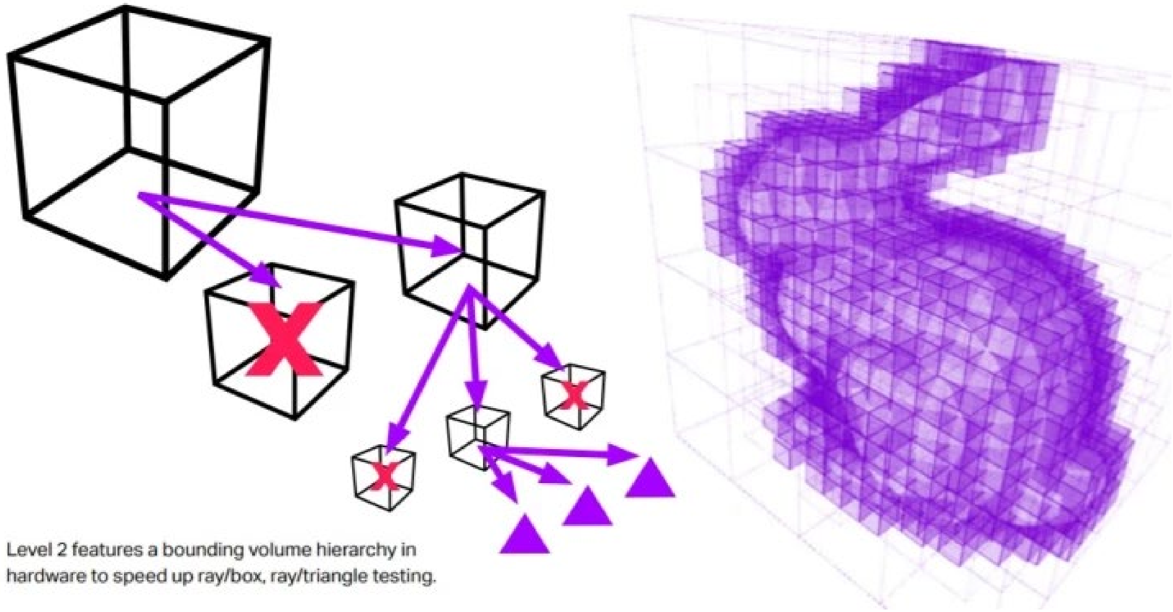
Monte Carlo Estimation:

$$\int f(x)dx = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} \qquad X_i \sim p(x)$$

Towards real-time ray tracing

Real-time ray tracing itself is an active research area. Several aspects can be used to improve the rendering efficiency:

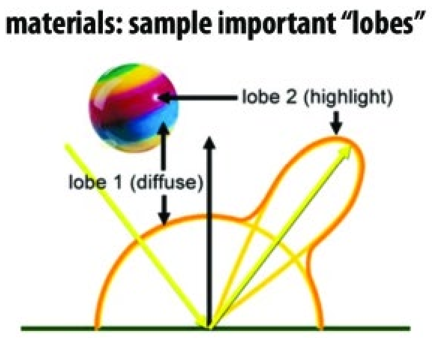
- Efficient data structure (e.g. bounding volume hierarchy (BVH))
  - Use a tree structure to partition the space which speed-up the ray intersection.
- Importance Sampling
- Learning based denoising



Towards real-time ray tracing

Real-time ray tracing itself is an active research area. Several aspects can be used to improve the efficiency:

- Efficient data structure (e.g. bounding volume hierarchy (BVH))
- Importance Sampling on Materials (BRDF)
  - Instead of uniformly sampling on the semi-sphere, we can sample more points based on the shape of BRDF (diffusion + specular).
  - It can also be combined with “sampling on light” which means “multiple importance sampling”.
- Learning based denoising



Towards real-time ray tracing

Real-time ray tracing itself is an active research area. Several aspects can be used to improve the efficiency:

- Efficient data structure (e.g. bounding volume hierarchy (BVH))
- Importance Sampling
- Learning based denoising/super-sampling
    - Use deep learning techniques, it is also possible to trace less paths for each pixel or lower resolution image, and then use neural network to get the final image.

Results of ray tracing: Photo-realistic

Other Modern Ray Tracing Algorithms

- Bidirectional path tracing
- Photon Mapping
- Metropolis light transport
- Multiple Importance Sampling (MIS)
- Quasi-Monte Carlo methods (QMC)
- Finite Element Radiosity
- ...

0. Fundamentals of Classical Rendering Techniques in Computer Graphics

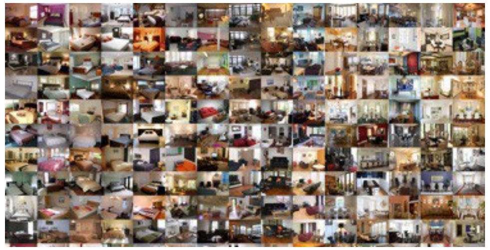
1. Three pioneering works in Neural Scene Representations and Neural Rendering
- Scene Representation Networks (SRN)
- Neural Volumes (before that: Deep Appearance Models)
- Neural Radiance Fields (NeRF)

2. Different Neural Scene Representations (Next Class)
- Uniform Grids -> Sparse Grids -> Multiresolution Grids -> Hash Grids
- Point Clouds
- Surface Mesh / Volumetric Mesh (Tetrahedron)
- Multiplane Images

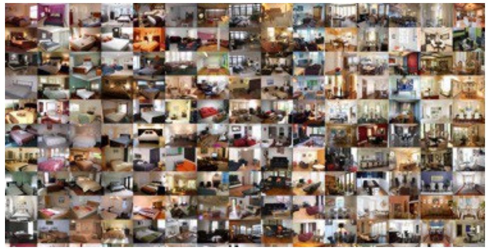
Scene Representation Networks (SRN)

Sitzmann, Zollhoefer, Wetzstein

NeurIPS 2019



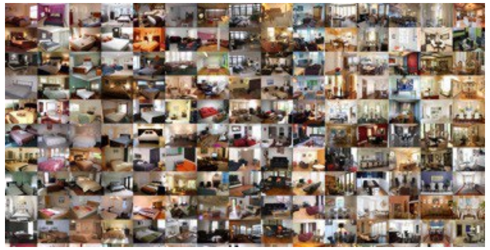
Infer Neural Scene Representation from 2D observations.



Neural Scene Representation

Learned feature representation of scene.

Formulate Neural Renderer.



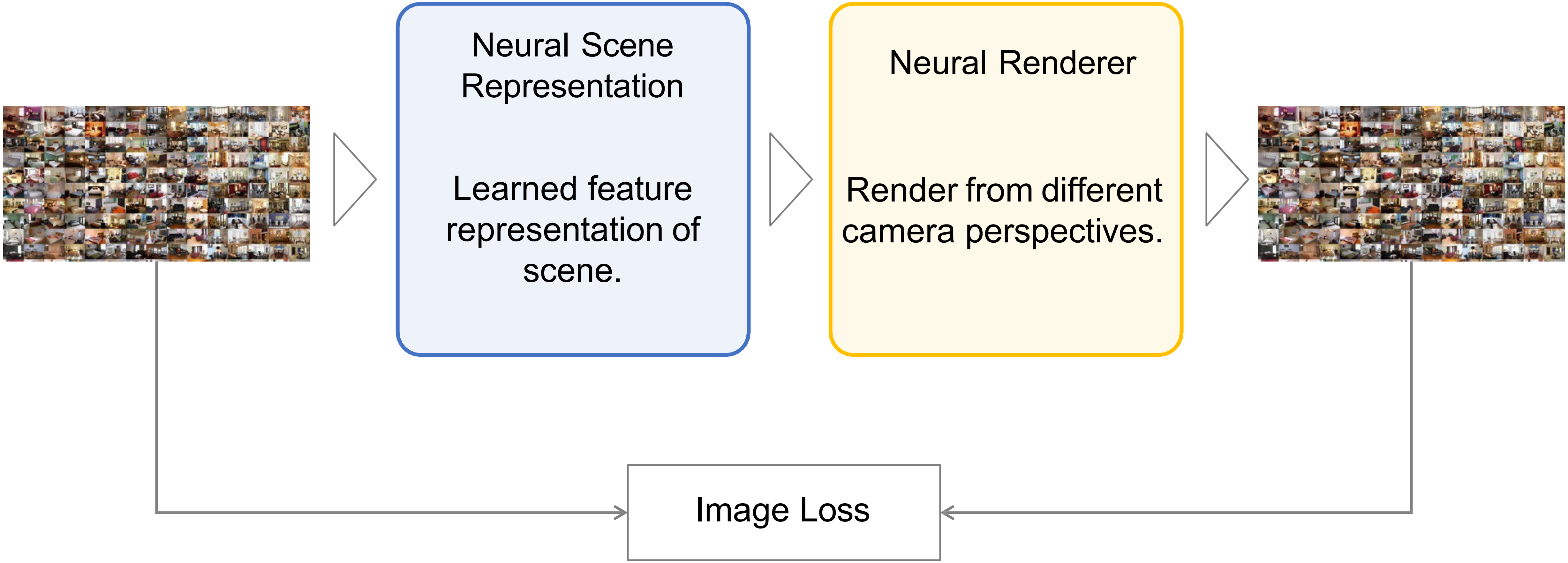
Neural Scene Representation

Learned feature representation of scene.

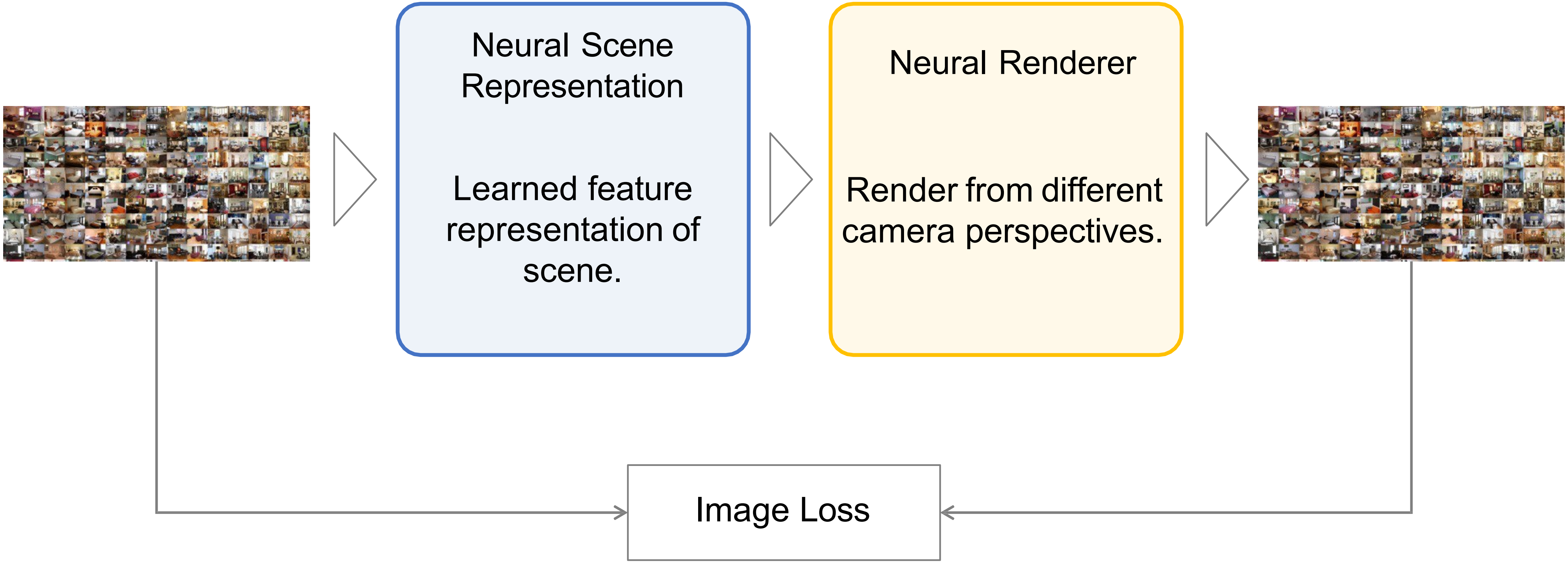
Neural Renderer

Render from different camera perspectives.

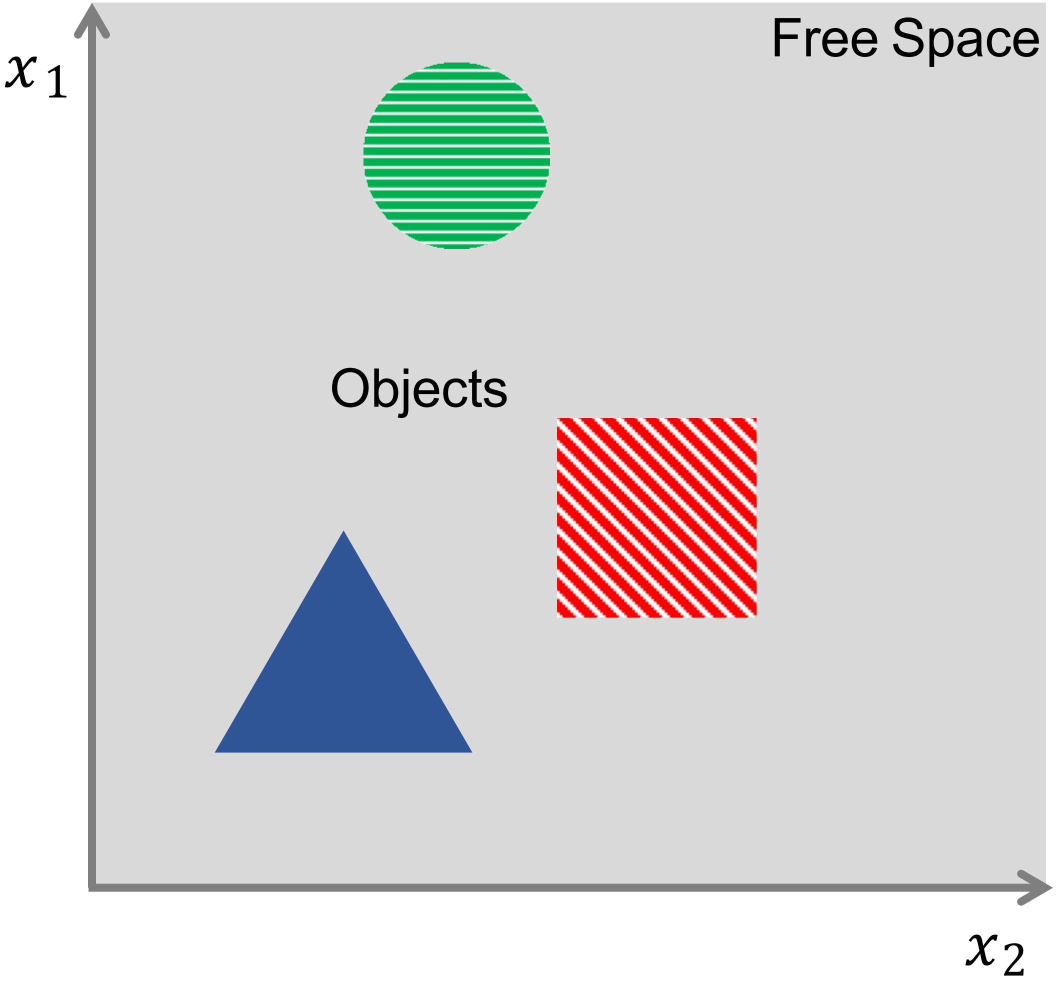
Finally: Predict training views & enforce loss on re-rendering error!

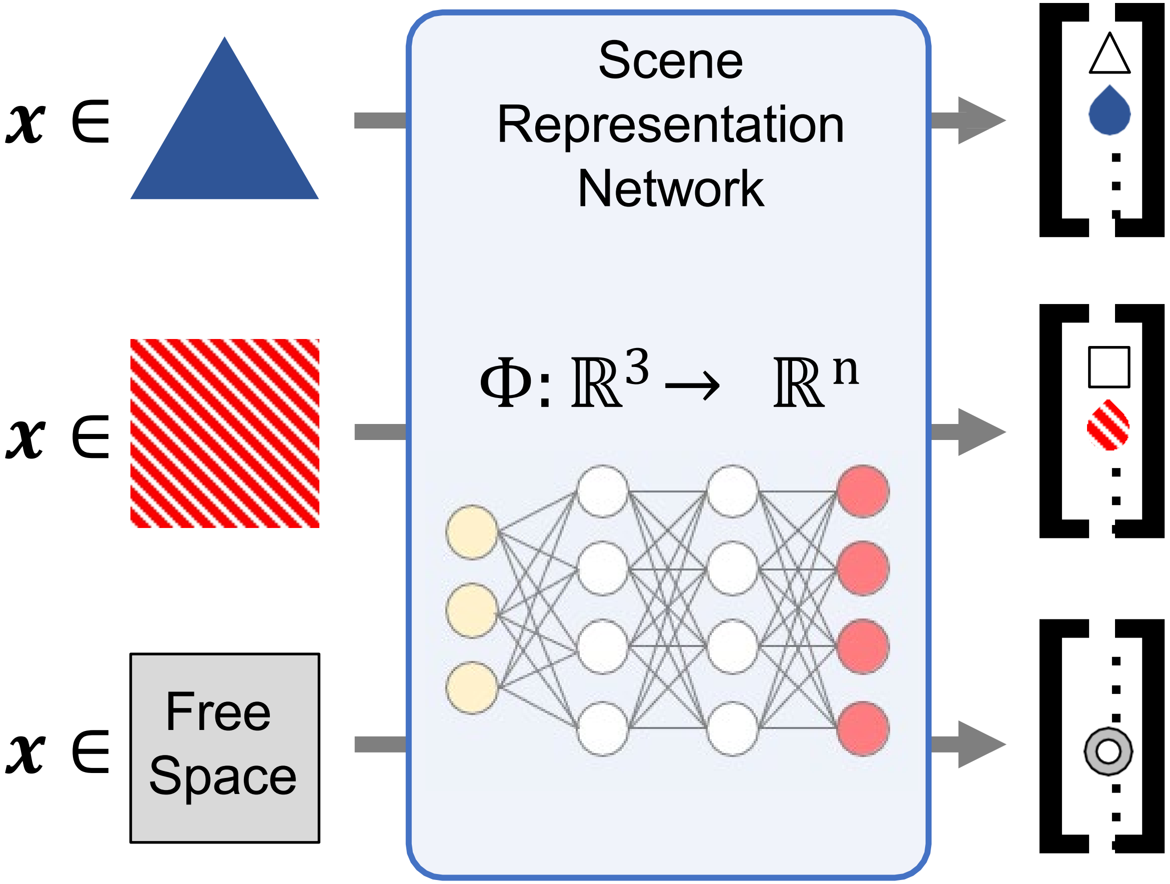


Self-supervised Scene Representation Learning



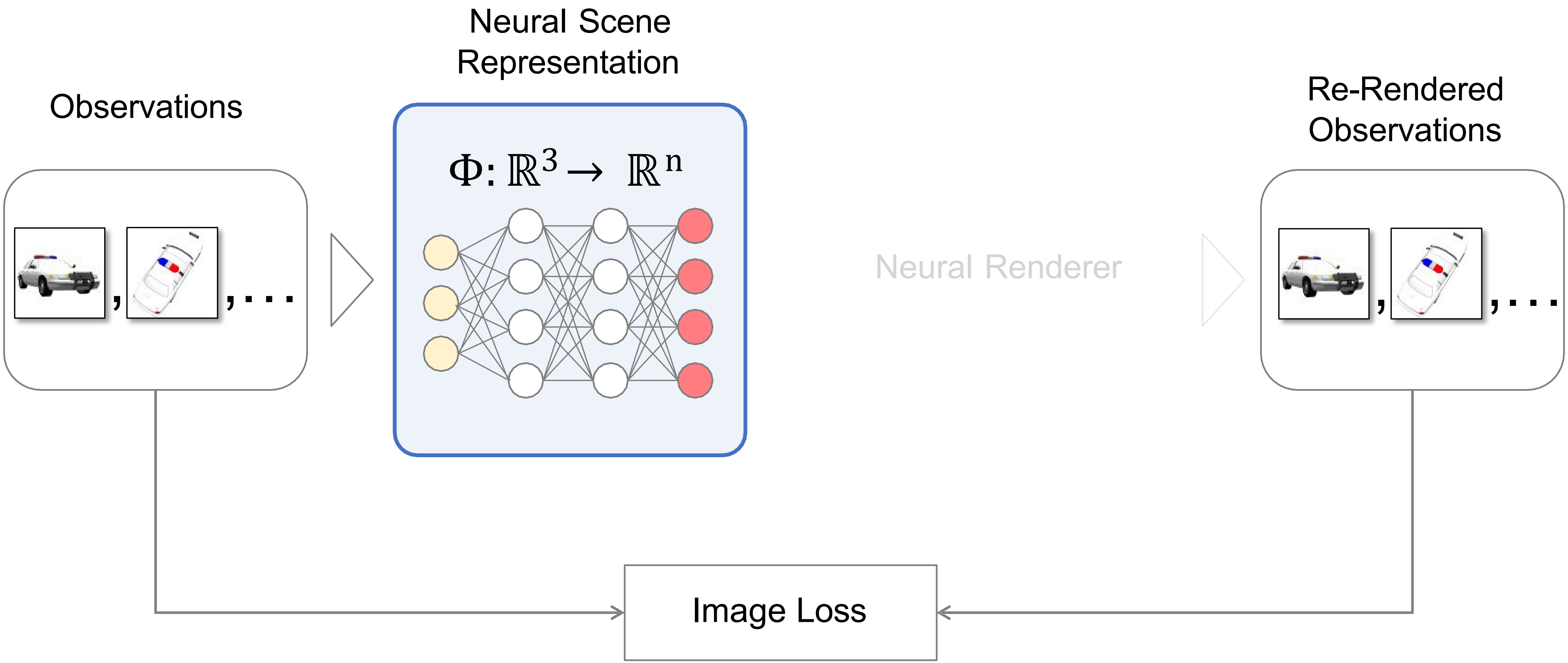
Scene Representation Network parameterizes scene as MLP.





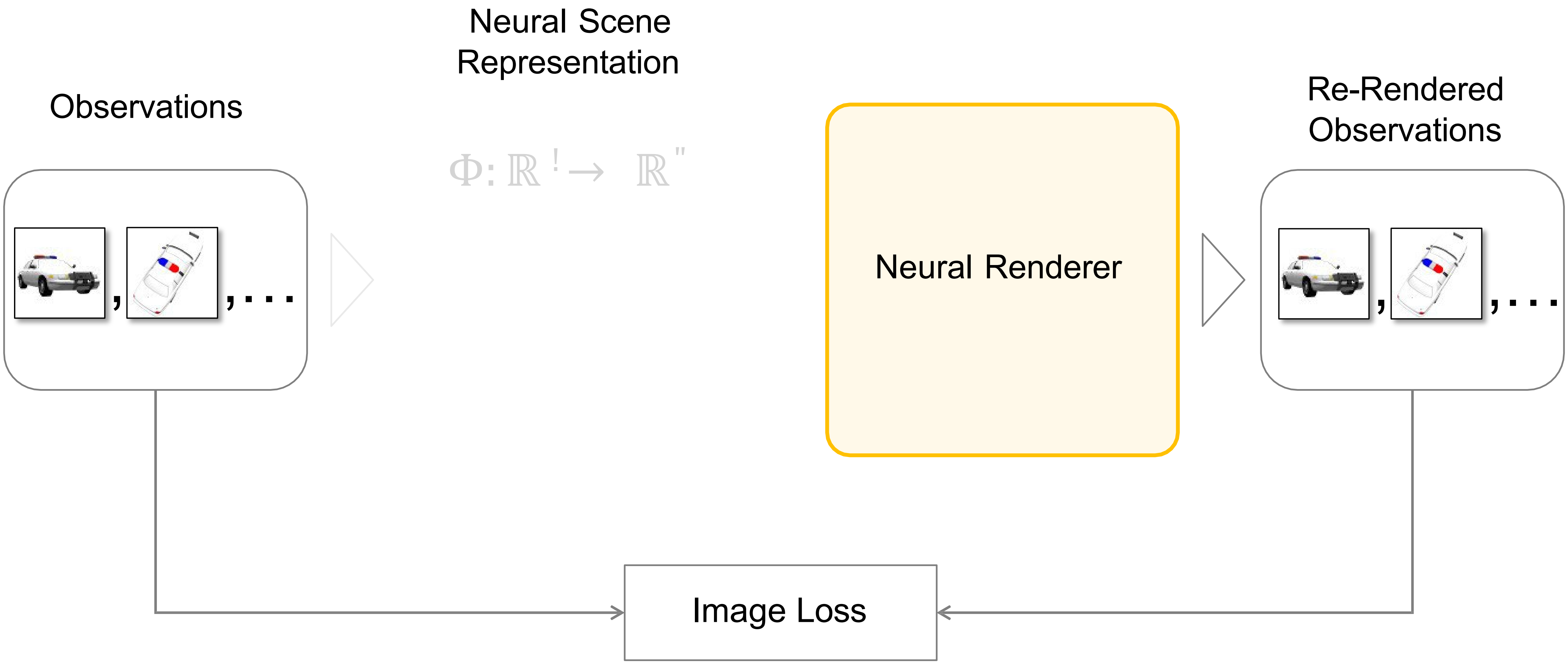
Sitzmann et al., NeurIPS 2019

Scene Representation Networks



Sitzmann et al., NeurIPS 2019

Scene Representation Networks



Sitzmann et al., NeurIPS 2019

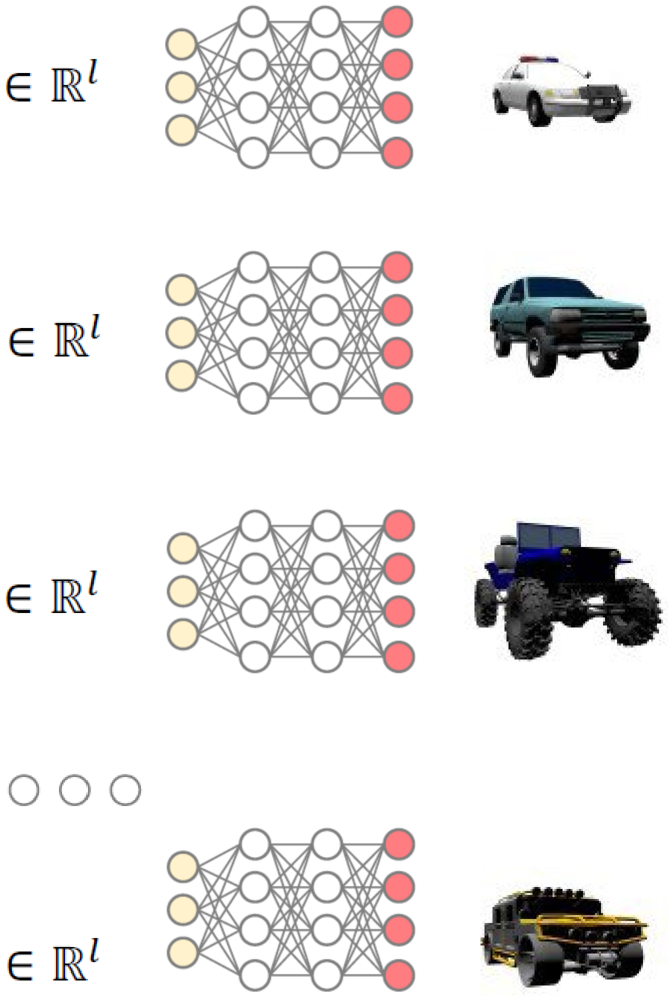
Each scene represented by its own SRN.

parameters $\phi_0 \in \mathbb{R}^l$

parameters $\phi_1 \in \mathbb{R}^l$

parameters $\phi_2 \in \mathbb{R}^l$

parameters $\phi_n \in \mathbb{R}^l$



Sitzmann et al., NeurIPS 2019

Manifold assumption.

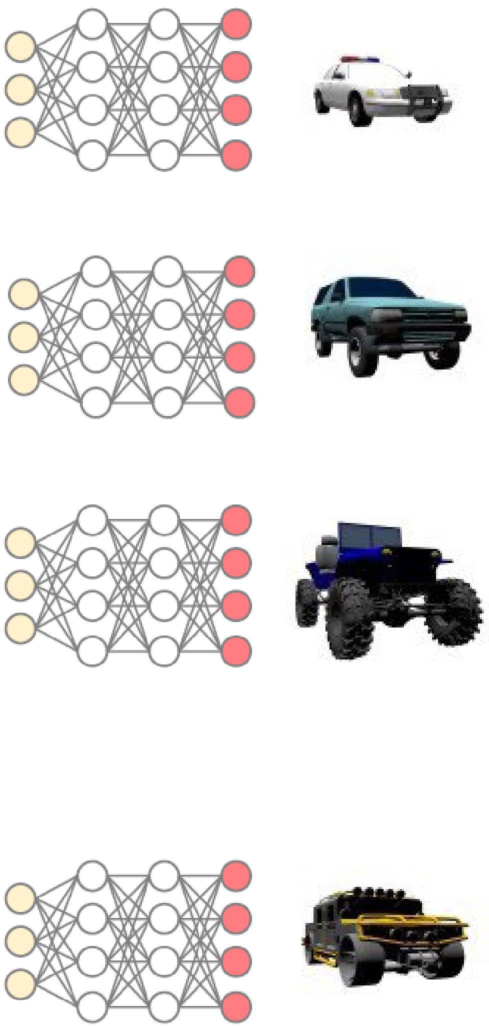
ϕ_i live on k-dimensional subspace of $\mathbb{R}^l$, $k < l$.

parameters $\phi_0 \in \mathbb{R}^l$

parameters $\phi_1 \in \mathbb{R}^l$

parameters $\phi_2 \in \mathbb{R}^l$

parameters $\phi_n \in \mathbb{R}^l$



Sitzmann et al., NeurIPS 2019

Represent each scene by low-dimensional embedding.

embedding $z_0 \in \mathbb{R}^k$

embedding $z_1 \in \mathbb{R}^k$

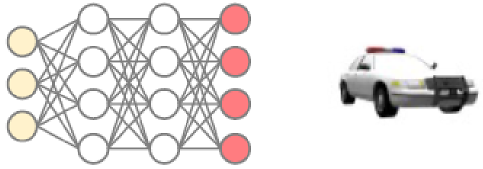
embedding $z_2 \in \mathbb{R}^k$

○○○

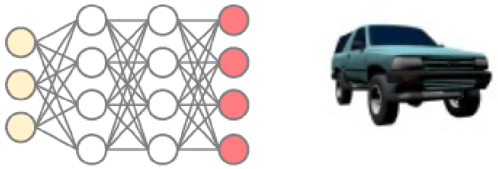
embedding $z_n \in \mathbb{R}^k$

Represent each scene with
low-dimensional embedding

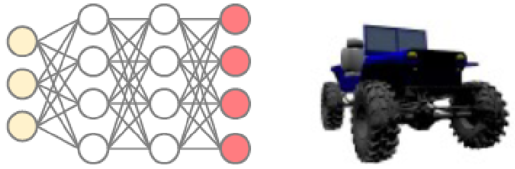
parameters $\phi_0 \in \mathbb{R}^l$



parameters $\phi_1 \in \mathbb{R}^l$

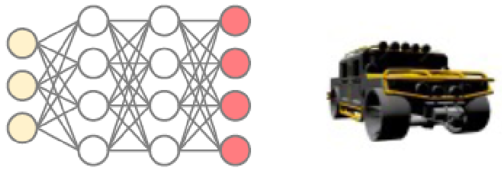


parameters $\phi_2 \in \mathbb{R}^l$



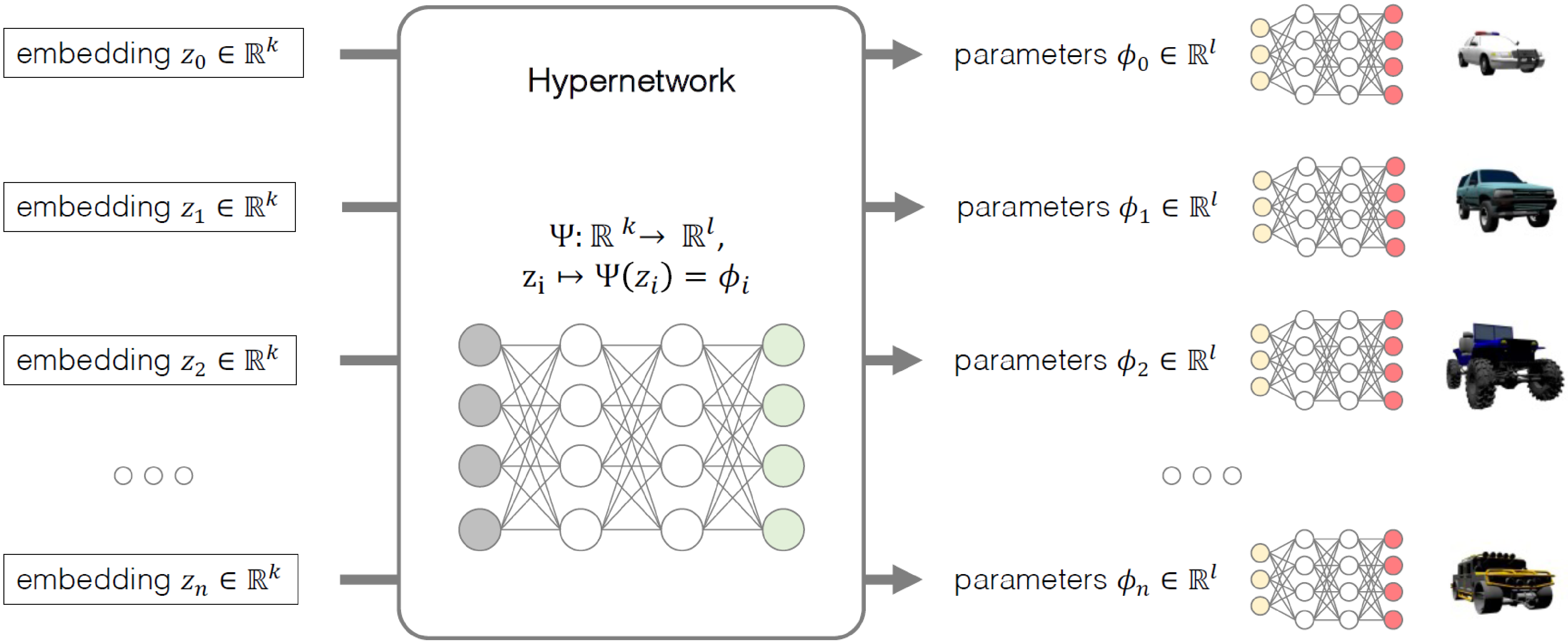
○○○

parameters $\phi_n \in \mathbb{R}^l$



Sitzmann et al., NeurIPS 2019

Map embeddings to SRN parameters via Hypernetwork.



Sitzmann et al., NeurIPS 2019

Novel View Synthesis – Baseline Comparison

Shapenet v2 cars – training set objects

Tatarchenko et al.

Worrall et al.

Deterministic GQN

SRNs

Training on:

- 2434 cars
- 50 observations each

Testing on:

- 2434 cars from training set
- 250 novel views rendered in Archimedean spiral around each object

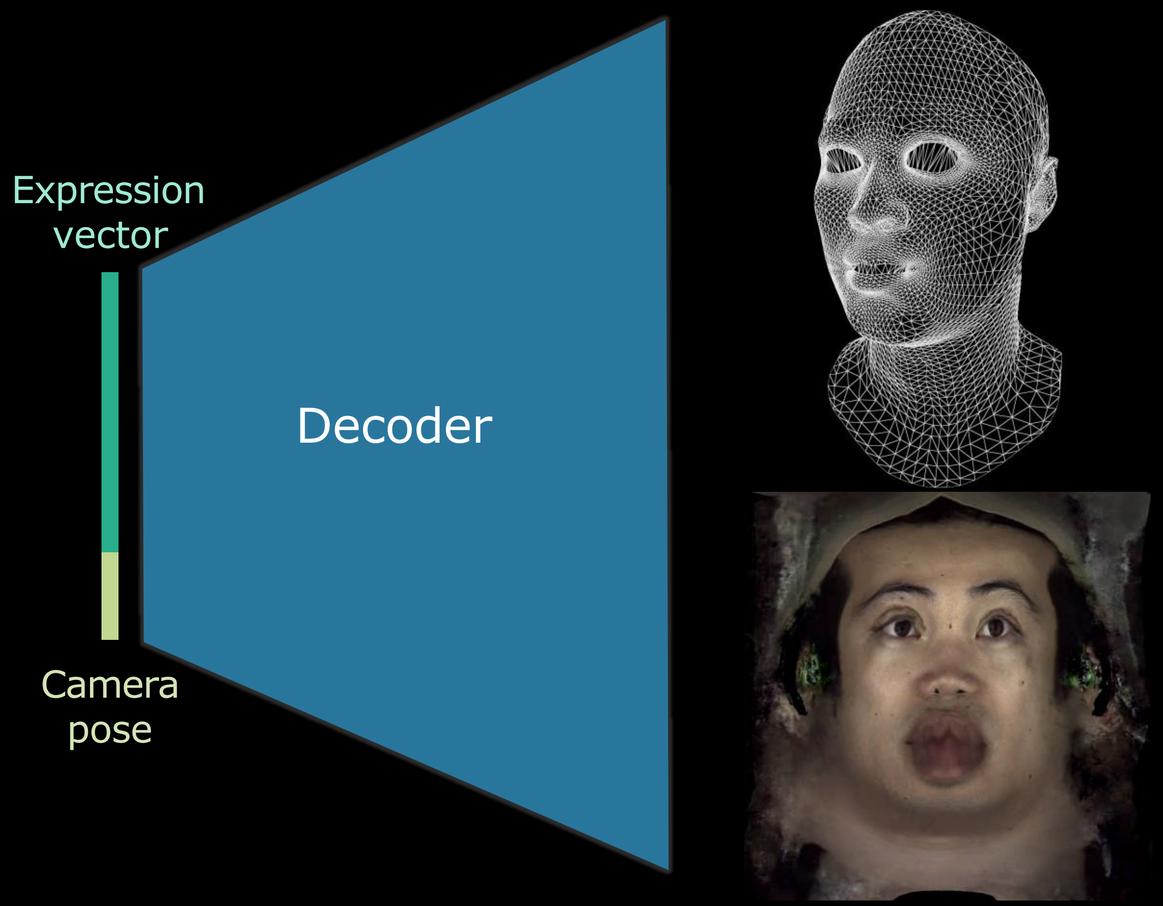
Neural Volumes

Lombardi, Simon, Saragih, Schwartz, Lehrmann, Sheikh

SIGGRAPH 2019

Deep Appearance Models

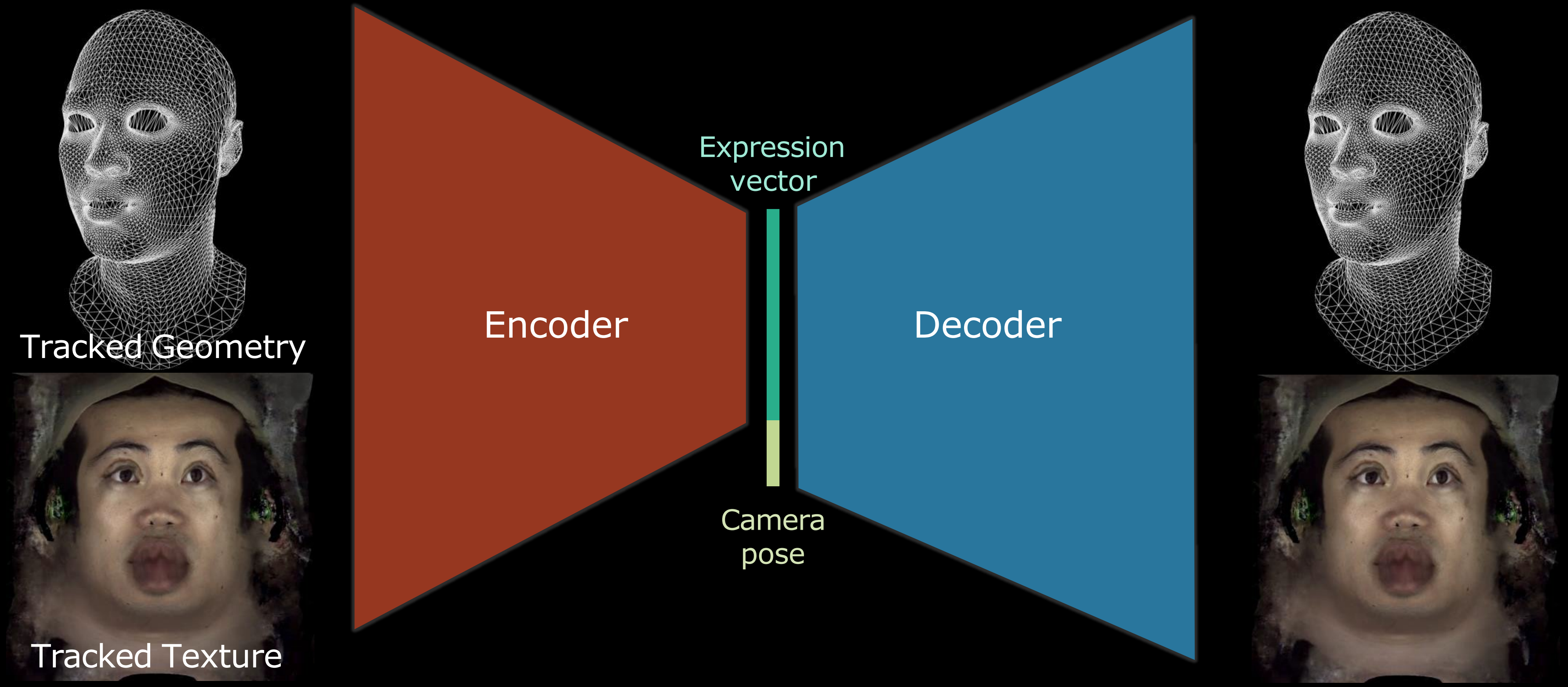
View-Conditioned Decoder



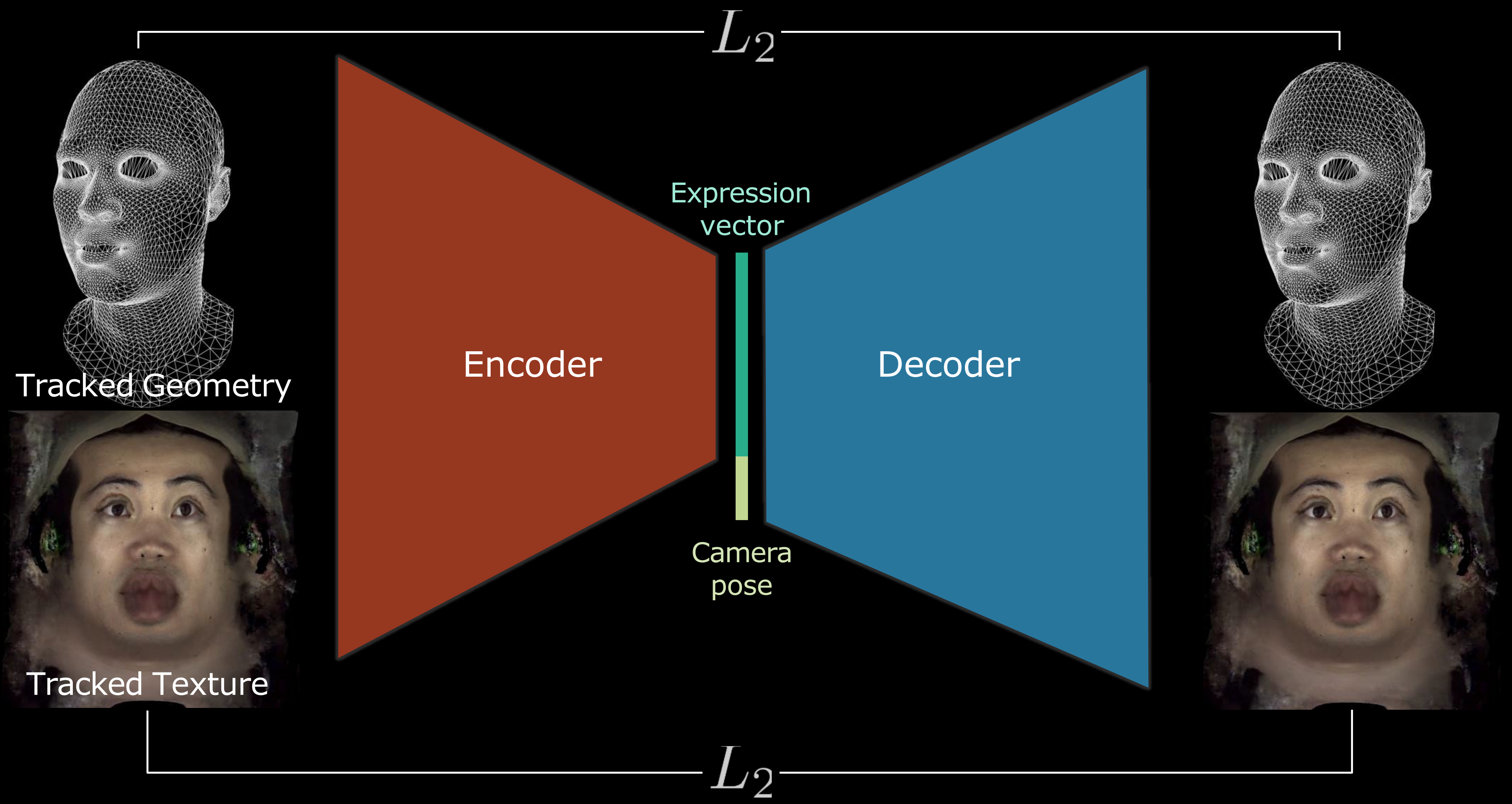
Lombardi, Saragih, Simon, Sheikh

SIGGRAPH 2018

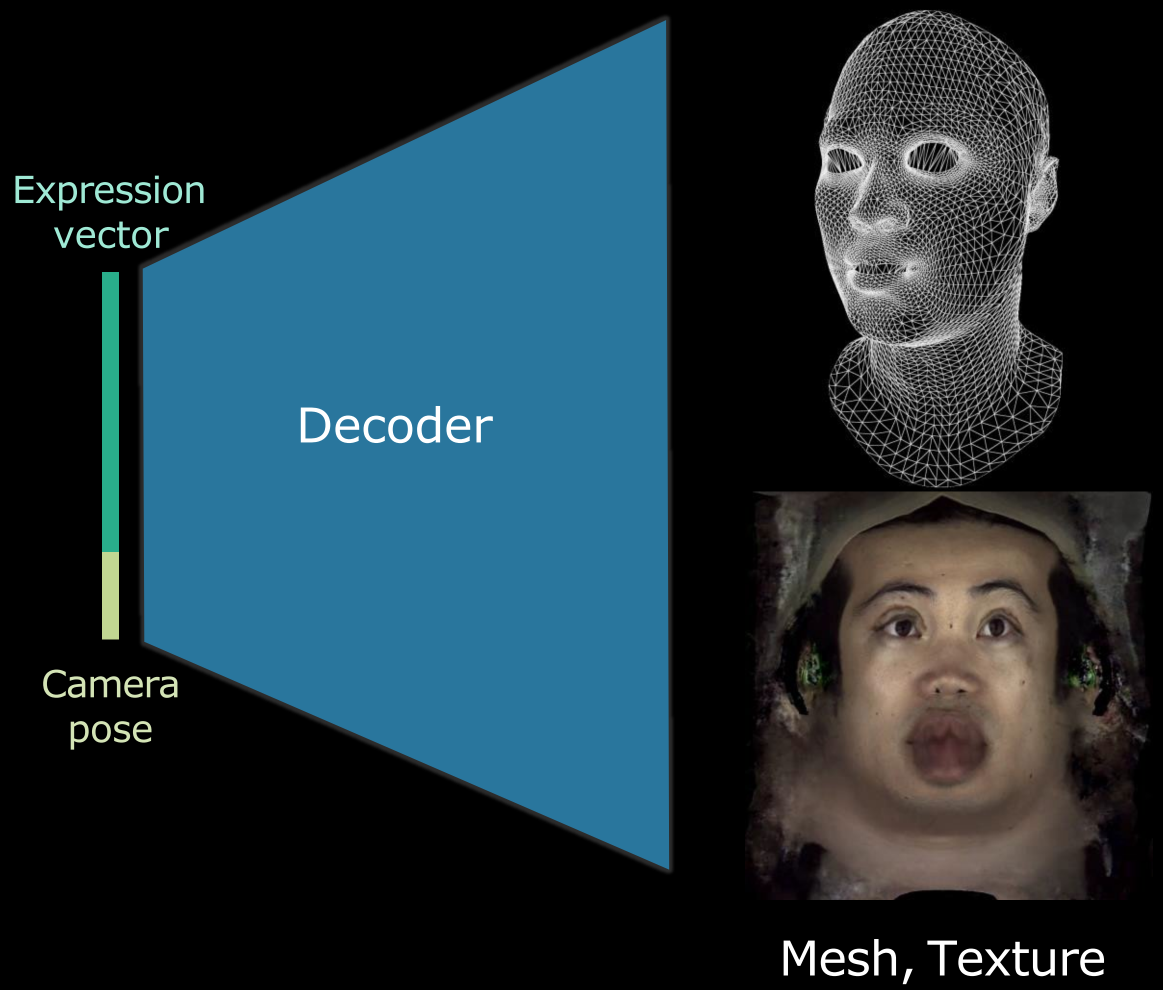
Deep Appearance Models



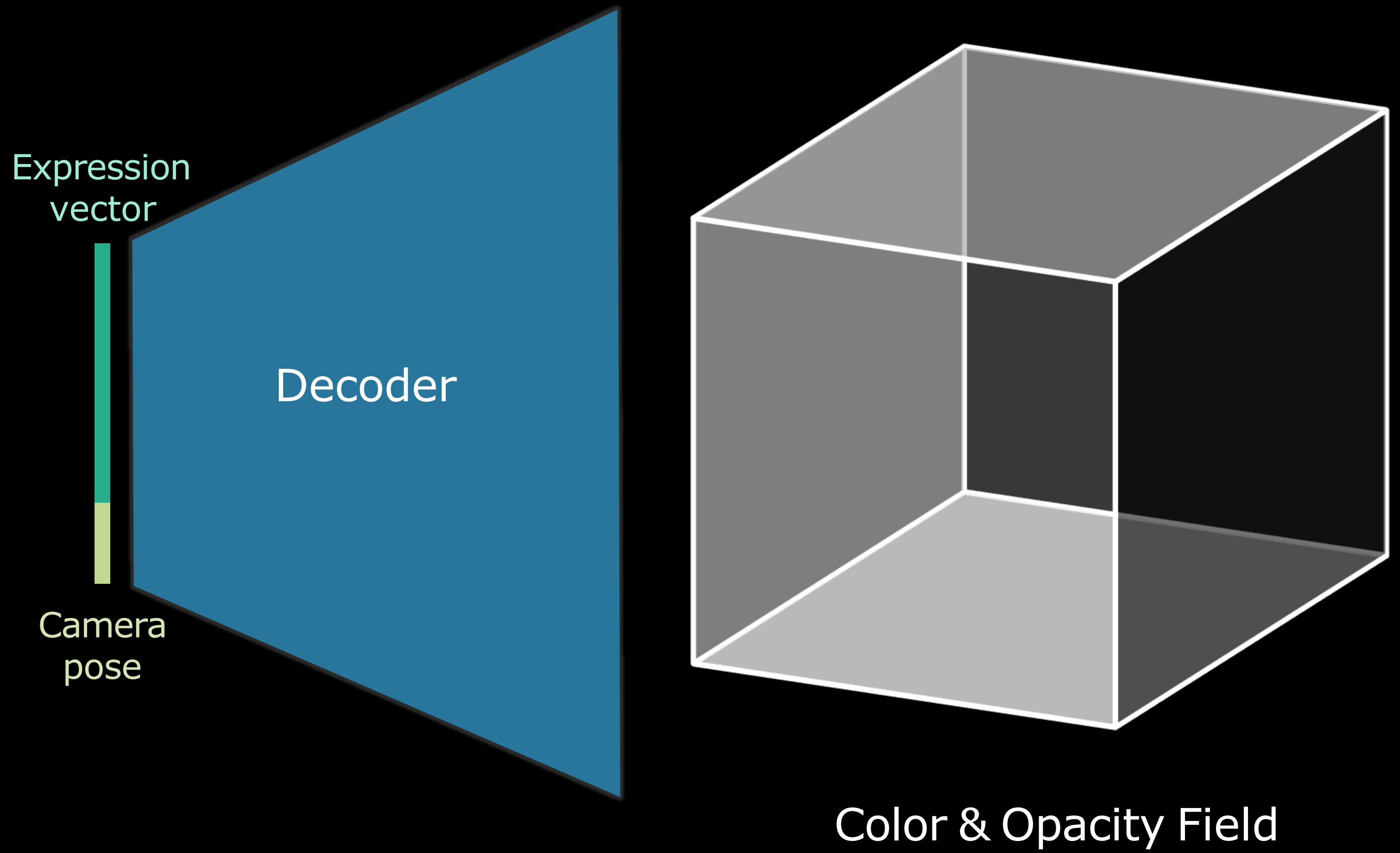
Deep Appearance Models



Mesh/Texture Decoder

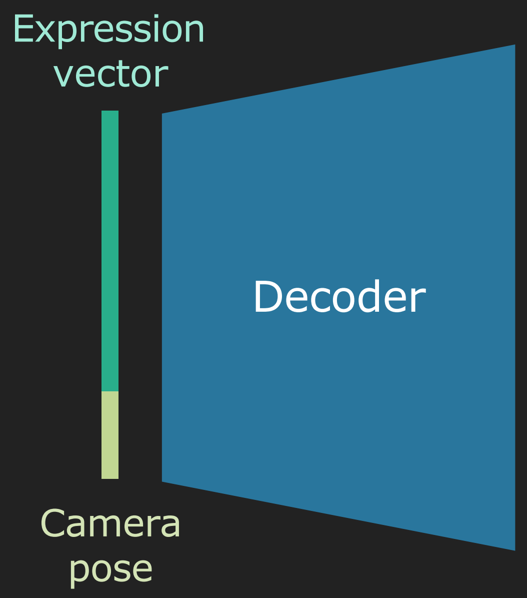


Volume Decoder

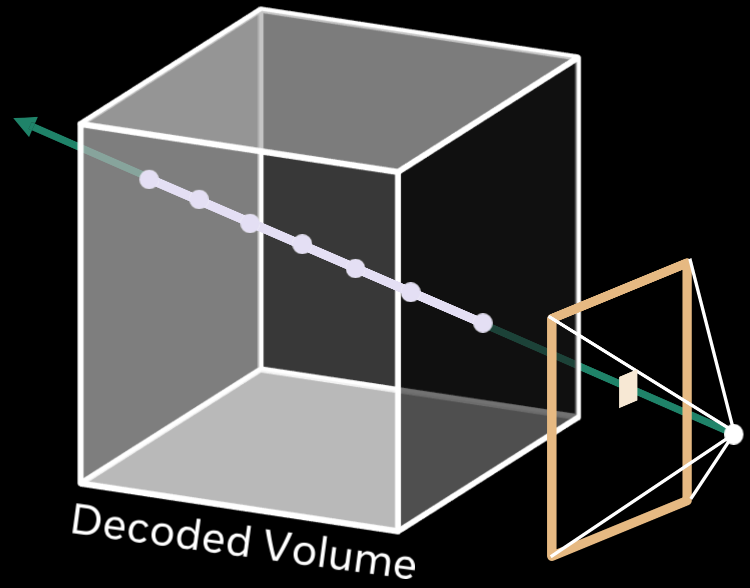


Volumetric Neural Rendering

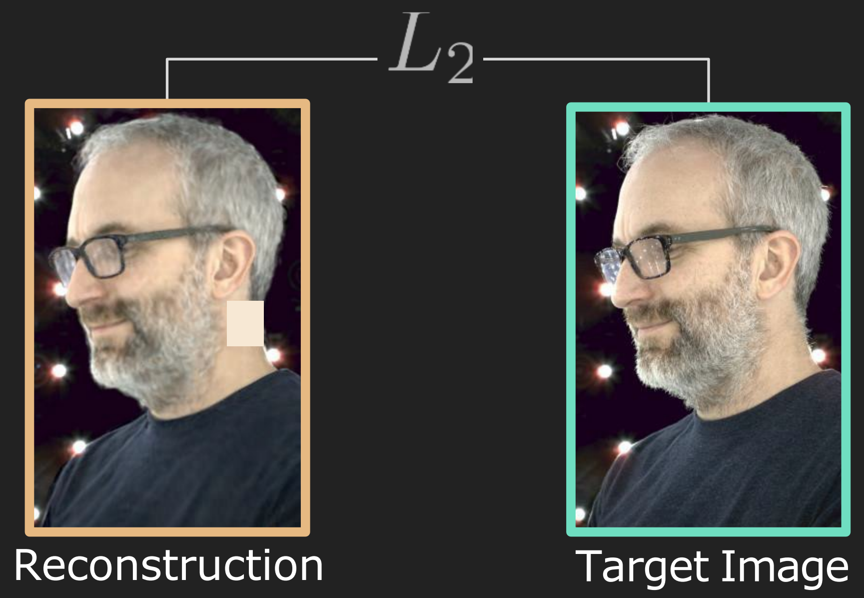
Decoding



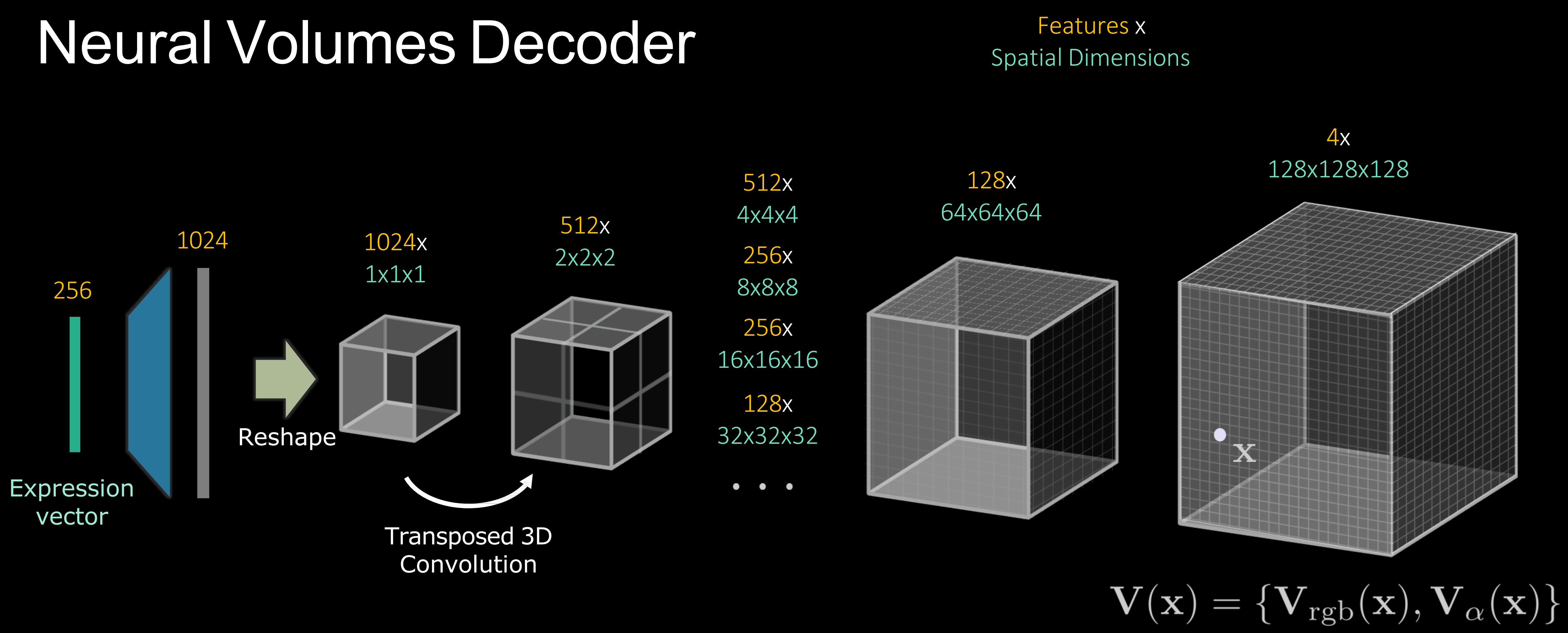
Raymarching



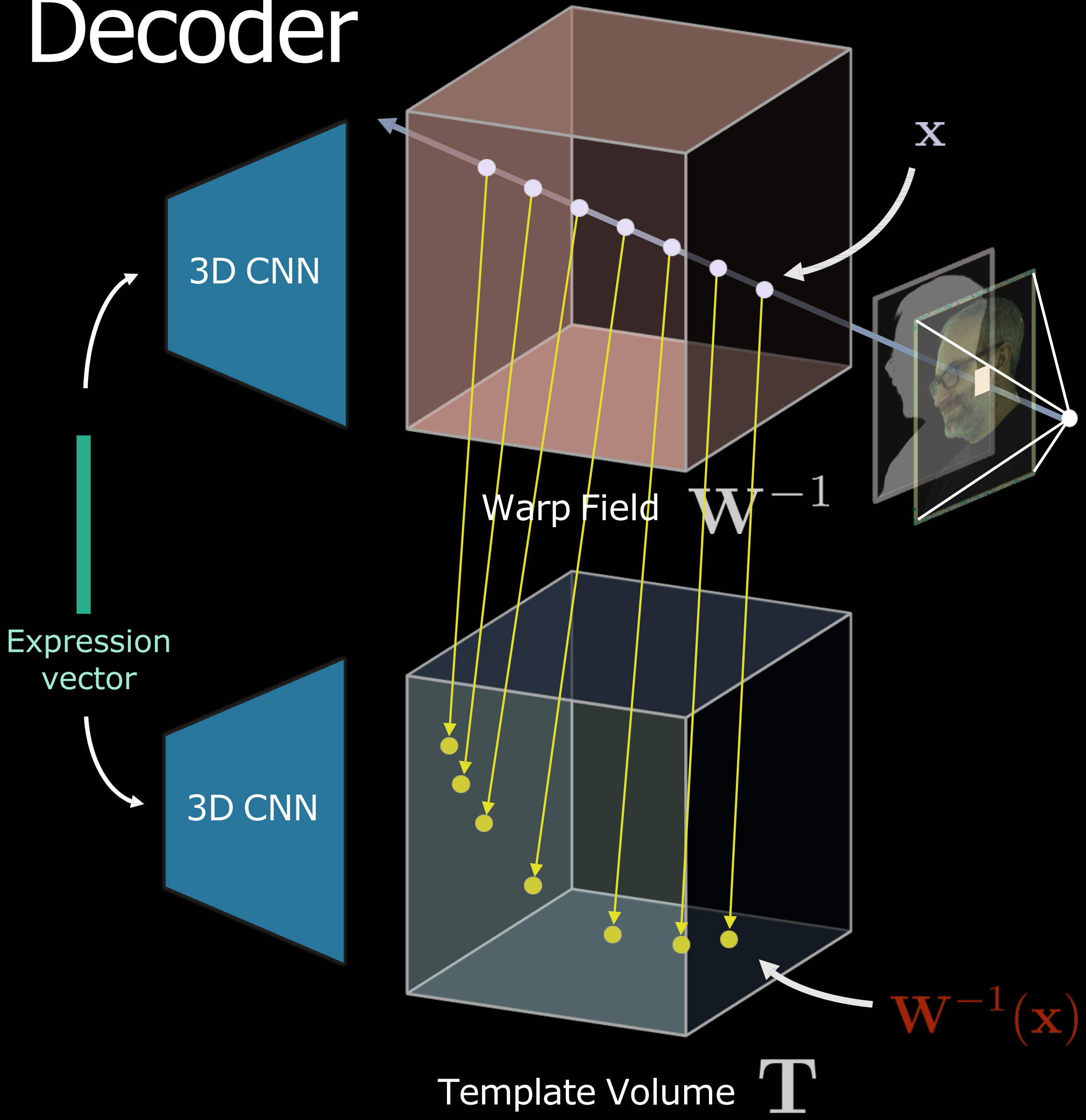
Training



Neural Volumes Decoder



Warp Field Decoder




Volume with Warping


Template

Example Reconstructions

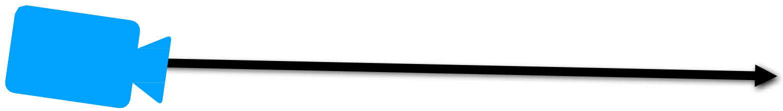
Neural Radiance Fields (NeRF)

Mildenhall, Srinivasan, Tancik, Barron, Ramamoorthi, Ng
ECCV 2020

Neural Volumetric Rendering

Neural Volumetric Rendering

querying the radiance value
along rays through 3D space

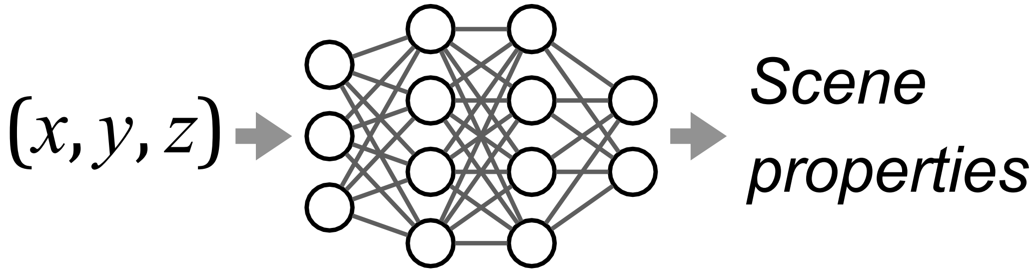


Neural Volumetric Rendering

continuous, differentiable
rendering model without
concrete ray/surface intersections

Neural Volumetric Rendering

using a neural network as a
scene representation, rather
than a voxel grid of data



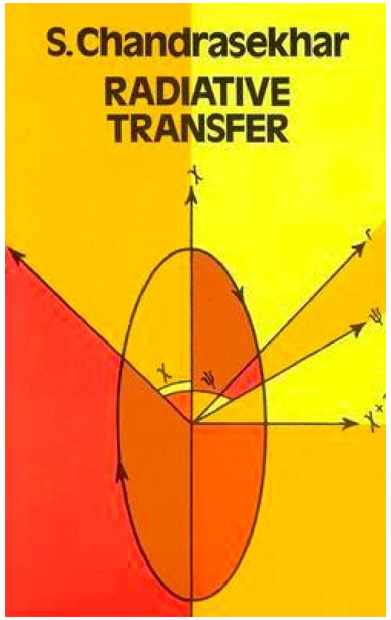
Inputs: sparse, unstructured photographs of a scene

Outputs: representation allowing us to render *new* views of that scene

Overview of NeRF

- Volumetric rendering math
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

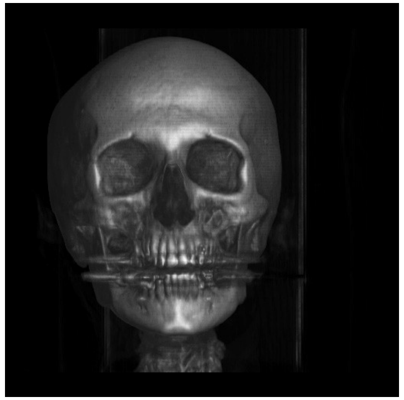
Traditional volumetric rendering



- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering
- Adapted for visualising medical data and linked with alpha compositing
- Modern path tracers use sophisticated Monte Carlo methods to render volumetric effects

Chandrasekhar 1950, *Radiative Transfer*
Kajia 1984, *Ray Tracing Volume Densities*
Levoy 1988, *Display of Surfaces from Volume Data*
Max 1995, *Optical Models for Direct Volume Rendering*
Porter and Duff 1984, *Compositing Digital Images*
Novak et al 2018, *Monte Carlo methods for physically based volume rendering*

Traditional volumetric rendering



Medical data visualisation [Levoy]

- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering
- Adapted for visualising medical data and linked with alpha compositing
- Modern path tracers use sophisticated Monte Carlo methods to render volumetric effects

Alpha compositing [Porter and Duff]

Chandrasekhar 1950, *Radiative Transfer*
Kajia 1984, *Ray Tracing Volume Densities*
Levoy 1988, *Display of Surfaces from Volume Data*
Max 1995, *Optical Models for Direct Volume Rendering*
Porter and Duff 1984, *Compositing Digital Images*
Novak et al 2018, *Monte Carlo methods for physically based volume rendering*

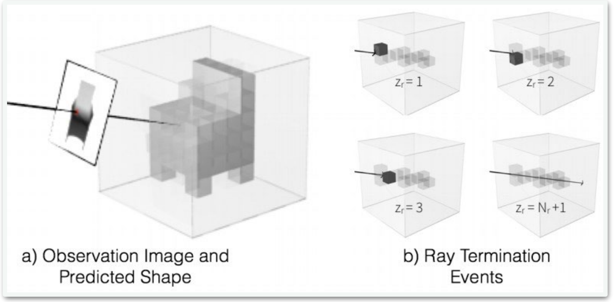
Traditional volumetric rendering

Physically-based Monte Carlo rendering [Novak et al]

- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering
- Adapted for visualising medical data and linked with alpha compositing
- Modern path tracers use sophisticated Monte Carlo methods to render volumetric effects

Chandrasekhar 1950, *Radiative Transfer*
Kajia 1984, *Ray Tracing Volume Densities*
Levoy 1988, *Display of Surfaces from Volume Data*
Max 1995, *Optical Models for Direct Volume Rendering*
Porter and Duff 1984, *Compositing Digital Images*
Novak et al 2018, *Monte Carlo methods for physically based volume rendering*

Volumetric rendering and machine learning



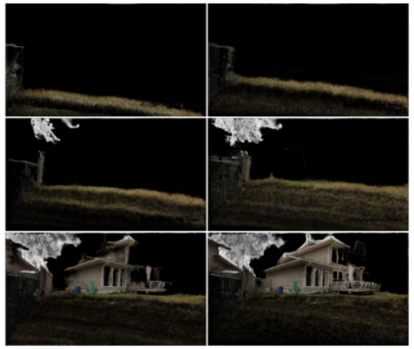
"Probabilistic" voxel grid rendering [Tulsiani et al]

- Various volume-rendering-esque methods devised for 3D shape reconstruction methods

- Scaled up to higher resolution volumes to achieve excellent view synthesis results

Tulsiani et al 2017, *Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency*
Henzler et al 2019, *Escaping Plato's Cave: 3D Shape From Adversarial Rendering*
Zhou et al 2018, *Stereo Magnification: Learning View Synthesis using Multiplane Images*
Lombardi et al 2019, *Neural Volumes: Learning Dynamic Renderable Volumes from Images*

Volumetric rendering and machine learning



Slices from a volumetric scene representation [Zhou et al]

View synthesis from a dynamic voxel grid [Lombardi et al]

- Various volume-rendering-esque methods devised for 3D shape reconstruction methods
- Scaled up to higher resolution voxel grids, ML methods can achieve excellent view synthesis results

Tulsiani et al 2017, *Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency*

Henzler et al 2019, *Escaping Plato's Cave: 3D Shape From Adversarial Rendering*

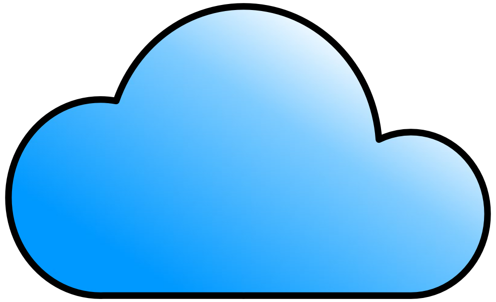
Zhou et al 2018, *Stereo Magnification: Learning View Synthesis using Multiplane Images*

Lombardi et al 2019, *Neural Volumes: Learning Dynamic Renderable Volumes from Images*

Volumetric formulation for NeRF

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

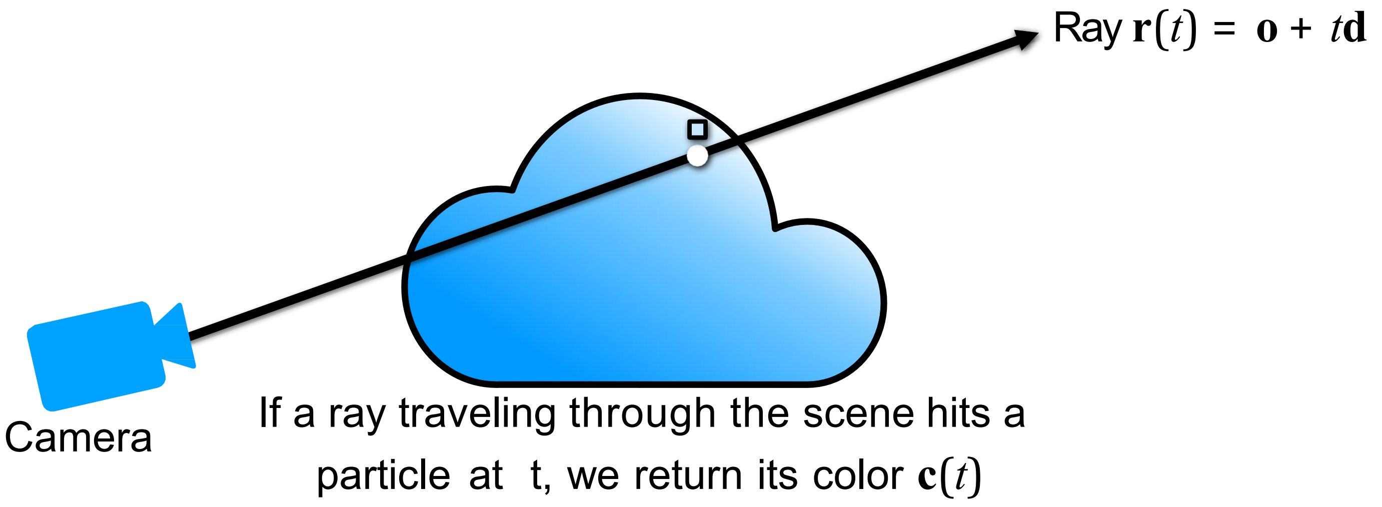
Volumetric formulation for NeRF



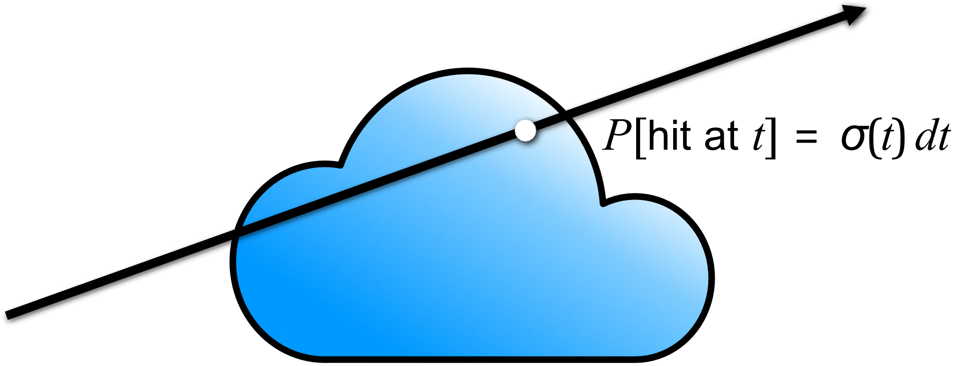
Scene is a cloud of tiny colored particles

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

Volumetric formulation for NeRF



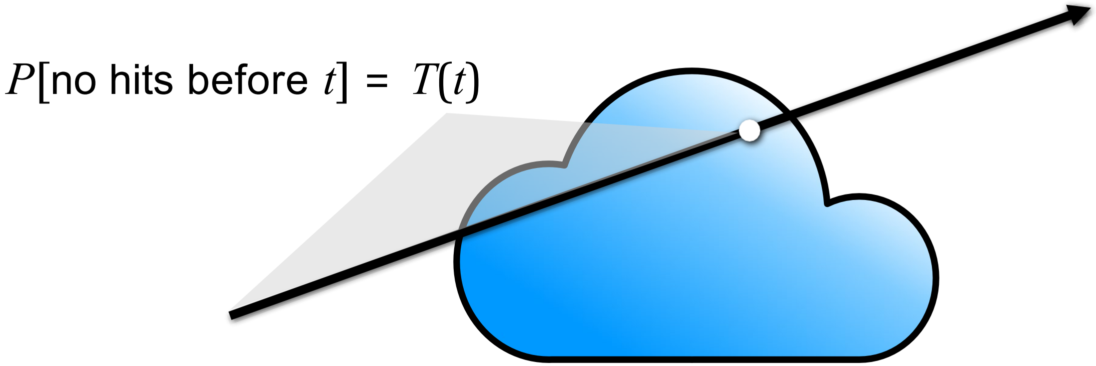
Volumetric formulation for NeRF



This notion is *probabilistic:* chance that ray stops in a small interval around t is $\sigma(t)\,dt$.

$\sigma(t)$ is known as the "volume density"

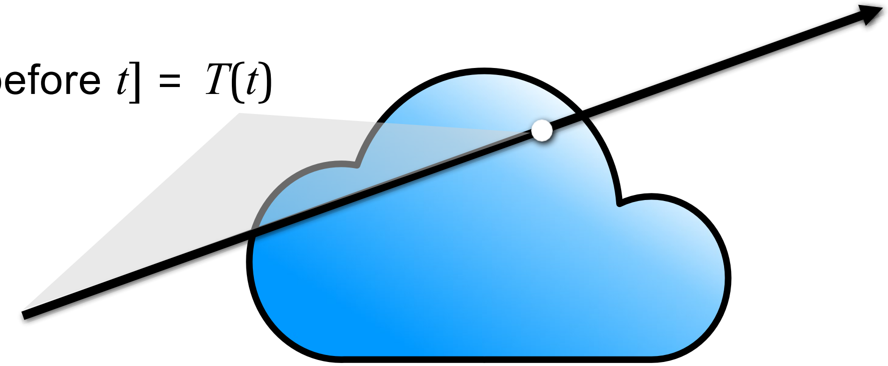
Volumetric formulation for NeRF



To determine if t is the *first* hit, need to know $T(t)$:
probability that the ray didn't hit any particles earlier.
$T(t)$ is called "transmittance"

Volumetric formulation for NeRF

$P[\text{no hits before } t] = T(t)$

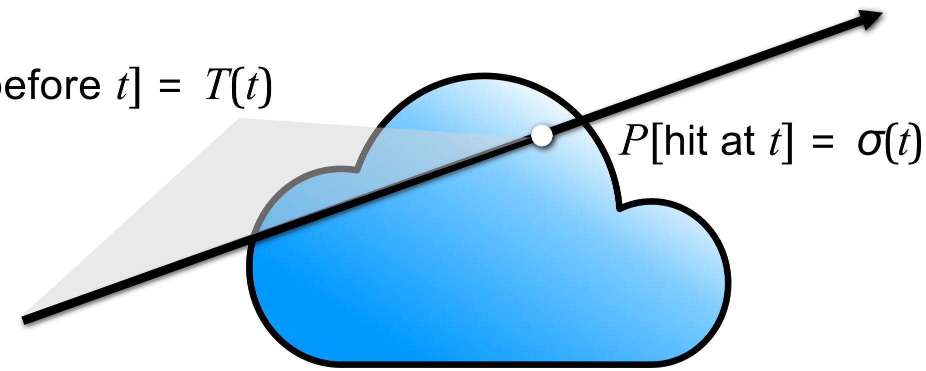


To determine if t is the *first* hit, need to know $T(t)$: probability that the ray didn't hit any particles earlier.

$T(t)$ is called "transmittance"

We assume σ is known and want to use it to calculate $T(t)$

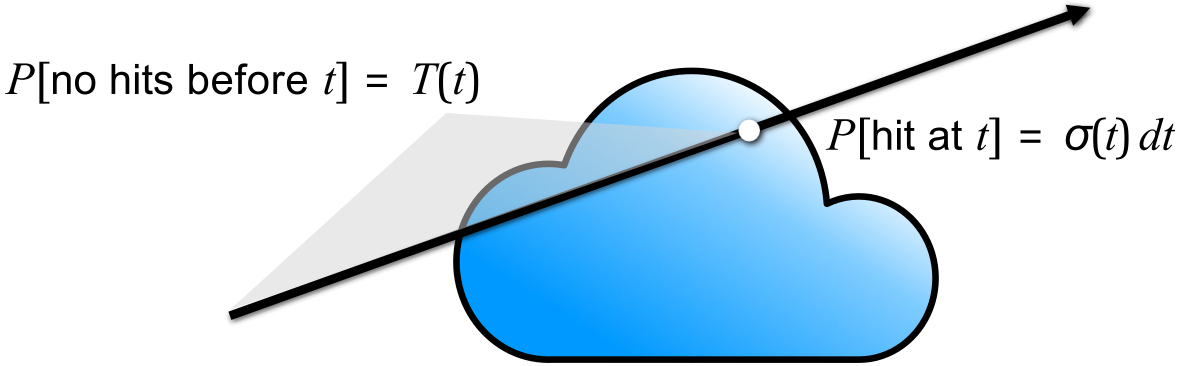
Volumetric formulation for NeRF



σ and T are related by the probability fact that

P[no hits before $t + dt$] = P[no hits before t] $\times P$[no hit at t]

Volumetric formulation for NeRF



These are related by the probability fact that

$$T(t + dt) = T(t) \times (1 - \sigma(t)dt)$$

Volumetric formulation for NeRF

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

Volumetric formulation for NeRF

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

Split up differential $\Rightarrow$ $$T(t) + T'(t)dt = T(t) - T(t)\sigma(t)dt$$

Volumetric formulation for NeRF

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

Split up differential $\Rightarrow$ $T(t) + T'(t)dt = T(t) - T(t)\sigma(t)dt$

Rearrange $\Rightarrow$ $\dfrac{T'(t)}{T(t)}dt = -\sigma(t)dt$

Volumetric formulation for NeRF

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

Split up differential $\Rightarrow$ $T(t) + T'(t)dt = T(t) - T(t)\sigma(t)dt$

Rearrange $\Rightarrow$ $\frac{T'(t)}{T(t)}dt = -\sigma(t)dt$

Integrate $\Rightarrow$ $\log T(t) = -\int_{t_0}^{t} \sigma(s)ds$

Volumetric formulation for NeRF

Thus, the probability that a ray first hits a particle at t is

$$T(t)\sigma(t)\,dt = \exp\left(-\int_{t_0}^{t}\sigma(t)\right)\sigma(t)\,dt$$

Volumetric formulation for NeRF

Thus, the probability that a ray first hits a particle at t is

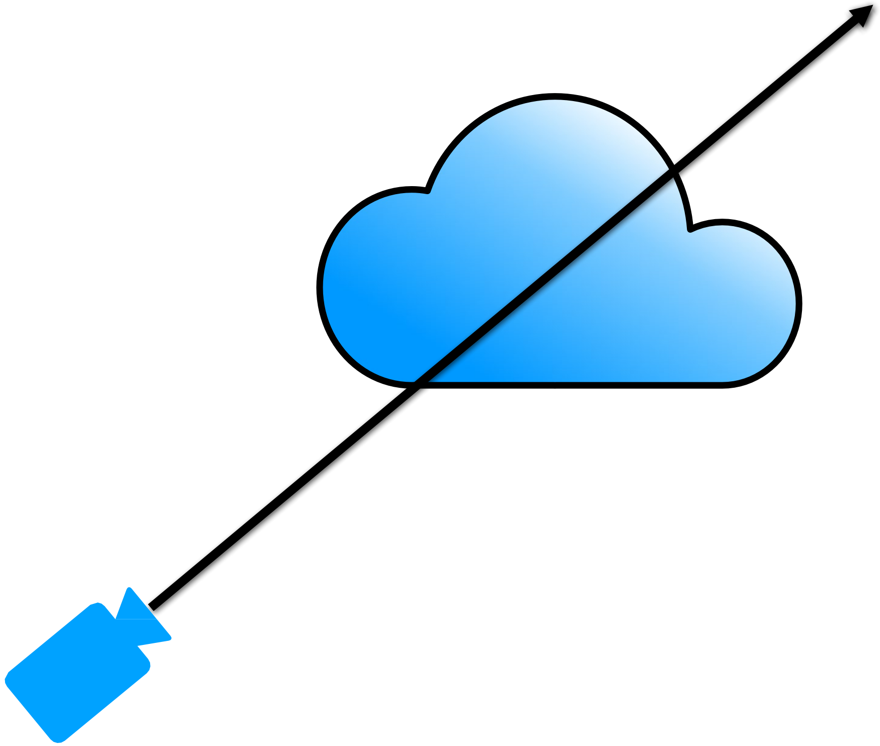
$$T(t)\sigma(t)\,dt = \exp\left(-\int_{t_0}^{t} \sigma(t)\right)\sigma(t)\,dt$$

And expected color returned by the ray will be

$$\int_{t_0}^{t_1} T(t)\sigma(t)\mathbf{c}(t)\,dt$$

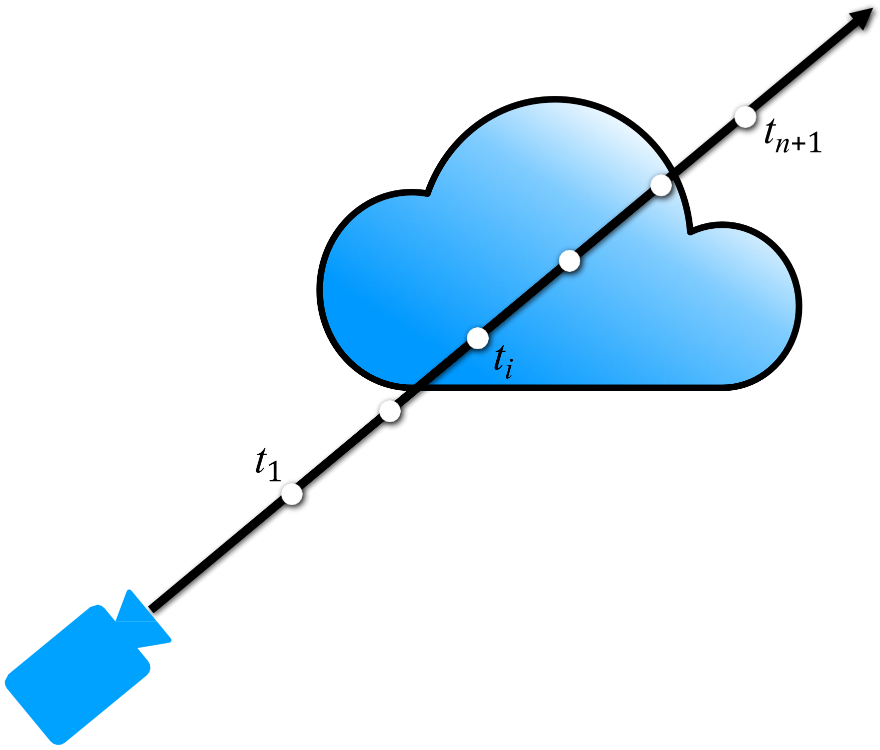
Note the nested integral!

Approximating the nested integral



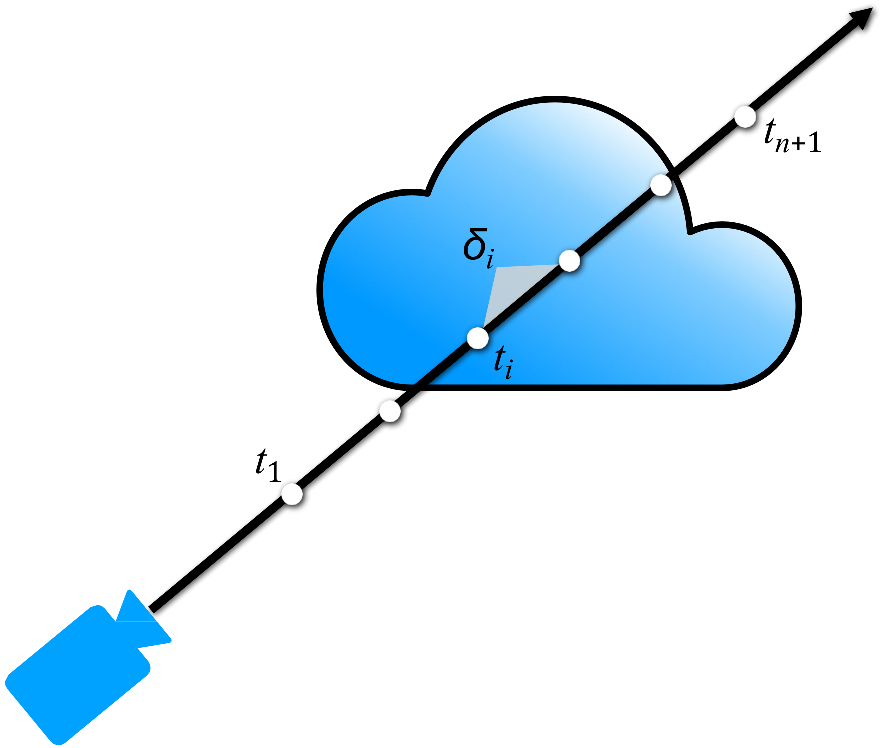
We use quadrature to approximate the nested integral,

Approximating the nested integral



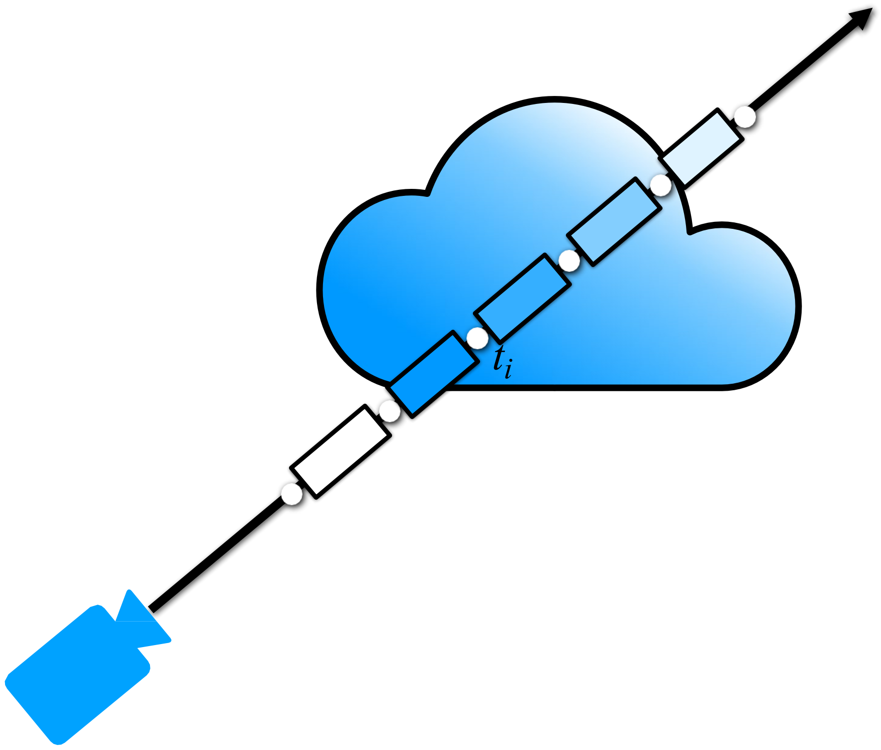
We use quadrature to approximate the nested integral,
splitting the ray up into n segments with endpoints $\{t_1, t_2, \ldots, t_{n+1}\}$

Approximating the nested integral



We use quadrature to approximate the nested integral,
splitting the ray up into n segments with endpoints $\{t_1, t_2, \ldots, t_{n+1}\}$
with lengths $\delta_i = t_{i+1} - t_i$

Approximating the nested integral



We assume volume density and color are roughly constant within each interval

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt$$

This allows us to break the outer integral

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i\mathbf{c}_i\,dt$$

This allows us to break the outer integral
into a sum of analytically tractable integrals

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} \boxed{T(t)}\sigma_i \mathbf{c}_i \,dt$$

Catch: piecewise constant density and color

do not imply constant transmittance!

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\, dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} \boxed{T(t)}\,\sigma_i \mathbf{c}_i \, dt$$

Catch: piecewise constant density and color
do not imply constant transmittance!

Important to account for how early part of a
segment blocks later part when σ_i is high

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i\,dt$$

For $t \in [t_i, t_{i+1}]$, $T(t) = \exp\left(-\int_{t_1}^{t_i} \sigma_i\,ds\right) \exp\left(-\int_{t_i}^{t} \sigma_i\,ds\right)$

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i\,dt$$

$$\text{For } t \in [t_i, t_{i+1}],\ T(t) = \exp\left(-\int_{t_1}^{t_i} \sigma_i\,ds\right)\exp\left(-\int_{t_i}^{t} \sigma_i\,ds\right)$$

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

"How much is blocked by all previous segments?"

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i\,dt$$

$$\text{For } t \in [t_i, t_{i+1}],\ T(t) = \exp\left(-\int_{t_1}^{t_i} \sigma_i\,ds\right) \exp\left(-\int_{t_i}^{t} \sigma_i\,ds\right)$$

"How much is blocked partway through the current segment?"

$$\exp\left(-\sigma_i(t - t_i)\right)$$

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i\,dt$$

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i\,dt$$

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

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i\,dt$$

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

Integrate

$$= \sum_{i=1}^{n} T_i \sigma_i \mathbf{c}_i \frac{\exp\left(-\sigma_i(t_{i+1} - t_i)\right) - 1}{-\sigma_i}$$

Approximating the nested integral

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i\mathbf{c}_i\,dt$$

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

$$= \sum_{i=1}^{n} T_i\sigma_i\mathbf{c}_i\,\frac{\exp\left(-\sigma_i(t_{i+1}-t_i)\right)-1}{-\sigma_i}$$

Cancel σ_i

$$= \sum_{i=1}^{n} T_i\mathbf{c}_i\,(1-\exp(-\sigma_i\delta_i))$$

Connection to alpha compositing

$$\alpha_i = 1 - \exp(-\sigma_i\delta_i)$$

$$= \sum_{i=1}^{n} T_i \mathbf{c}_i (1 - \exp(-\sigma_i\delta_i))$$

Connection to alpha compositing

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i) \quad \Rightarrow$$

$$\text{color} = \sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i = \sum_{i=1}^{n} T_i \mathbf{c}_i \left(1 - \exp(-\sigma_i \delta_i)\right)$$

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

$$= \sum_{i=1}^{n} T_i \mathbf{c}_i \left(1 - \exp(-\sigma_i \delta_i)\right)$$

Summary: volume rendering integral estimate

Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

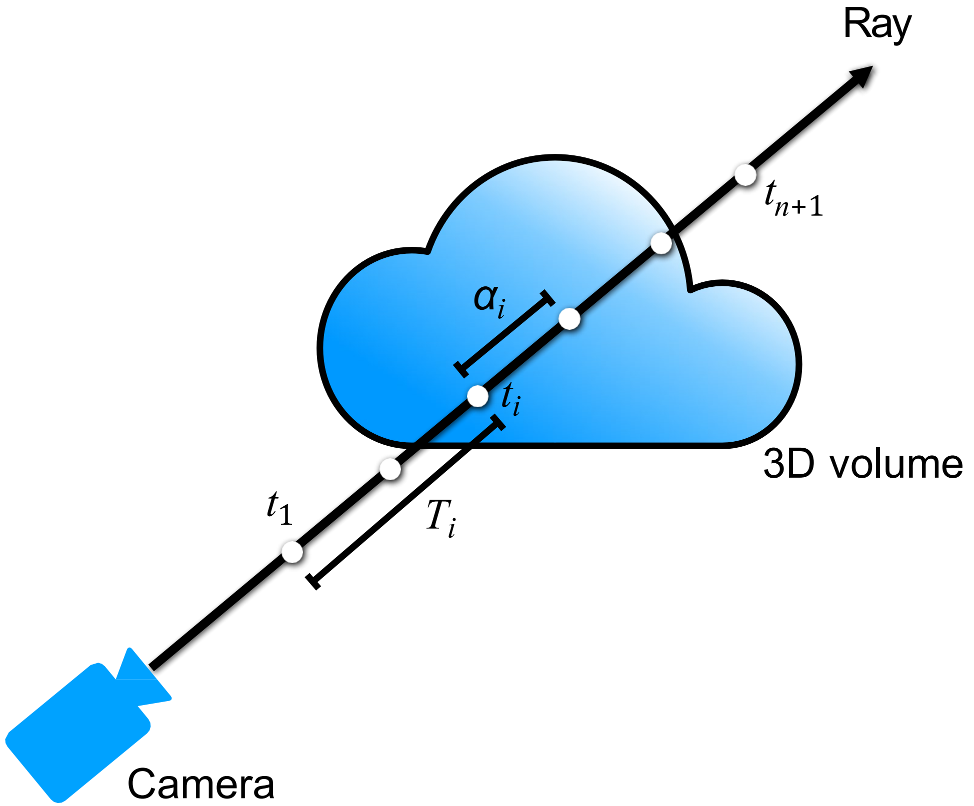
$$\mathbf{c} \approx \sum_{i=1}^{n} \underbrace{T_i \alpha_i}_{\text{weights}} \underbrace{\mathbf{c}_i}_{\text{colors}}$$

How much light is blocked earlier along ray:

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

How much light is contributed by ray segment i:

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$



Summary: volume rendering integral estimate

Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

$$\mathbf{c} \approx \sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i$$

weights

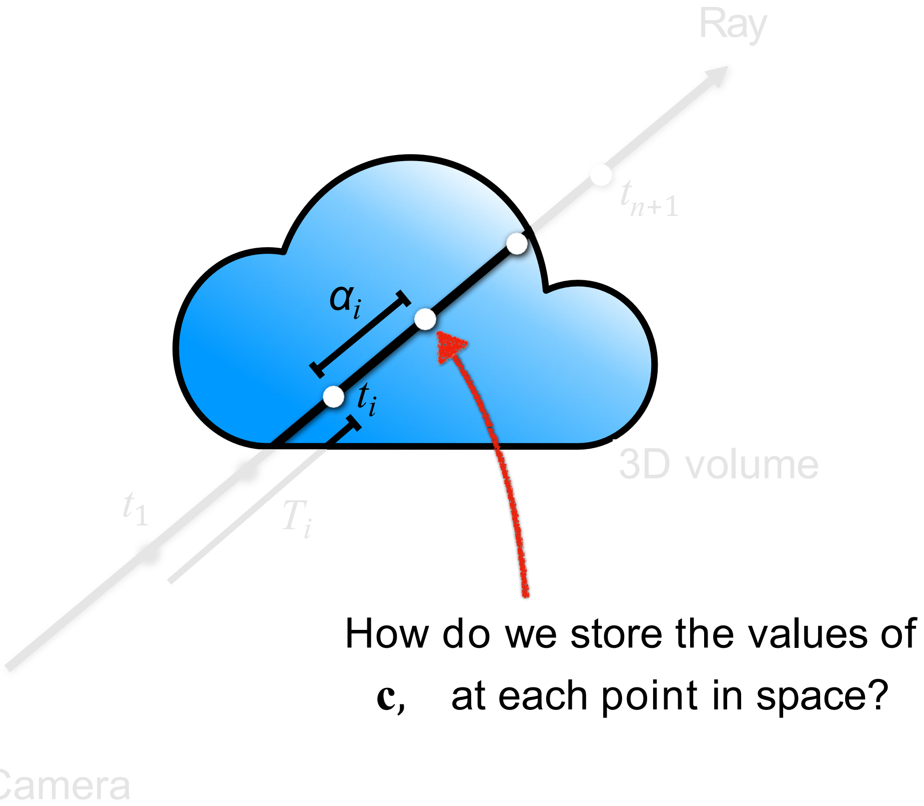
colors

How much light is blocked earlier along ray:

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

How much light is contributed by ray segment i:

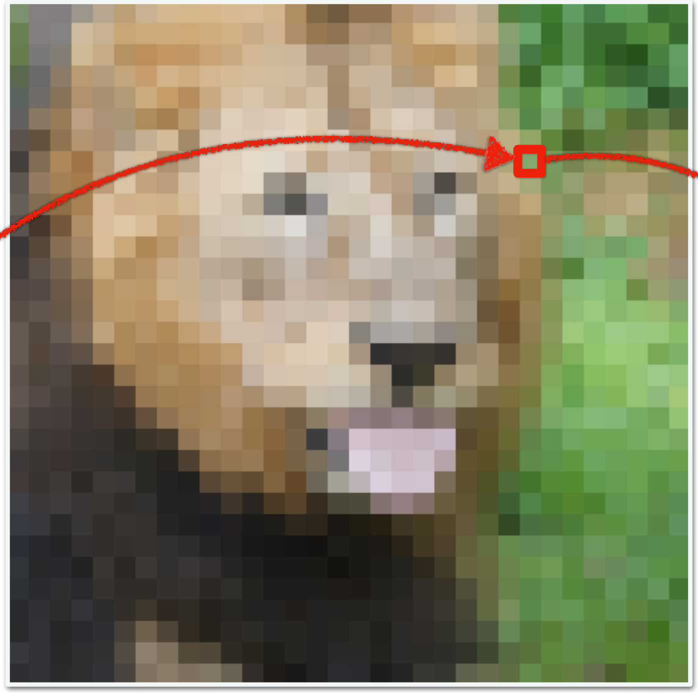
$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$



Overview

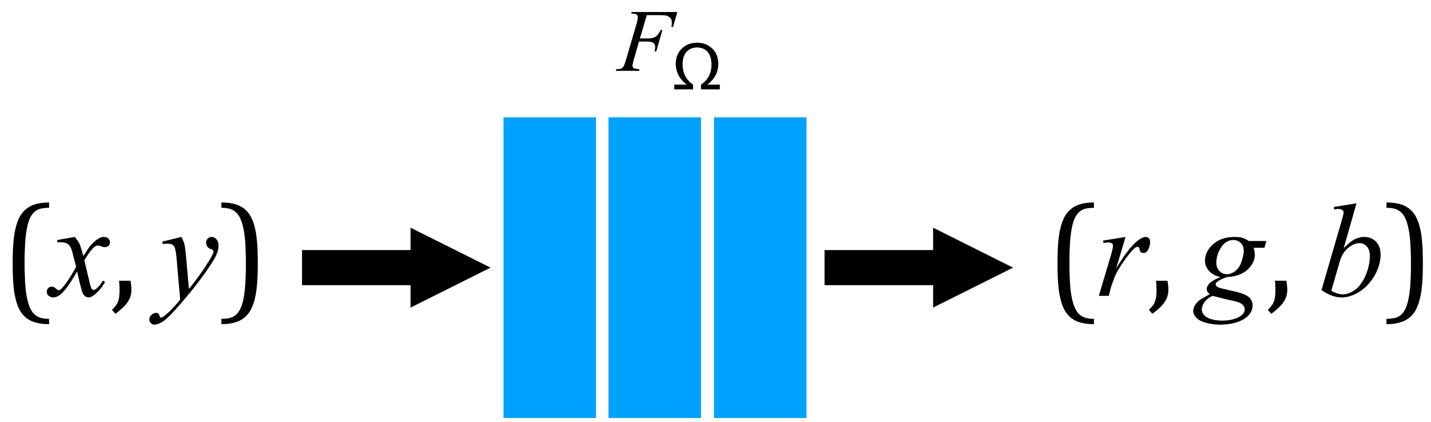
- Volumetric rendering math
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

Toy problem: storing 2D image data

(x, y)  (r, g, b)

Usually we store an image as a
2D grid of RGB color values

Toy problem: storing 2D image data

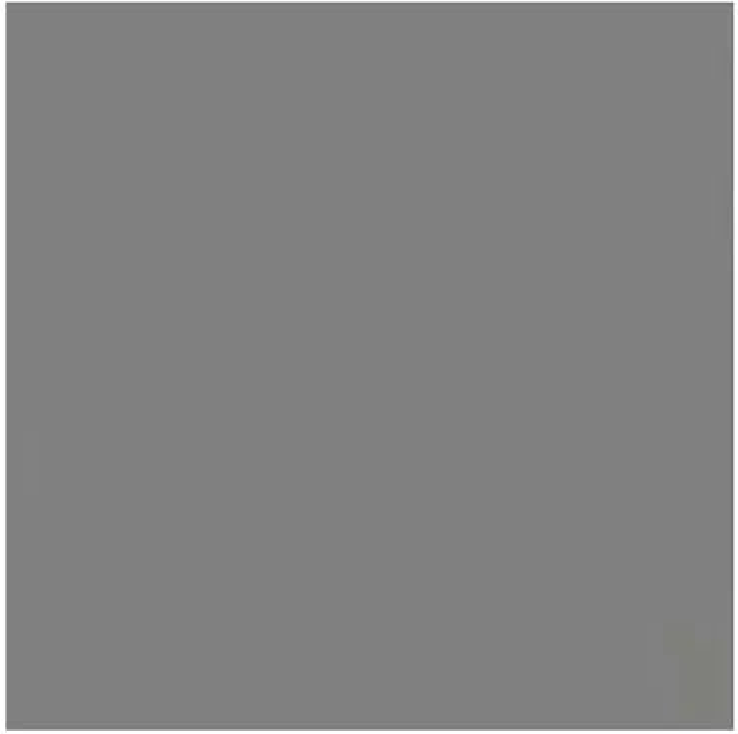


What if we train a simple fully-connected network (MLP) to do this instead?

Naive approach fails!

Ground truth image

Standard fully-connected net



Problem:

“Standard” coordinate-based MLPs cannot represent high-frequency functions

Solution:

Pass input coordinates through a
high frequency mapping first

Input coordinate mapping

- Simple formula: apply a tall skinny matrix **B** to input coordinate vector **x**, then pass through *sin* and *cos*:

$$\gamma(\mathbf{x}) = (\sin(2\pi\mathbf{B}\mathbf{x}), \cos(2\pi\mathbf{B}\mathbf{x}))$$

- Passing network a subset of the Fourier basis functions. Same effect from:
  - Positional encoding
  - Fourier features
  - SIREN

Problem solved

Ground truth image

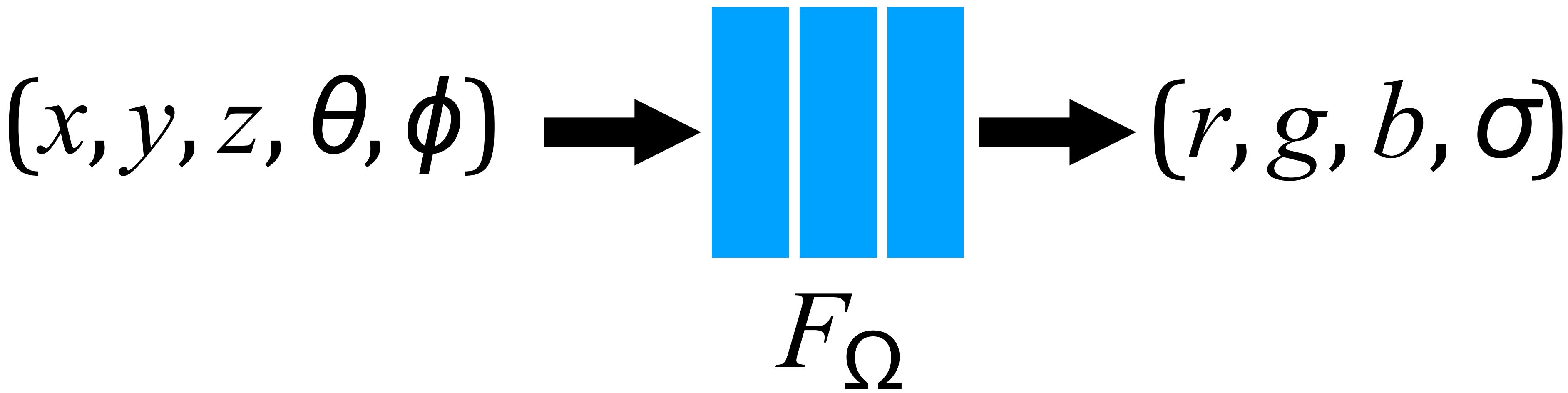
Standard fully-connected net

With "positional encoding"

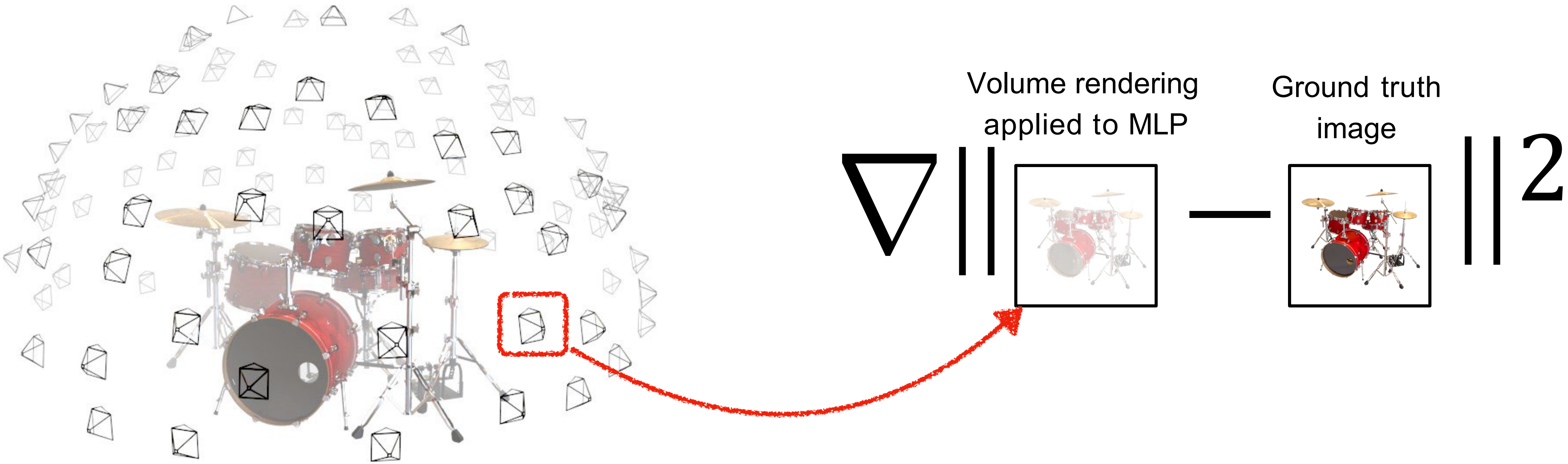
Overview

- Volumetric rendering math
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

NeRF = volume rendering + coordinate-based network


(x, y, z, θ, ϕ)
F_Ω
(r, g, b, σ)

Train network to reproduce input views of scene using gradient descent



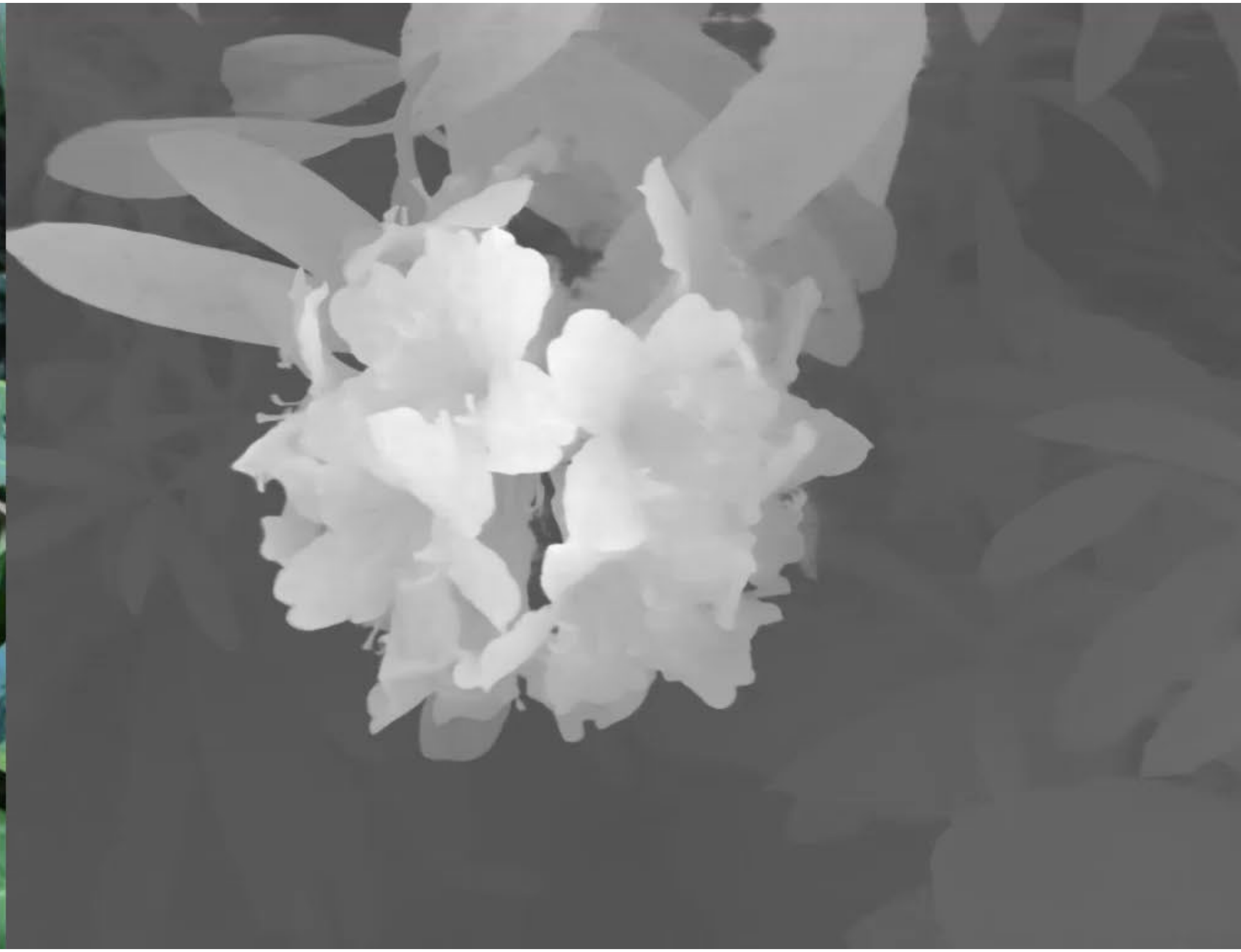
Visualizing view-dependent effects

Regular NeRF rendering

Manipulating input viewing directions

Visualizing learned density field as geometry

Regular NeRF rendering



Expected ray termination depth

Visualizing learned density field as geometry

Regular NeRF rendering

Expected ray termination depth

Acknowledgments

- Advances in Neural Rendering
- Neural Fields in Visual Computing and Beyond
- awesome-NeRF: a curated list of awesome neural radiance fields papers
- MPII Summer Semester 2023: Computer Vision and Machine Learning for Computer Graphics
- Lingqi Yan’s Slides for Rendering

Any Questions?