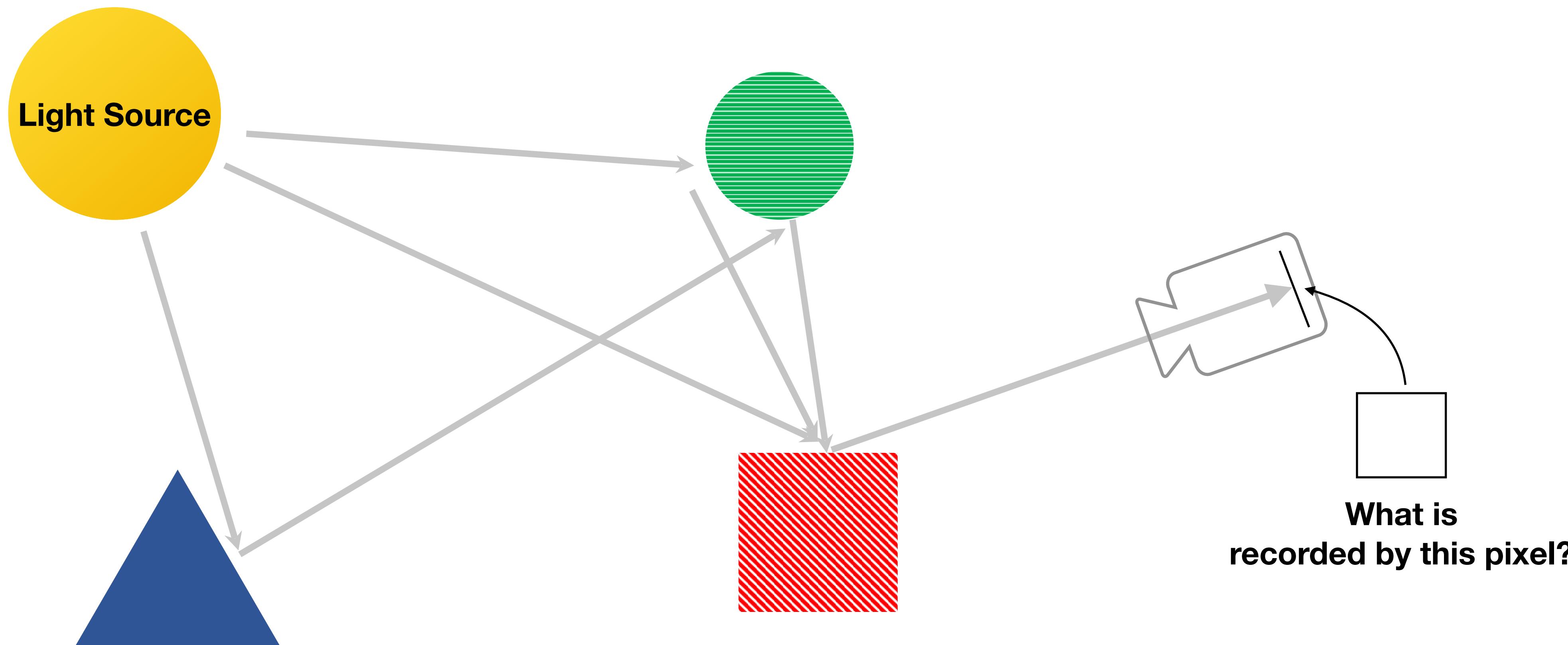


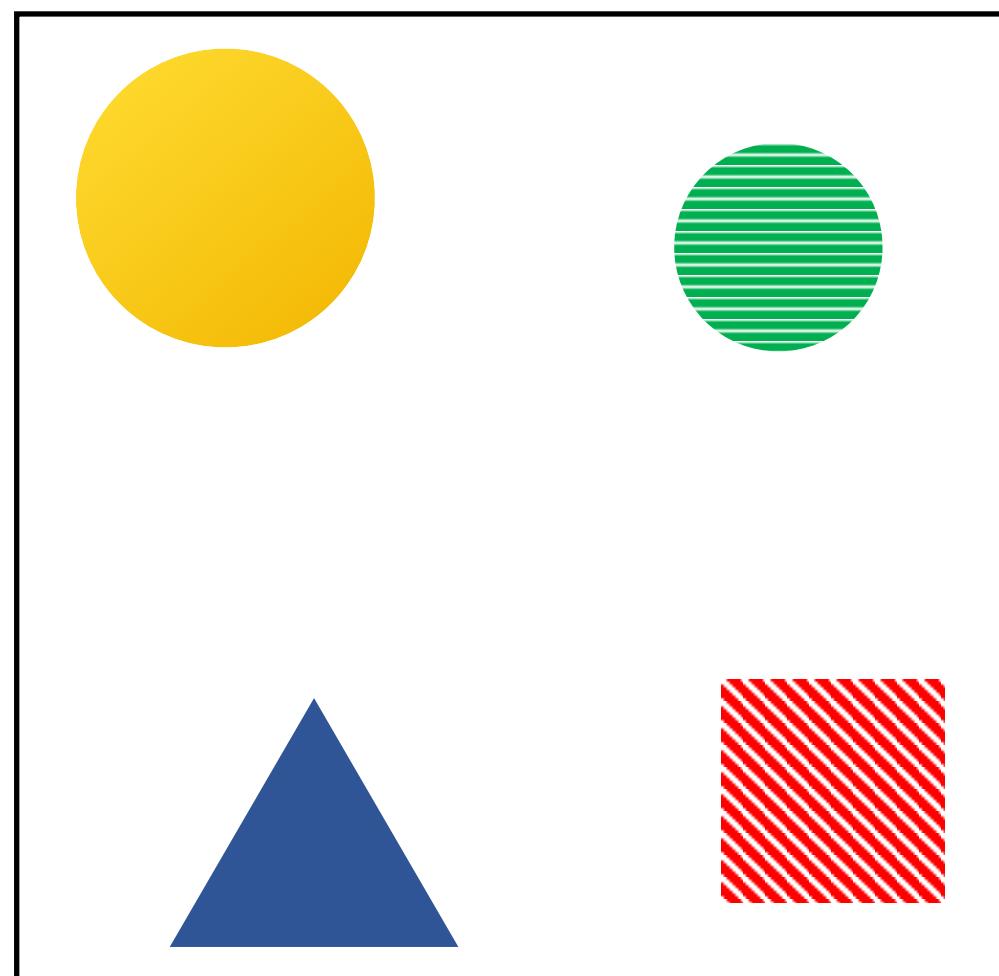
RENDERING

Inverse Graphics & Differentiable Rendering

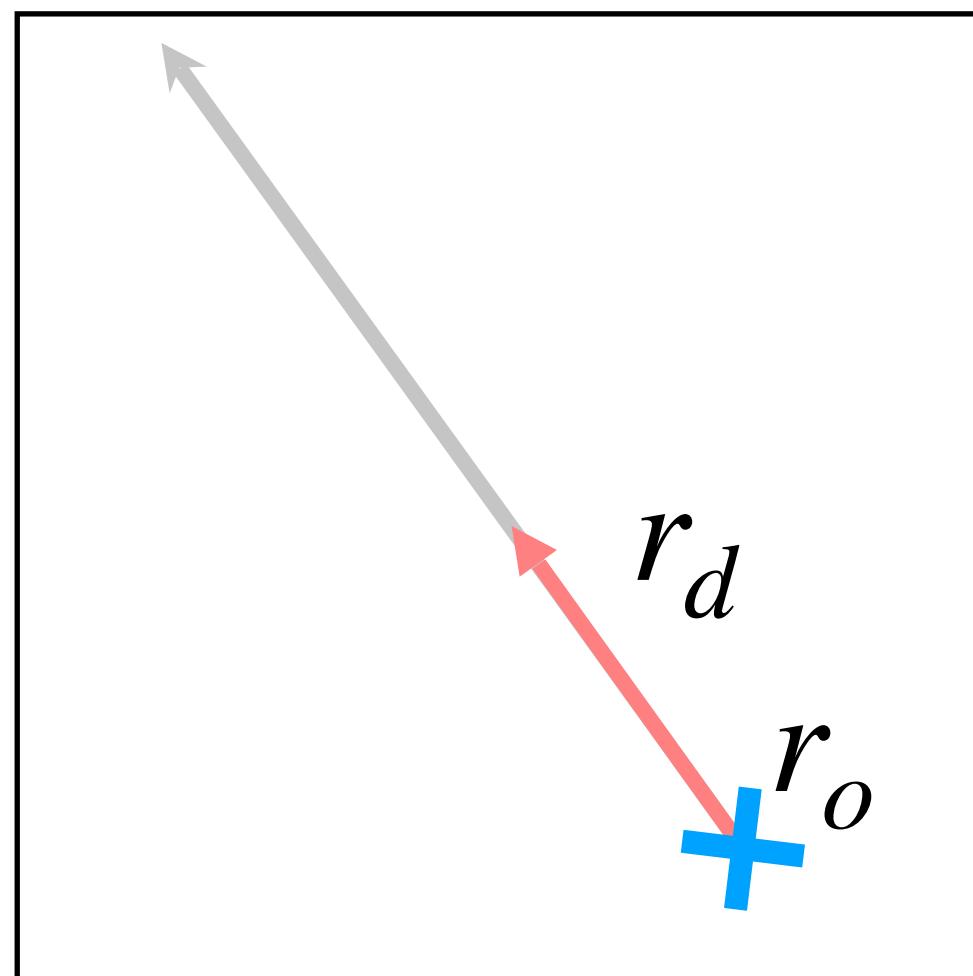
Recap - Light Transport & Cameras



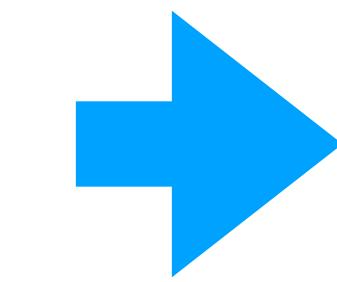
Recap: Function Signature of Rendering



Scene : Light sources,
materials, shapes, ...



Camera Ray



Radiance of the ray

Radiance

- Radiance is a fundamental quantity of light.
- Radiance is density of photons at a point, traveling in the same direction, at given wavelength.
- Radiance is energy per time, per unit area, per unit solid angle, per wavelength.

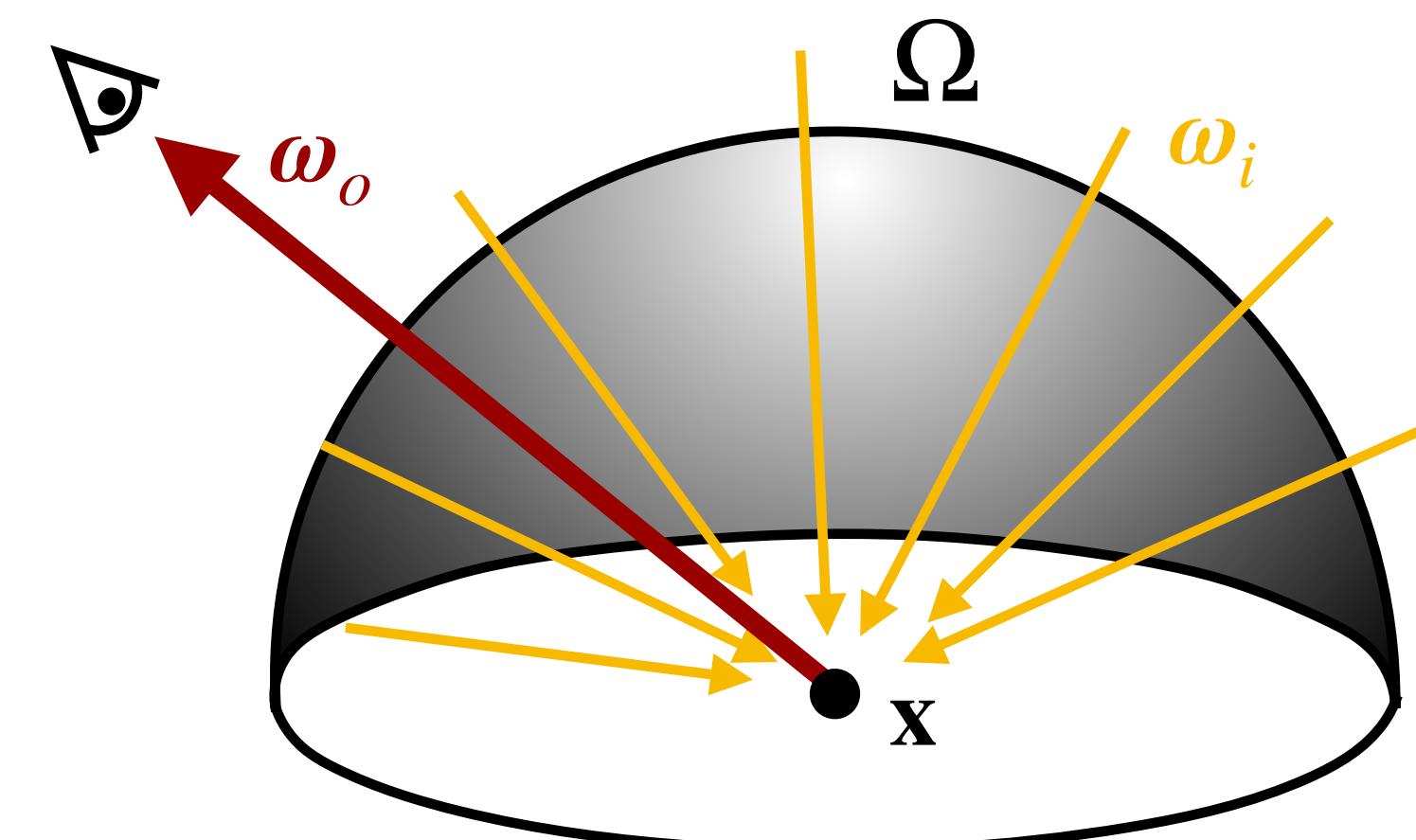
Recap: The rendering equation

$$L_o(\mathbf{x}, \omega_o, \lambda) = L_e(\mathbf{x}, \omega_o, \lambda) + \int_{\Omega} f(\mathbf{x}, \omega_i, \omega_o, \lambda) L_i(\mathbf{x}, \omega_i, \lambda) (\omega_i \cdot \mathbf{n}) d\omega_i$$

emitted radiance
outgoing radiance in ω_o
fraction of incoming light that is reflected into ω_o
incoming radiance
angle weight of photon density
solid angle differential

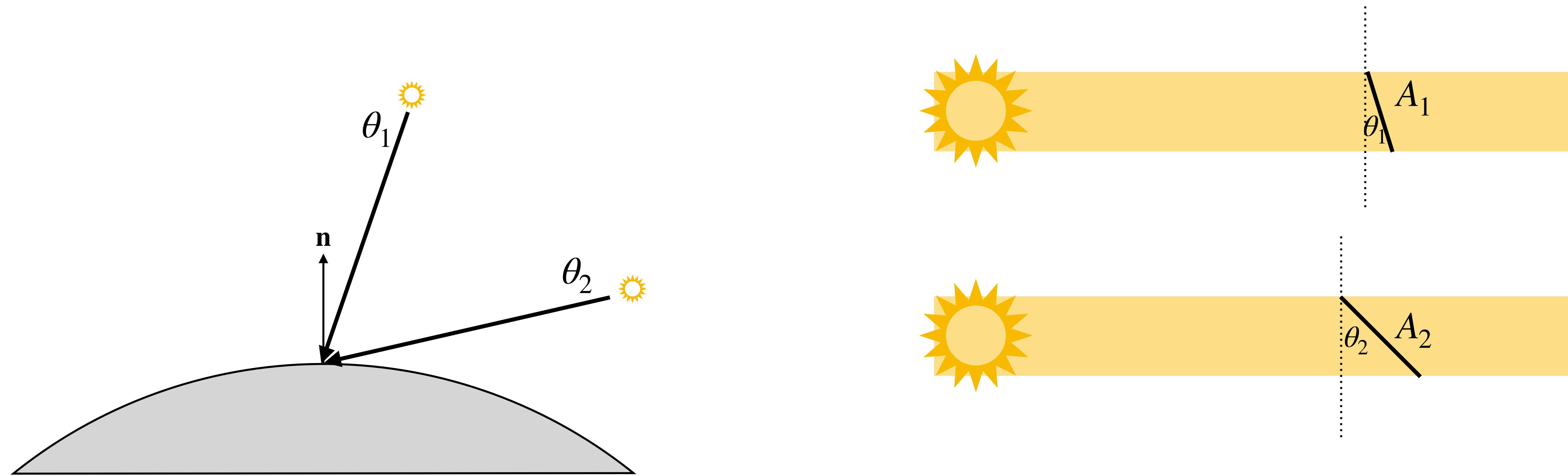
Conservation of energy:

$$\int_{\omega_i \in \Omega} f(\omega_i, \omega_o) (\omega_i \cdot \mathbf{n}) d\omega_i \leq 1$$

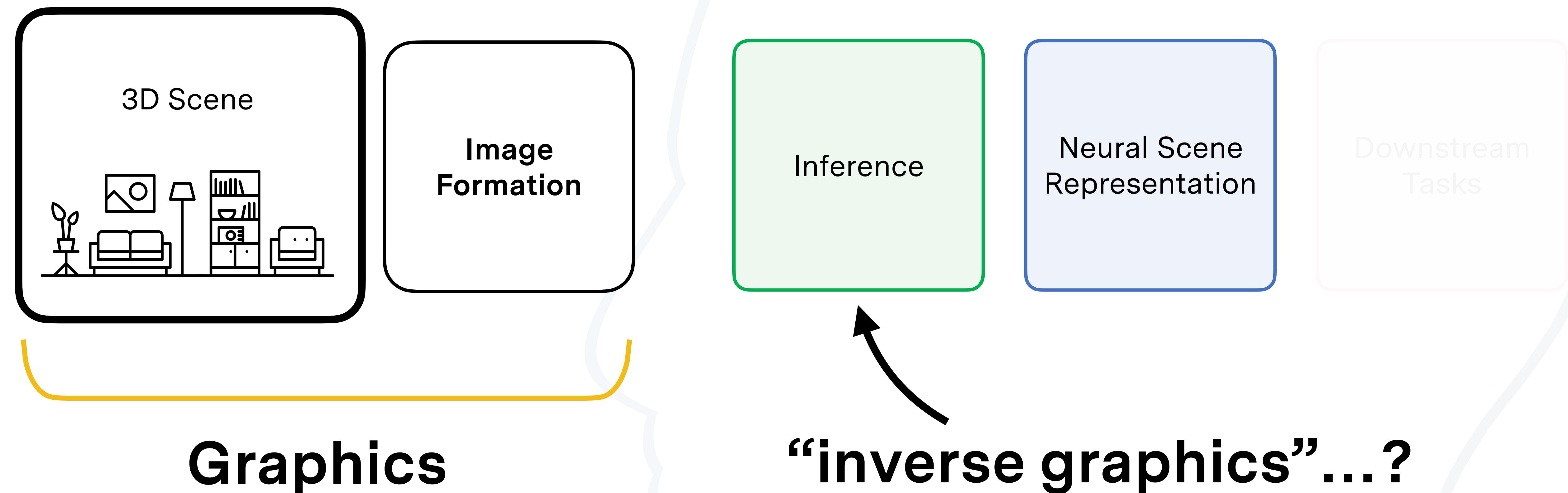


Angle Weight of Photon density

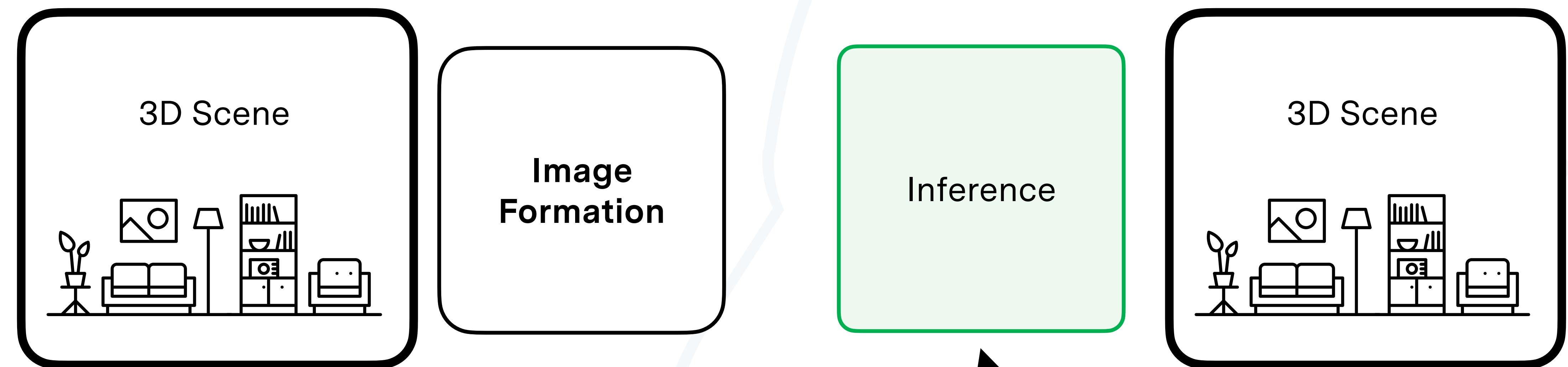
Which incoming light results in more photon density at the surface?



Today: Differentiable Rendering, AKA “Inverse Graphics”



Today: Differentiable Rendering, AKA “Inverse Graphics”



Graphics

“inverse graphics”...?

ownstream
Tasks

Differentiable Rendering

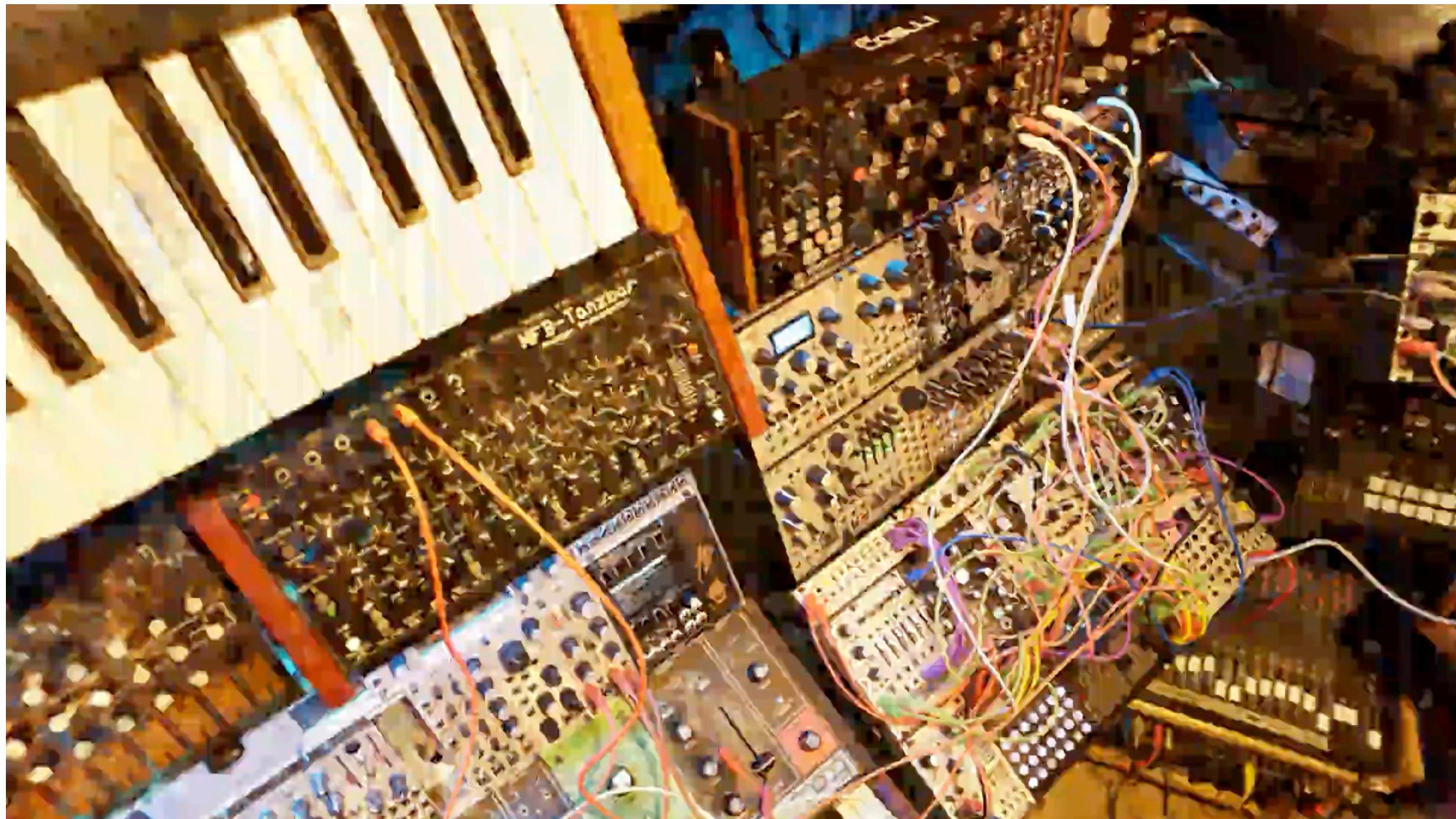
Optimization

Optimizing shape, albedo
& roughness



Iteration 0

Differentiable Signed Distance Function Rendering,
Vicini et al. 2022



InstantNGP, Müller et al. 2022

Why?

Part of our problem relates to **inverting** the rendering process: Given 2D images, we want to reconstruct 3D scenes. Differentiable rendering is one way of exactly inverting the rendering process.

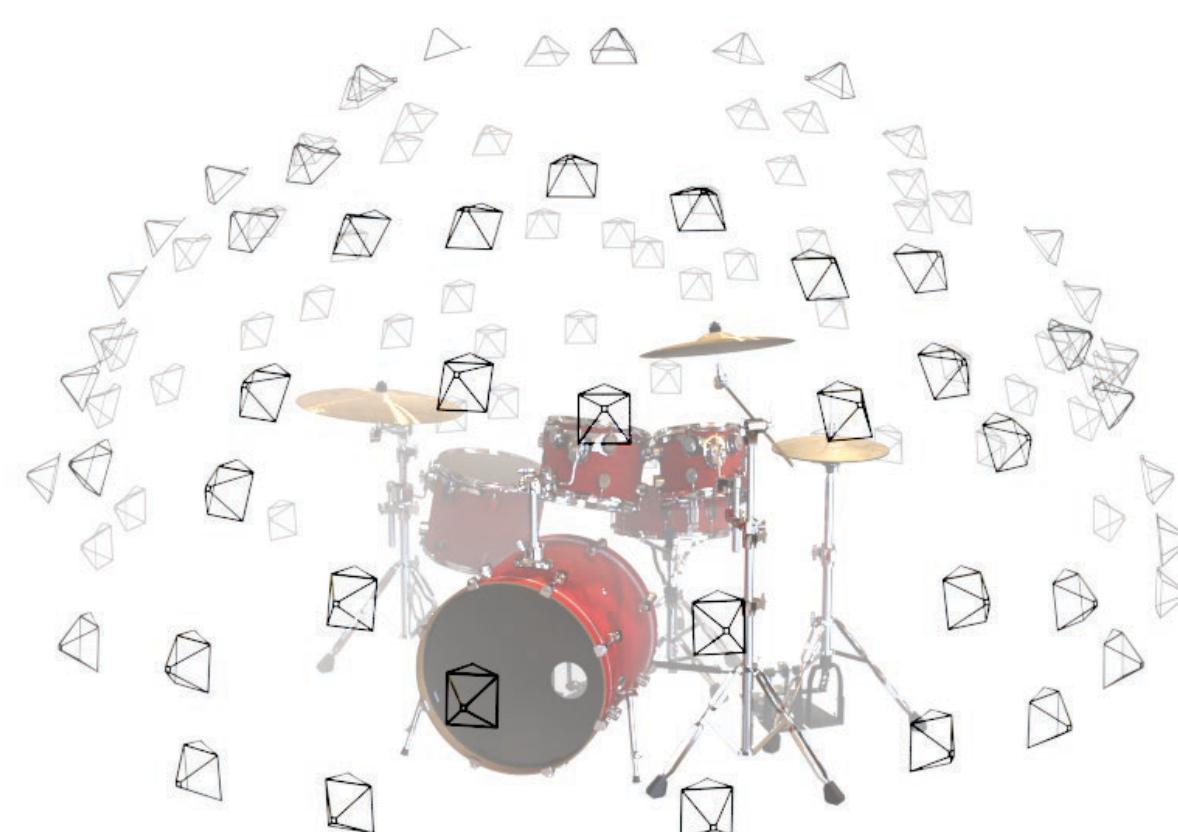
What you'll learn.

The structure of differentiable renderers, pros and cons and assumptions of different algorithms.

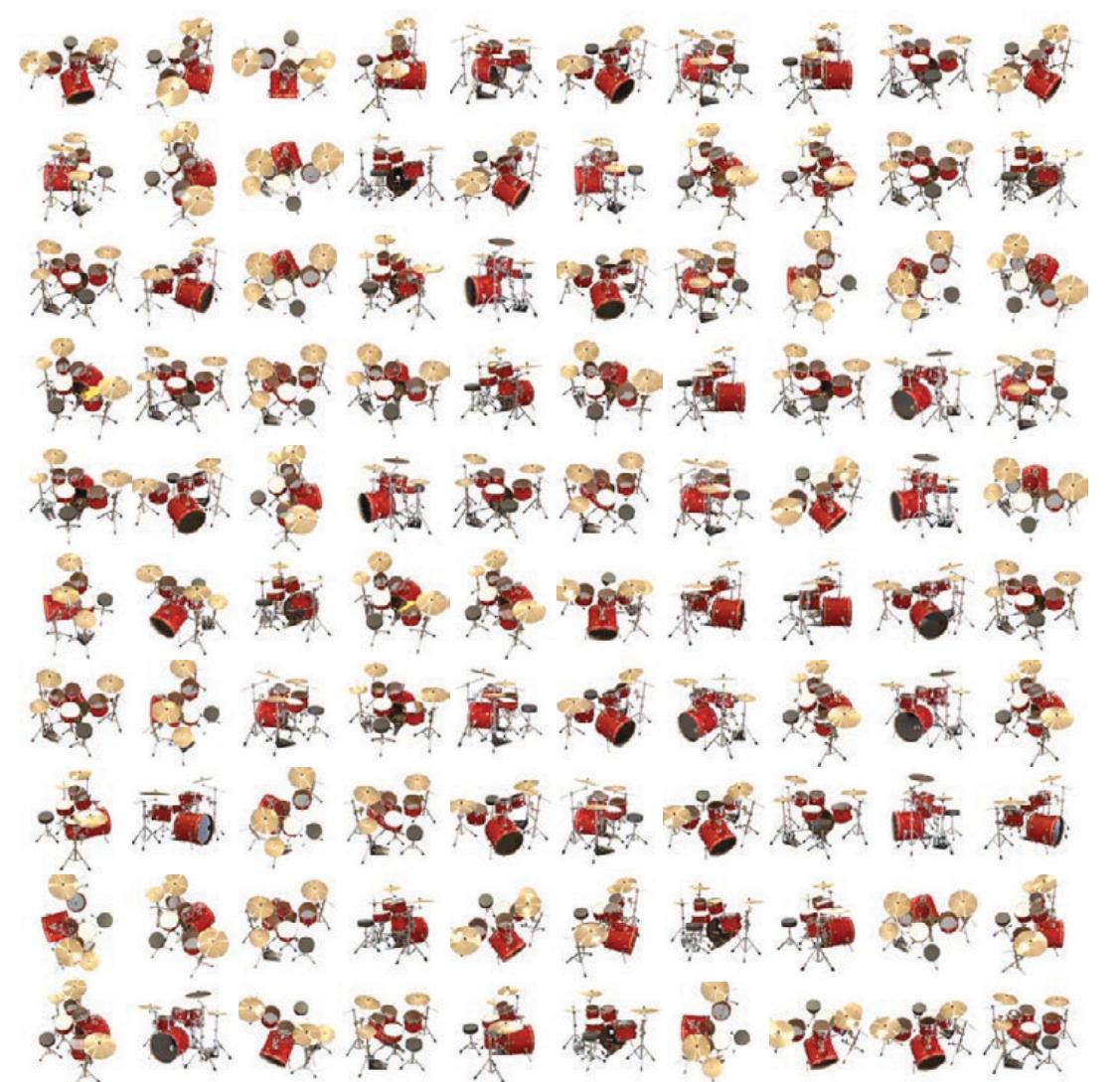
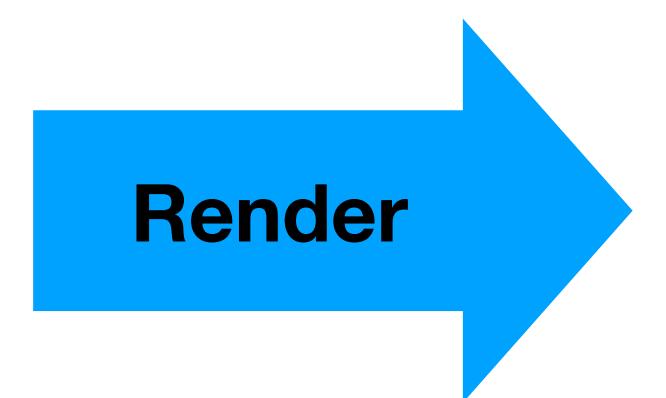
Rendering (Graphics): Given 3D Scene + Camera parameters, yield images



3D Scene



Camera Poses



Images

Inverse Graphics: Given Images, Infer Camera Poses & 3D Scene!



Reconstruct

Images



3D Scene



Camera Poses

How to get camera poses?



Reconstruct



Images

3D Scene



Camera Poses

Bundle Adjustment (Multi-View Geometry Lecture)!

Bundle adjustment minimizes the reprojection error of all observations:

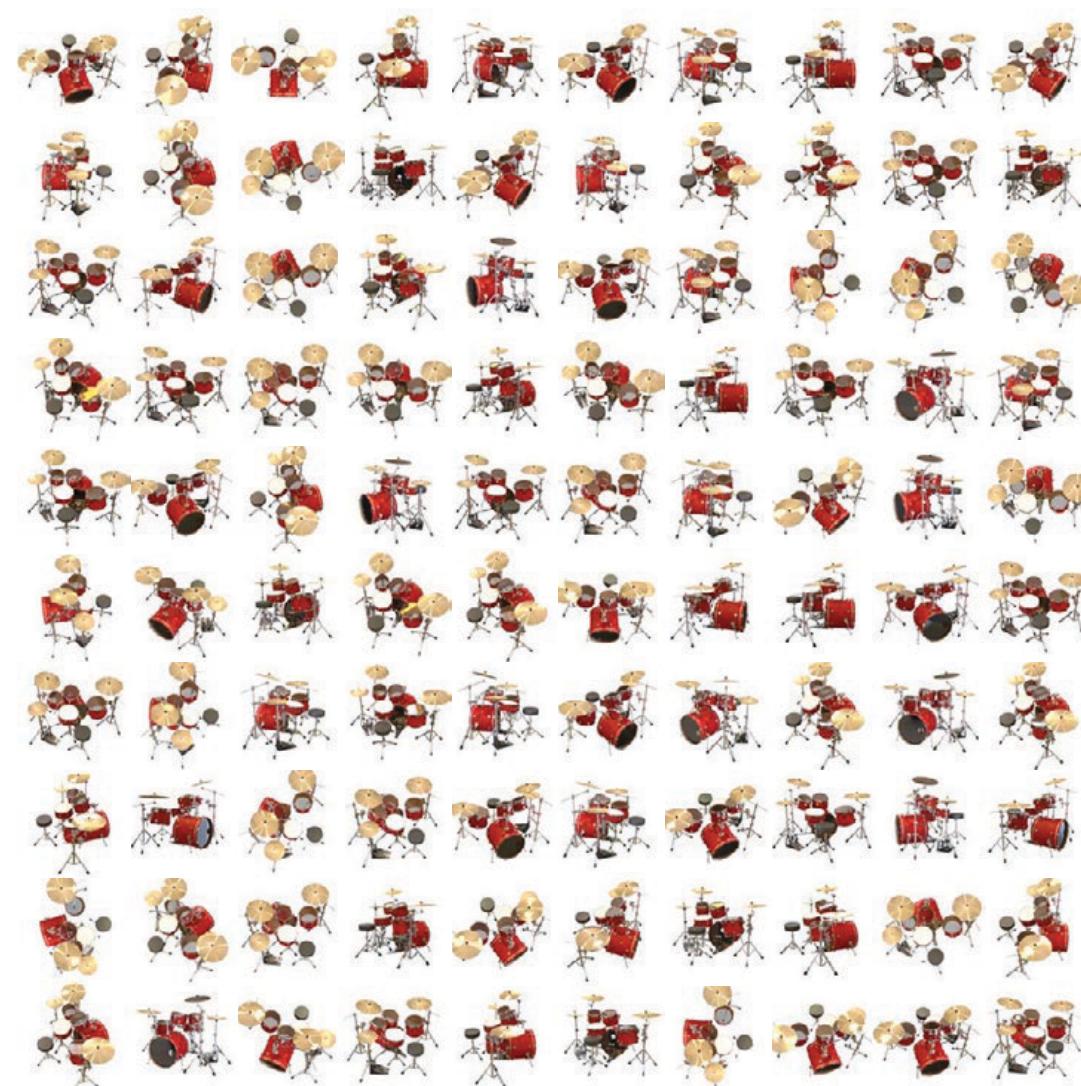
$$\Pi^*, \mathcal{X}_w^* = \operatorname{argmin}_{\Pi, \mathcal{X}_w} \sum_{i=1}^N \sum_{p=1}^P w_{ip} \|\mathbf{x}_{ip}^s - \pi_i(\mathbf{x}_p^w)\|_2^2$$

Images

3D Scene

Camera Poses

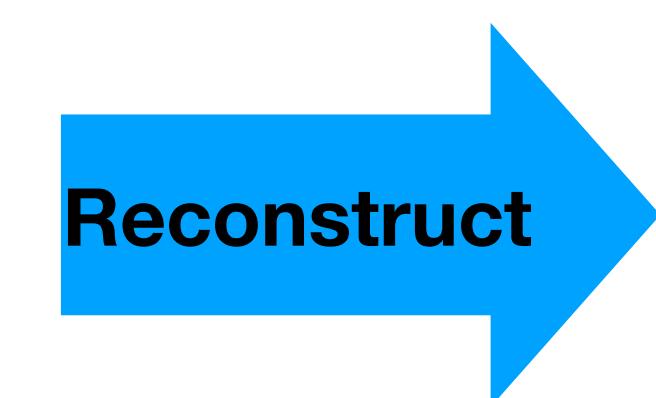
Can assume we know the camera poses.



Images

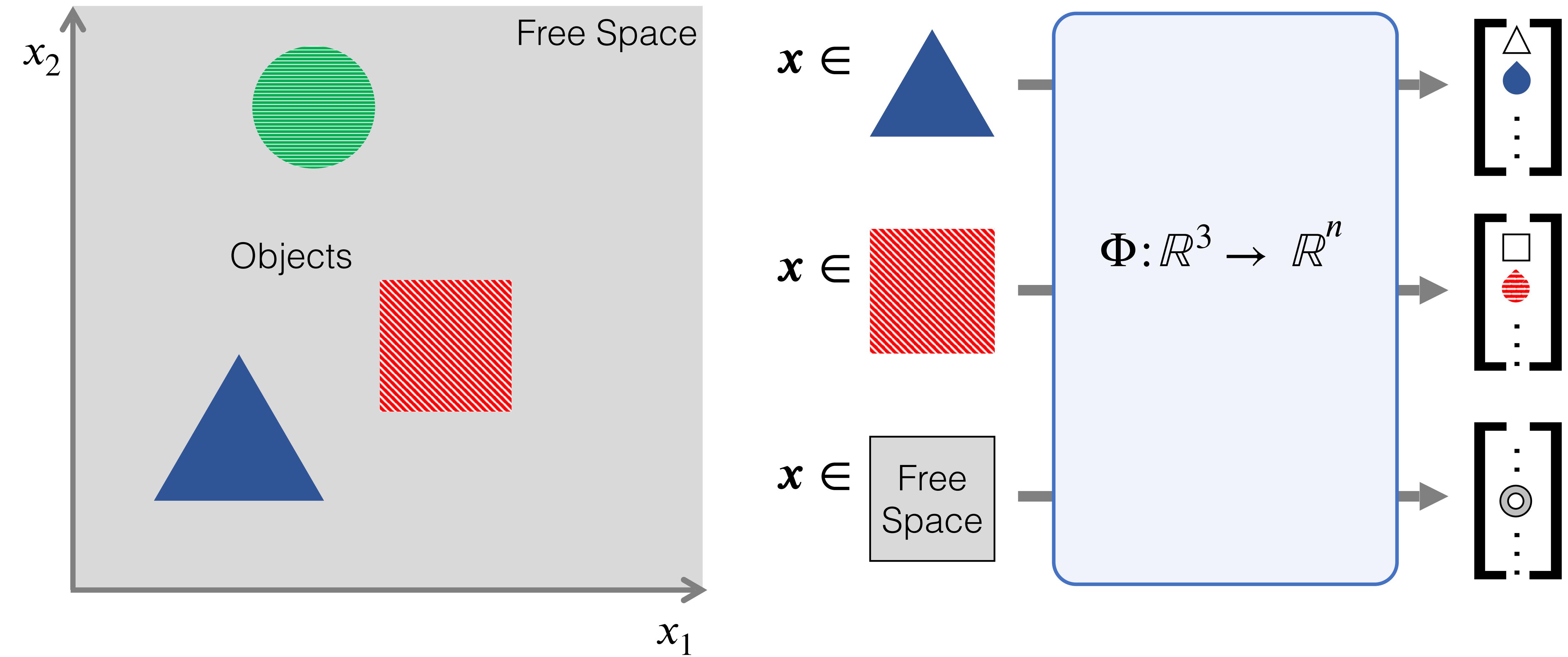


Camera Poses

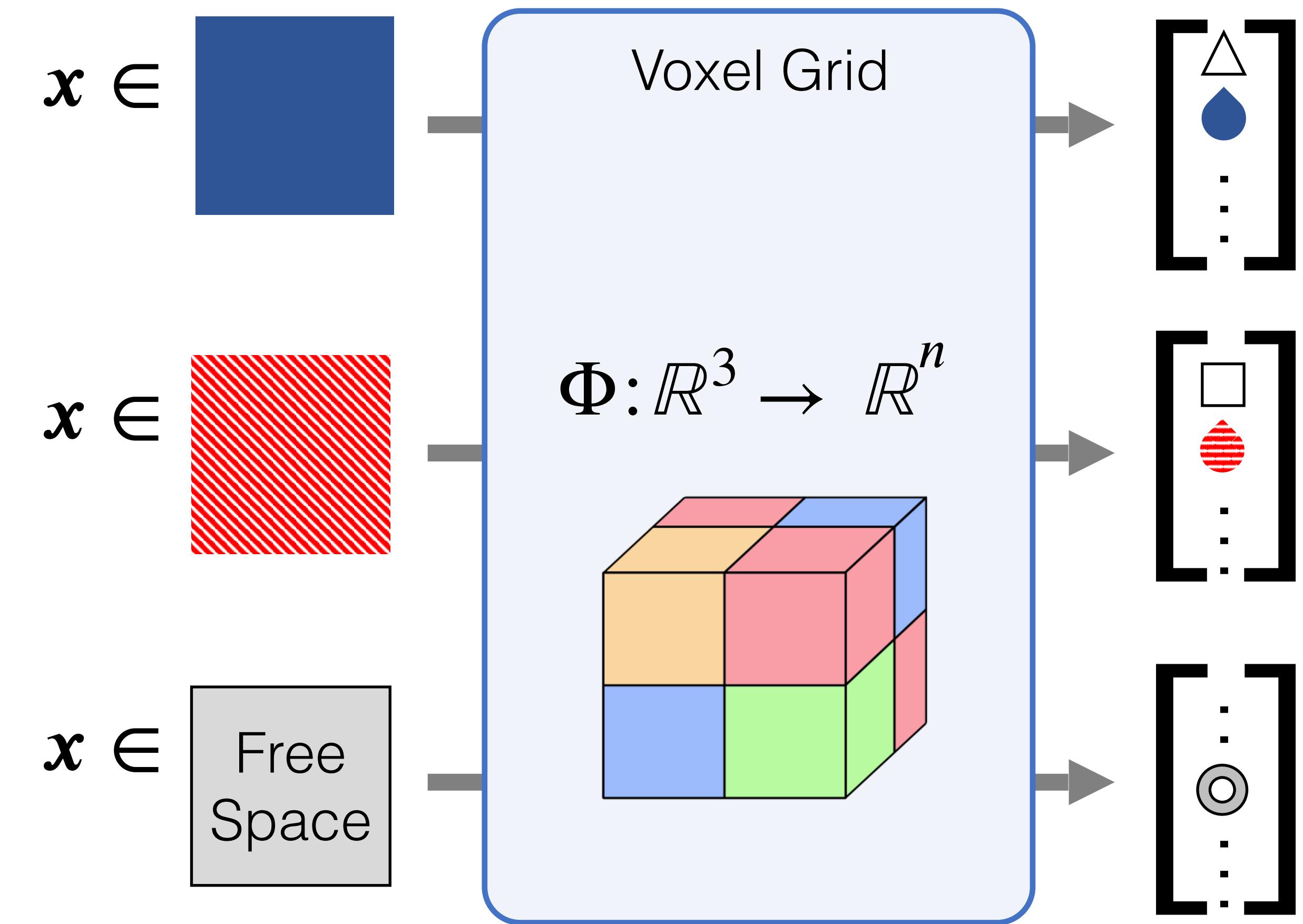
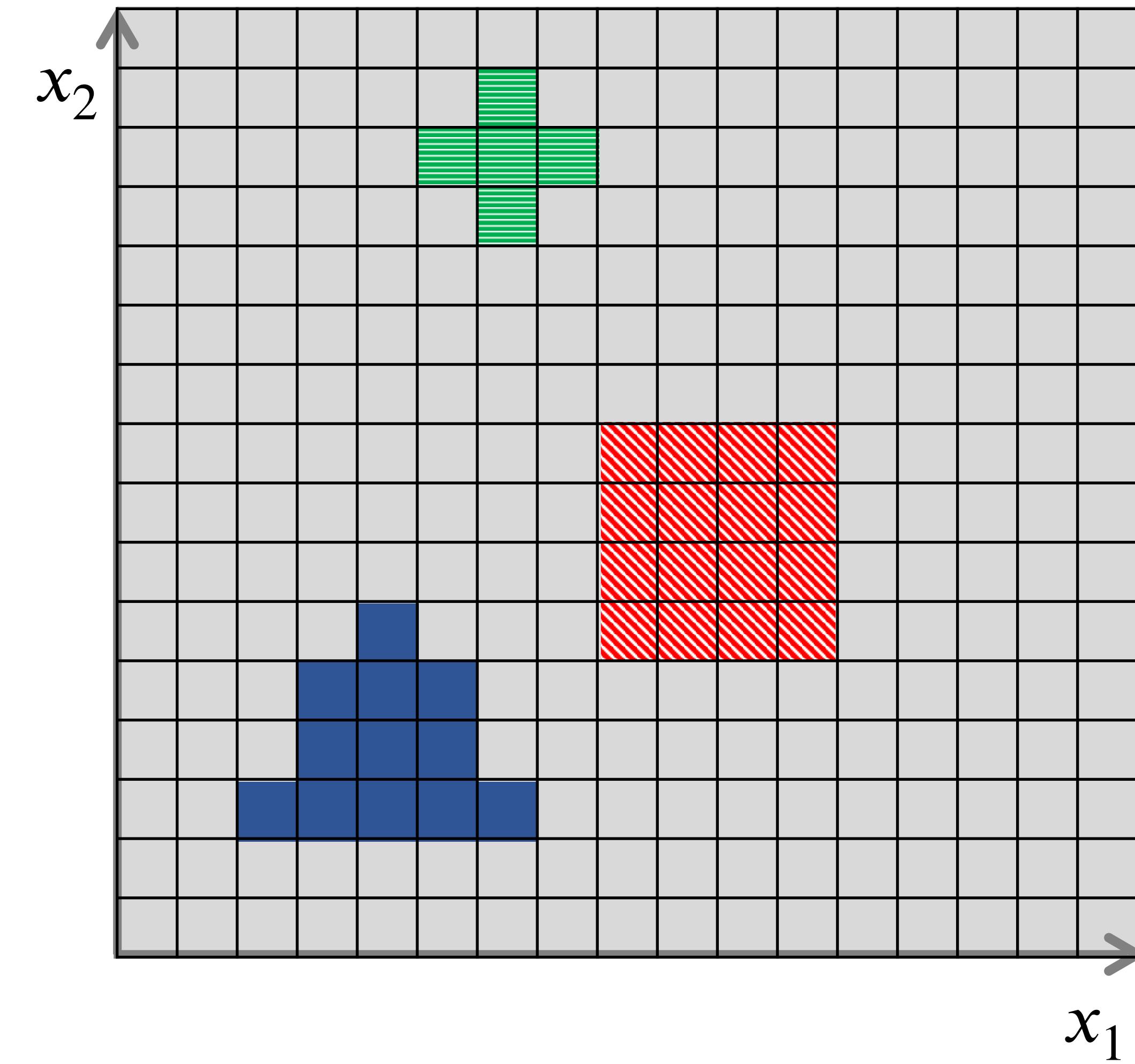


3D Scene

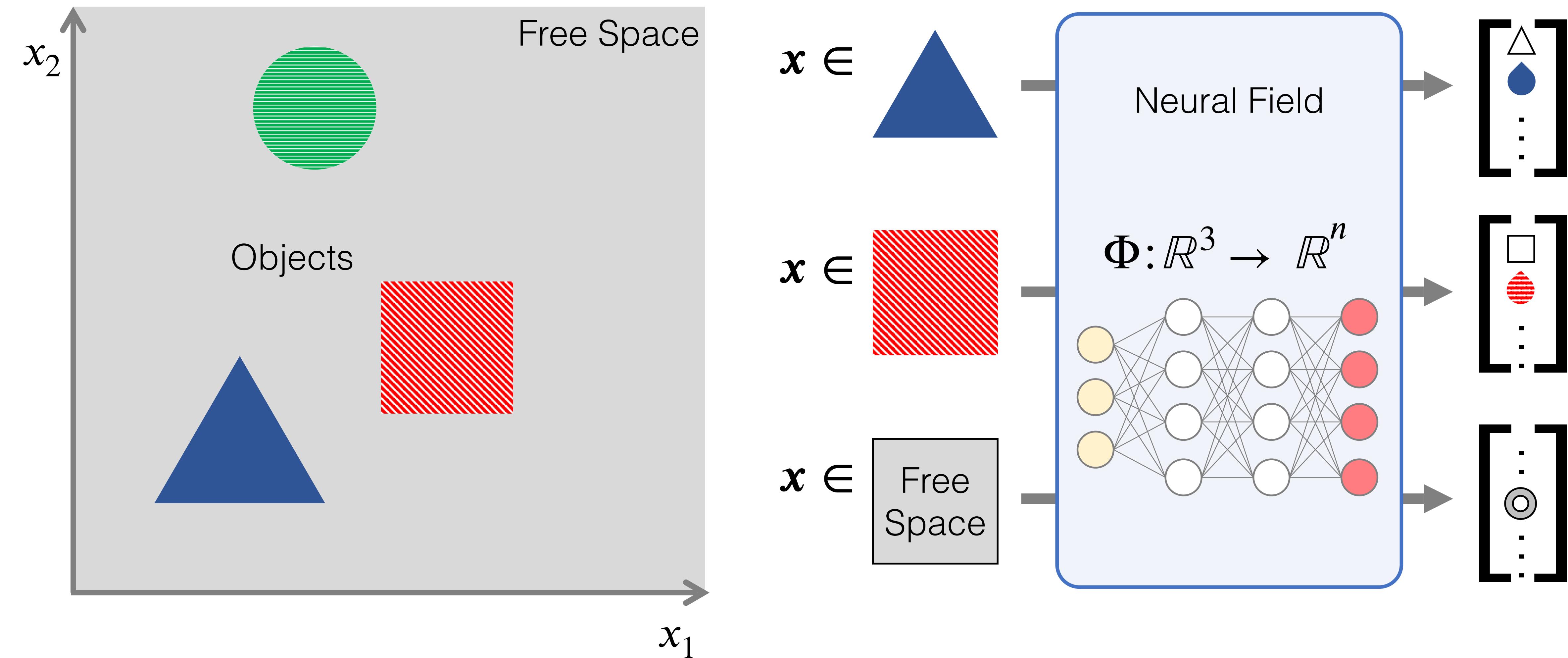
Starting Point: 3D Scene Representation



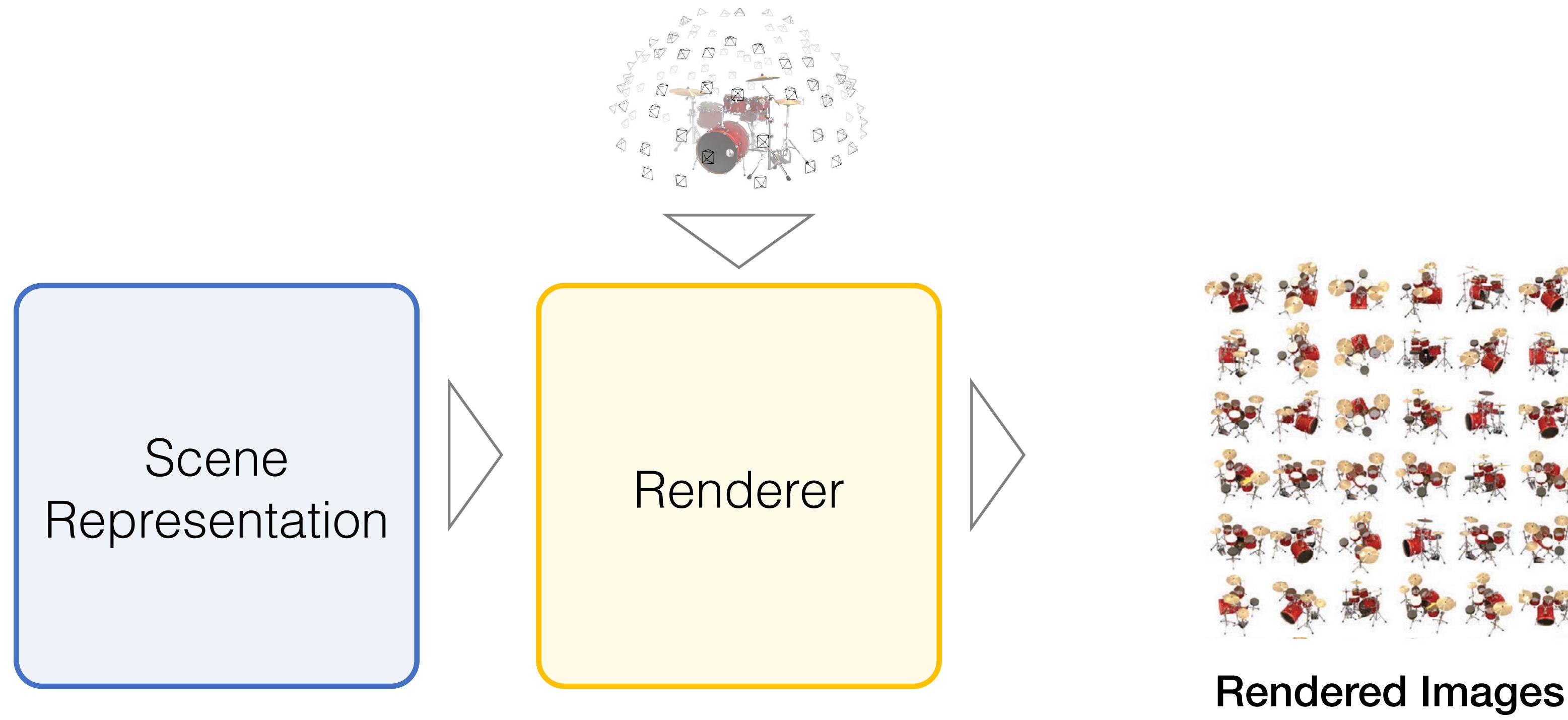
Starting Point: 3D Scene Representation



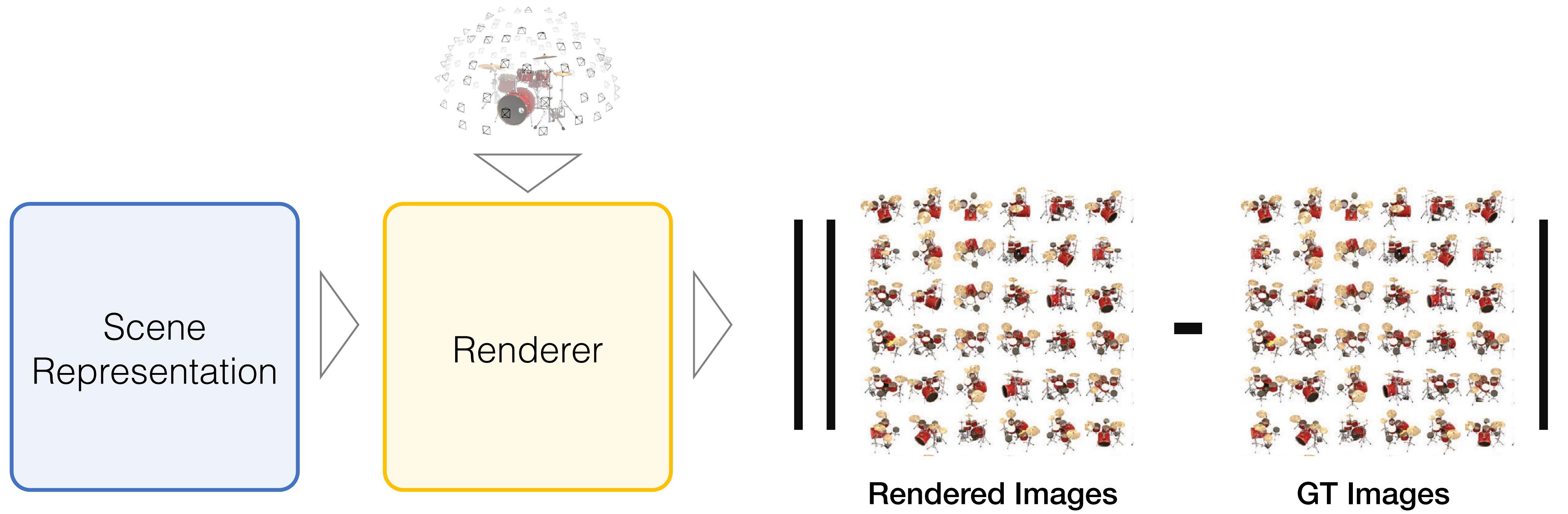
Starting Point: 3D Scene Representation



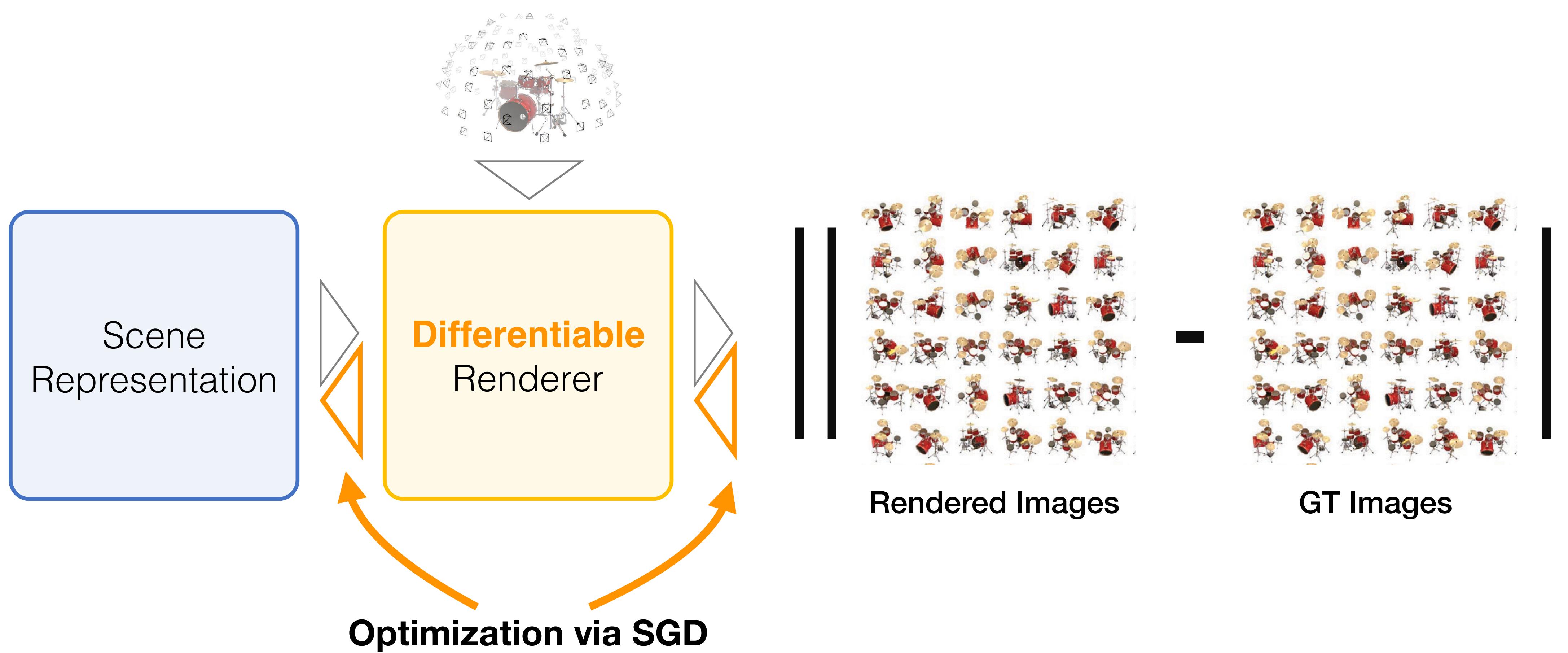
Differentiable Rendering



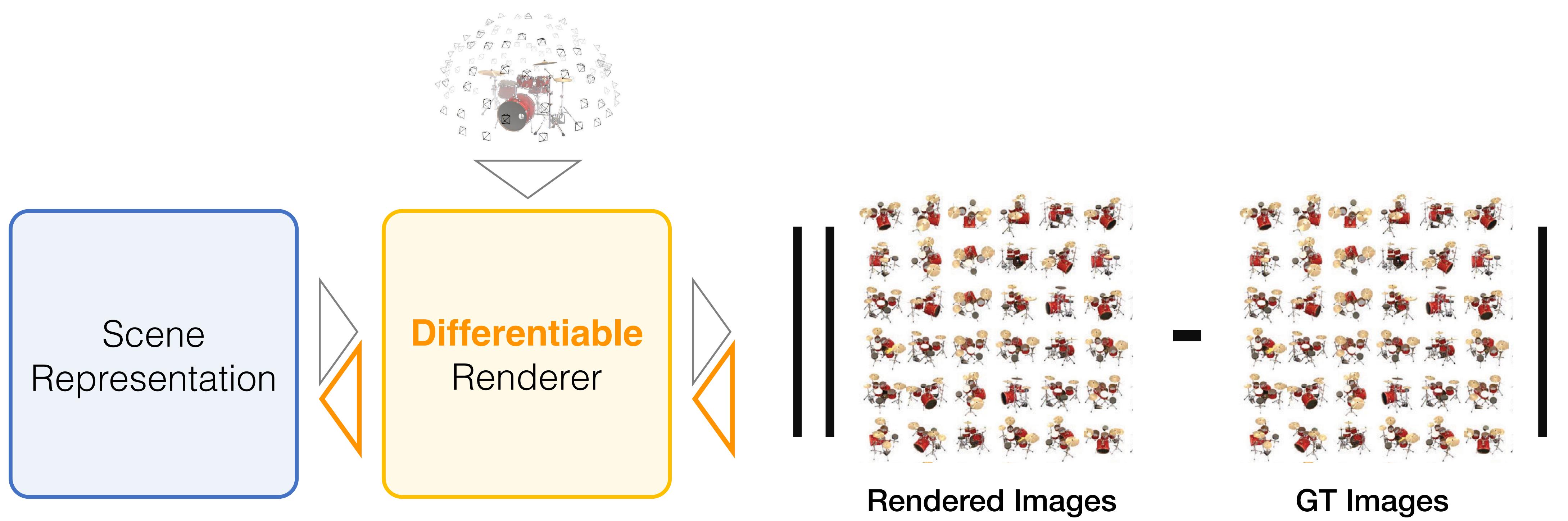
Differentiable Rendering



Differentiable Rendering



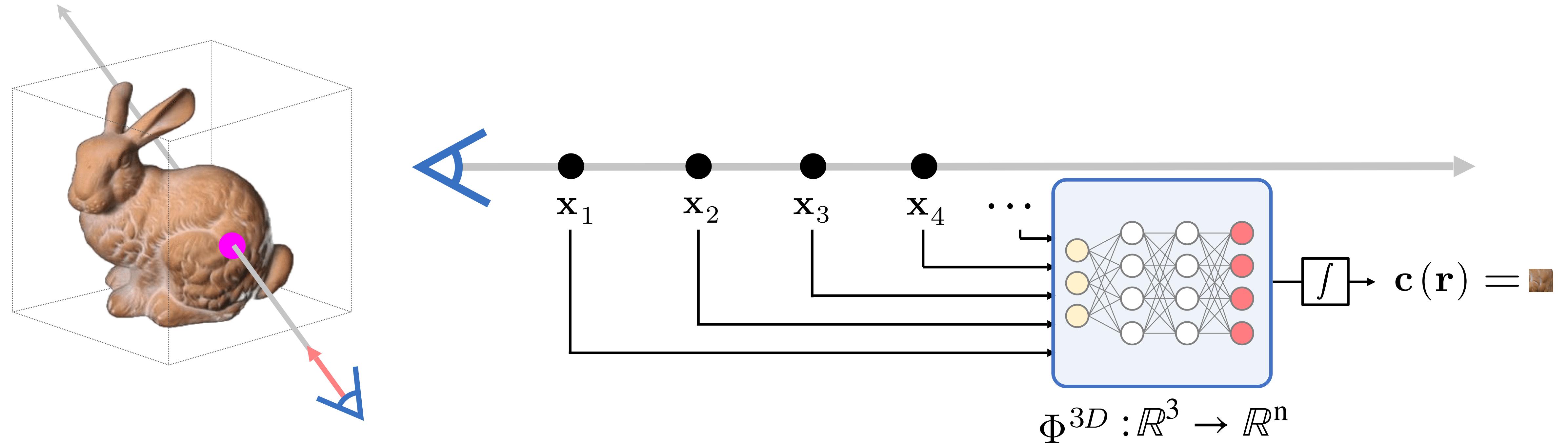
Differentiable Rendering



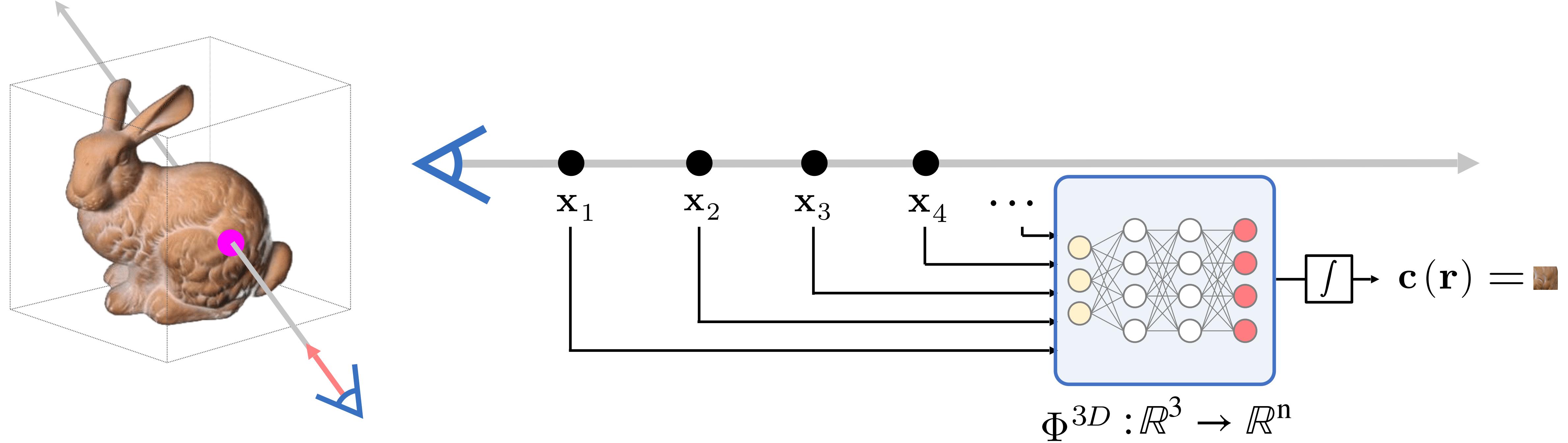
Given an *observable* variable (pixel colors), we will build a differentiable forward model that we then use to estimate *unobserved (latent) variables* (geometry, appearance)!

Questions?

General structure of Neural Renderers for 3D Fields



General structure of Neural Renderers for 3D Fields



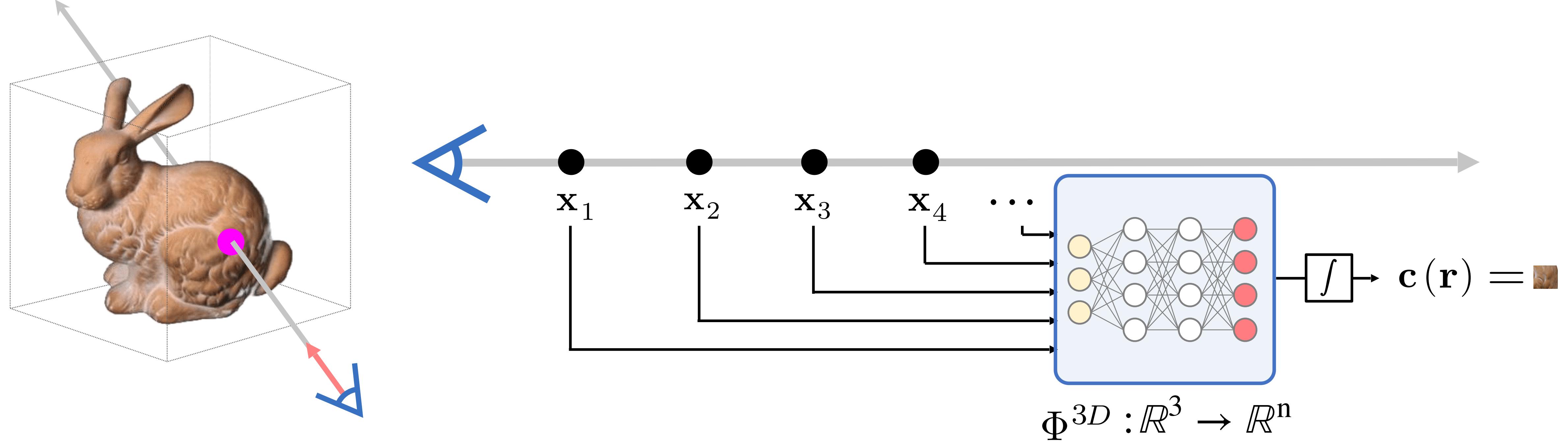
Sphere-Tracing
[JC Hart, 1996]

Volumetric

Hybrid implicit-volumetric

Learned aggregation

General structure of Neural Renderers for 3D Fields



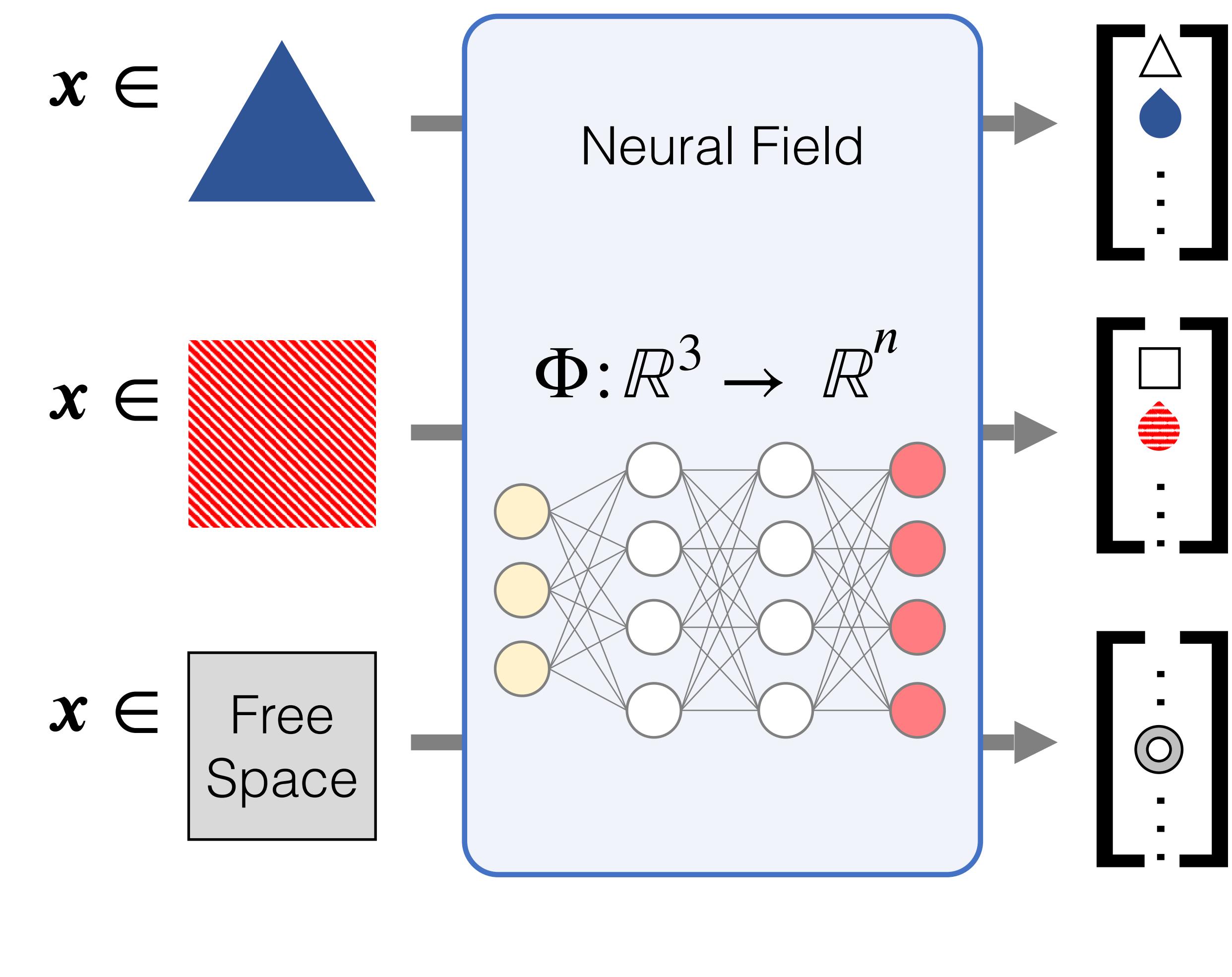
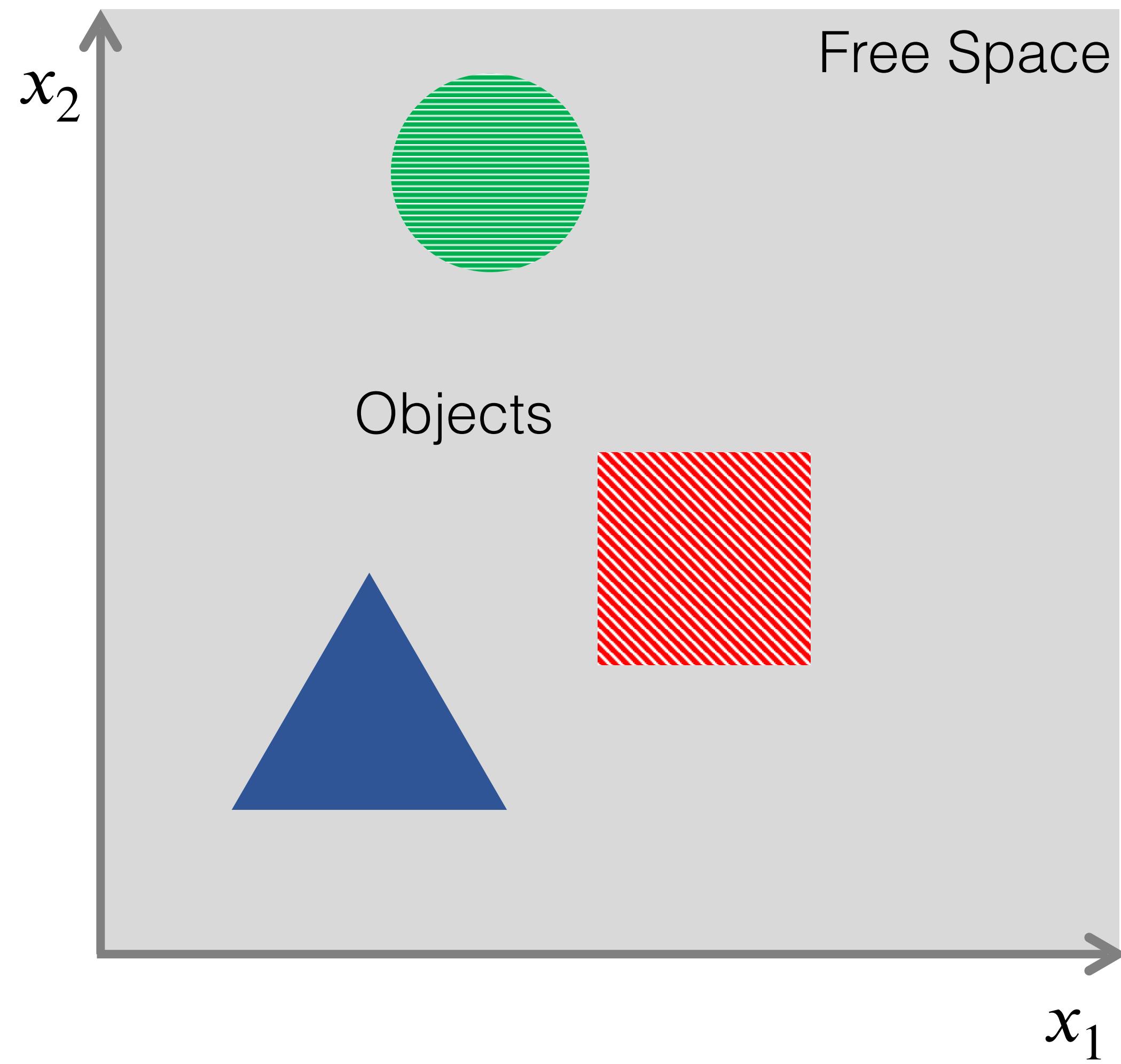
Sphere-Tracing
[JC Hart, 1996]

Volumetric

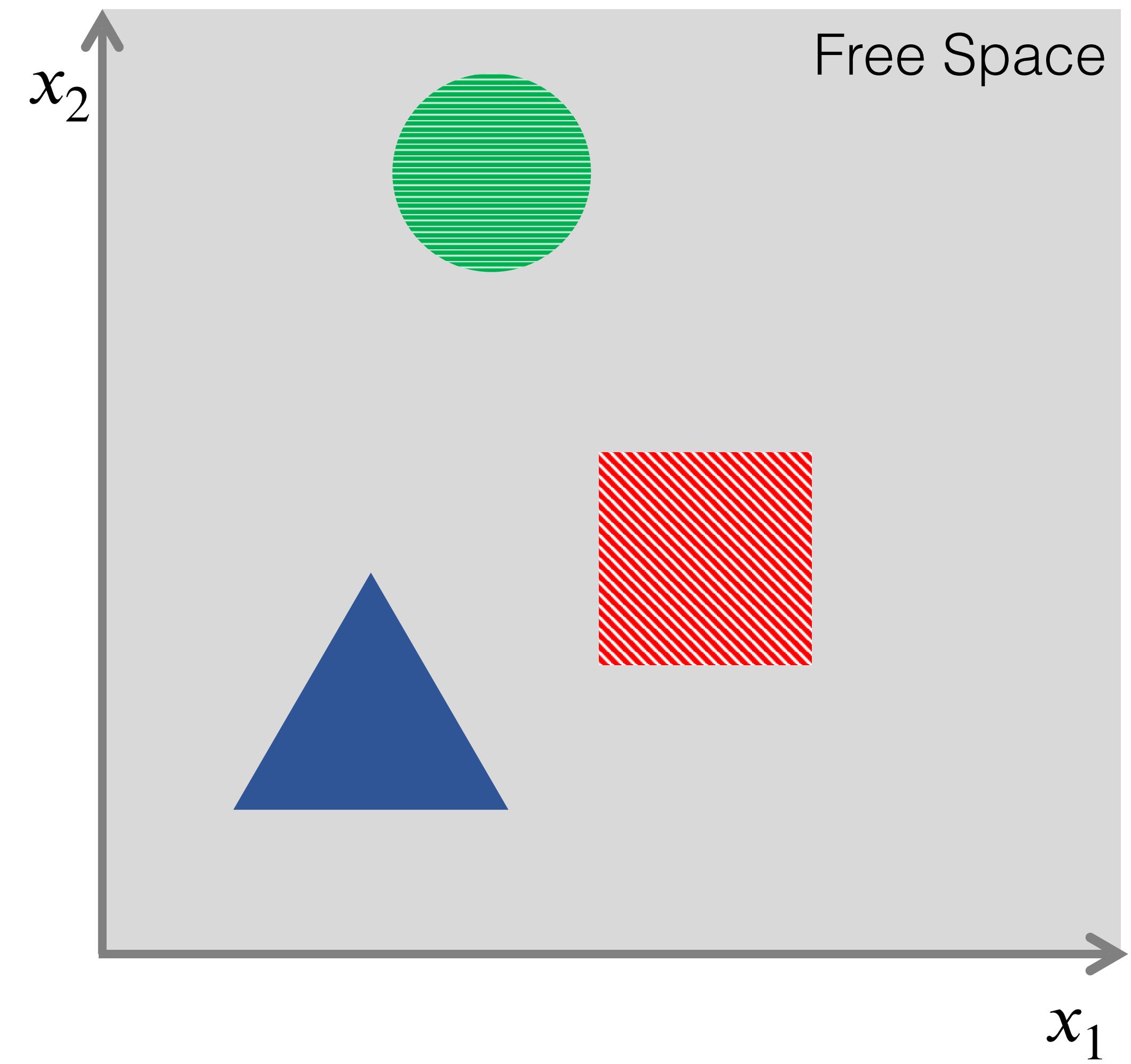
Hybrid implicit-volumetric

Learned aggregation

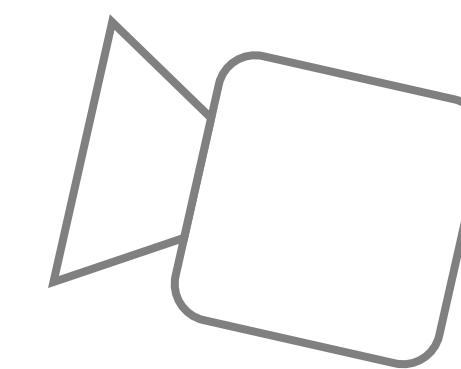
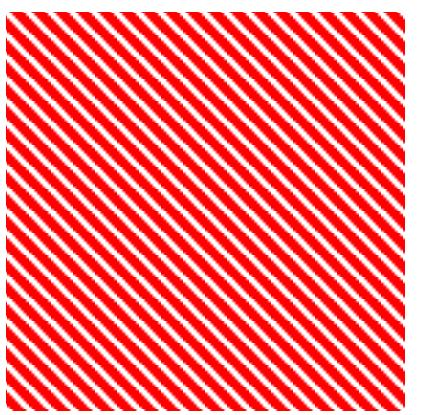
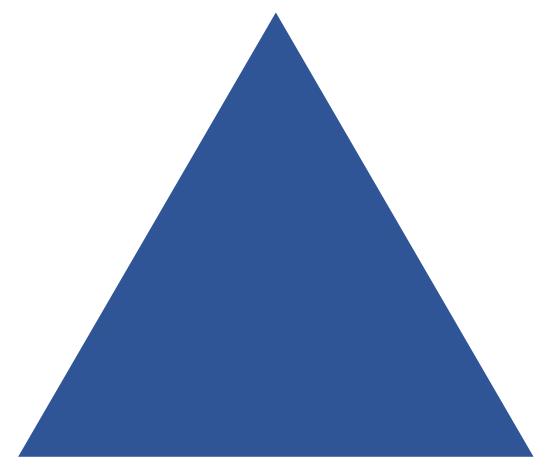
Scene Representation Networks (Sitzmann et al. 2019)



Sphere Tracing

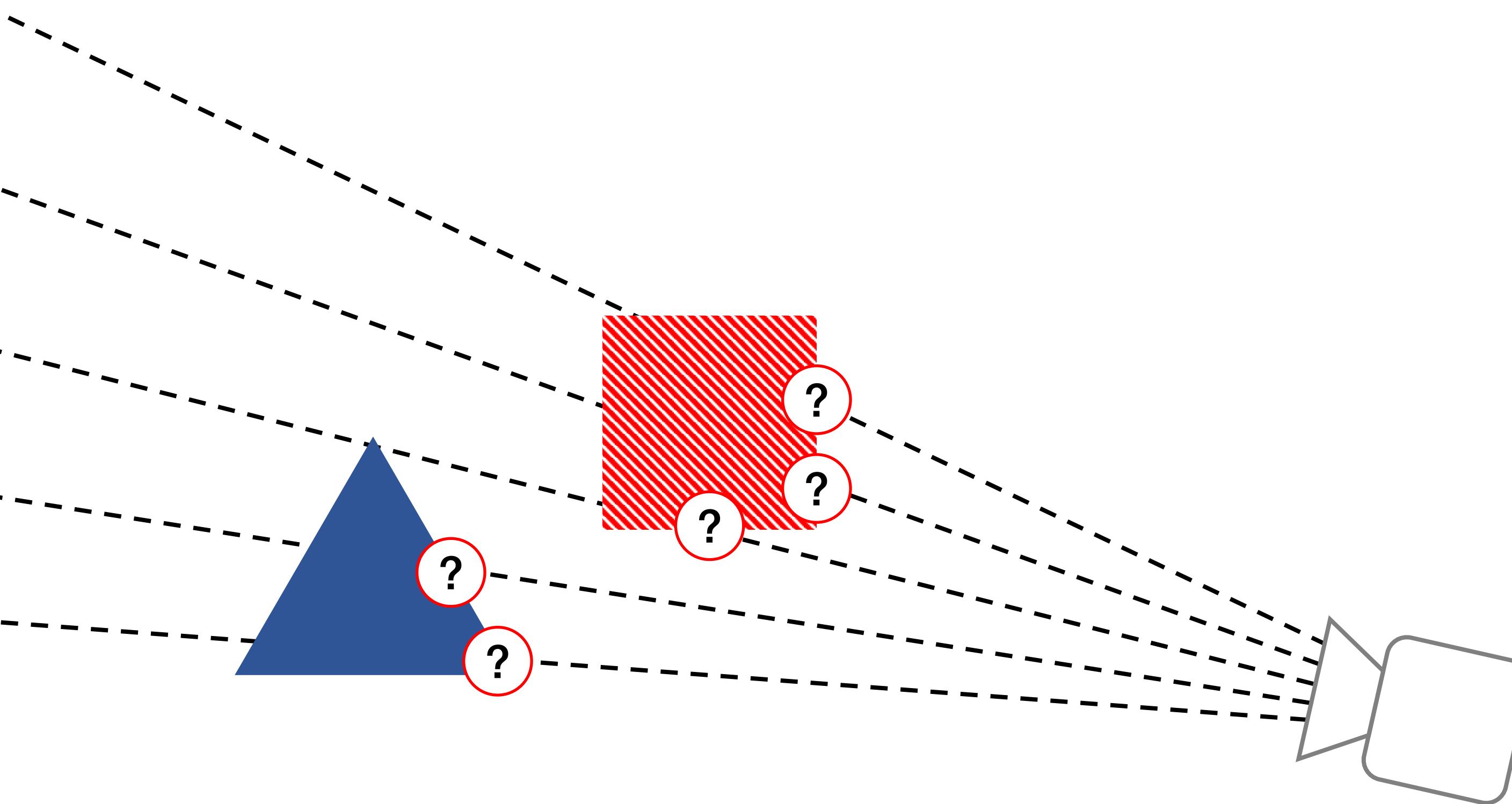


Sphere Tracing



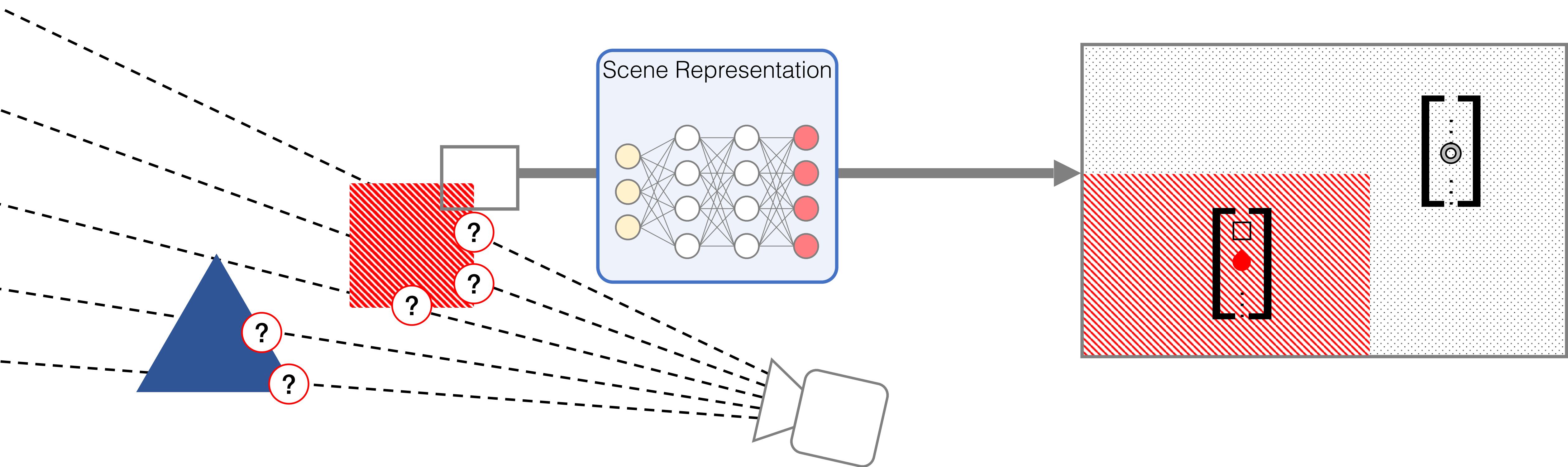
Sphere Tracing Step 1: Intersection Testing.

- ▶ How to parameterize scene geometry?



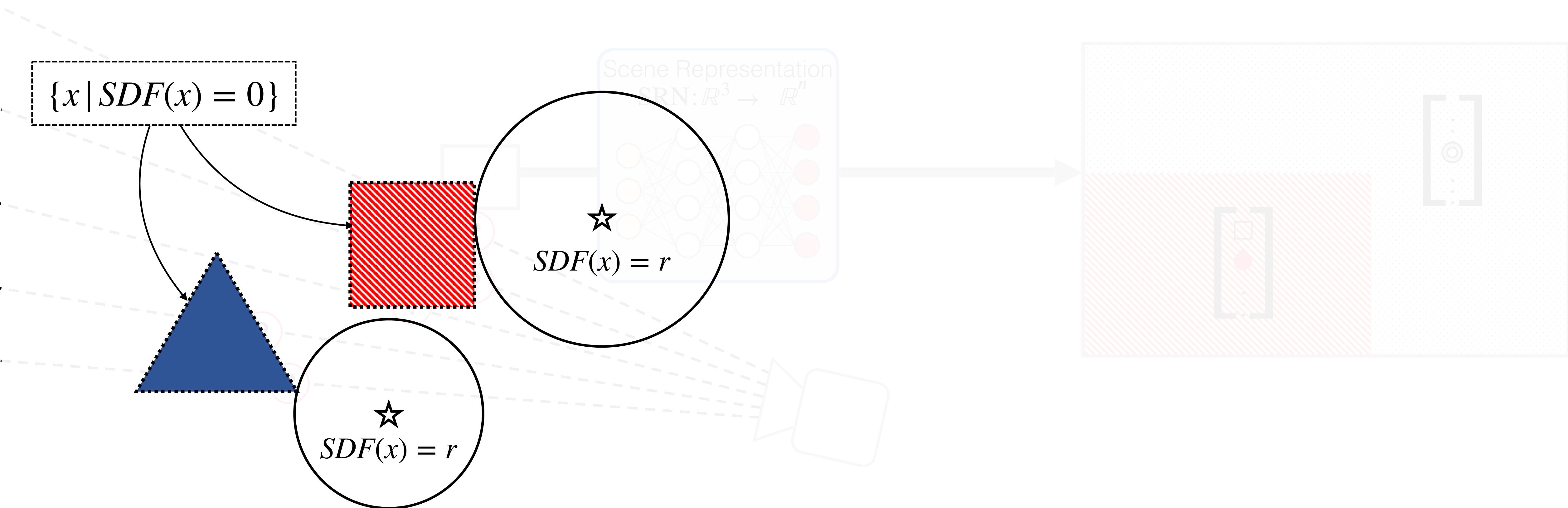
Sphere Tracing Step 1: Intersection Testing.

- ▶ How to parameterize scene geometry?



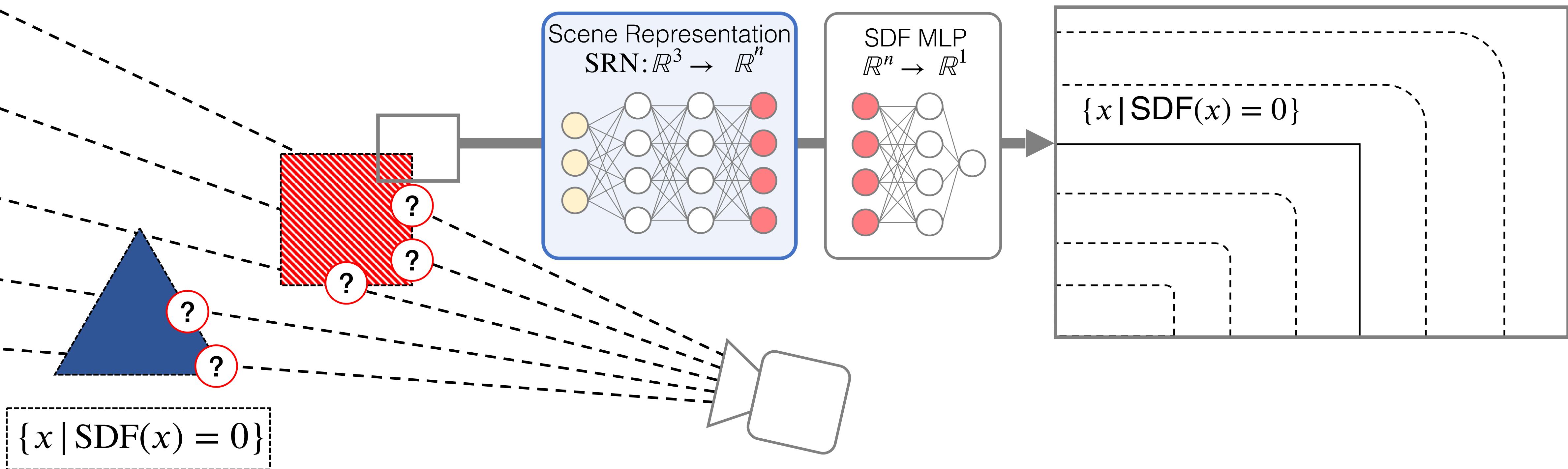
Neural Renderer Step 1: Intersection Testing.

- ▶ How to parameterize scene geometry?



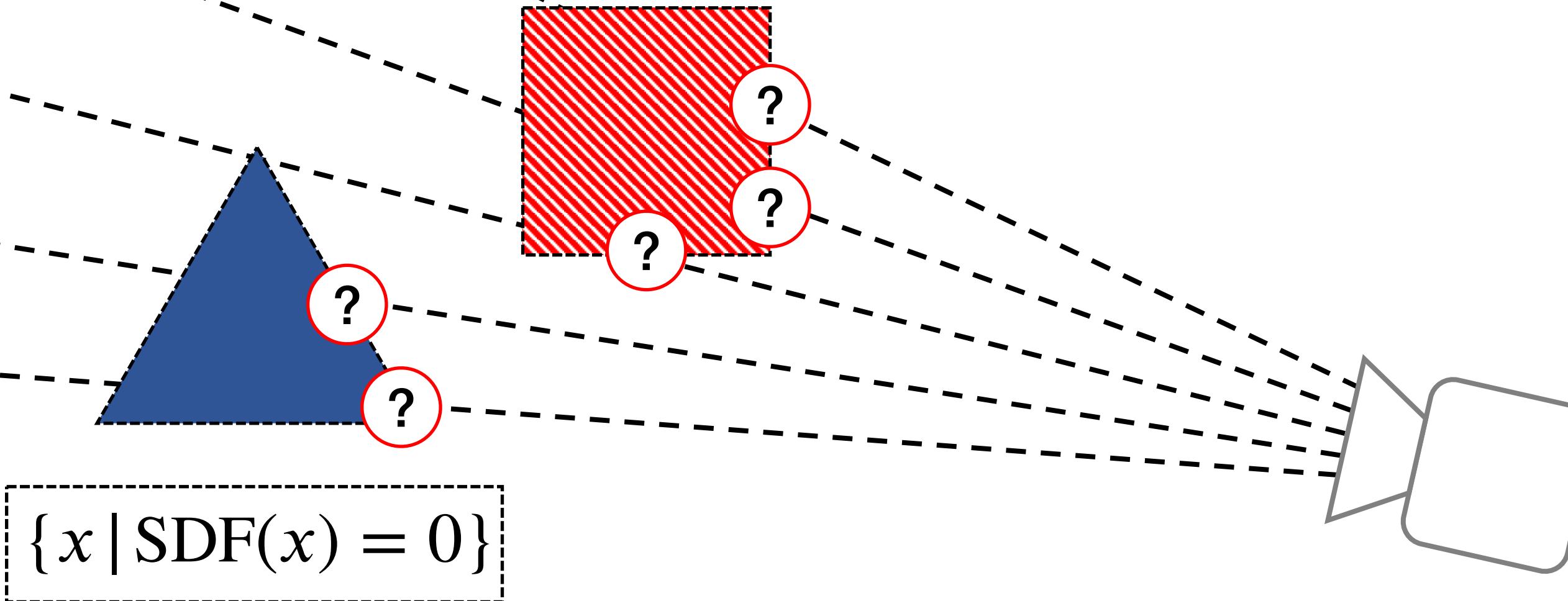
Sphere Tracing Step 1: Intersection Testing.

- ▶ How to parameterize scene geometry?
- ▶ Signed Distance Function maps point to distance to closest surface.

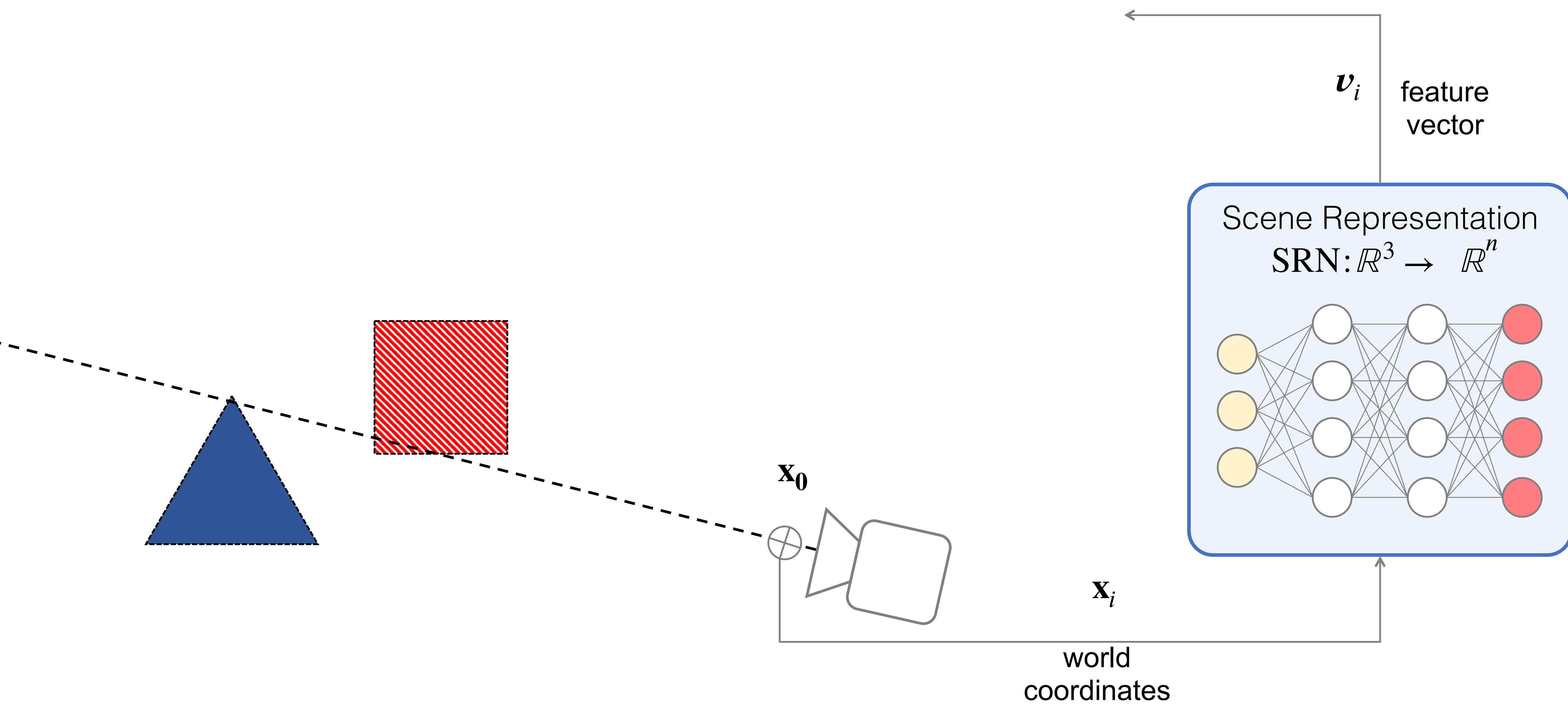


Sphere Tracing Step 1: Intersection Testing.

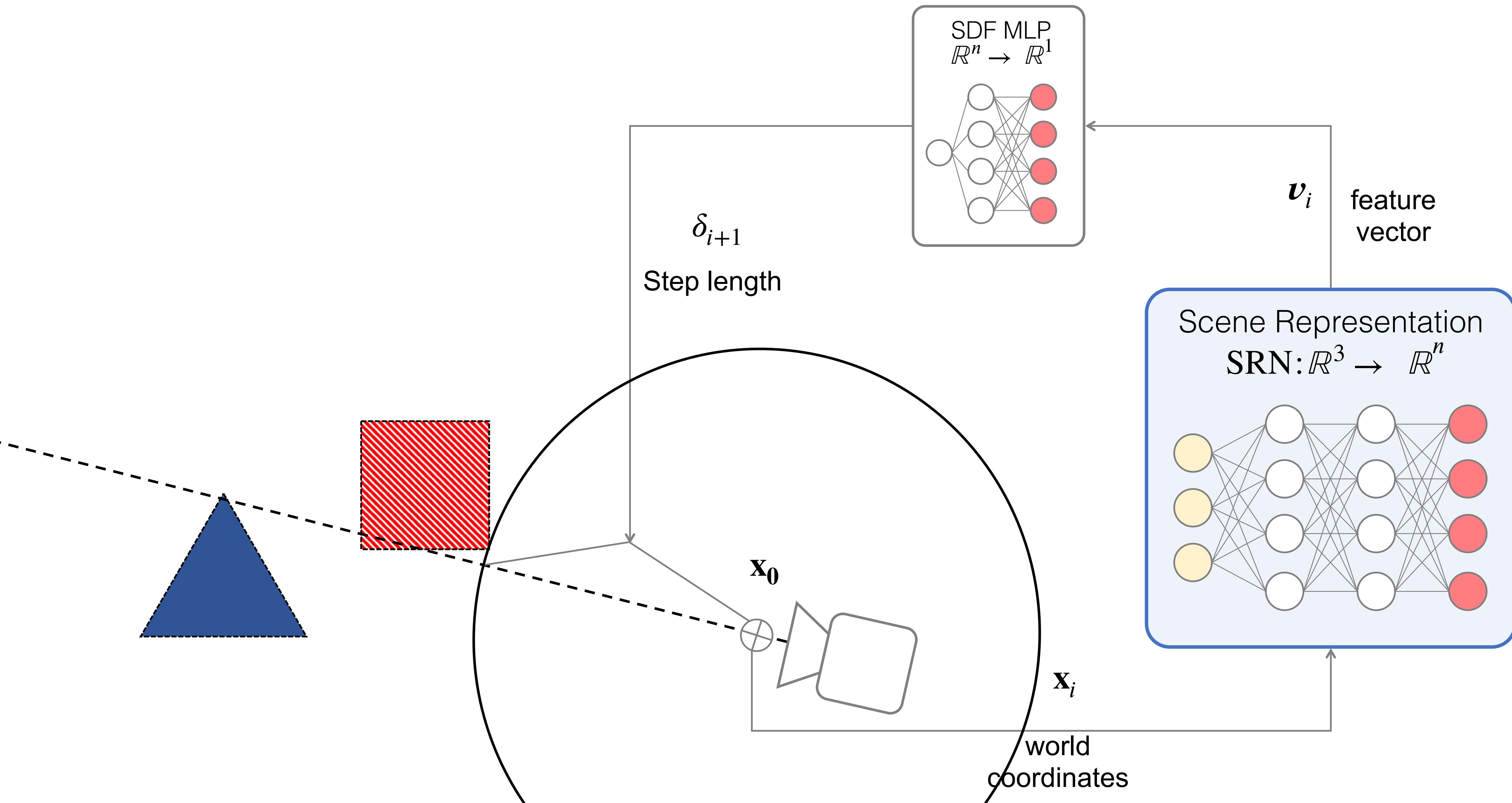
- ▶ How to find intersections?
- ▶ March along ray until we find zero-levelset.
- ▶ Naïve solution: constant steplength. Too expensive!



Sphere Tracing Step 1: Intersection Testing.

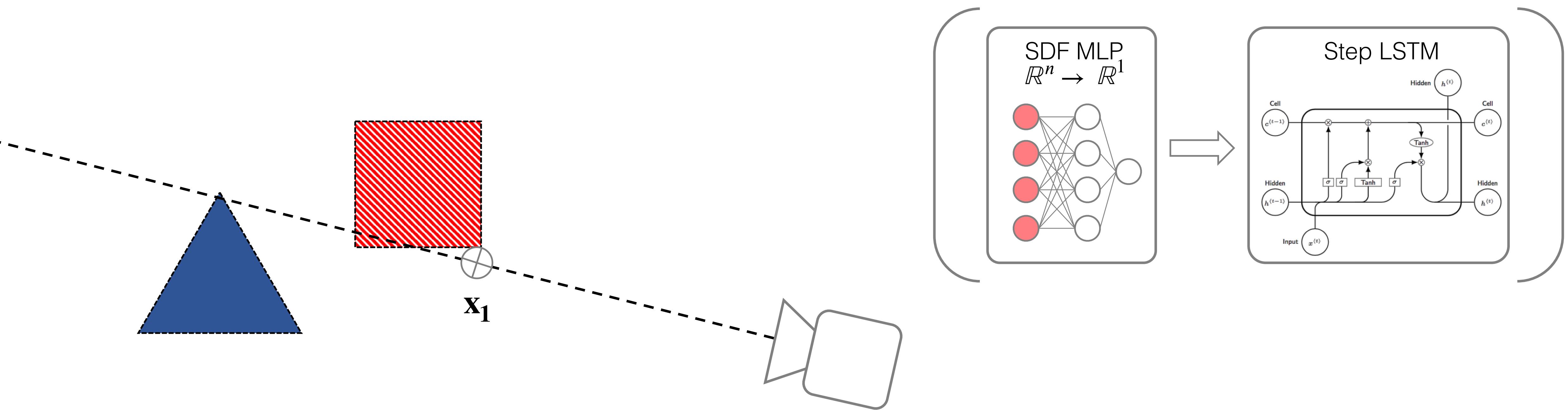


Sphere Tracing Step 1: Intersection Testing.

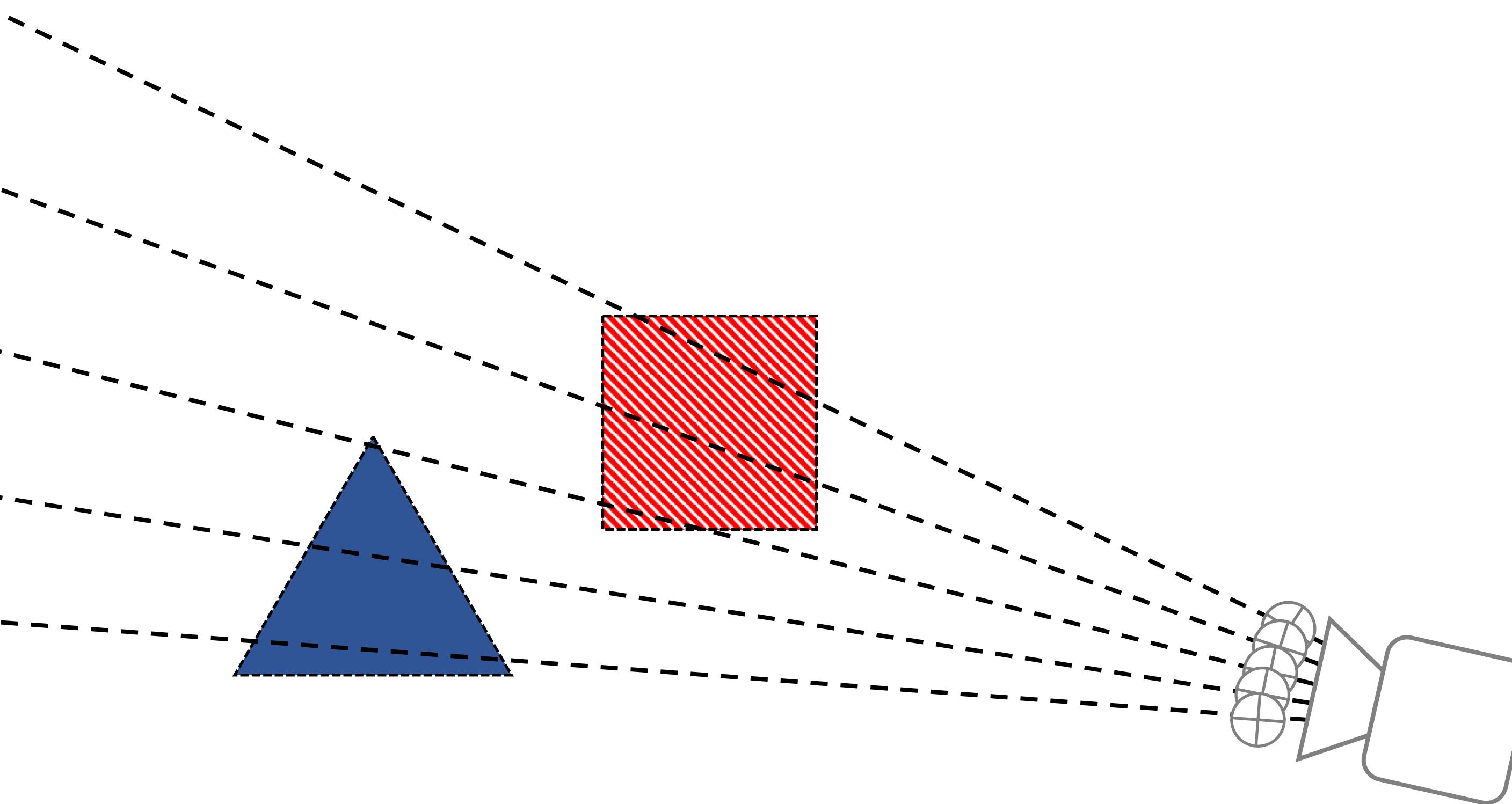


Sphere Tracing Step 1: Intersection Testing.

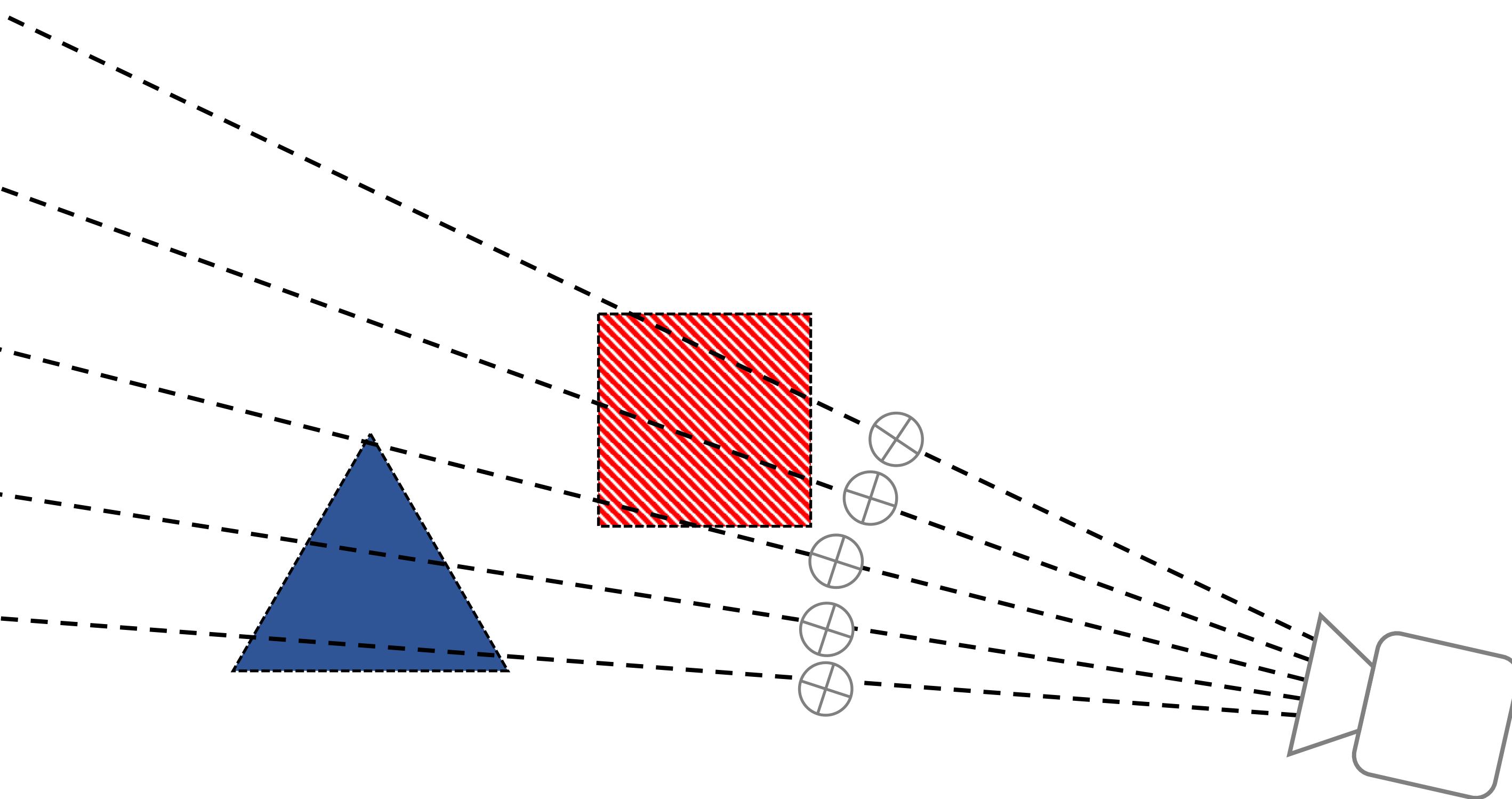
- Guaranteed to converge to surface.
- Guaranteed to not step *through* a surface.



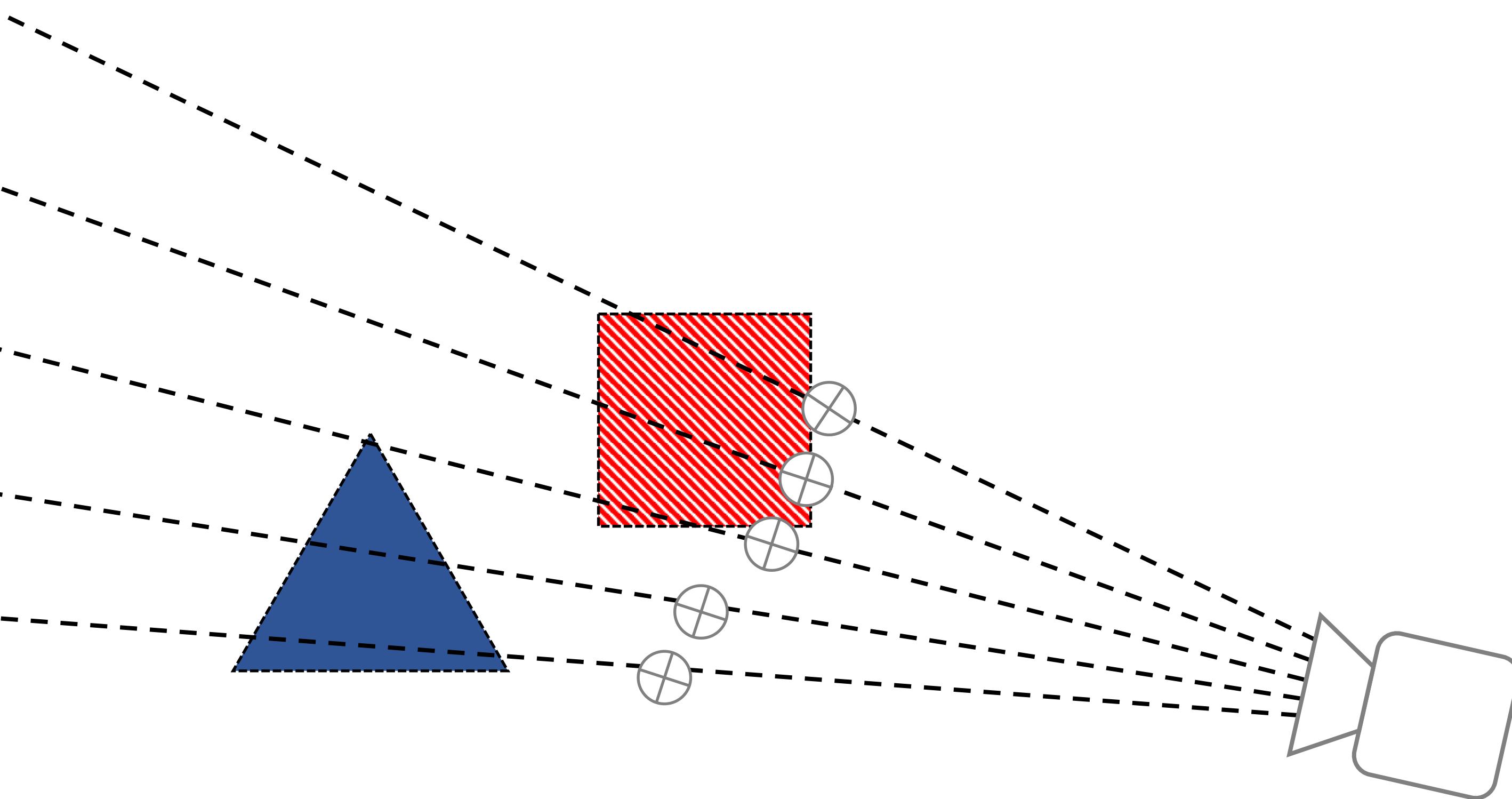
Sphere Tracing Step 1: Intersection Testing.



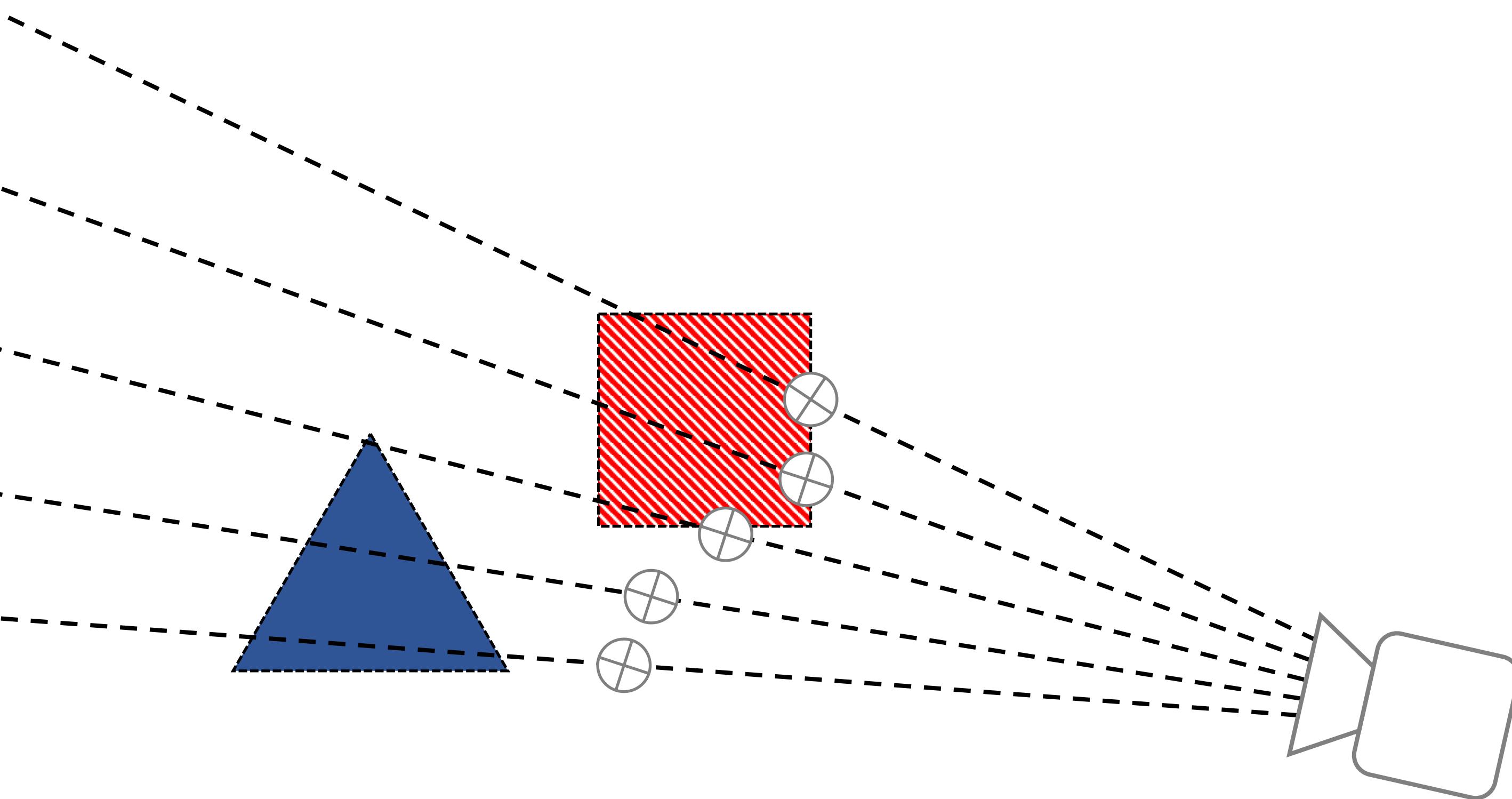
Sphere Tracing Step 1: Intersection Testing.



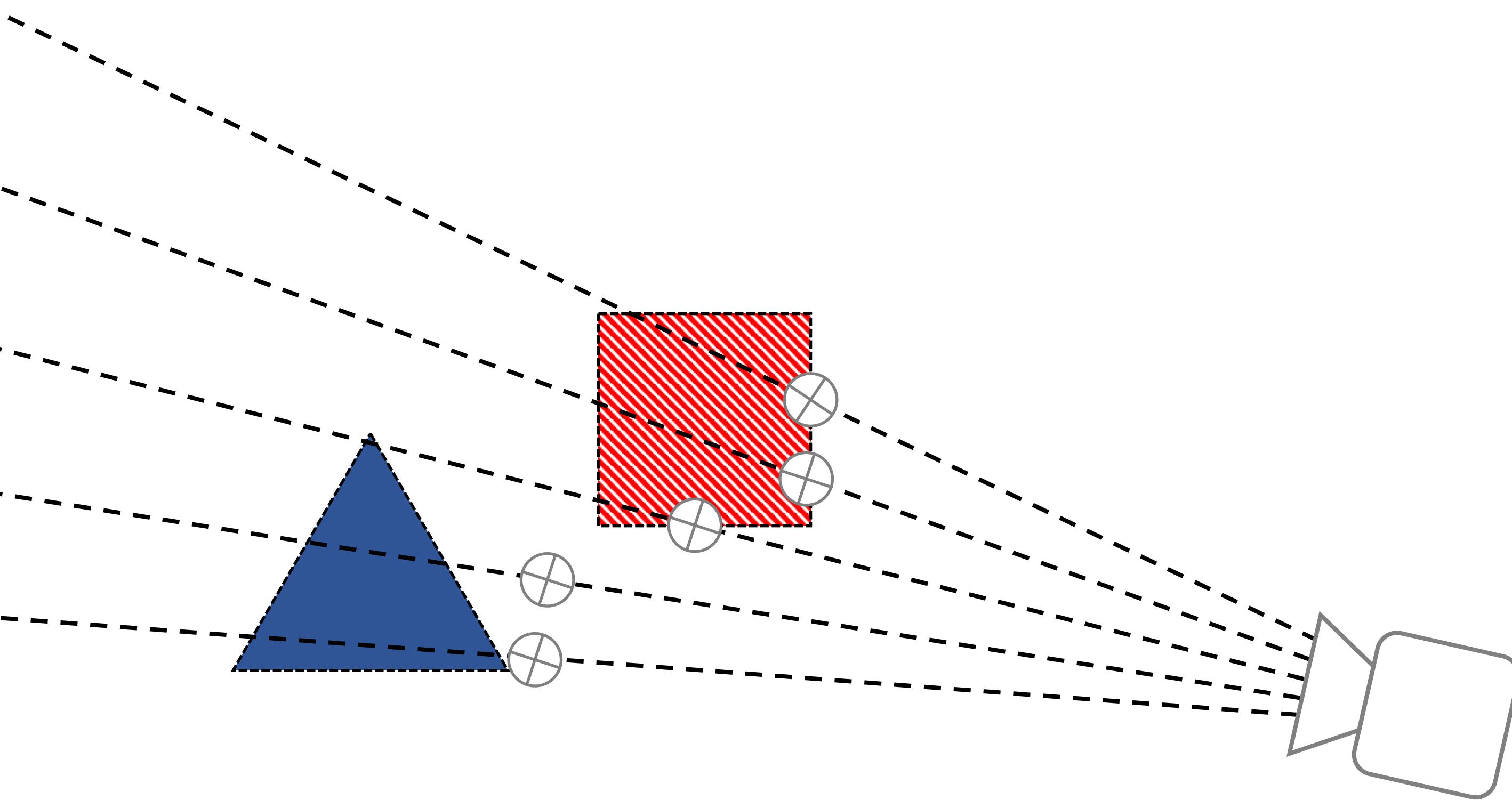
Sphere Tracing Step 1: Intersection Testing.



Sphere Tracing Step 1: Intersection Testing.

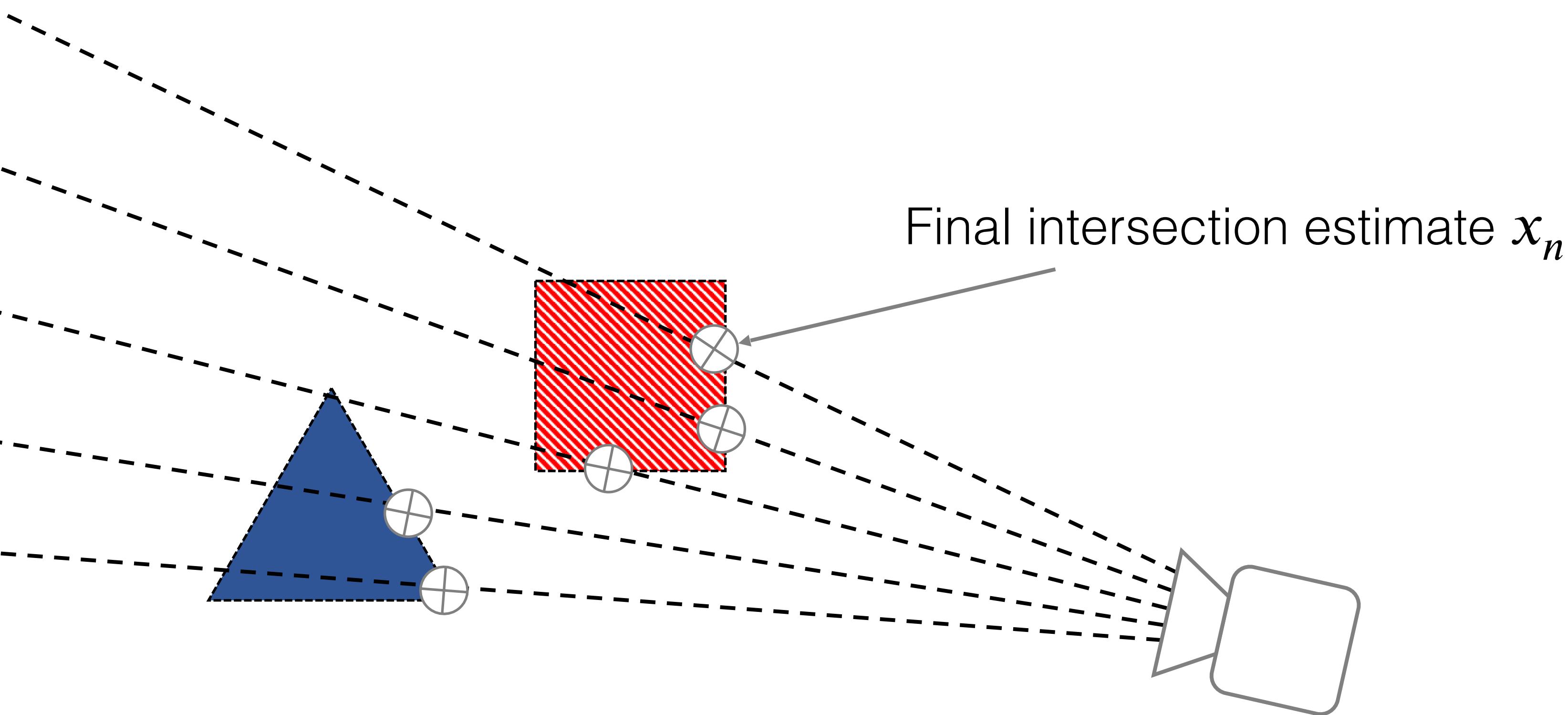


Sphere Tracing Step 1: Intersection Testing.

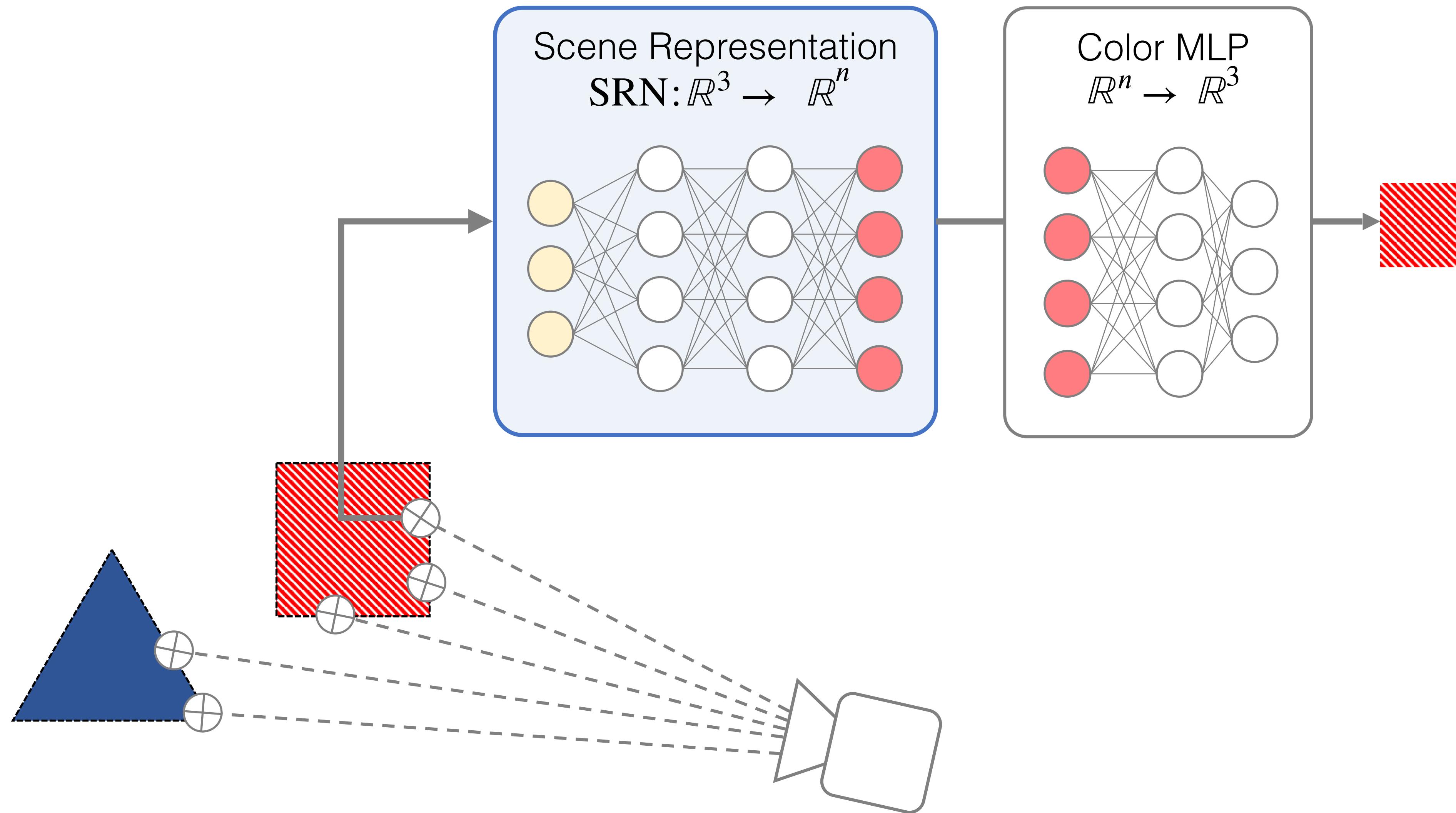


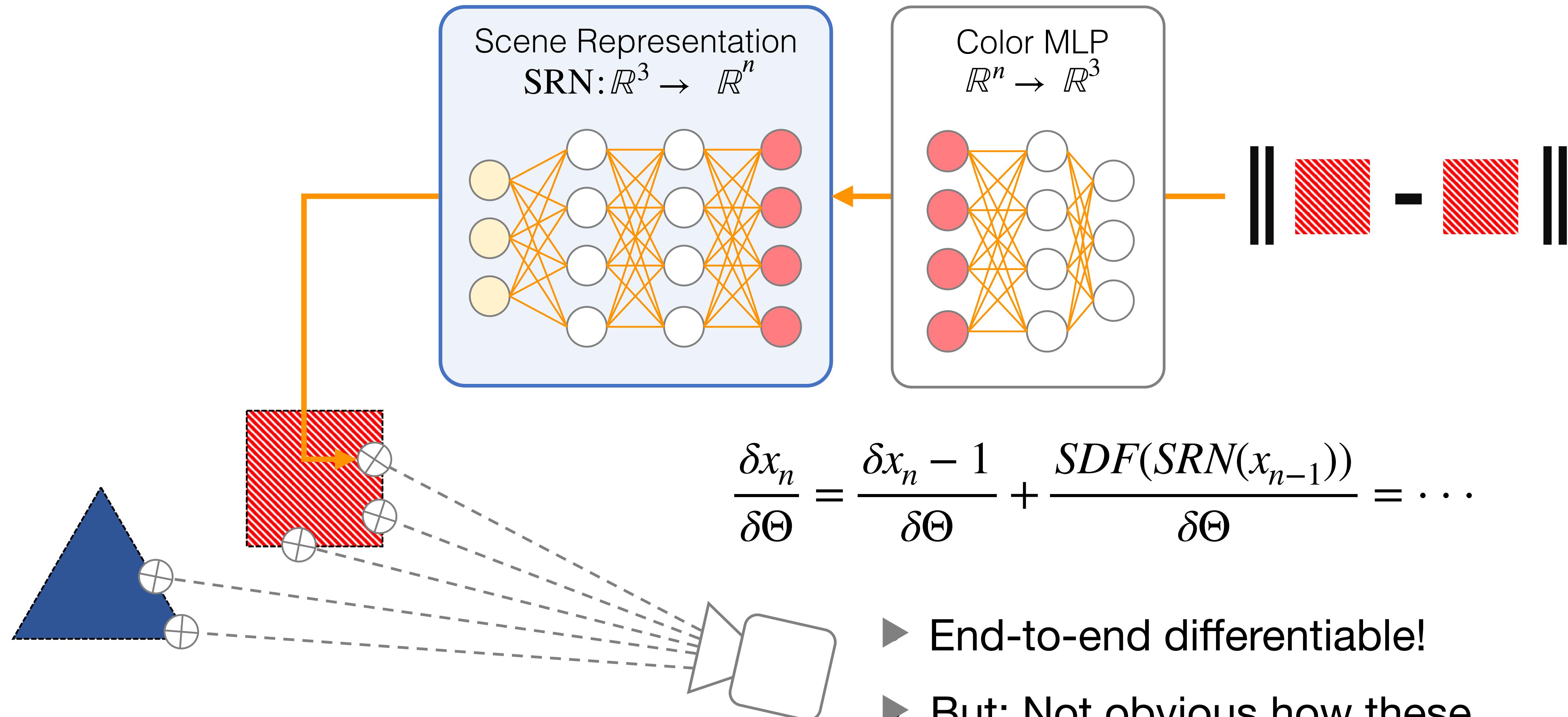
Sphere Tracing Step 1: Intersection Testing.

- Length of each ray yields the depth map!



Neural Renderer Step 2: Color Generation





Questions?

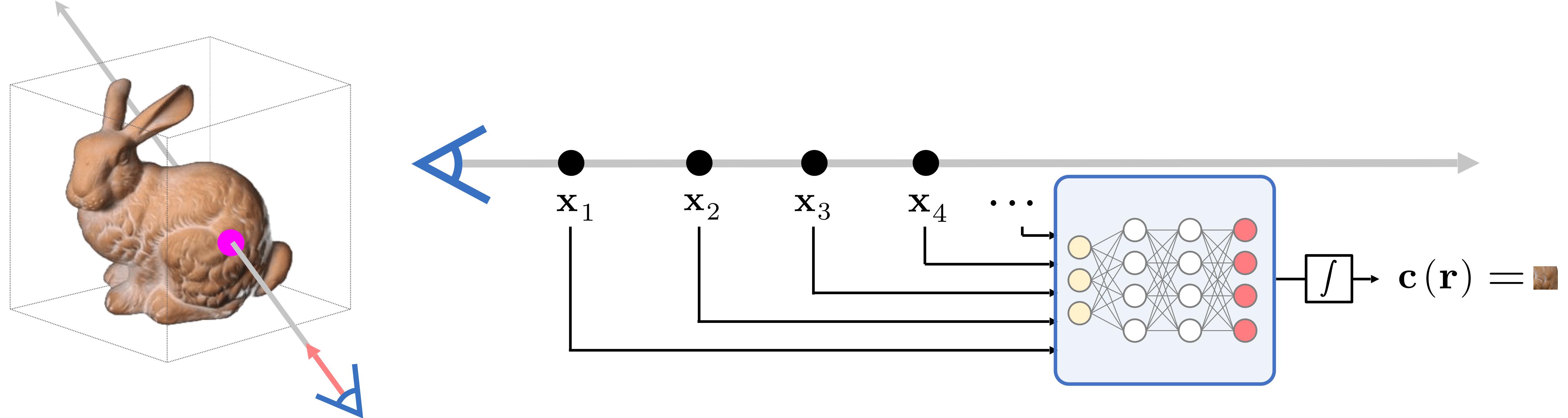
Further reading on computing gradients of sphere-tracing based renderers

Differentiable Rendering of Neural SDFs
through Reparameterization
Bangaru et al. 2022

Differentiable Volumetric Rendering
Niemeyer et al. 2020

Differentiable signed distance function
rendering
Vicini et al. 2022

General structure of Neural Renderers for 3D Fields



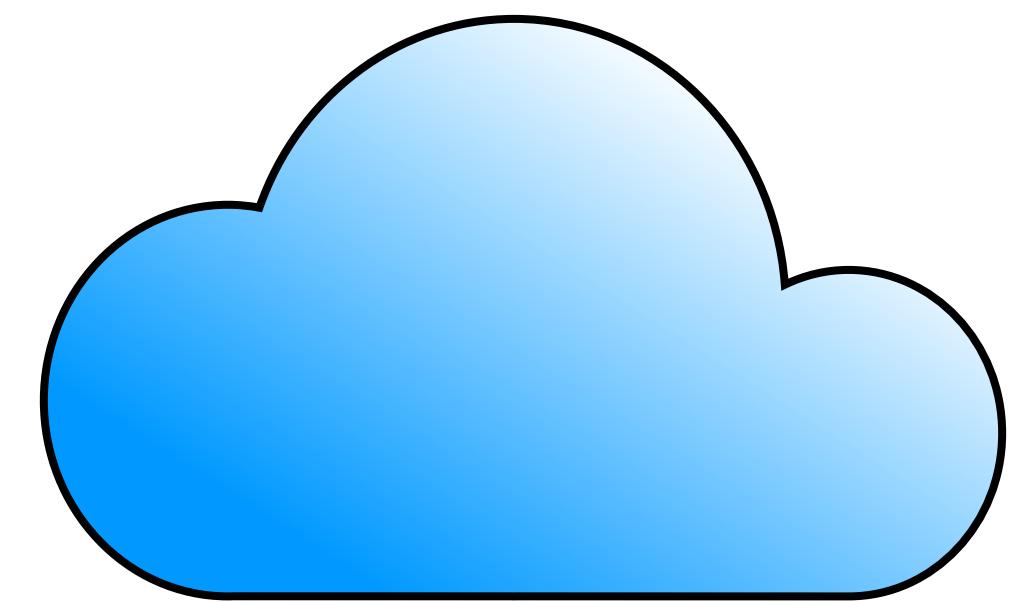
Sphere-Tracing
[JC Hart, 1996]

Volumetric

Hybrid implicit-volumetric

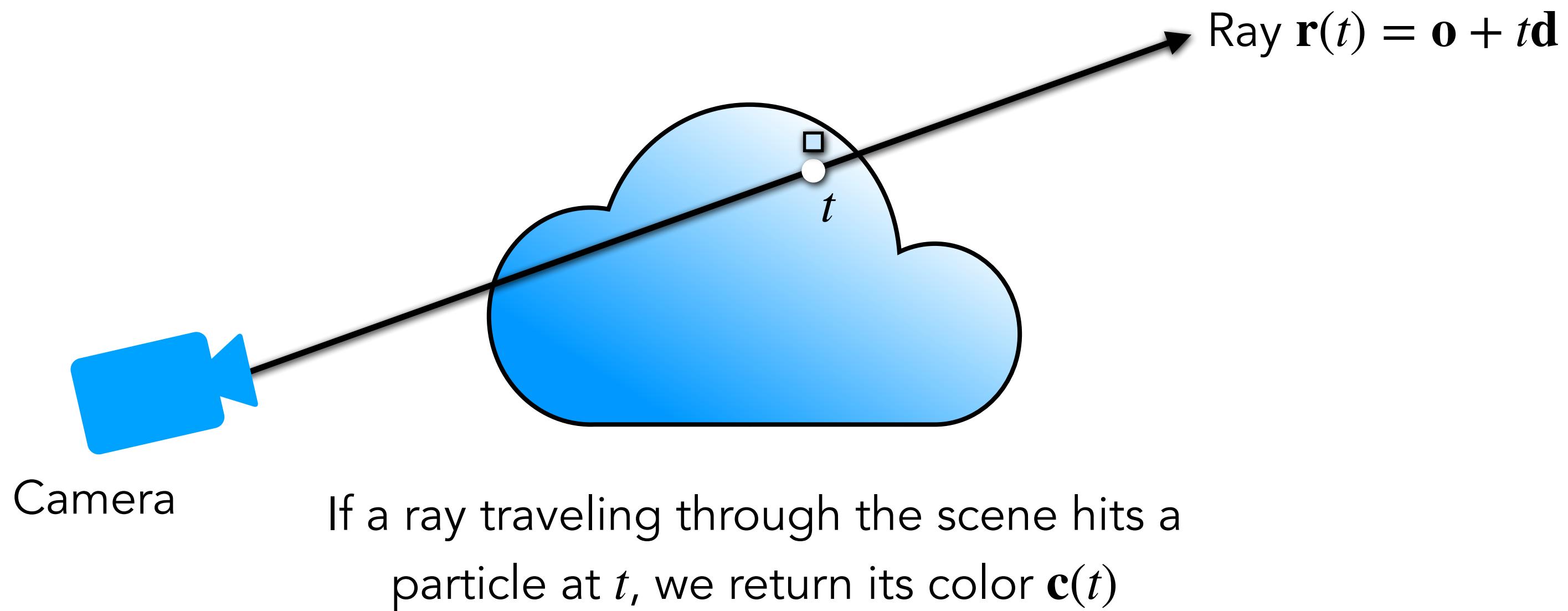
Learned aggregation

Volume Rendering

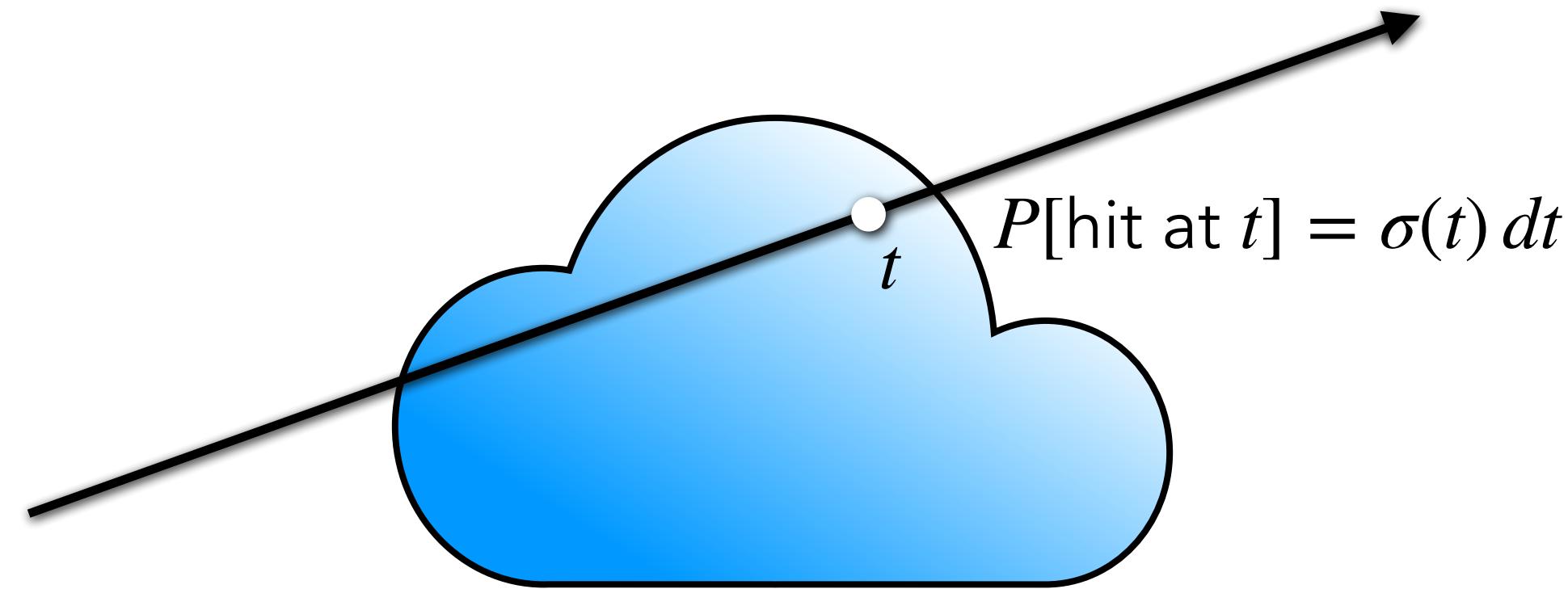


Scene is a cloud of tiny colored particles

Volume Rendering

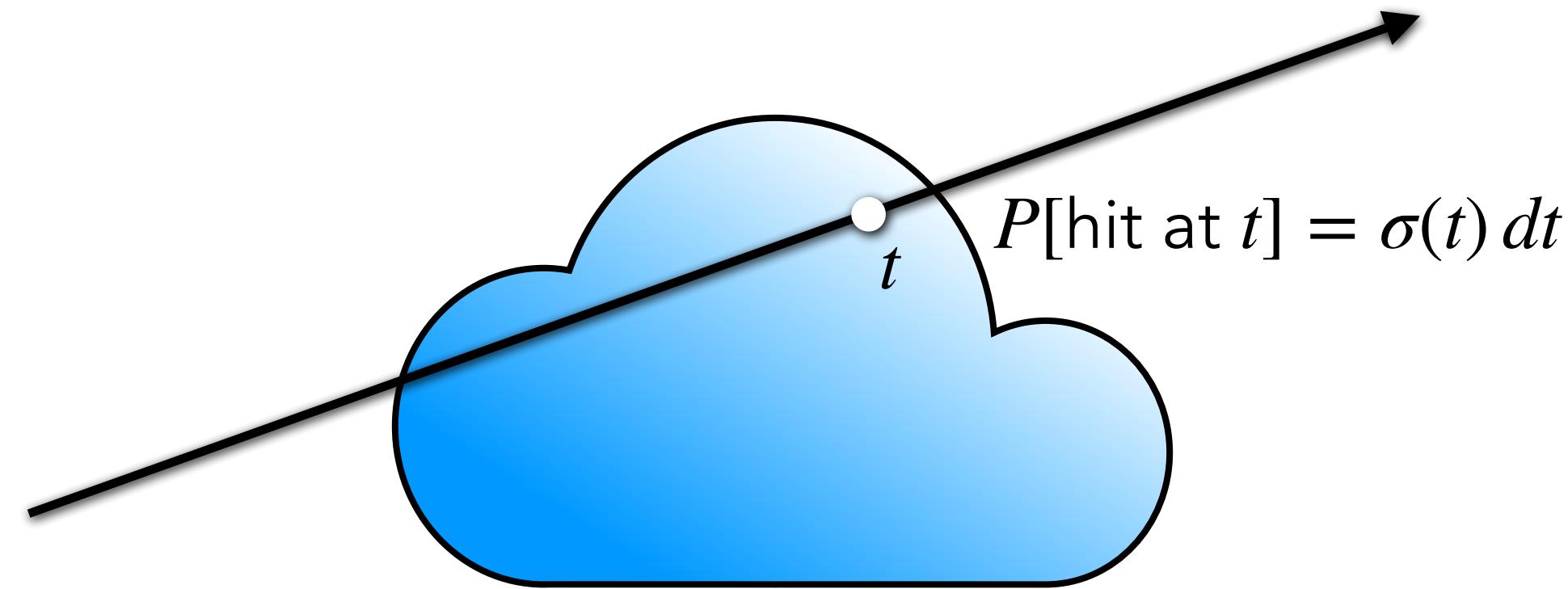


Volume Rendering



Probability that ray stops in a small interval around t is $\sigma(t) dt$.
 σ is known as the **Volume Density**.

Volume Rendering

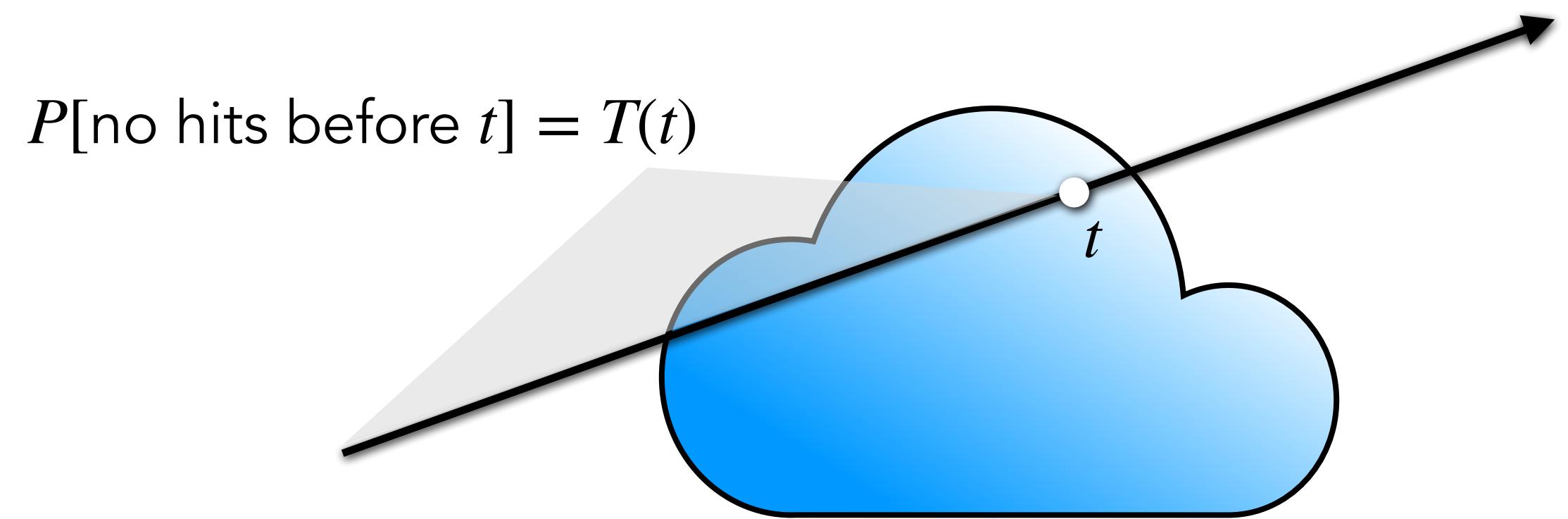


Probability that ray stops in a small interval around t is $\sigma(t) dt$.
 σ is known as the **Volume Density**.

Our field thus has the function signature:

$$\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \times \mathbb{R}^+; \quad \Phi(\mathbf{x}) = (\sigma, \mathbf{c})$$

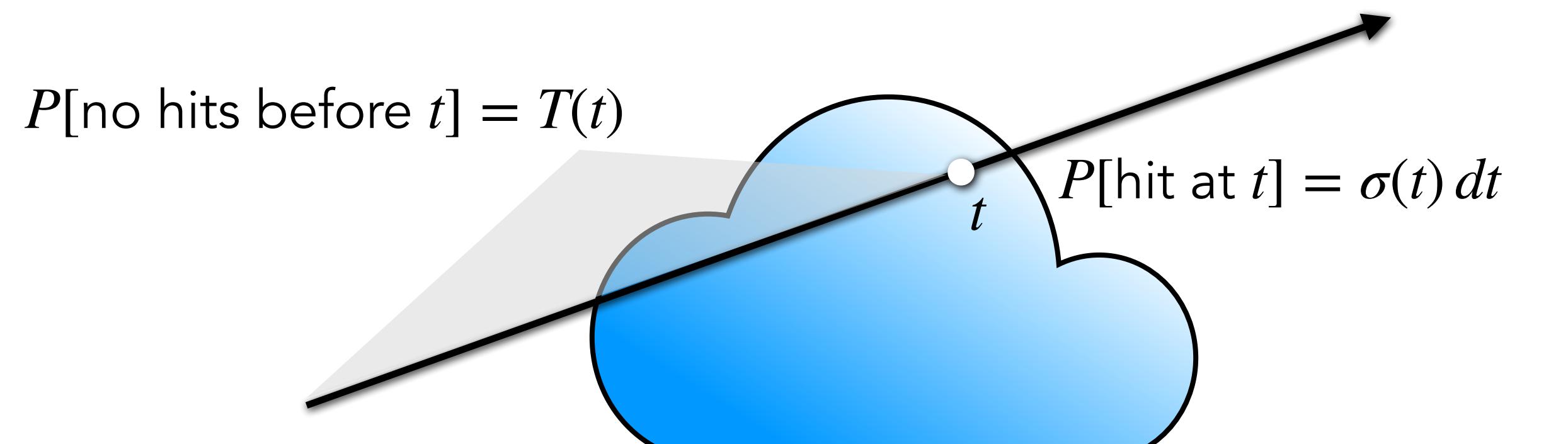
Volume Rendering



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**.

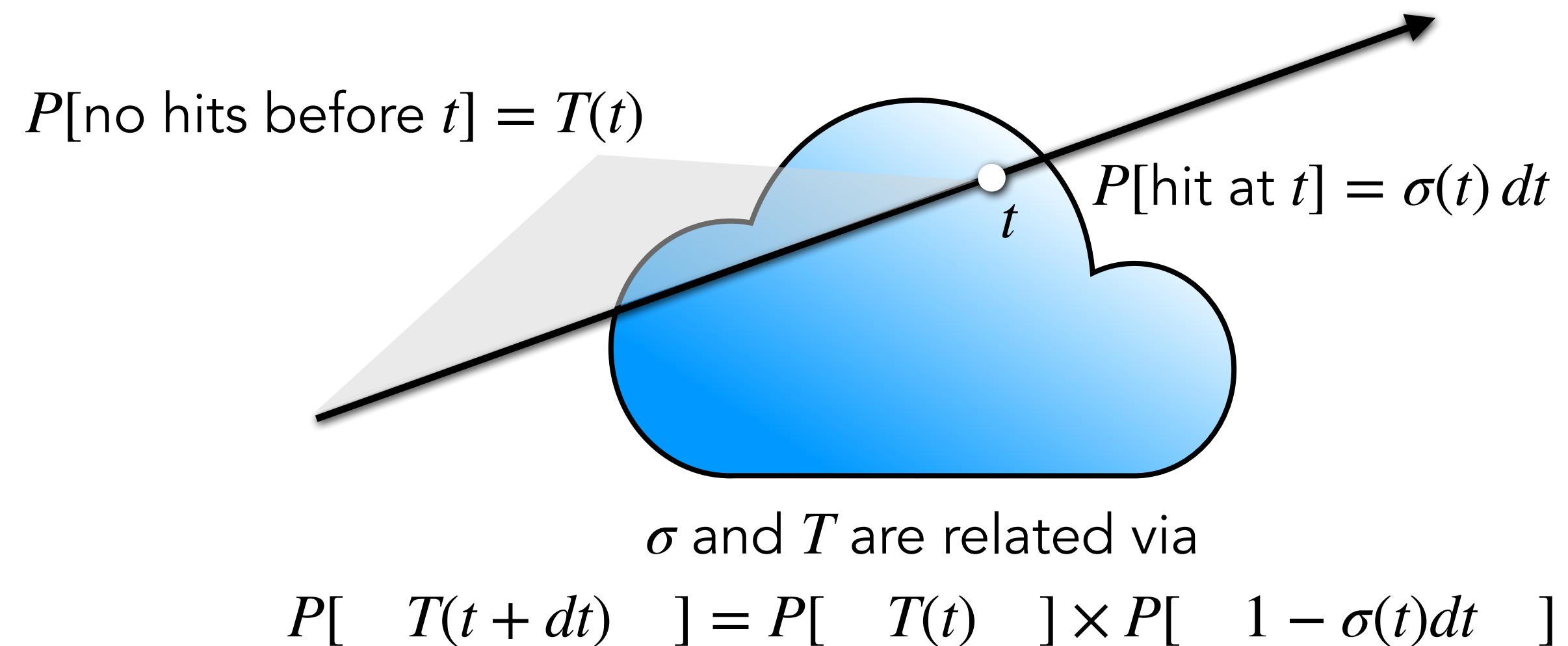
Volume Rendering



σ and T are related via

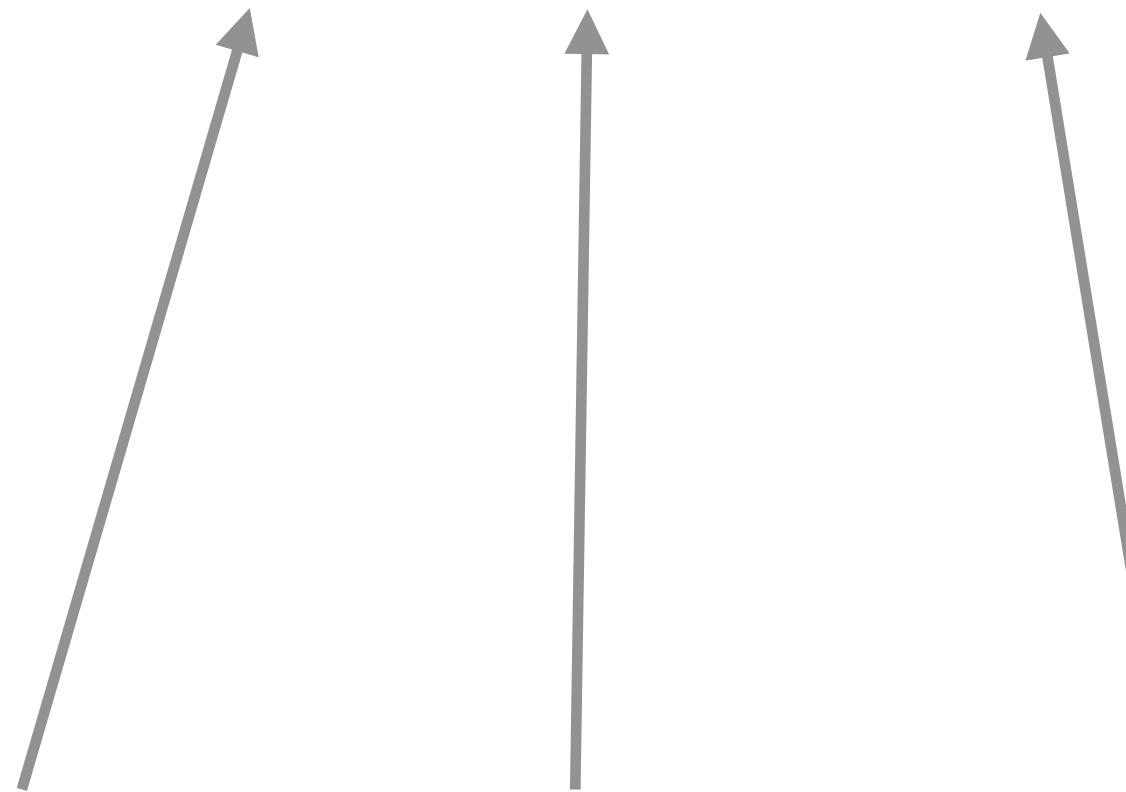
$$P[\text{no hit before } t + dt] = P[\text{no hit before } t] \times P[\text{no hit at } t]$$

Volume Rendering



Volume Rendering

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



$$P[\quad T(t + dt) \quad] = P[\quad T(t) \quad] \times P[\quad 1 - \sigma(t)dt \quad]$$

Volume Rendering

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

$$\frac{T(t + dt) - T(t)}{dt} = -T(t)\sigma(t)$$

$$T'(t) = -T(t)\sigma(t)$$

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

$$\int_a^b \frac{T'(t)}{T(t)} dt = - \int_a^b \sigma(t) dt$$

$$\log T(t) \Big|_a^b = - \int_a^b \sigma(t) dt$$

Volume Rendering

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

$$\frac{T(t + dt) - T(t)}{dt} = - T(t)\sigma(t)$$

$$T'(t) = - T(t)\sigma(t)$$

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

$$\int_a^b \frac{T'(t)}{T(t)} dt = - \int_a^b \sigma(t) dt$$

$$\log T(t) \Big|_a^b = - \int_a^b \sigma(t) dt$$

**Solving simple differential equation
(for great explanation, see “Volume Rendering Digest”, Tagliasacchi et al.)**

Volume Rendering

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

$$\frac{T(t + dt) - T(t)}{dt} = -T(t)\sigma(t)$$

$$T'(t) = -T(t)\sigma(t)$$

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

$$\int_a^b \frac{T'(t)}{T(t)} dt = - \int_a^b \sigma(t) dt$$

$$\log T(t) \Big|_a^b = - \int_a^b \sigma(t) dt$$

$$T(a \rightarrow b) := \frac{T(b)}{T(a)} = \exp \left(- \int_a^b \sigma(t) dt \right)$$

Volume Rendering

$$T(a \rightarrow b) := \frac{T(b)}{T(a)} = \exp\left(-\int_a^b \sigma(t)dt\right)$$

Volume Rendering

$$T(a \rightarrow b) := \frac{T(b)}{T(a)} = \exp\left(-\int_a^b \sigma(t)dt\right)$$

$$T(t) := T(0 \rightarrow t) = \exp\left(-\int_0^t \sigma(a)da\right)$$

Volume Rendering

No hits before t is equal to integral over density up until t .

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

Volume Rendering

No hits before t is equal to integral over density up until t .

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

$1 - T(t)$ can be seen as cumulative distribution function of probability that ray hits something before reaching t .

Volume Rendering

No hits before t is equal to integral over density up until t .

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

$1 - T(t)$ can be seen as cumulative distribution function of probability that ray hits something before reaching t .

Then $T'(t) = T(t)\sigma(t)$ is probability that ray stops *exactly* at t .

Questions?

Volume Rendering

Then $T'(t) = T(t)\sigma(t)$ is probability that ray stops *exactly* at t .

Volume Rendering

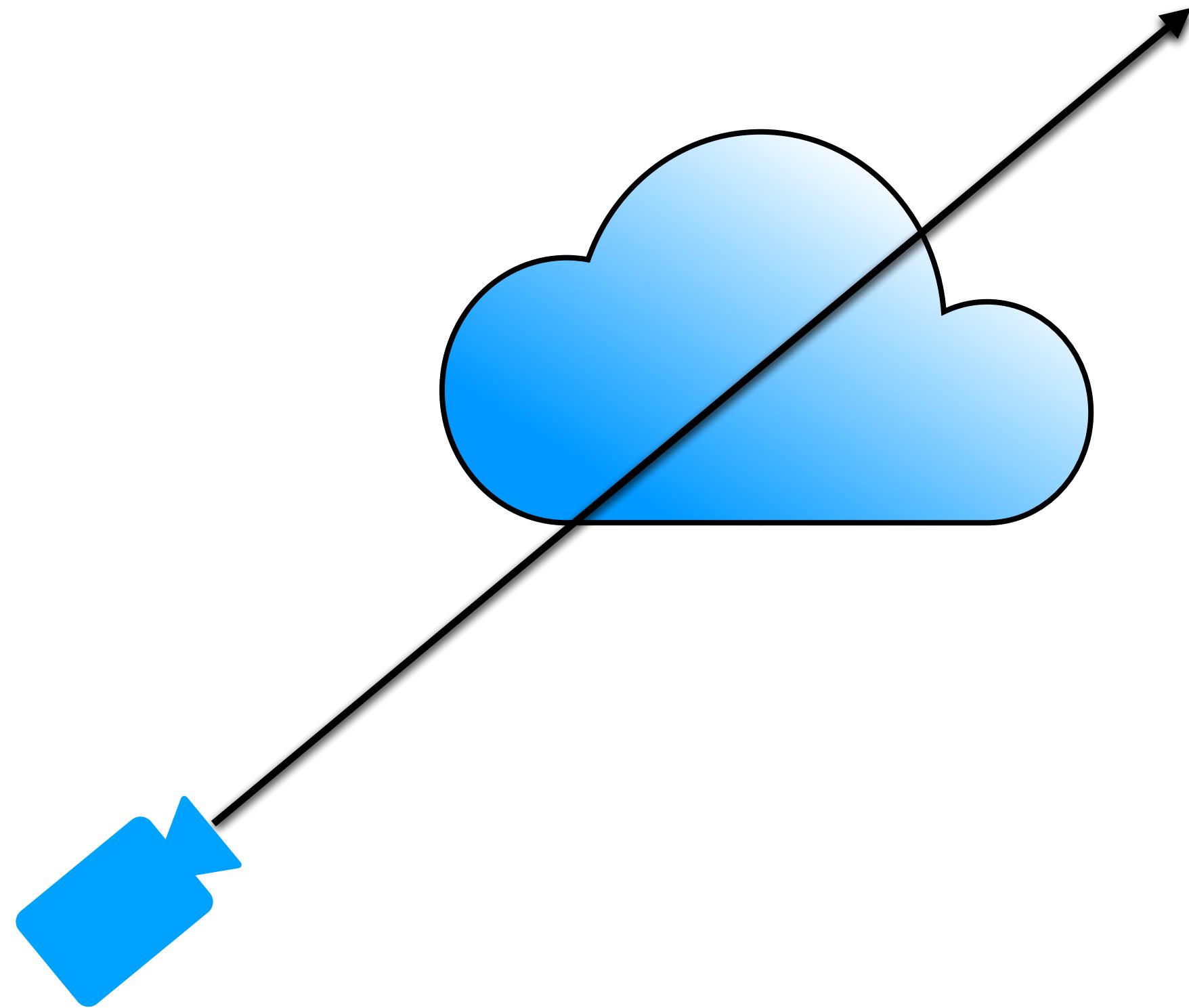
Then $T'(t) = T(t)\sigma(t)$ is probability that ray stops *exactly* at t .

So the expected color returned by the ray will be

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

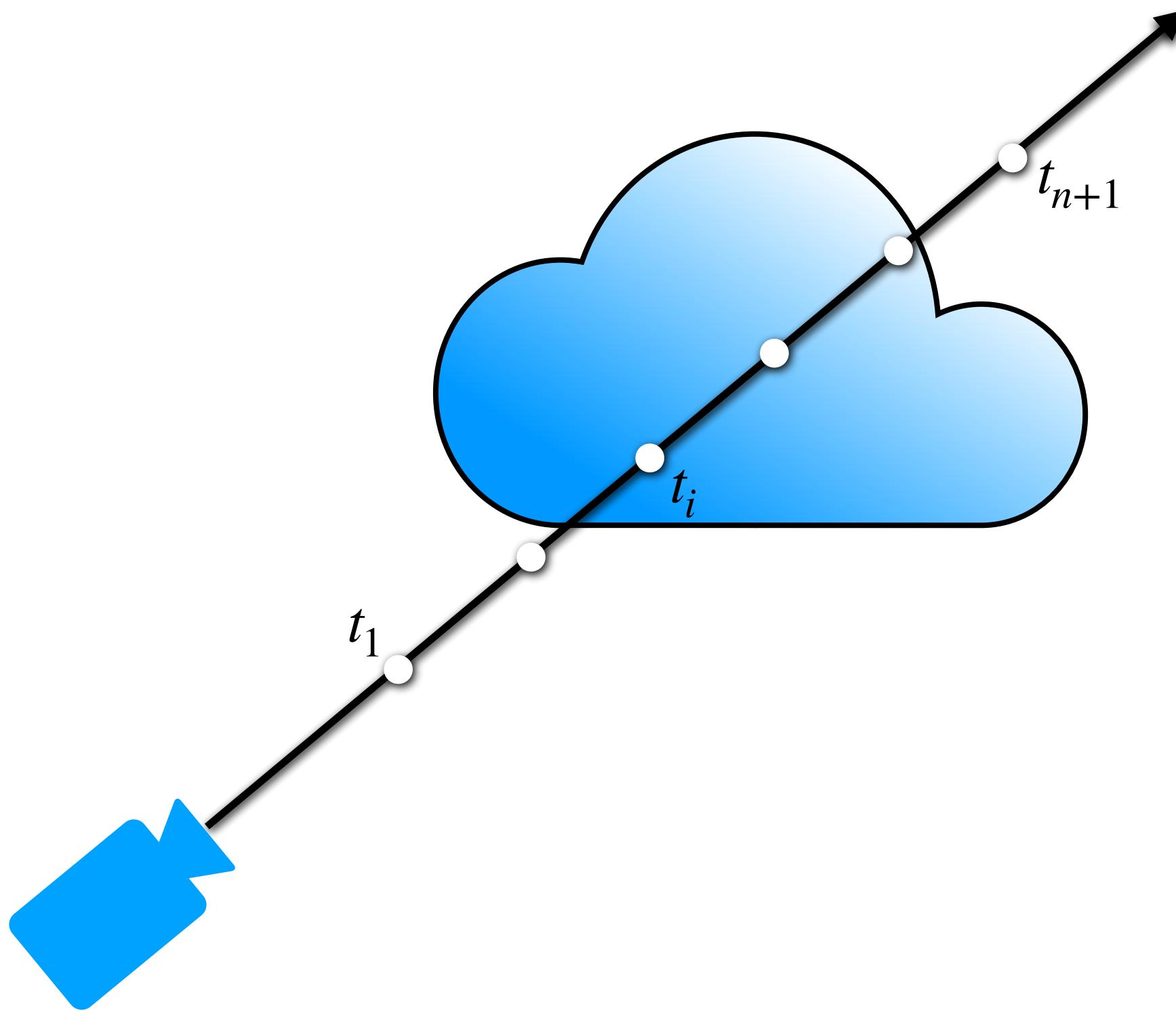
Note the nested integral!

Approximating the nested integral



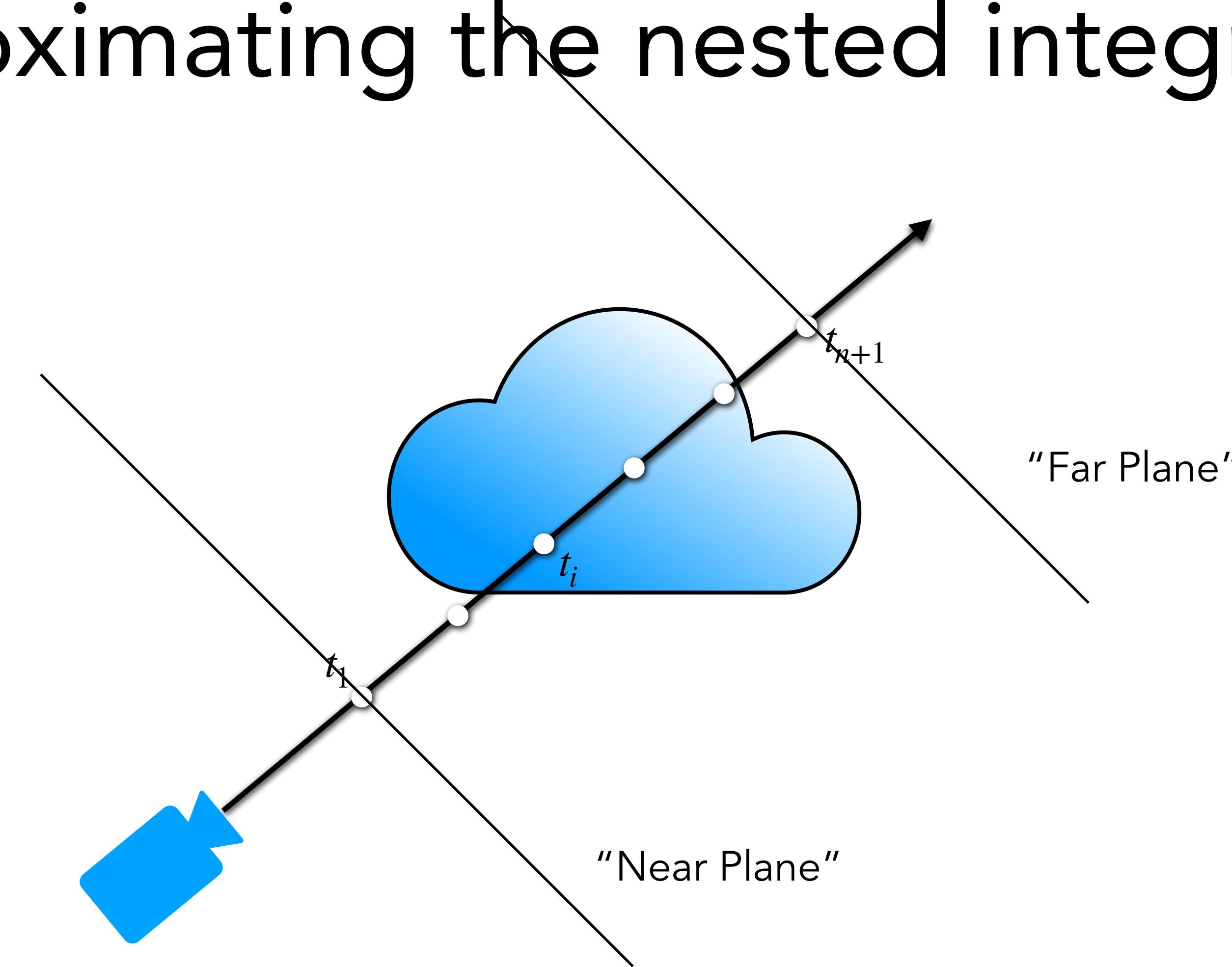
We use quadrature to approximate the nested integral,

Approximating the nested integral



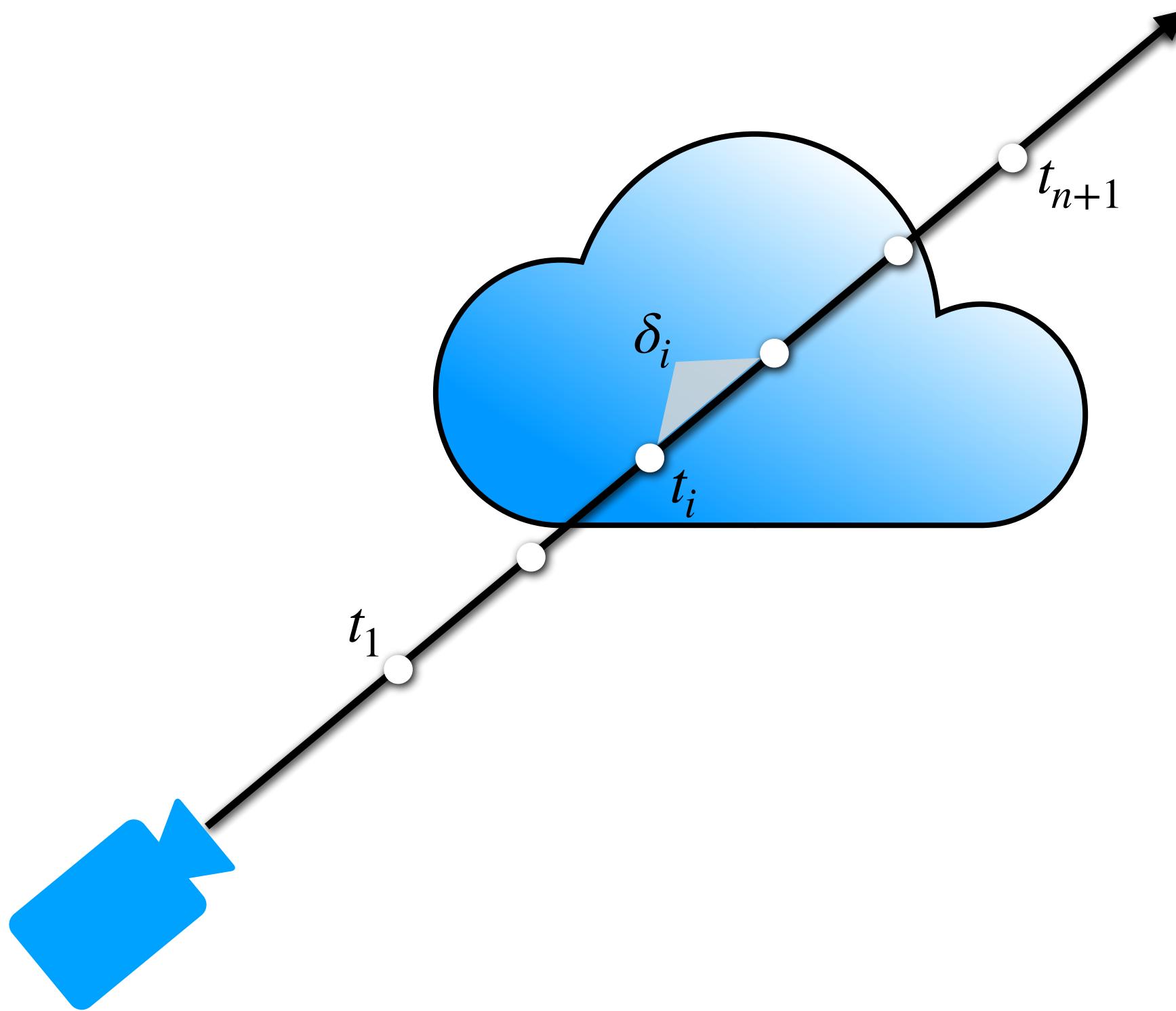
We use quadrature to approximate the nested integral,
splitting the ray up into n segments with endpoints $\{t_1, t_2, \dots, t_{n+1}\}$

Approximating the nested integral



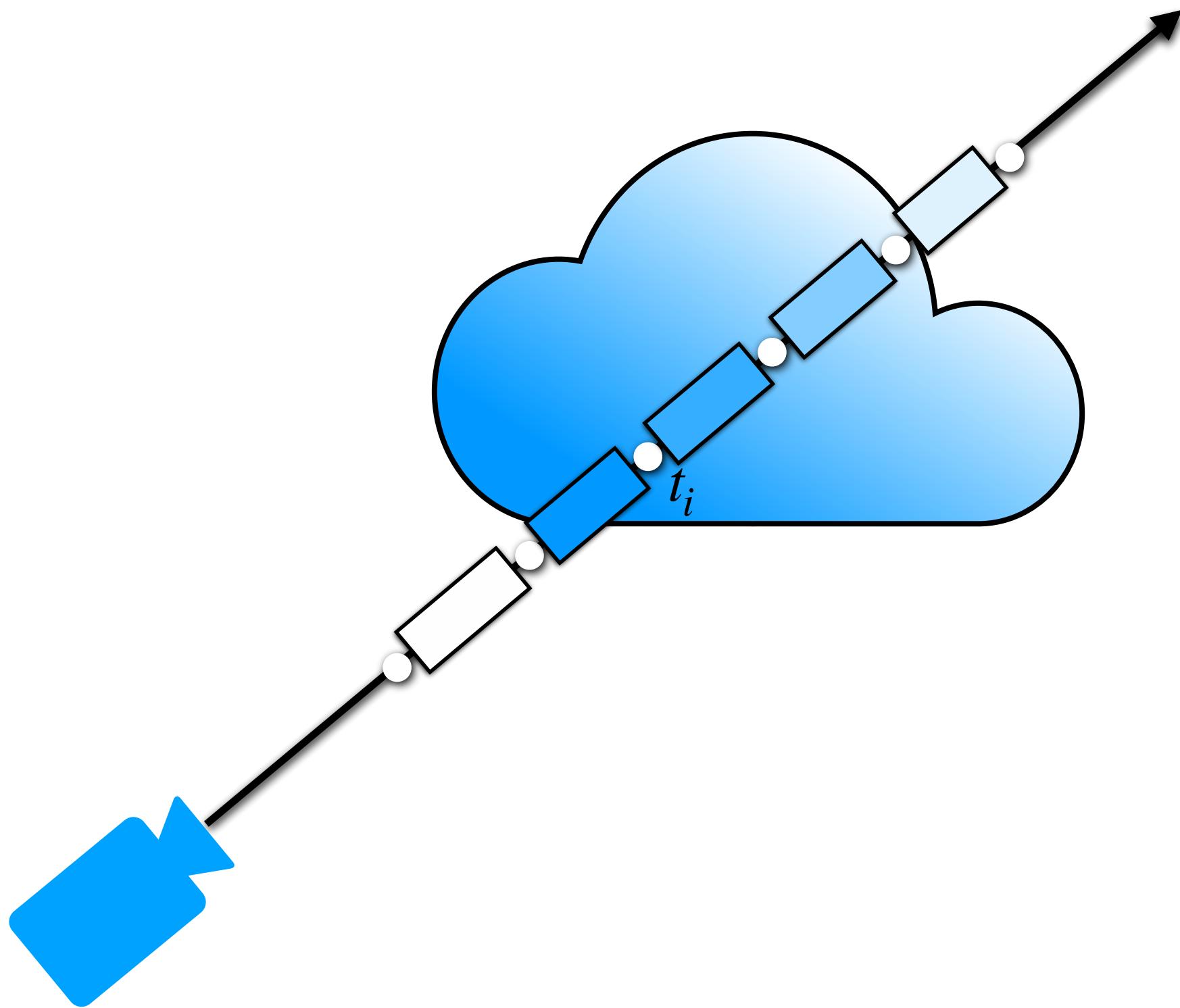
We use quadrature to approximate the nested integral,
splitting the ray up into n segments with endpoints $\{t_1, t_2, \dots, t_{n+1}\}$

Approximating the nested integral



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

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

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

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

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

Approximating the nested integral

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

Caveat: 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

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

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



$$\exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) = T_i \quad \text{"How much is blocked by all previous segments?"}$$

Approximating the nested integral

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

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?"

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

Approximating the nested integral

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

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

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

Approximating the nested integral

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

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

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

Approximating the nested integral

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

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

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

Cancel σ_i

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

Summary: volume rendering integral estimate

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

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

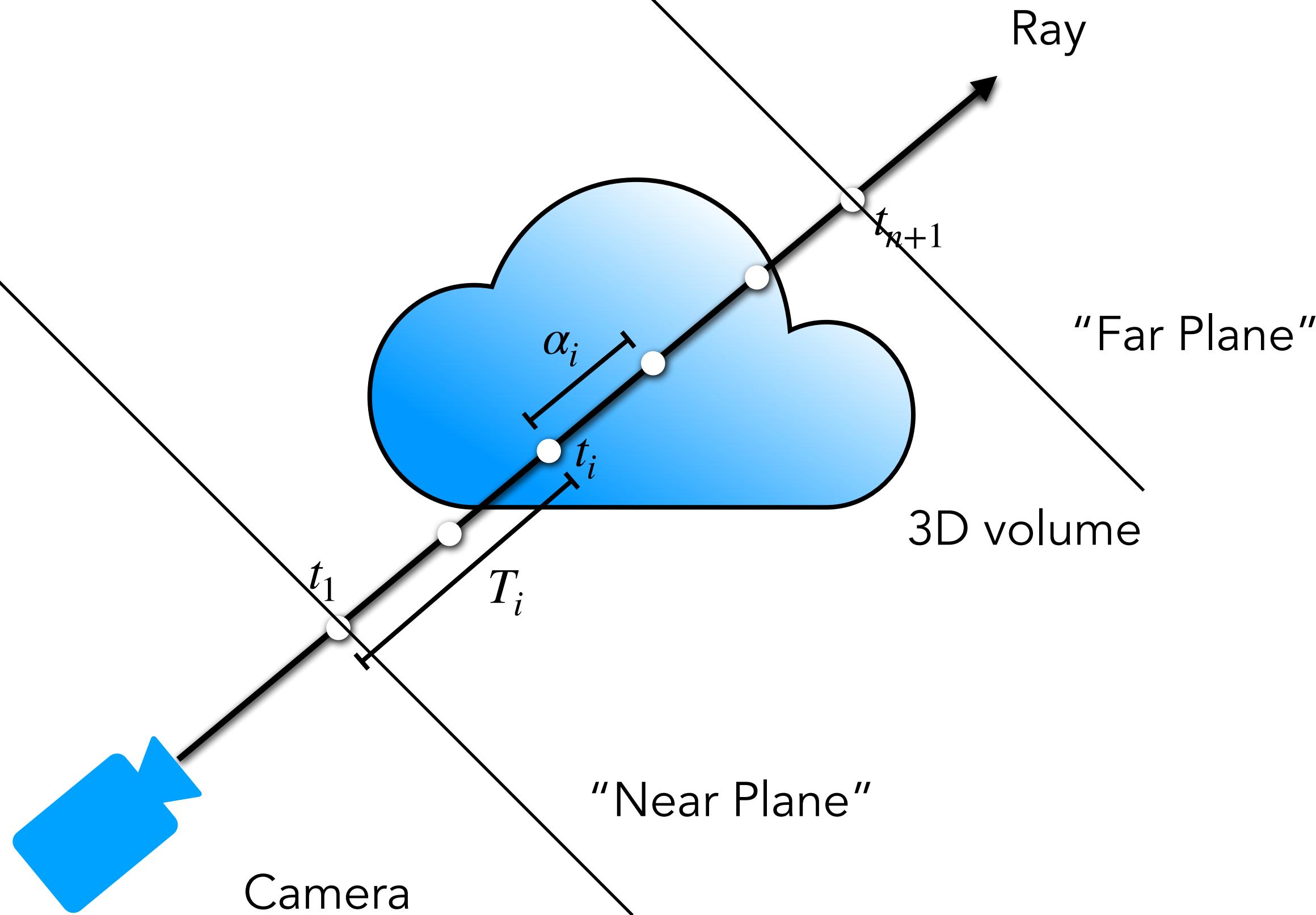
↑
weights colors

How much light is blocked earlier along ray:

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

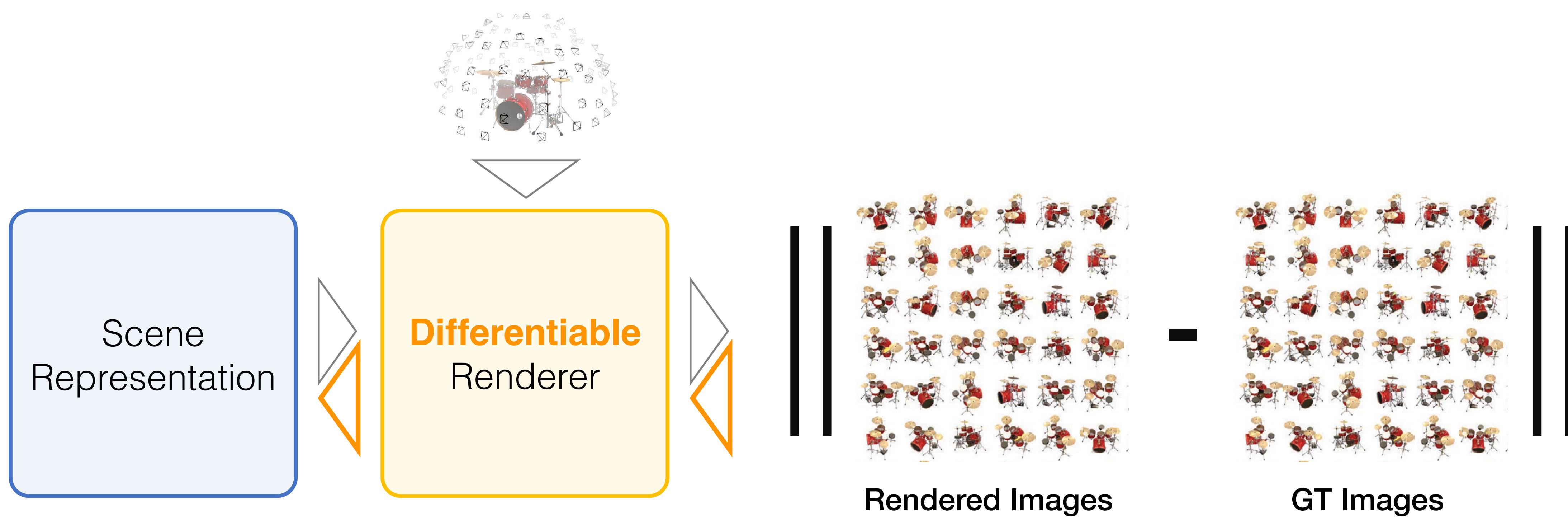
How much light is contributed by ray segment i :

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



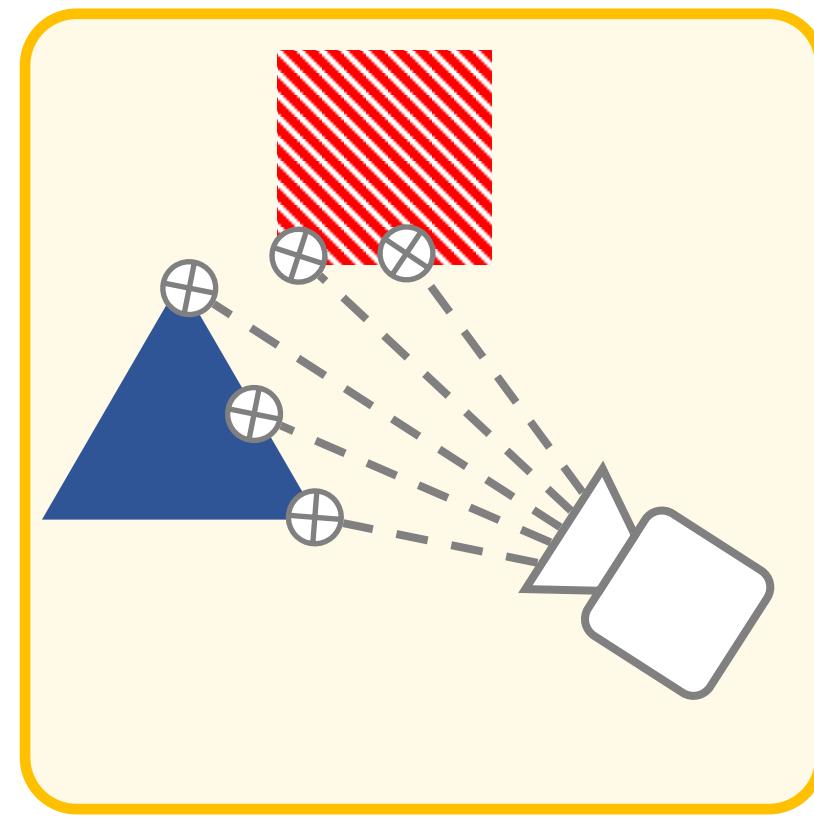
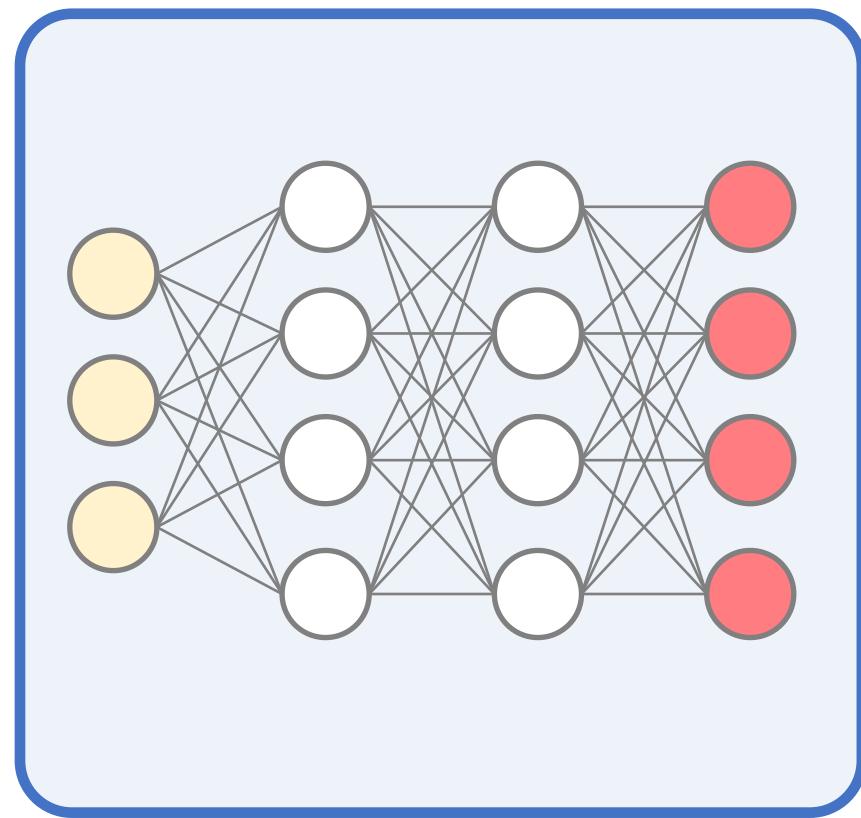
Questions?

Differentiable Renderer is done!

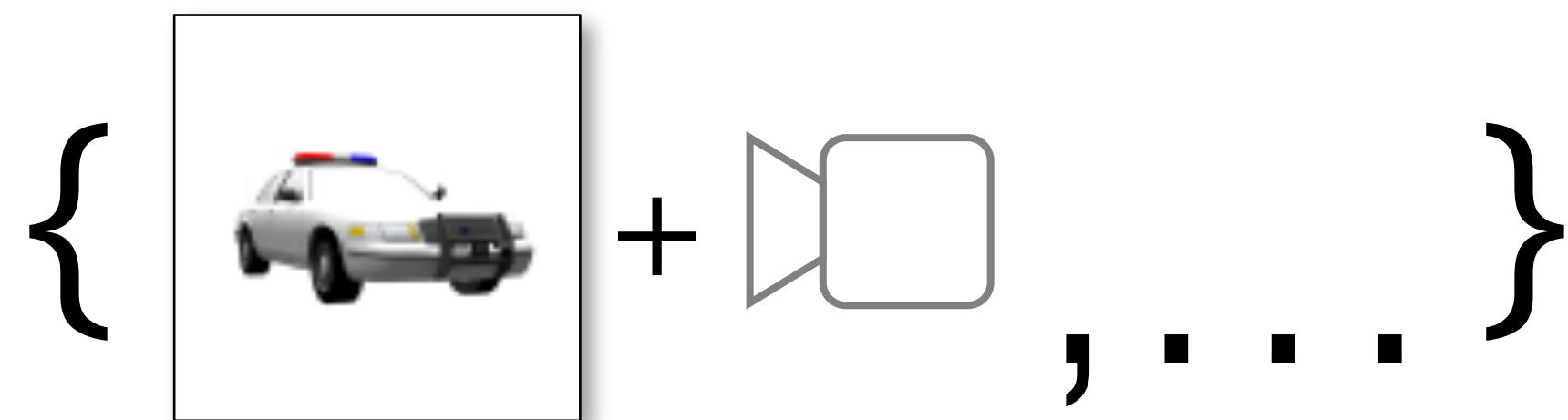


Given an *observable* variable (pixel colors), we will build a differentiable forward model that we then use to estimate *unobserved (latent) variables* (geometry, appearance)!

Test case: Single scene with many observations.



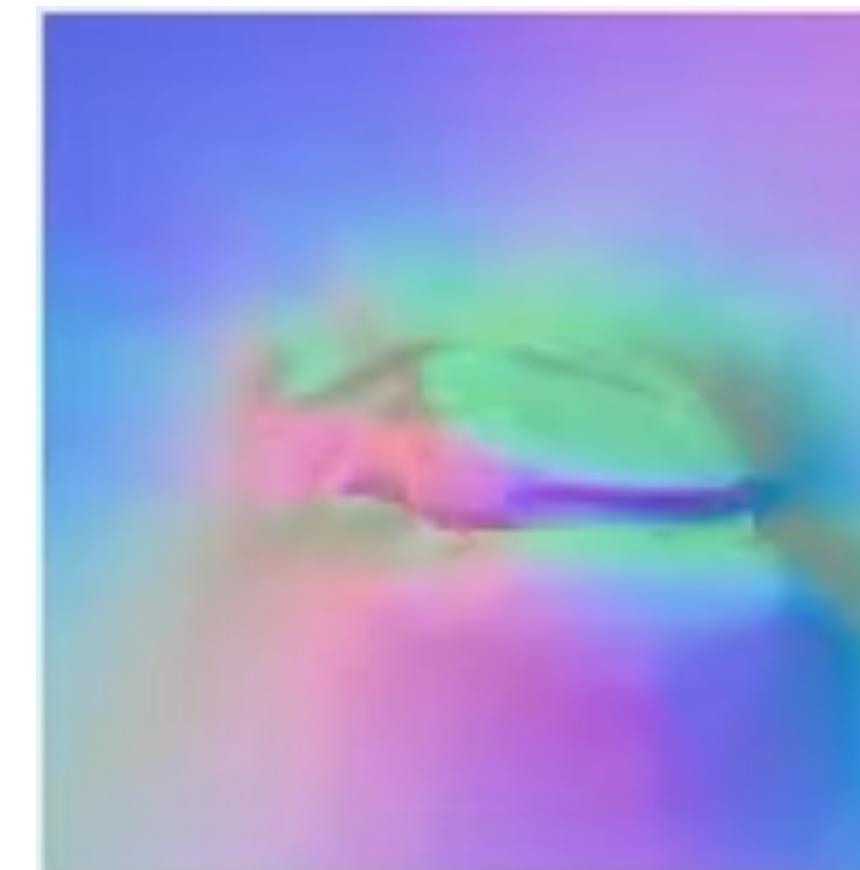
Optimization



$$\left\{ (\mathcal{J}, \xi)_i \right\}_{i=1}^N$$

$$\operatorname{argmin}_{\phi, \theta} \sum_i \left\| \text{Render}_{\theta}(\text{Srn}_{\phi}, \xi_i) - \mathcal{J}_i \right\|$$

Novel View Synthesis



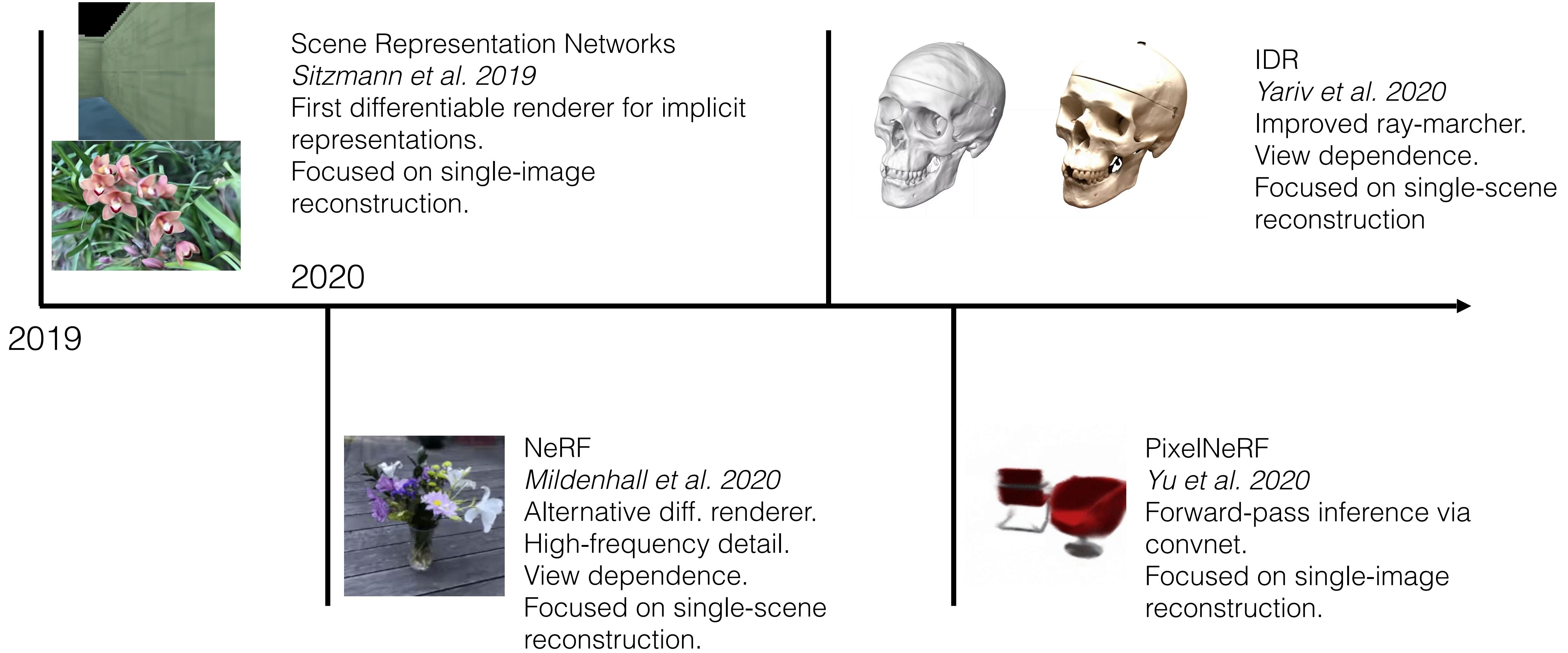
Normal map



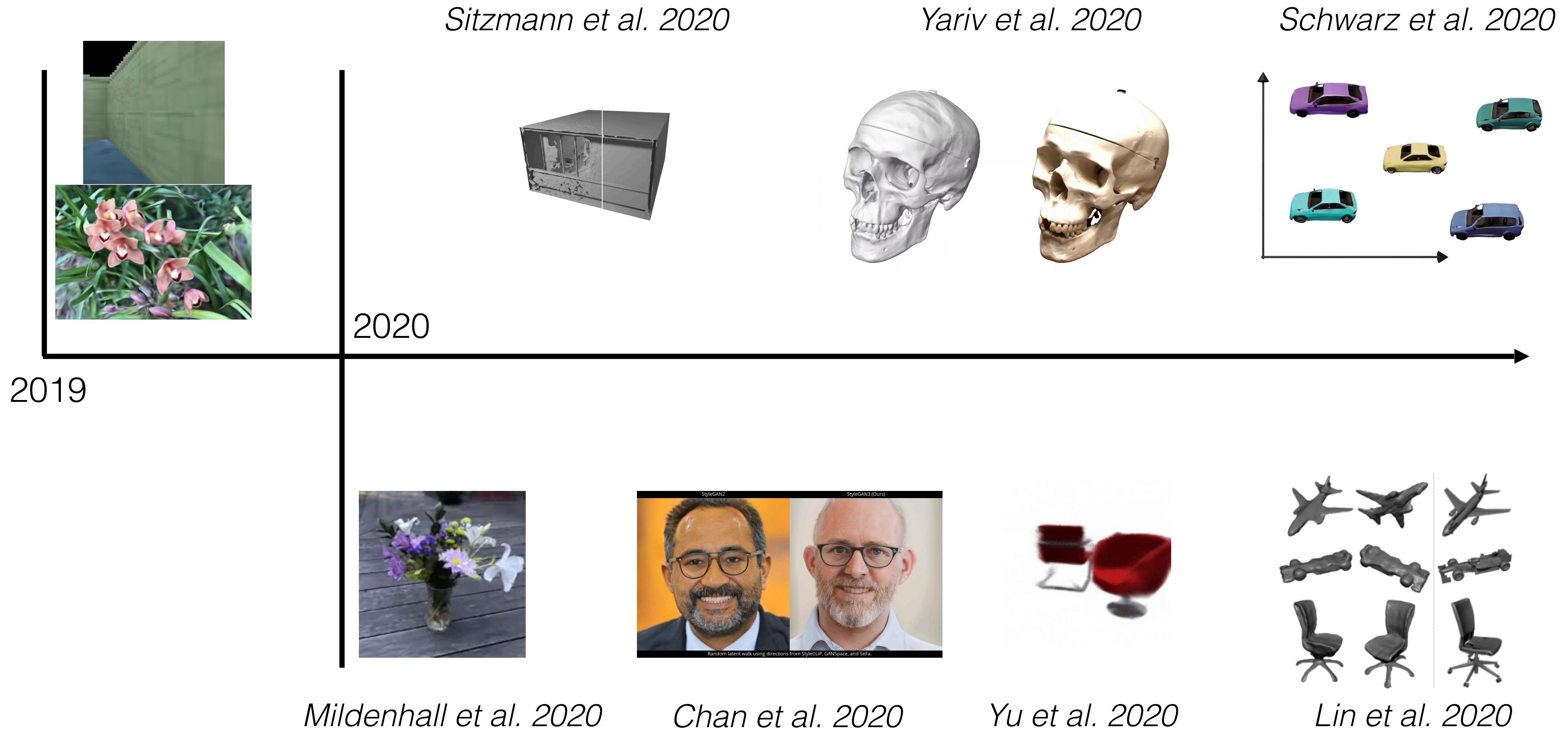
RGB

Questions?

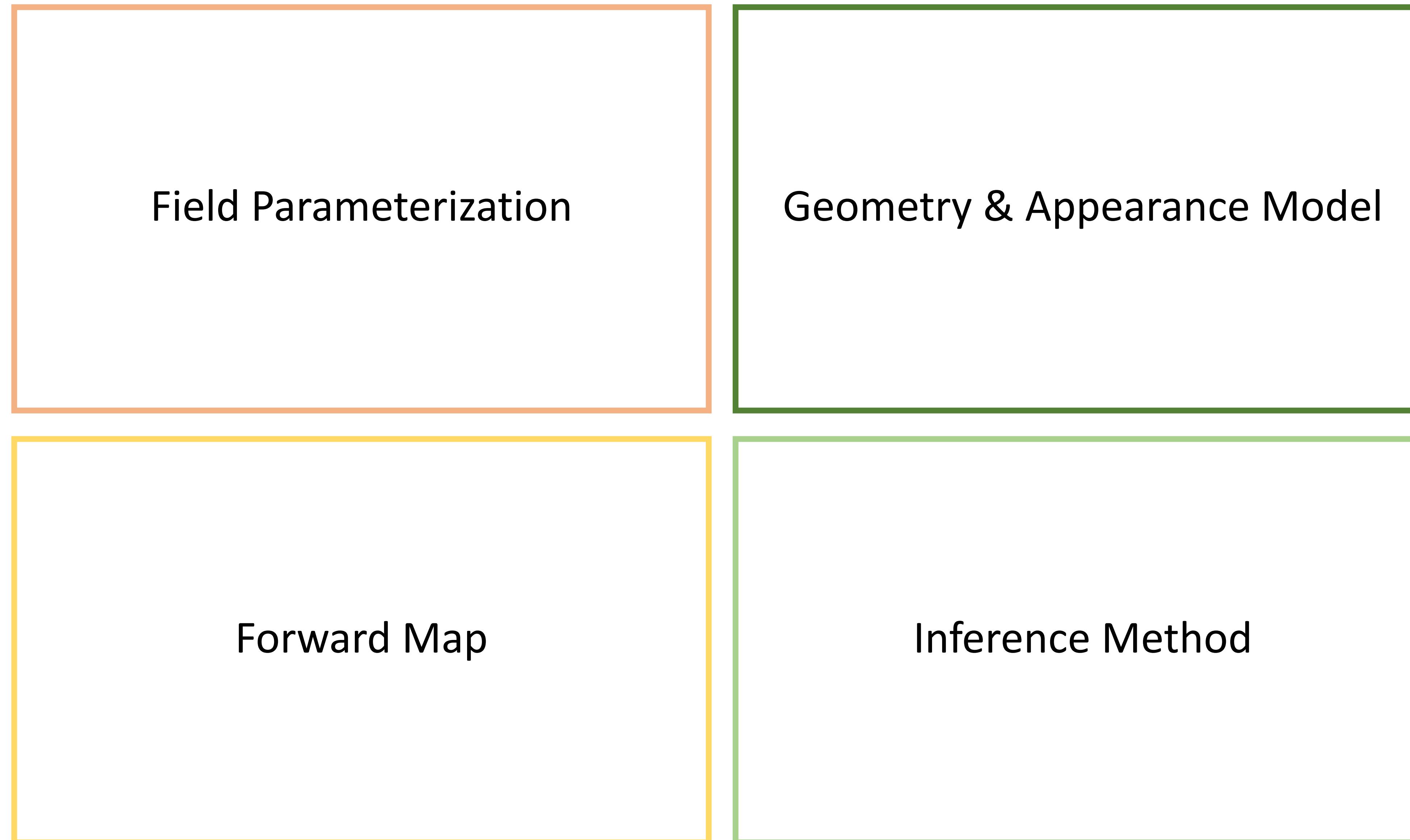
The explosion of neural implicit representations in inverse graphics



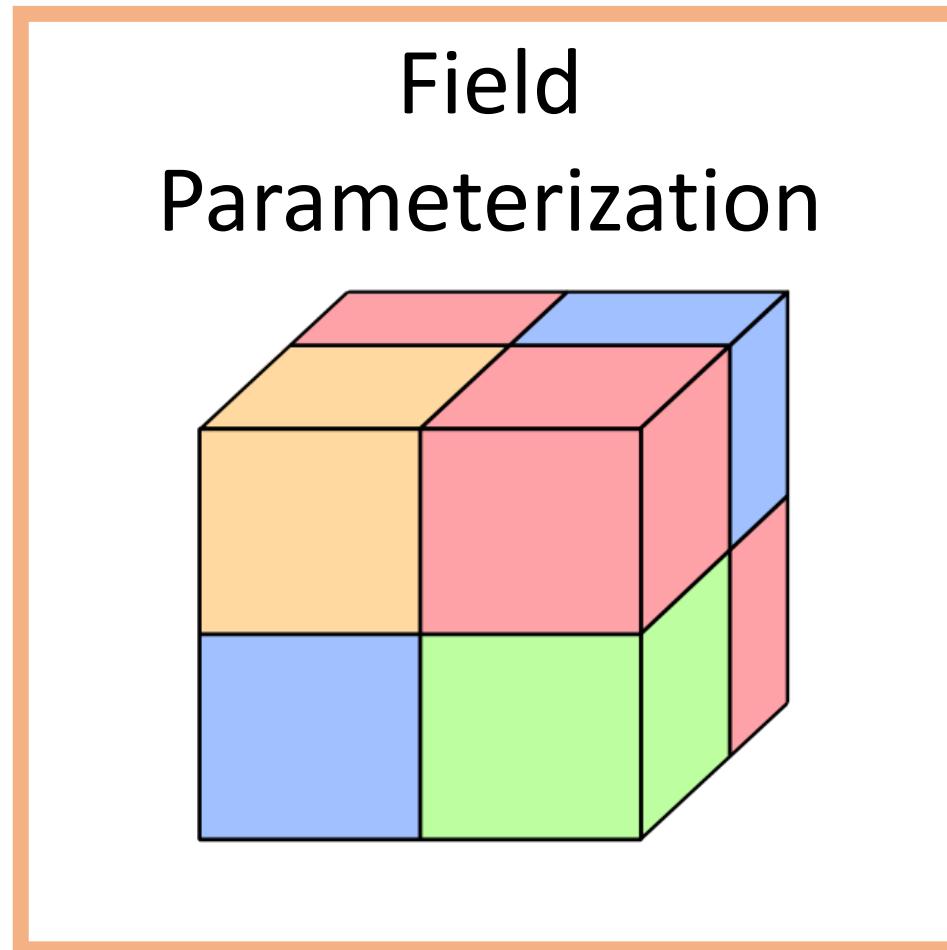
The explosion of neural implicit representations in inverse graphics



3D Inverse Graphics Taxonomy



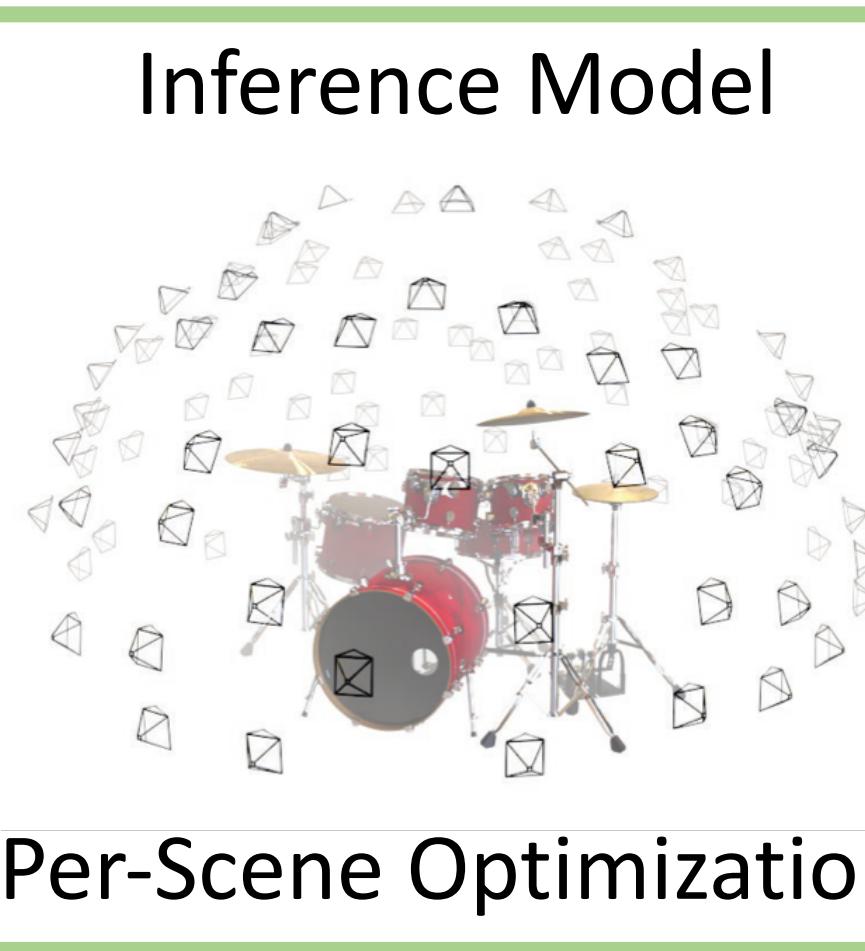
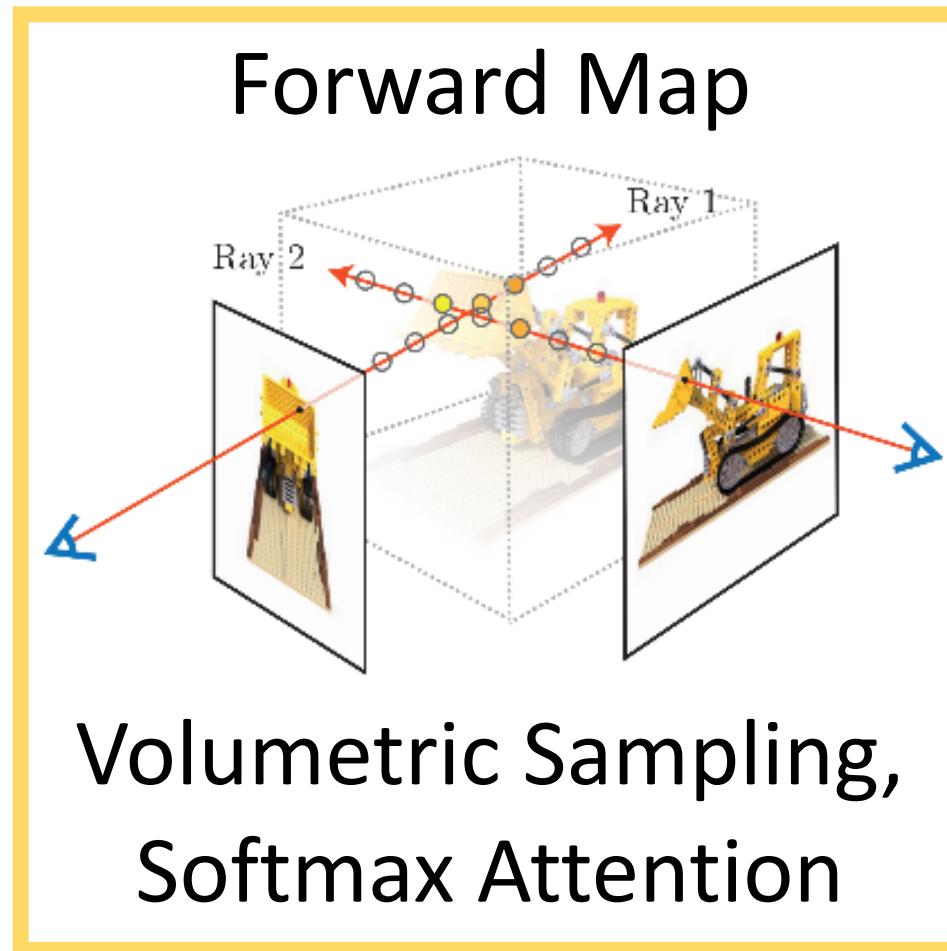
DeepVoxels (CVPR 2018)



Geometry & Appearance Model

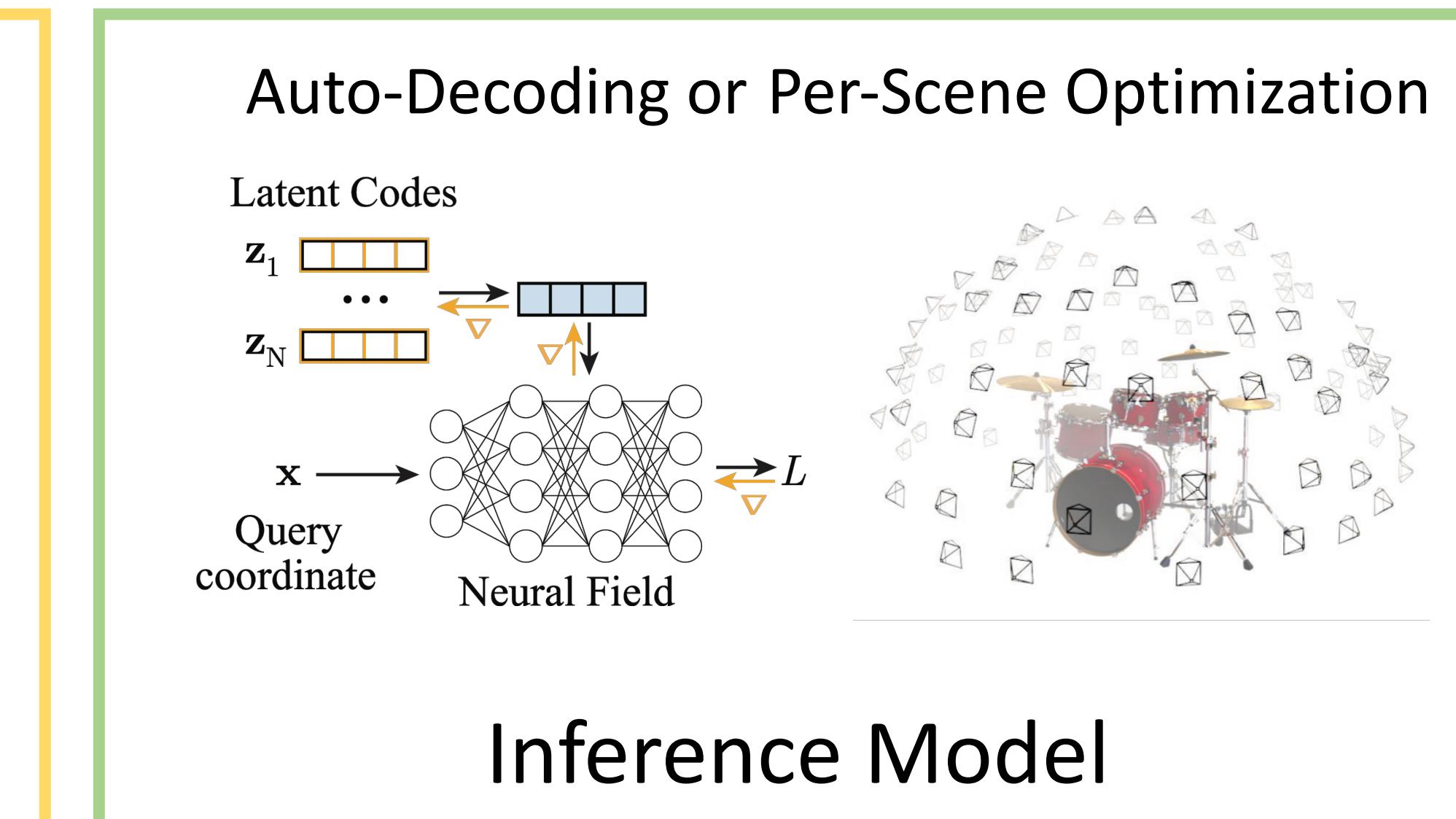
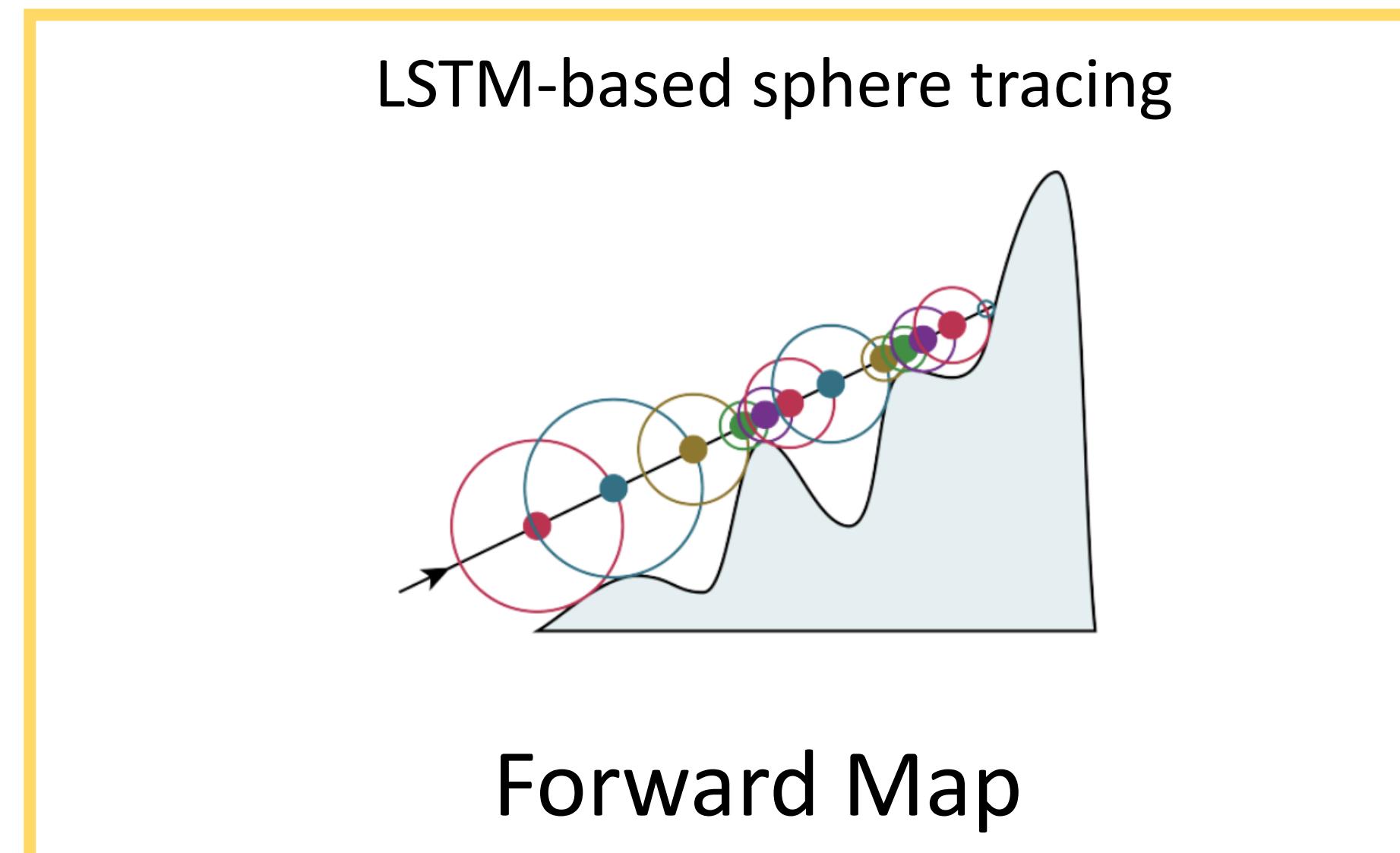
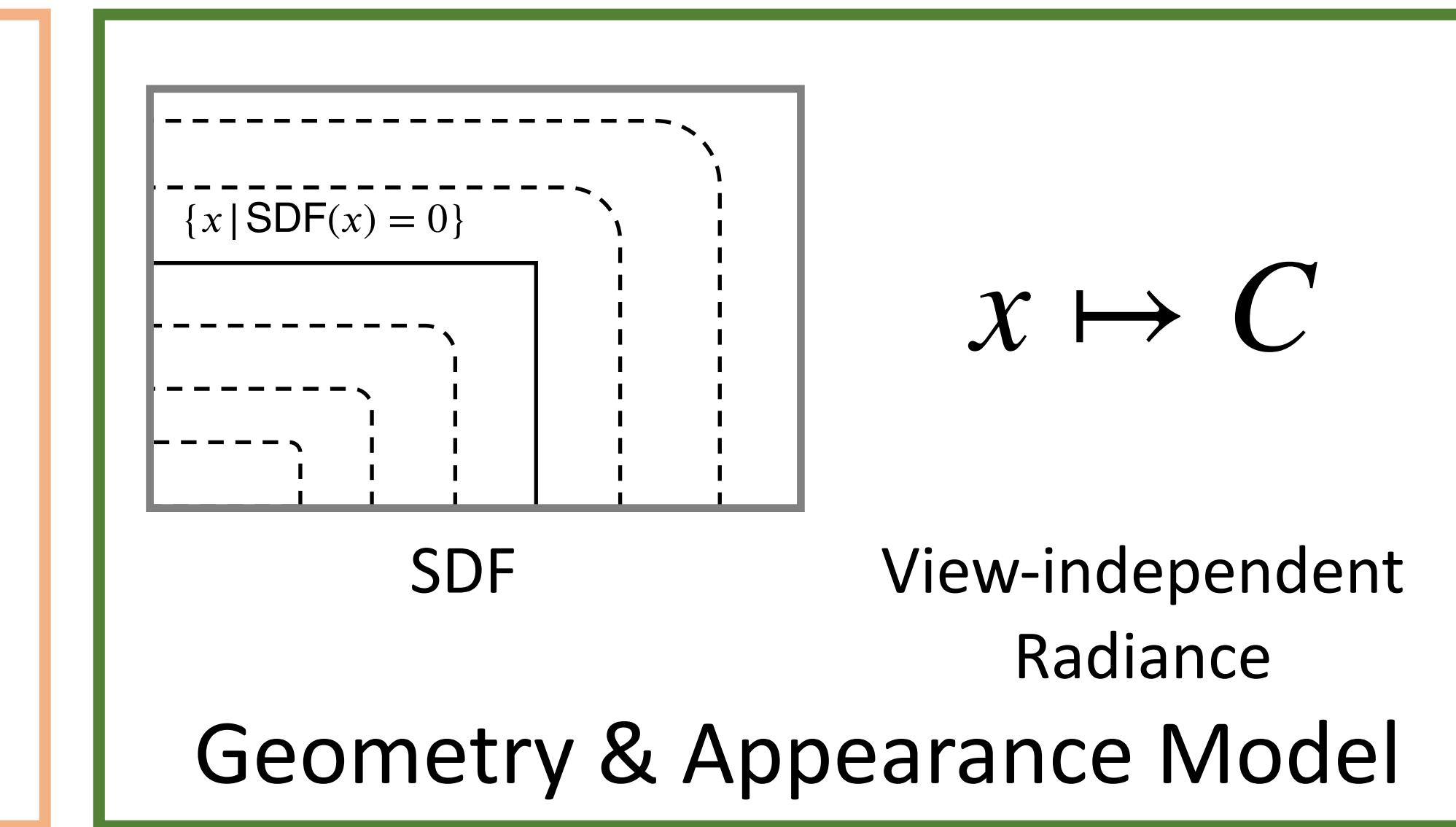
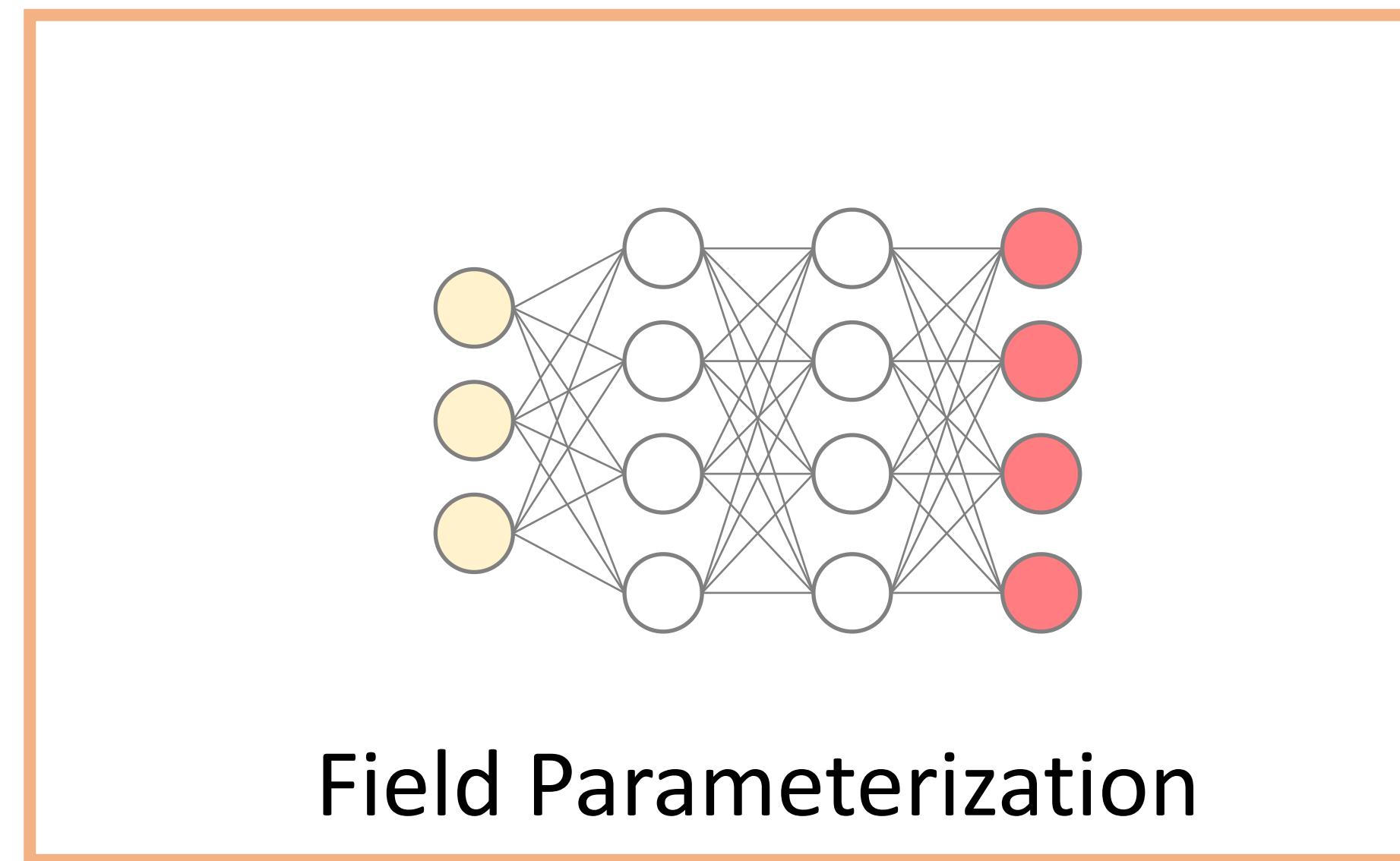
$$\Phi(\mathbf{x}) = \mathbf{f}$$
$$\mathbf{f} \in \mathbb{R}^n$$

Feature-based Representation

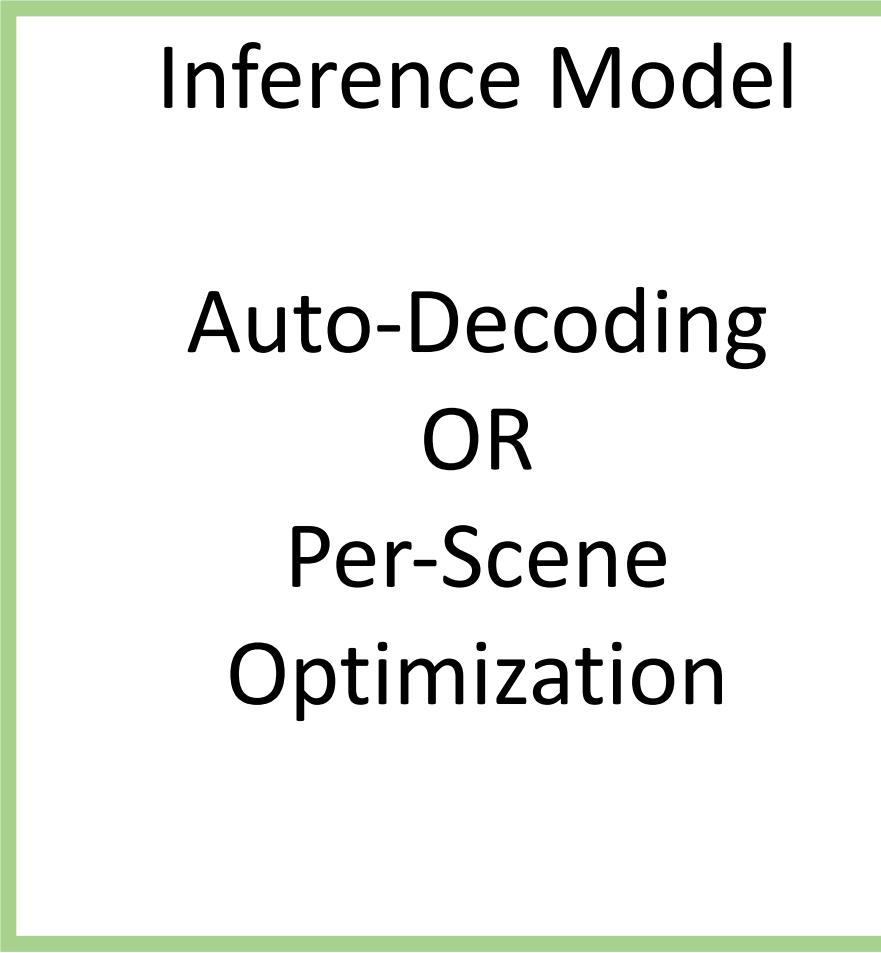
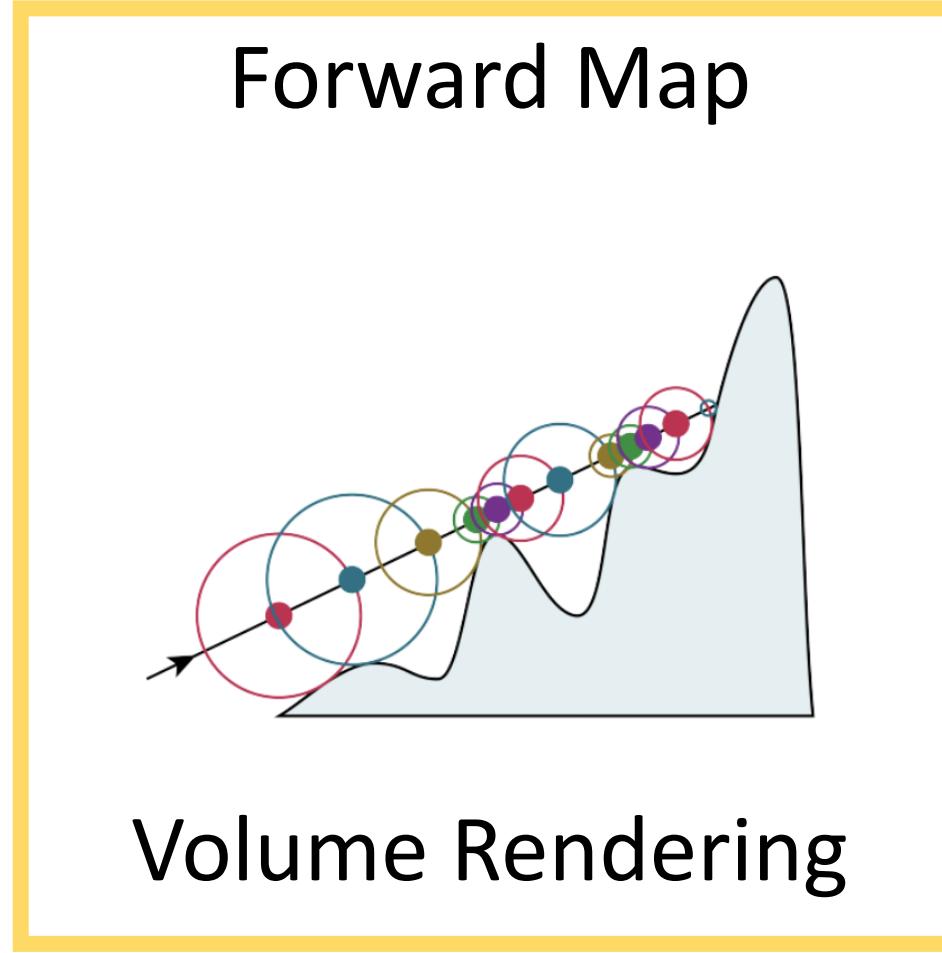
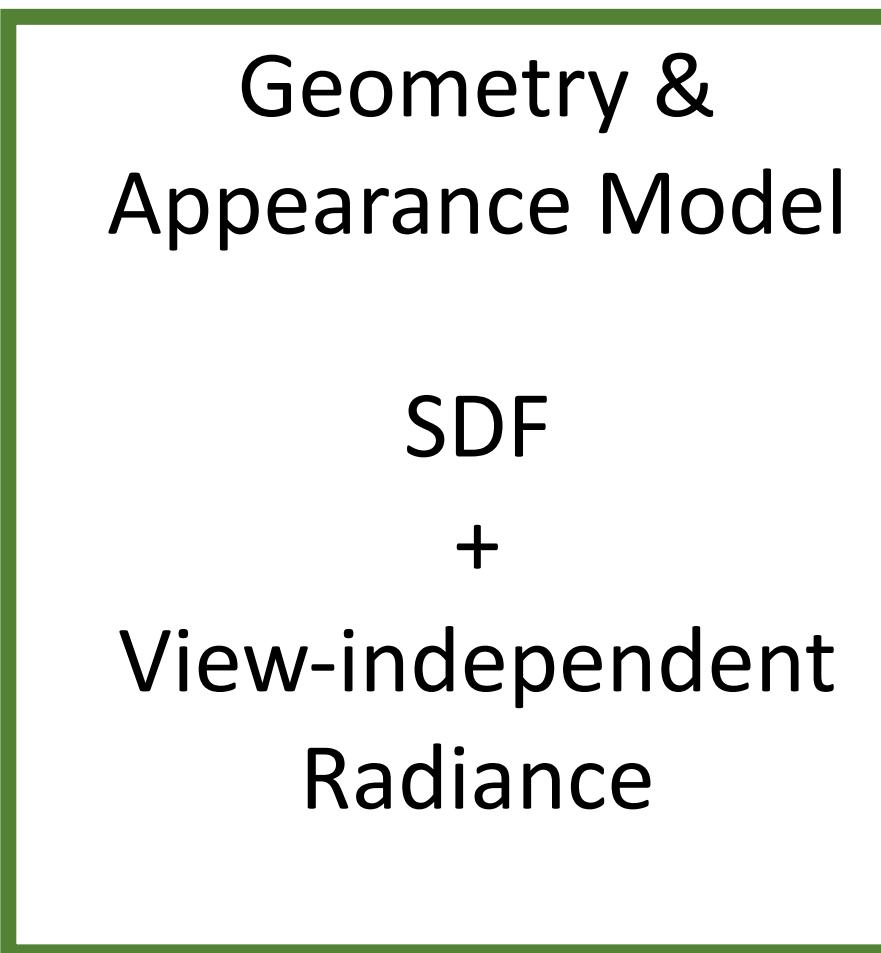
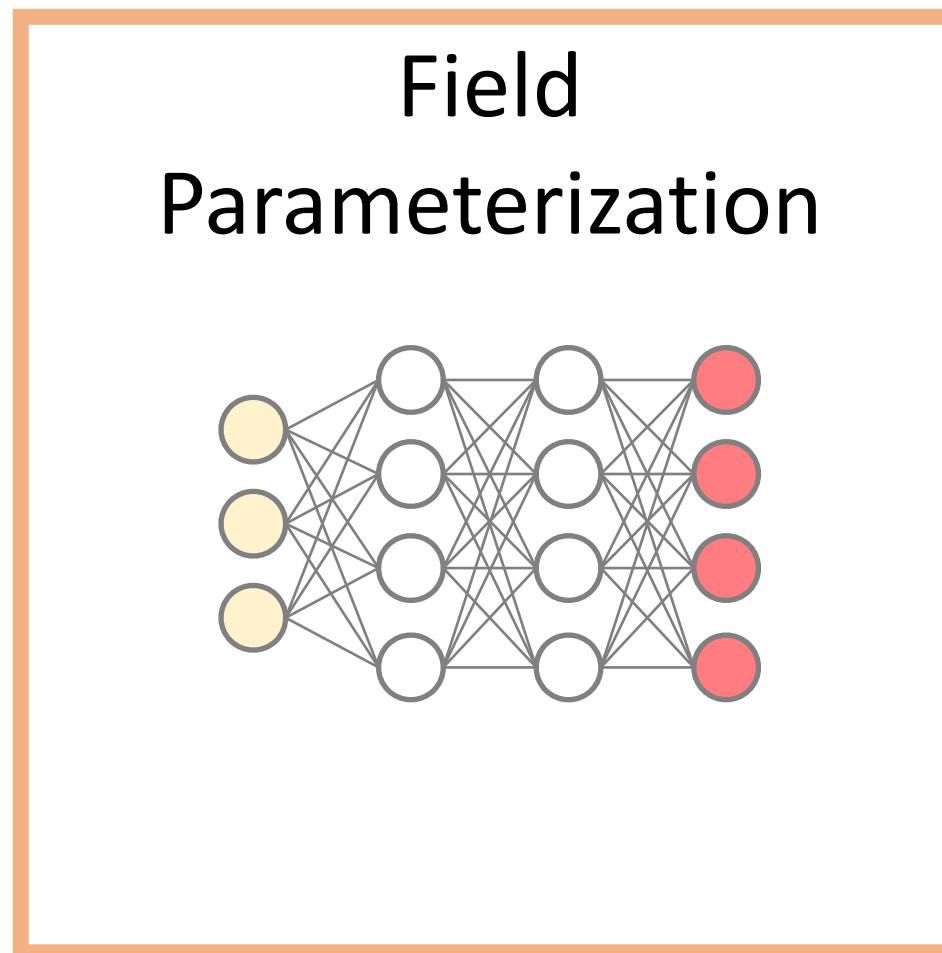


- Reasonable quality overfitting
- Fast rendering
- Noisy geometry
- Requires Many Images

Scene Representation Networks (NeurIPS 2019)



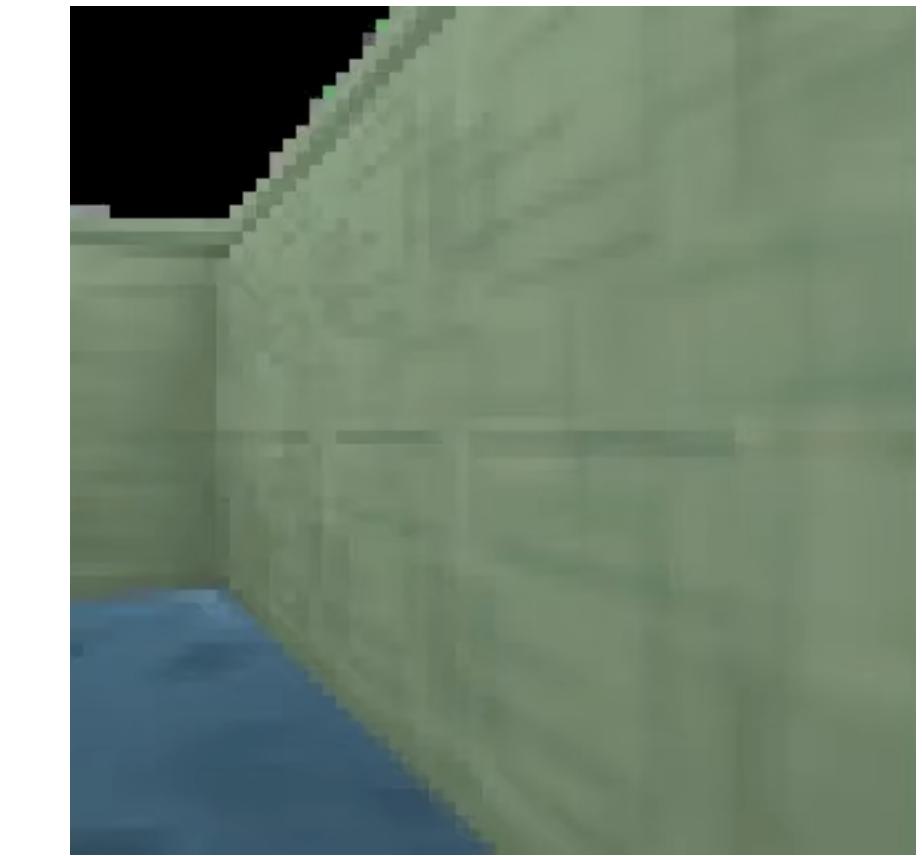
Scene Representation Networks (NeurIPS 2019)



Overfitting



Single Image



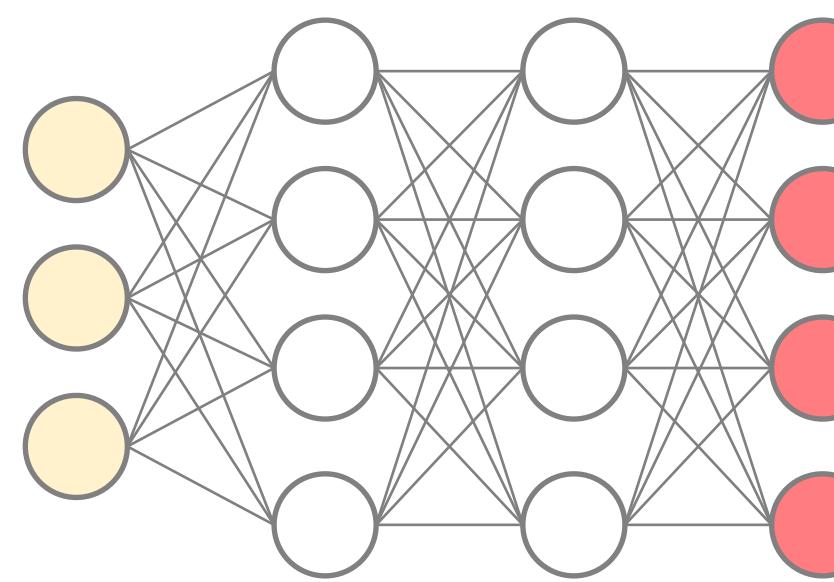
- When overfitting: Not photorealistic
- Non-compositional scenes:
Single-image reconstruction
- Well-defined geometry
- Low storage cost (weights)

Scene Representation Networks (NeurIPS 2019)



Scenes from NeRF (Mildenhall et al. 2021)

NeRF (Mildenhall et al. ECCV 2020)



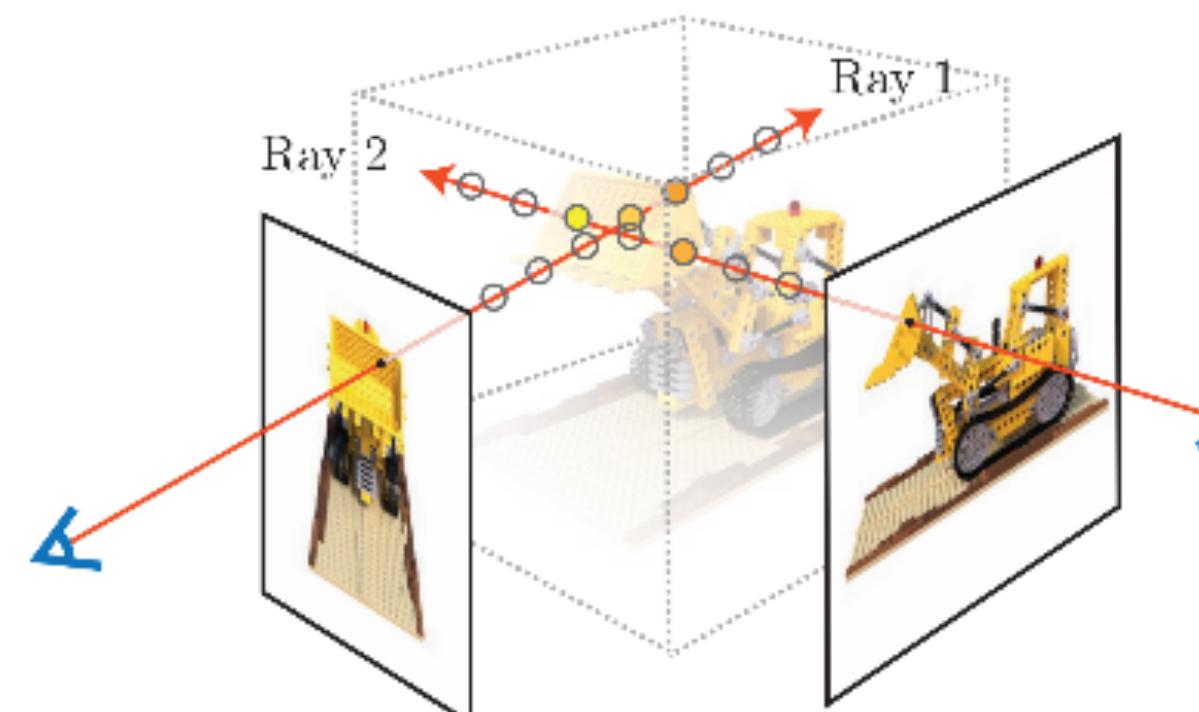
Field Parameterization

$$(x, d) \mapsto (\sigma, C)$$

View-Dependent Density + Radiance

Geometry & Appearance Model

Volume Rendering



Forward Map

Per-Scene Optimization

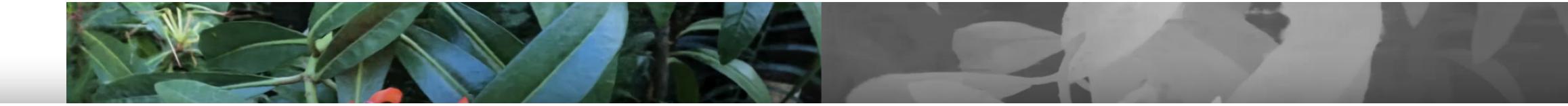


Inference Model

NeRF (Mildenhall et al. ECCV 2020)

Field

Geometry &



Great out-of-the-box performance
Great looking results for single-scene reconstruction
Ben donated slides ;)

ic!

**Inspired a boom in 3D reconstruction via differentiable
volumetric rendering!**



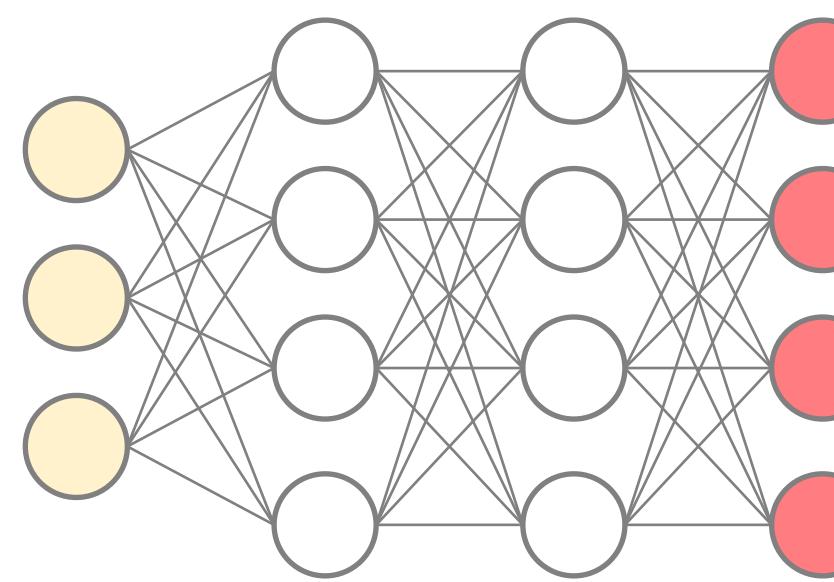
Volume Rendering



Per-Scene Optimization

- Noisy or Cloudy Geometry
- Requires Many Images

NeRF (Mildenhall et al. ECCV 2020)



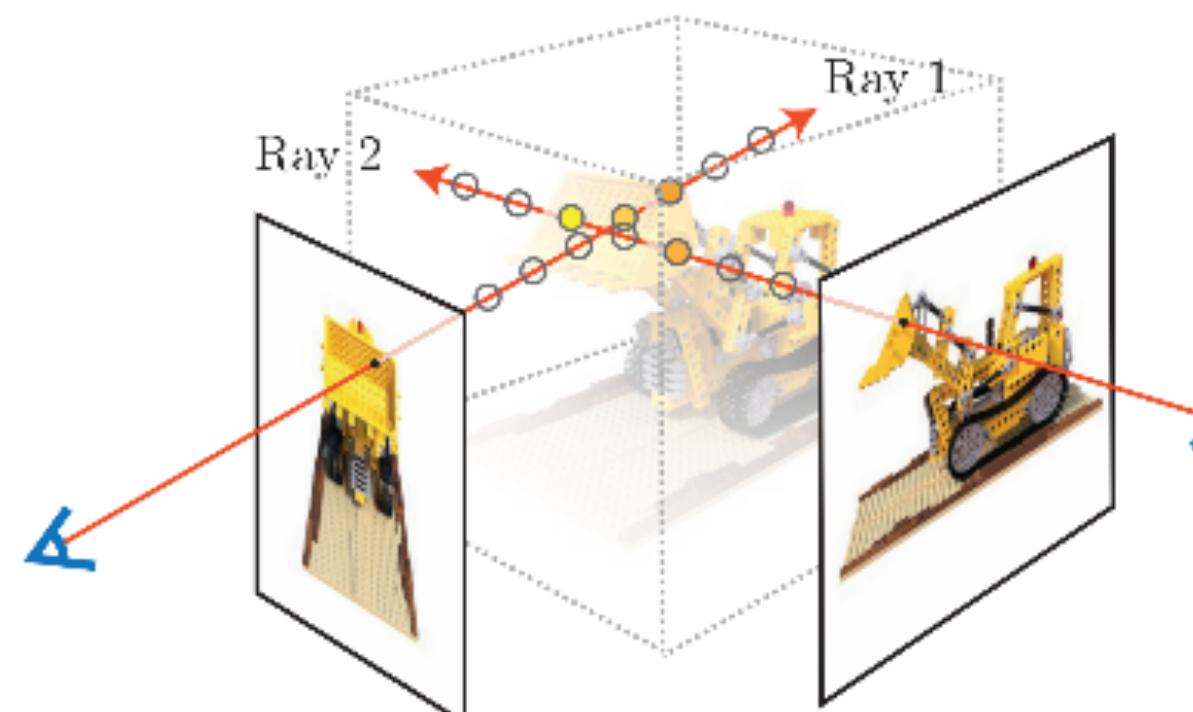
Field Parameterization

$$(x, d) \mapsto (\sigma, C)$$

View-Dependent Density + Radiance

Geometry & Appearance Model

Volume Rendering



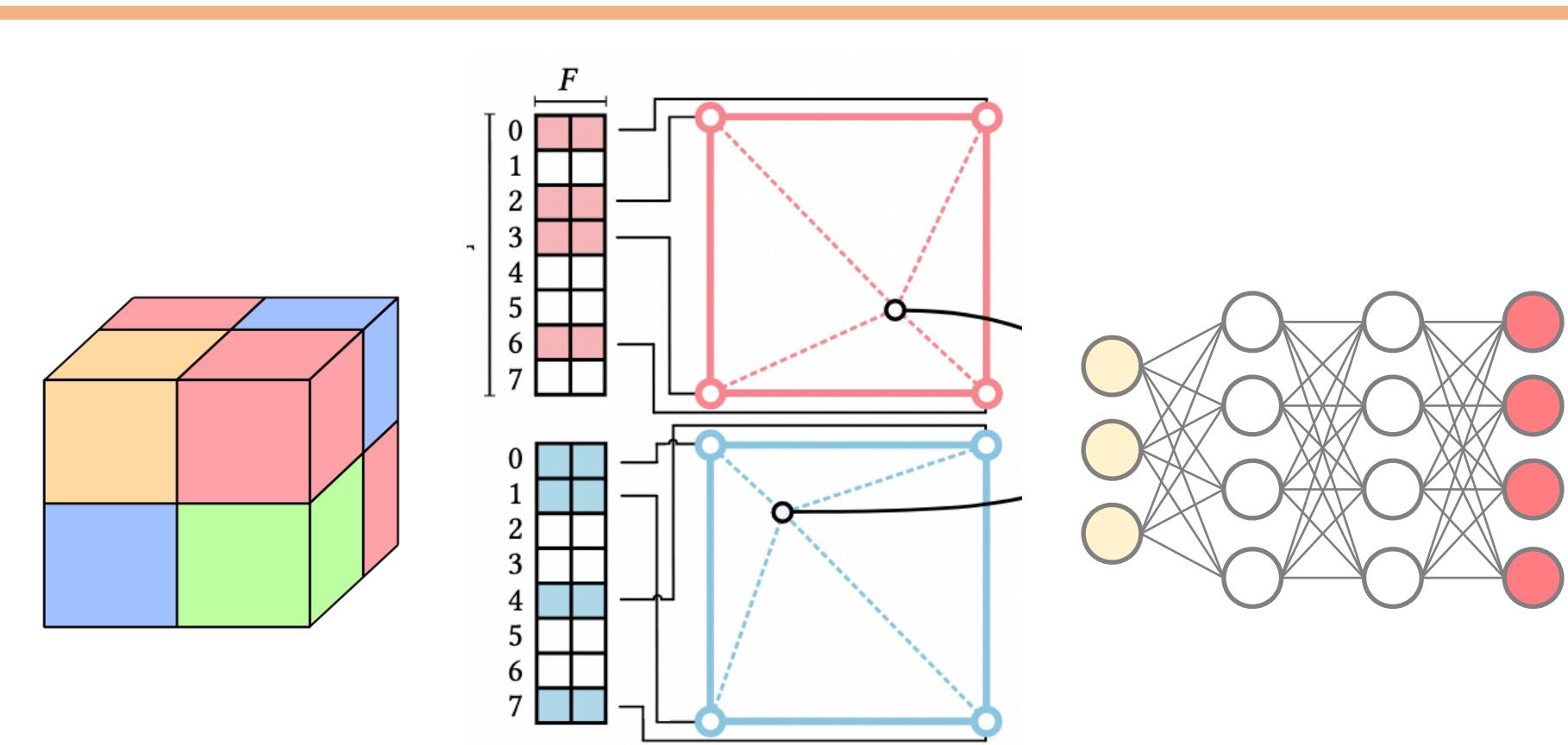
Forward Map

Per-Scene Optimization



Inference Model

Faster Radiance Fields w/ Hybrid Data Structures

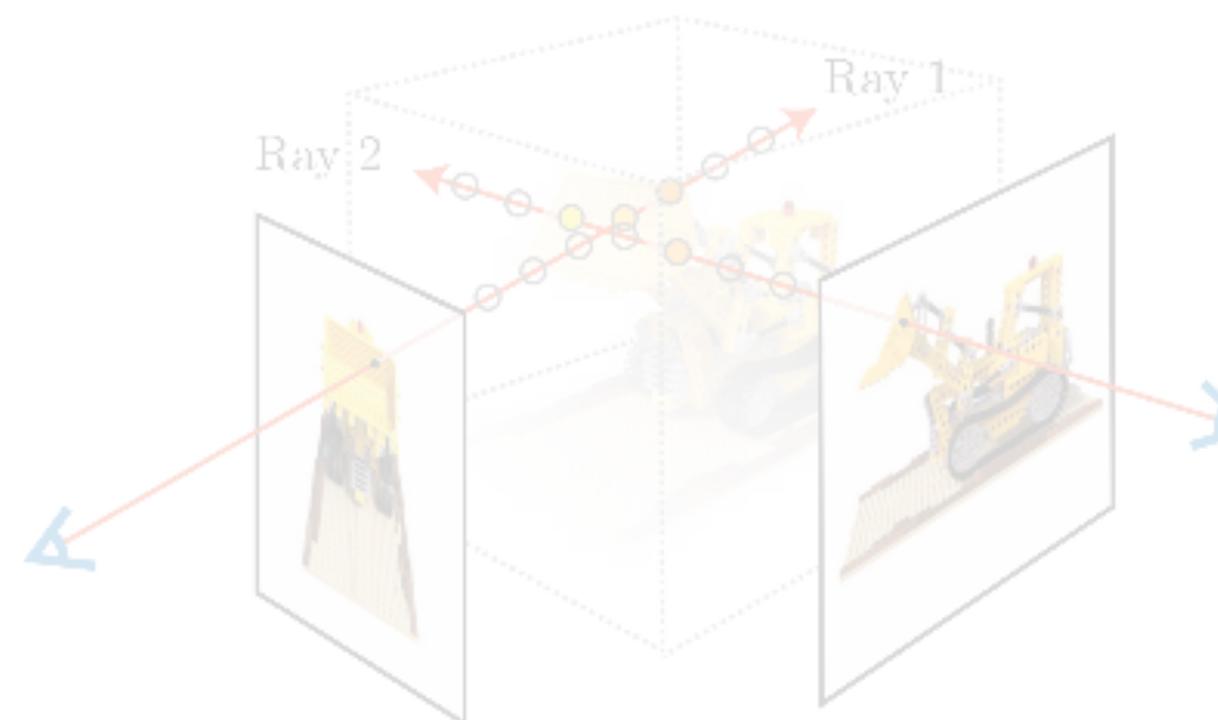


$$(x, d) \mapsto (\sigma, C)$$

View-Dependent Density + Radiance

Geometry & Appearance Model

Volume Rendering

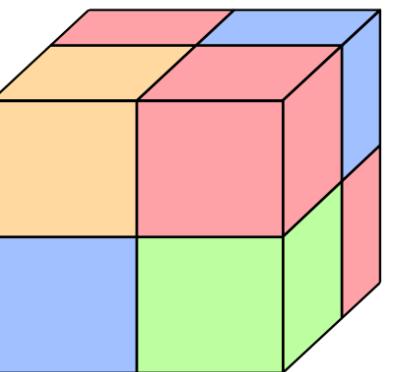


Forward Map

Per-Scene Optimization



Inference Model

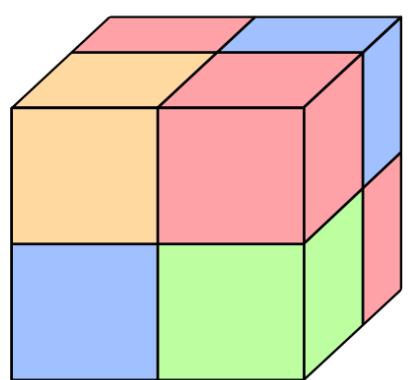


Plenoxels: Radiance Fields ...

[Yu et al. 2022]

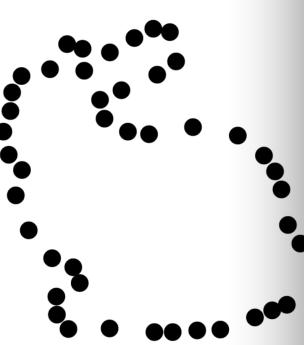
Direct Voxel Grid Optimization

[Sun et al. 2021]

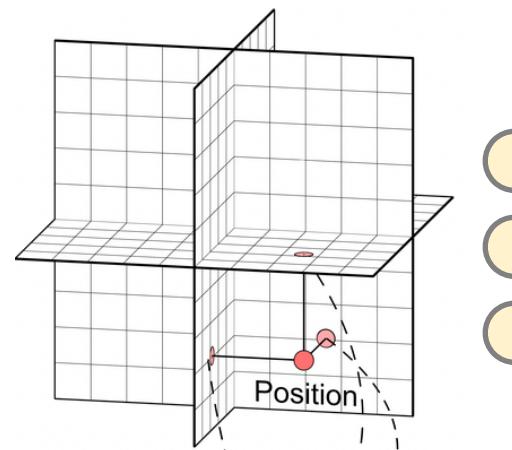
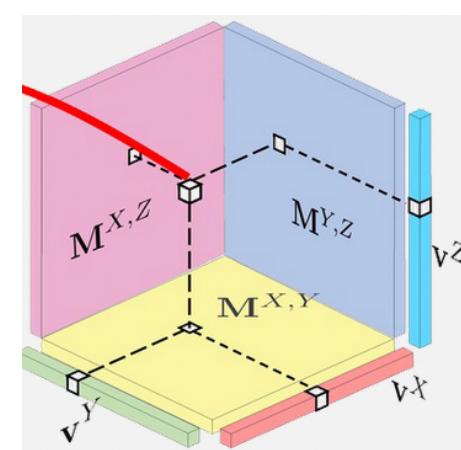


Voxel Grid

- Faster fitting
- Faster rendering
- Lower forward pass memory footprint
- May achieve higher quality than vanilla NeRF
- Voxels: Memory intense



Point Cloud + MLP



Tensor Factorization / Triplane + MLP

(b) Point-NeRF Representation with Volume



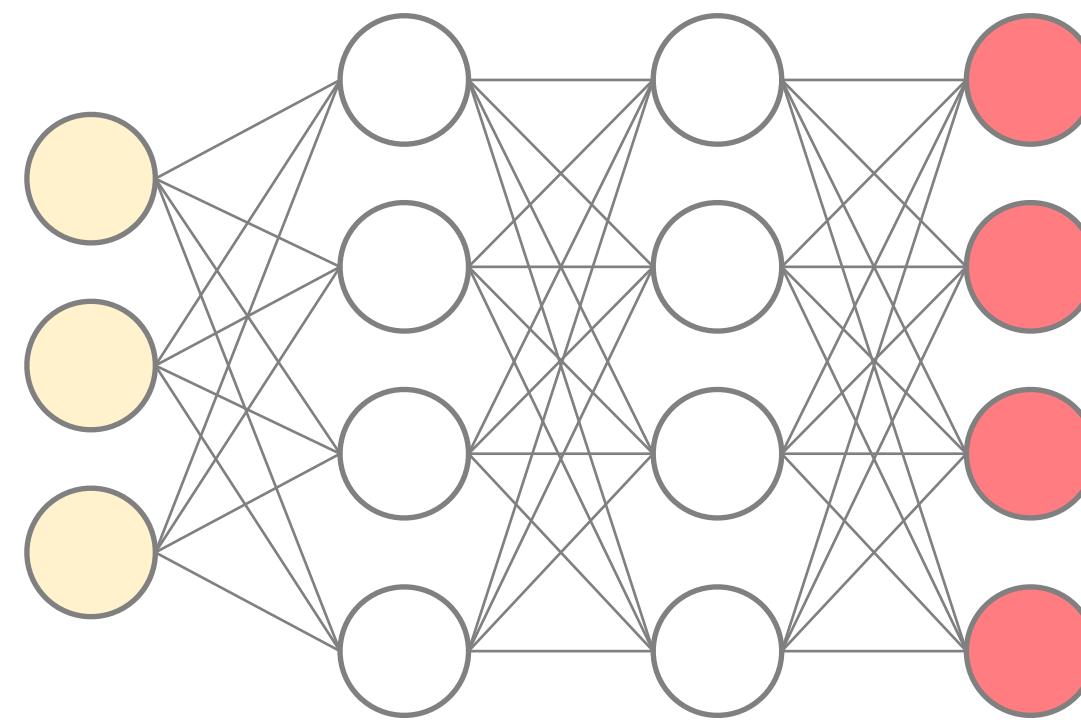
Efficient Geometry-aware 3D...

[Chan et al. 2022]

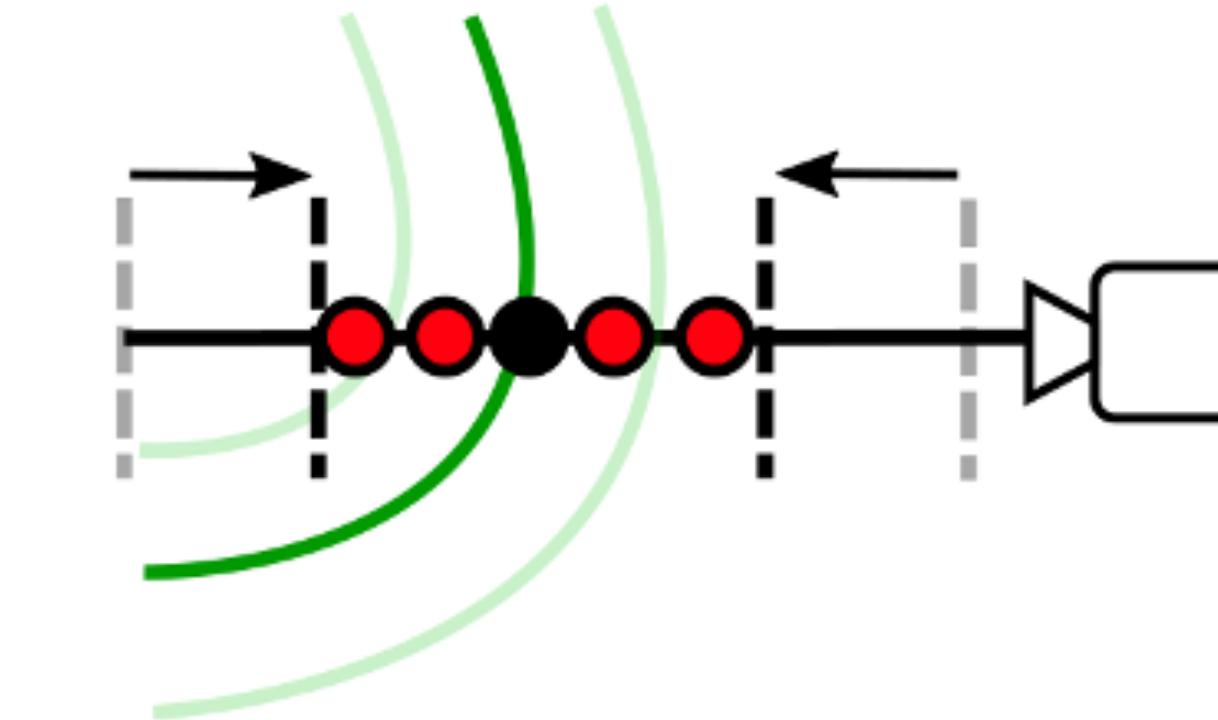
TensorRF: Tensor Radiance Fields

[Chen & Xu et al. 2022]

Hybrid Volumetric / Level Set Representations

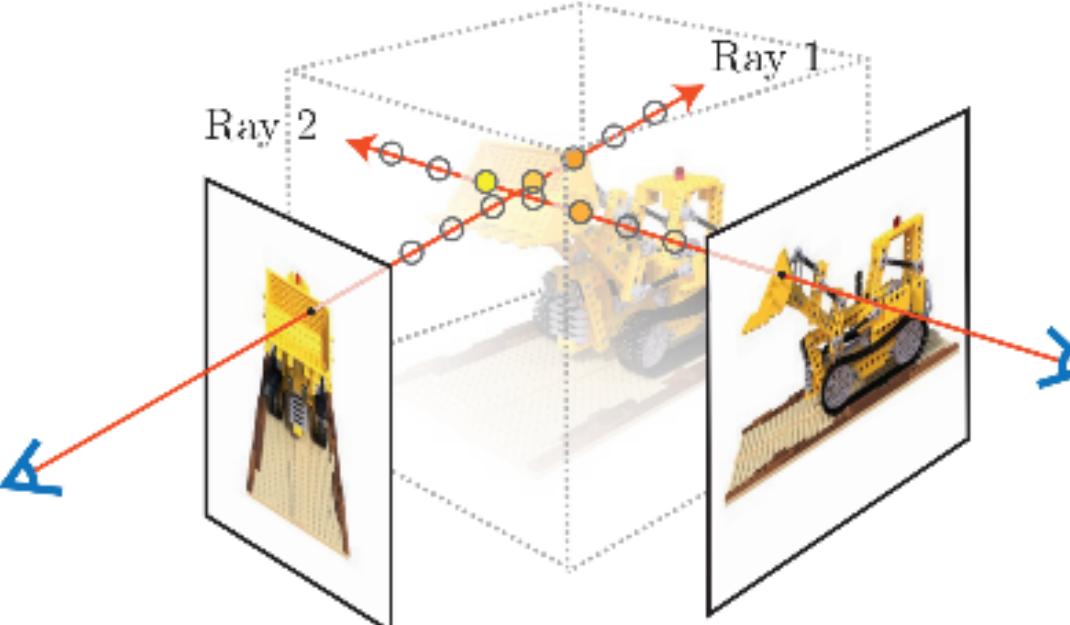


Field Parameterization



Geometry & Appearance Model

Volume Rendering



Forward Map

Per-Scene Optimization



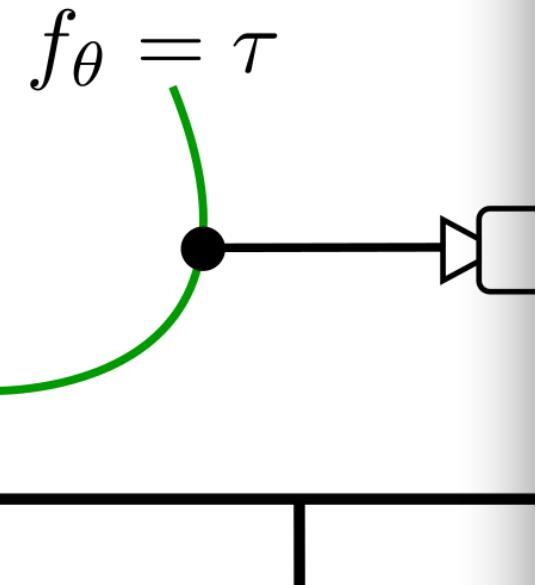
Inference Model

Hybrid Volumetric / Level Set Representations

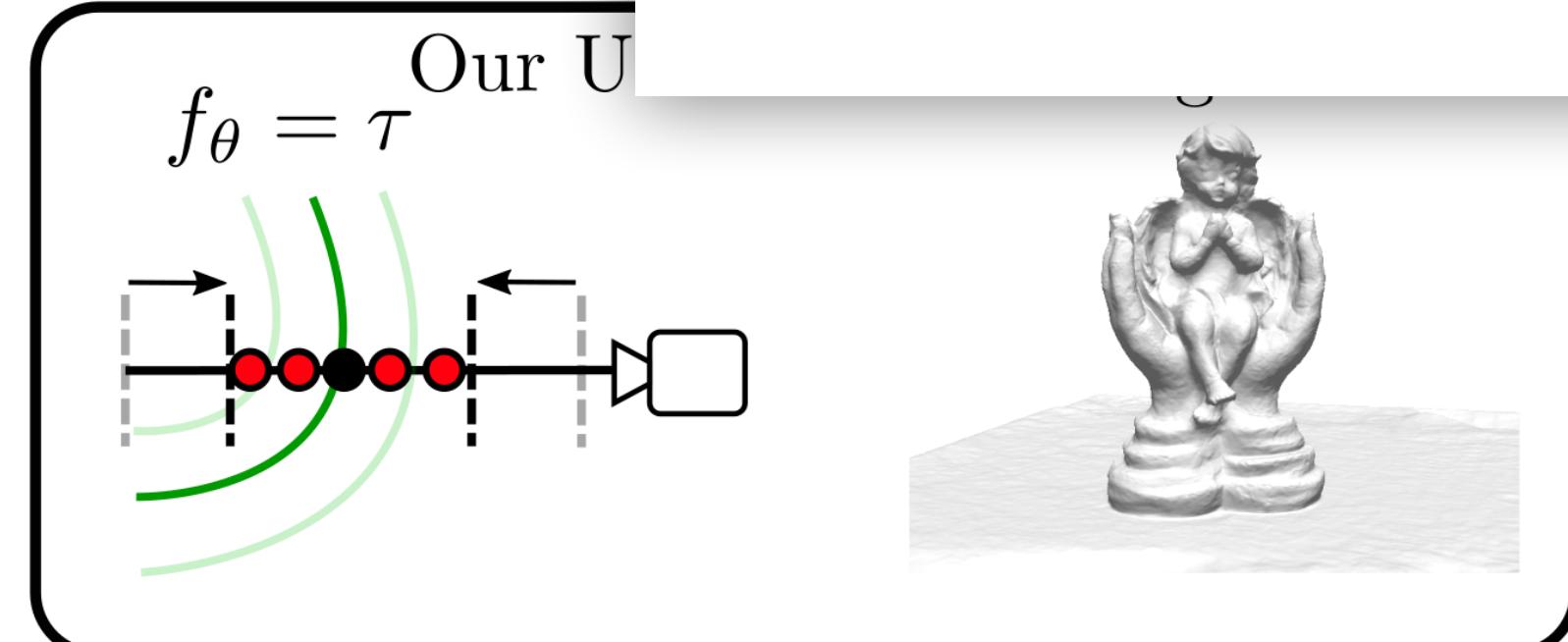
UNISURF

[Oechsle et al., ICCV 2021]

Surface Rendering



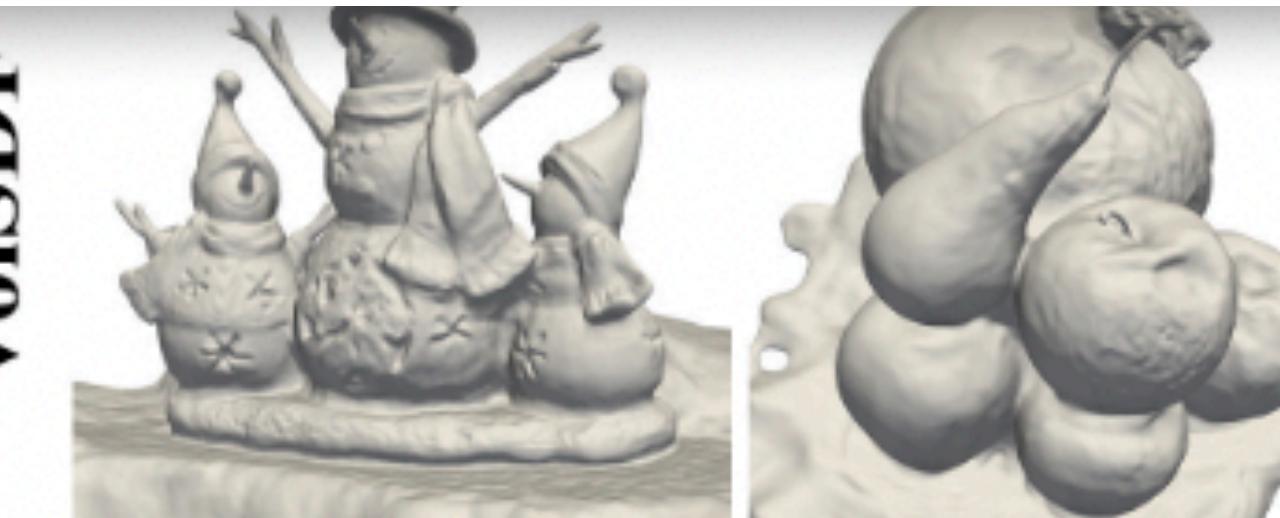
- Surface is well defined
- No noisy or cloudy geometry
- Good for solid objects
- Not good for complex scene phenomena (transparency)



VolSDF

[Yariv et al., NeurIPS 2021]

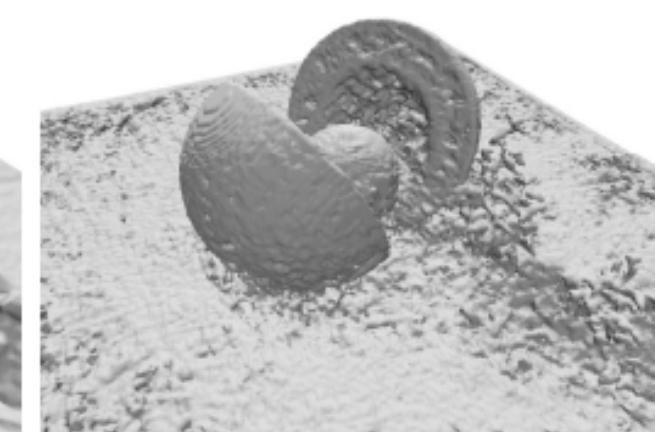
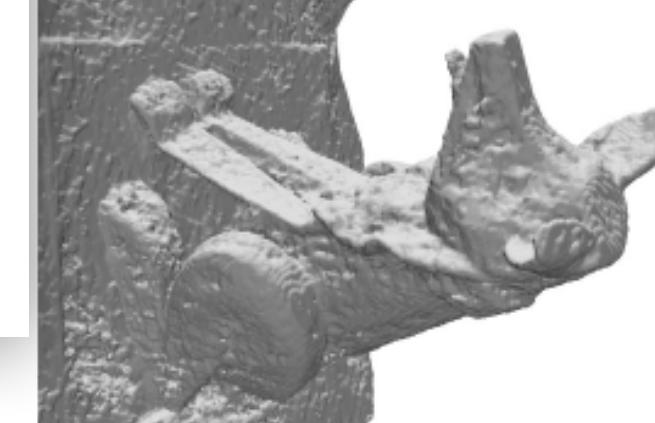
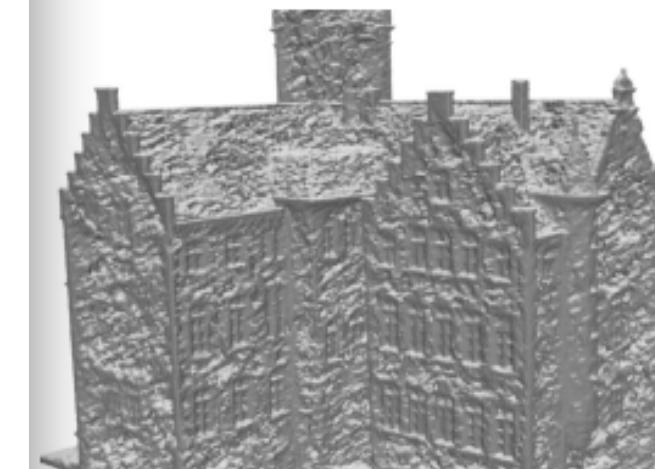
VolSDF



NeuS

[Wang et al., NeurIPS 2021]

NeRF



Neural Fields in Visual Computing and Beyond

Yiheng Xie^{1,2} Towaki Takikawa^{3,4} Shunsuke Saito⁵ Or Litany⁴ Shiqin Yan¹ Numair Khan¹ Federico Tombari^{6,7}
 James Tompkin¹ Vincent Sitzmann⁸ Srinath Sridhar^{1†}

¹Brown University ²Unity Technologies ³University of Toronto ⁴NVIDIA ⁵Meta Reality Labs Research ⁶Google ⁷Technical University of Munich
⁸Massachusetts Institute of Technology [†]Equal advising

<https://neuralfields.cs.brown.edu/>

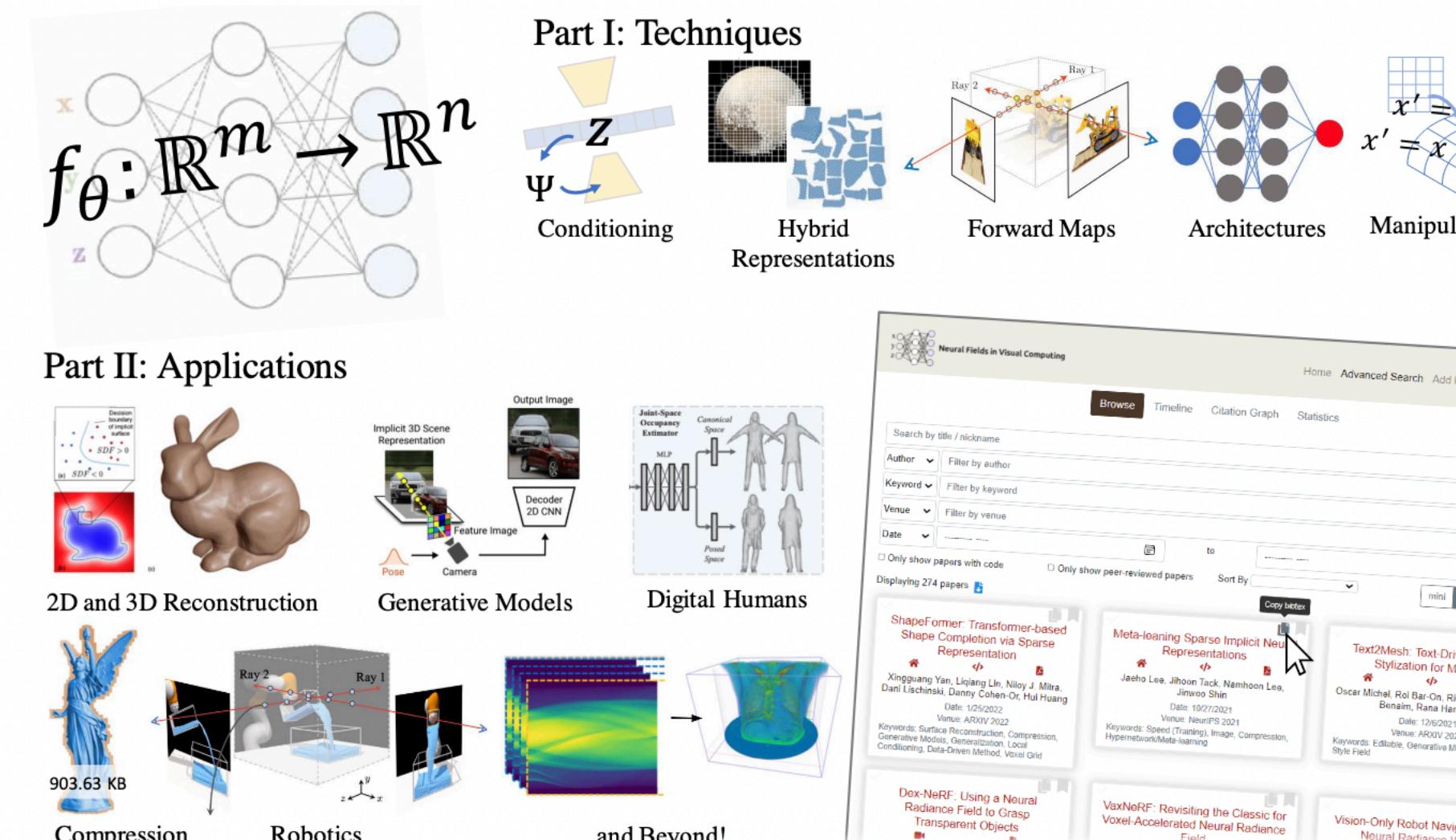


Figure 1: **Contribution of this report.** Following a survey of over 250 papers, we provide a review of (**Part I**) techniques in neural fields such as prior learning and conditioning, representations, forward maps, architectures, and manipulation, and of (**Part II**) applications in visual computing including 2D image processing, 3D scene reconstruction, generative modeling, digital humans, compression, robotics, and beyond. This report is complemented by a [community-driven website](https://neuralfields.cs.brown.edu/) with search, filtering, bibliographic, and visualization features.

How far does this get us?

$$\sigma = \sigma(\mathbf{x})$$

$$\mathbf{c} = \mathbf{c}(\mathbf{x})$$

$$L_o(\mathbf{x}, \omega_o, \lambda) = L_e(\mathbf{x}, \cancel{\omega}_o, \lambda) + \int_{\Omega} f(\mathbf{x}, \omega_i, \omega_o, \lambda) L_i(\mathbf{x}, \omega_i, \lambda) (\omega_i \cdot \mathbf{n}) d\omega_i$$

$$L_o(\mathbf{x}, \omega_o, \lambda) = L_e(\mathbf{x}, \lambda)$$

No “true” reflections, no diffuse inter-reflections, no re-lighting, no materials, no...

How far does this get us?

$$\begin{aligned}\sigma &= \sigma(\mathbf{x}) \\ \mathbf{c} &= \mathbf{c}(\mathbf{x}, \omega_o)\end{aligned}$$

$$L_o(\mathbf{x}, \omega_o, \lambda) = L_e(\mathbf{x}, \omega_o, \lambda) + \int_{\Omega} f(\mathbf{x}, \omega_i, \omega_o, \lambda) I_i(\mathbf{x}, \omega_i, \lambda) (\omega_i \cdot \mathbf{n}) d\omega_i$$

$$L_o(\mathbf{x}, \omega_o, \lambda) = L_e(\mathbf{x}, \omega_o, \lambda)$$

Half-way fix: allow direction-dependent color, but keep geometry direction-independent.

How far does this get us?

$$\begin{aligned}\sigma &= \sigma(\mathbf{x}) \\ \mathbf{c} &= \mathbf{c}(\mathbf{x}, \omega_o)\end{aligned}$$

$$L_o(\mathbf{x}, \omega_o, \lambda) = L_e(\mathbf{x}, \omega_o, \lambda) + \int_{\Omega} f(\mathbf{x}, \omega_i, \omega_o, \lambda) I_i(\mathbf{x}, \omega_i, \lambda) (\omega_i \cdot \mathbf{n}) d\omega_i$$

$$L_o(\mathbf{x}, \omega_o, \lambda) = L_e(\mathbf{x}, \omega_o, \lambda)$$

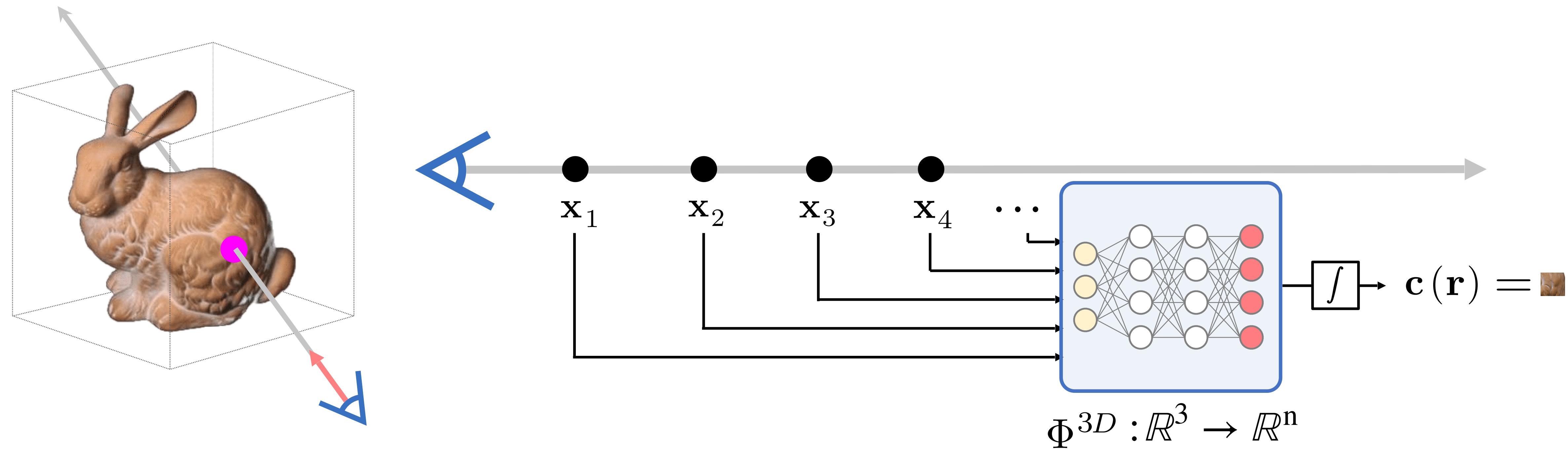
Can now approximate everything, but materials etc. are not enforced.

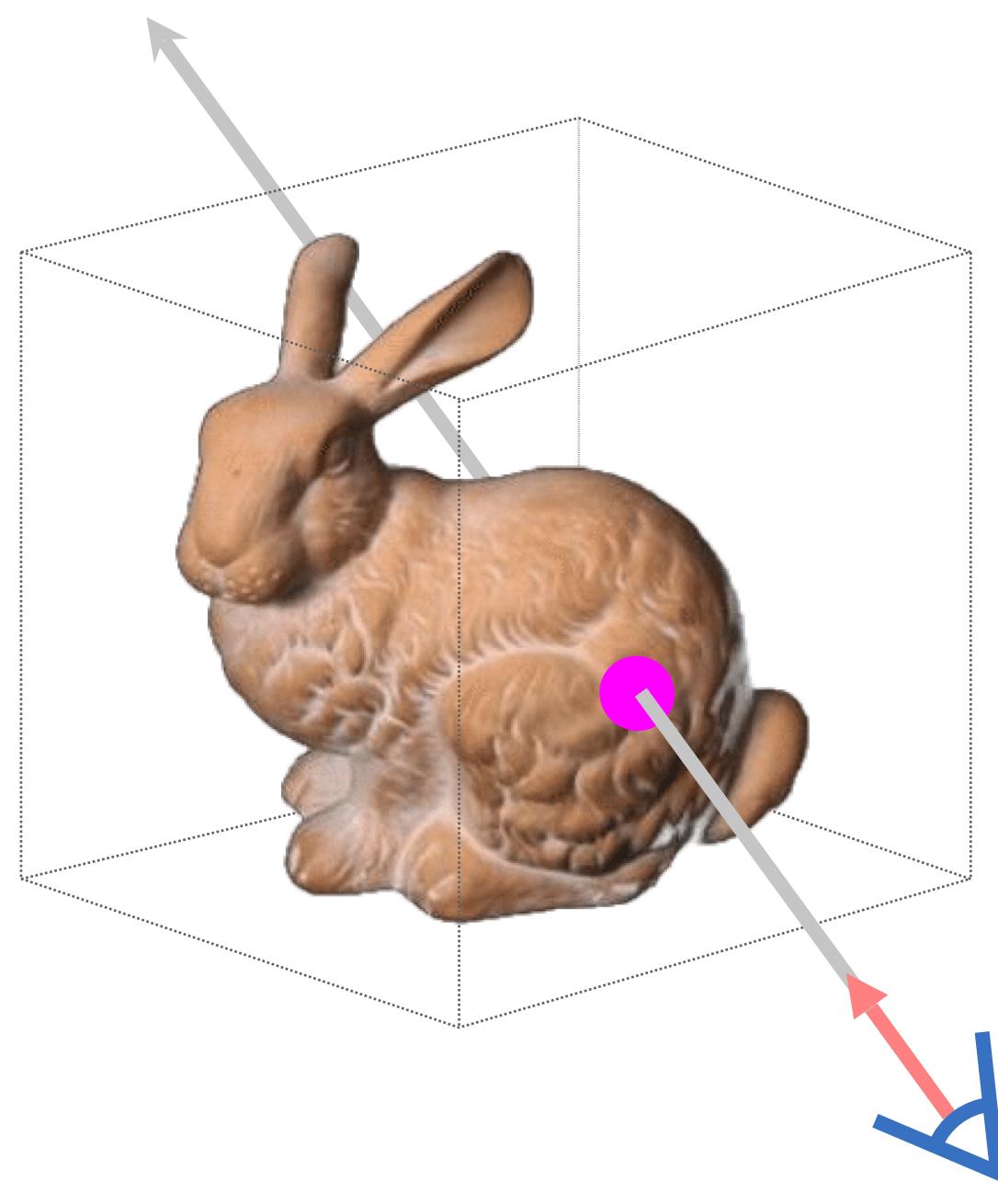
Questions?

Direction-dependent color



General structure of Neural Renderers for 3D-structured Representations





A

x_1

x_2

x_3

x_4

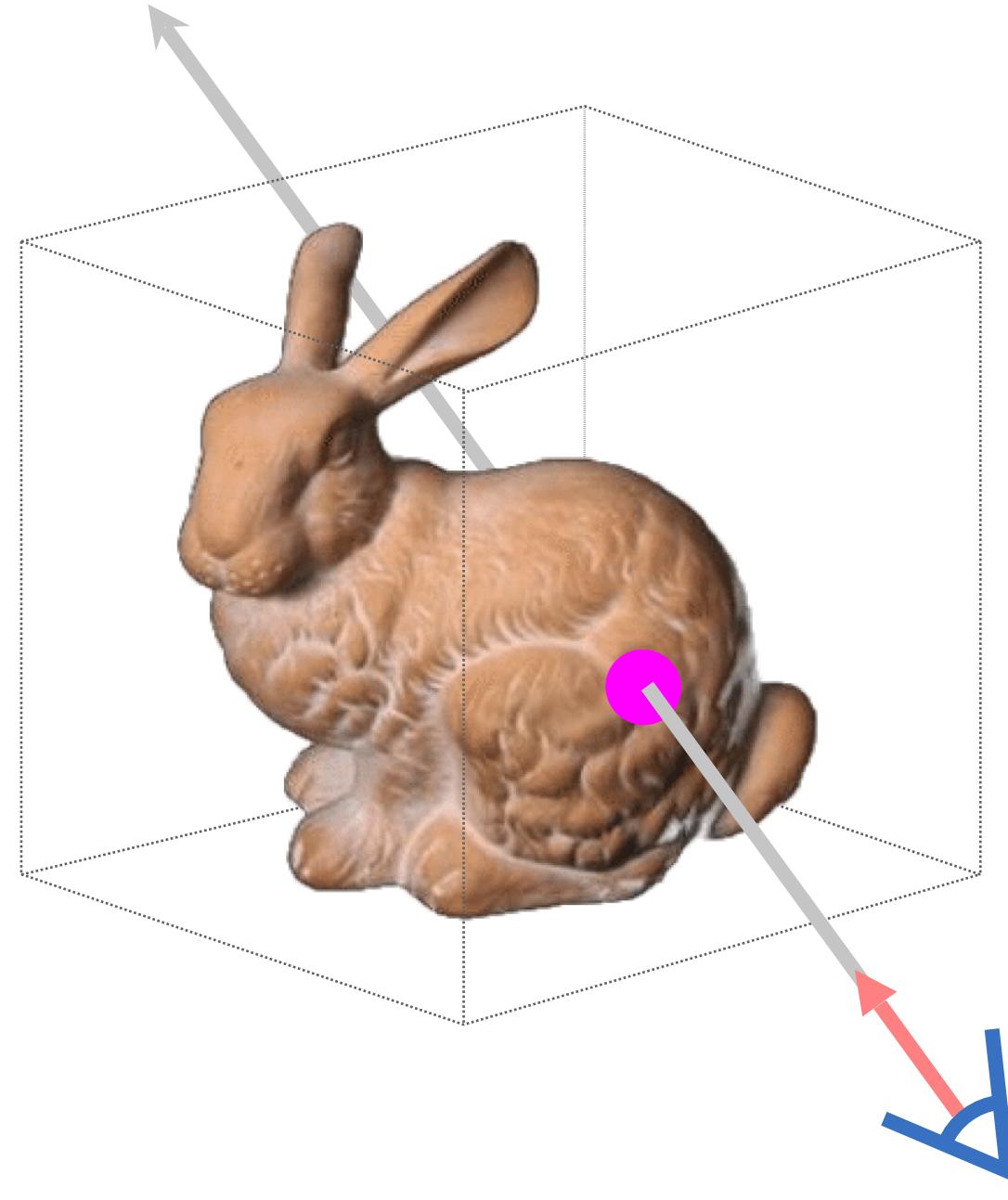
...

Light Field

$$\Phi^{3D} : \mathbb{R}^3 \rightarrow \mathbb{R}^n$$



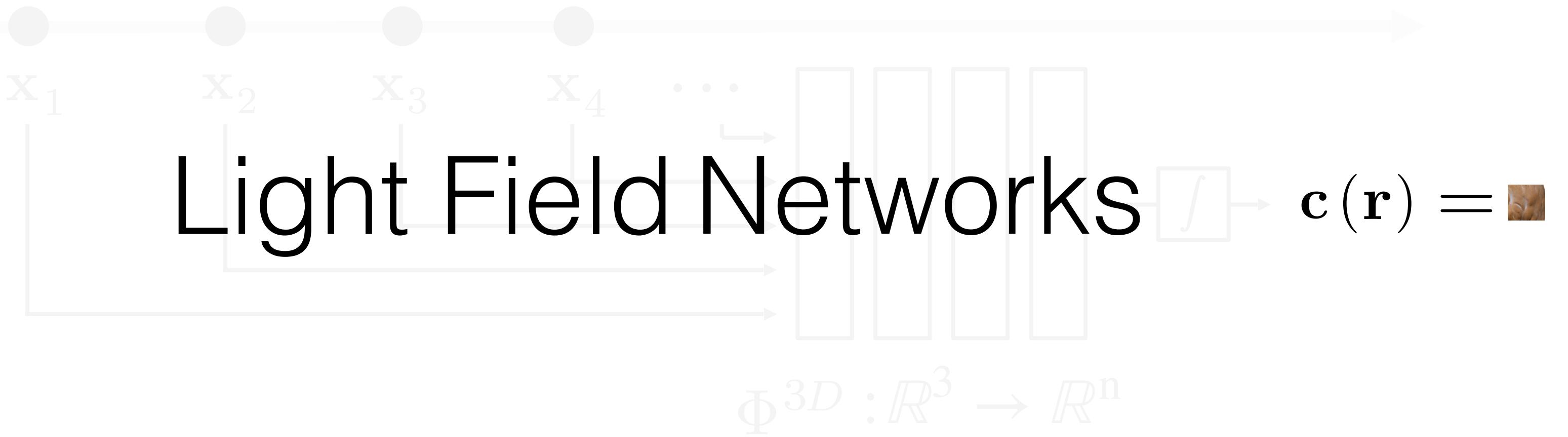
$$c(\mathbf{r}) =$$



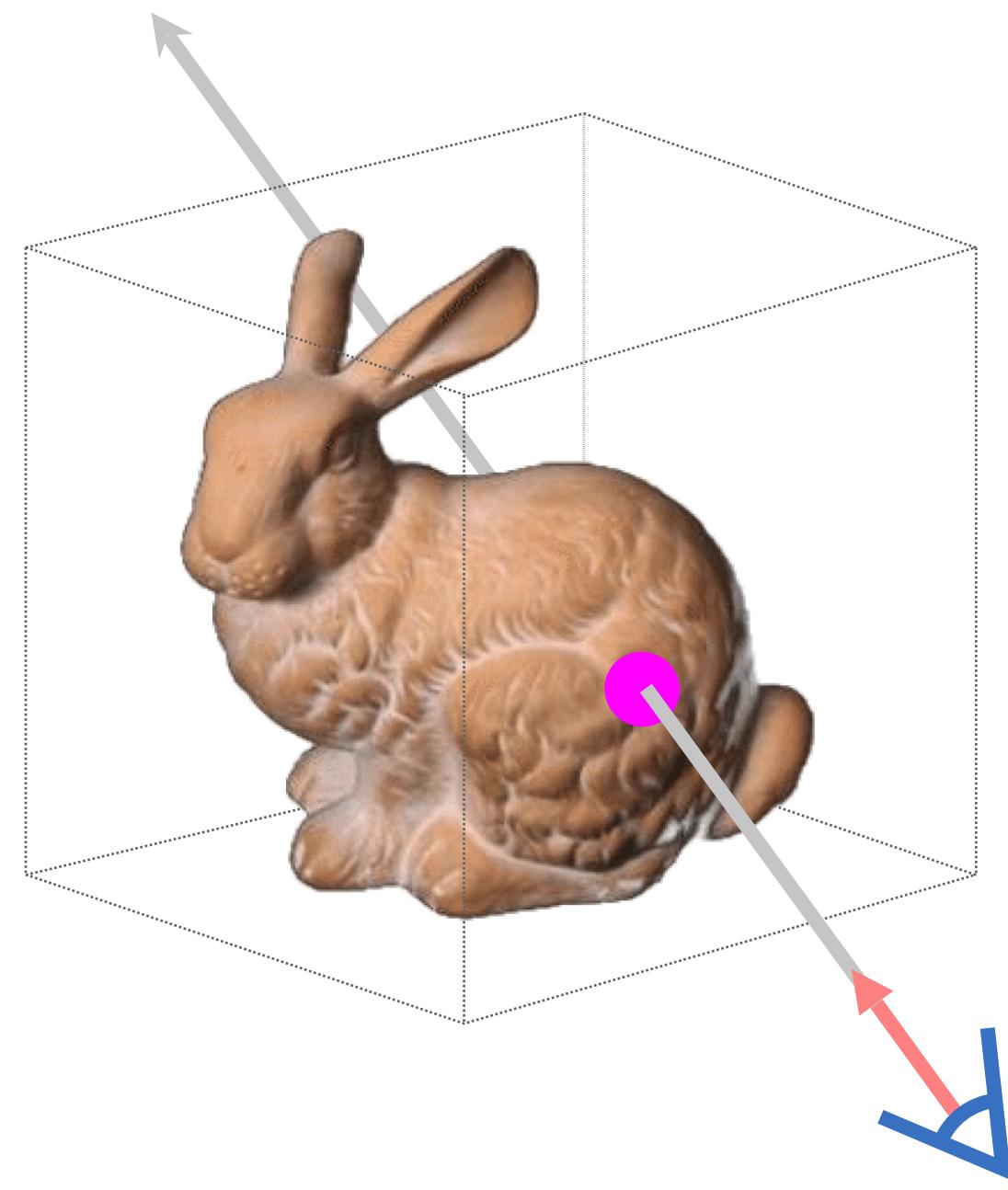
A

x_1

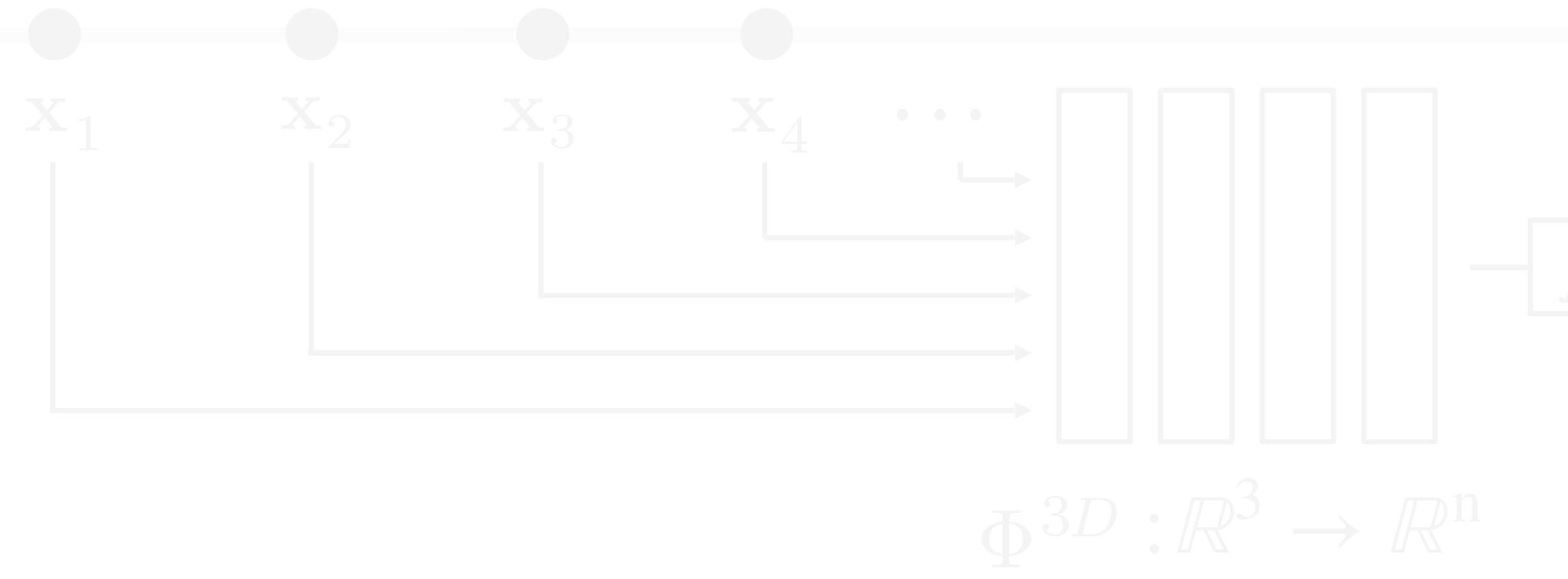
Light Field Networks



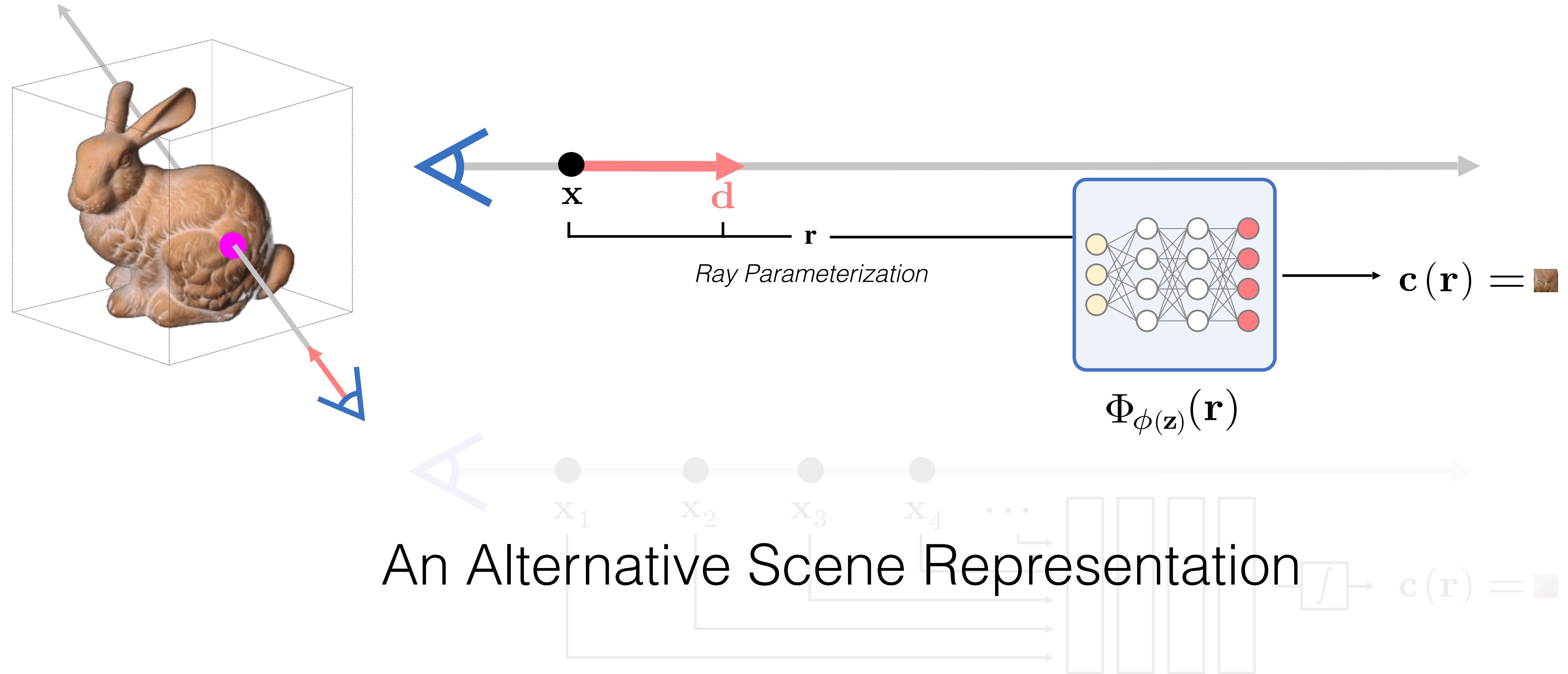
Light Field Networks



A



Light Field Networks



Overfitting doesn't work out-of-the-box: No built-in multi-view consistency!



Context Views



Intermediate Views

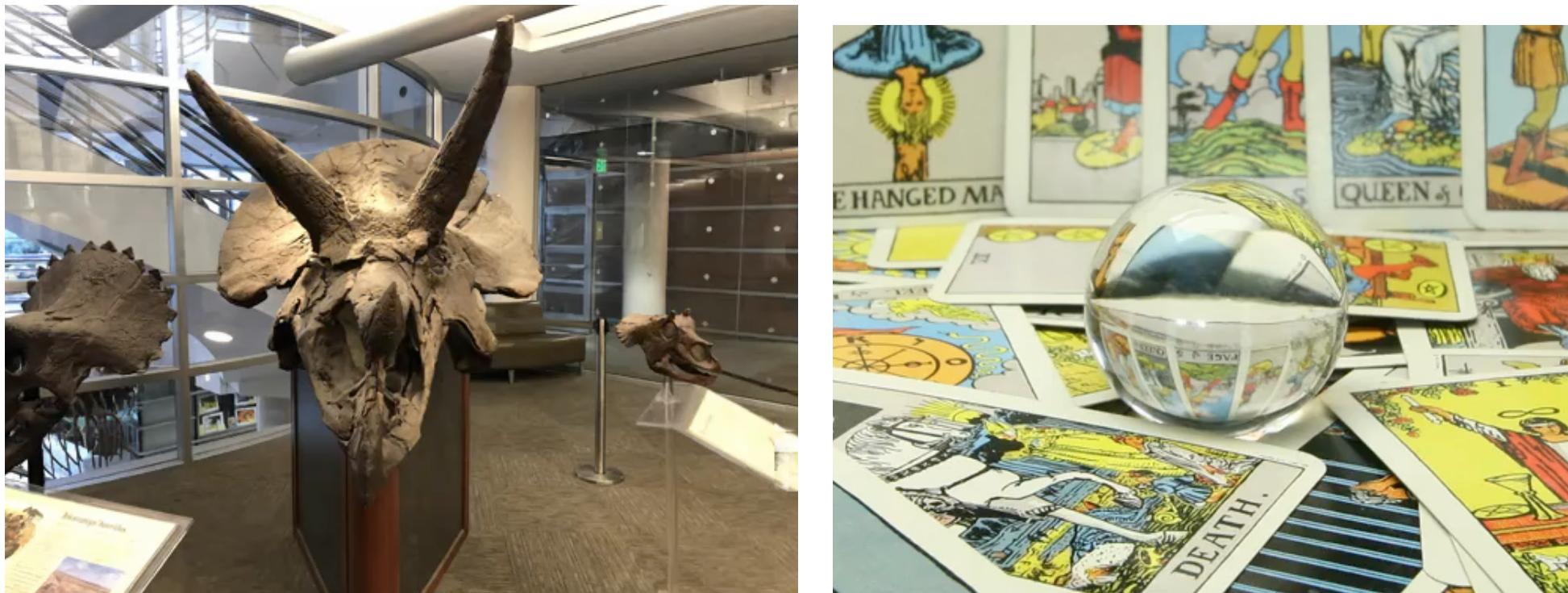
Achieving multi-view consistency in Light Fields

Single-Scene Overfitting

Regularization & Hybrid 3D / Light Field



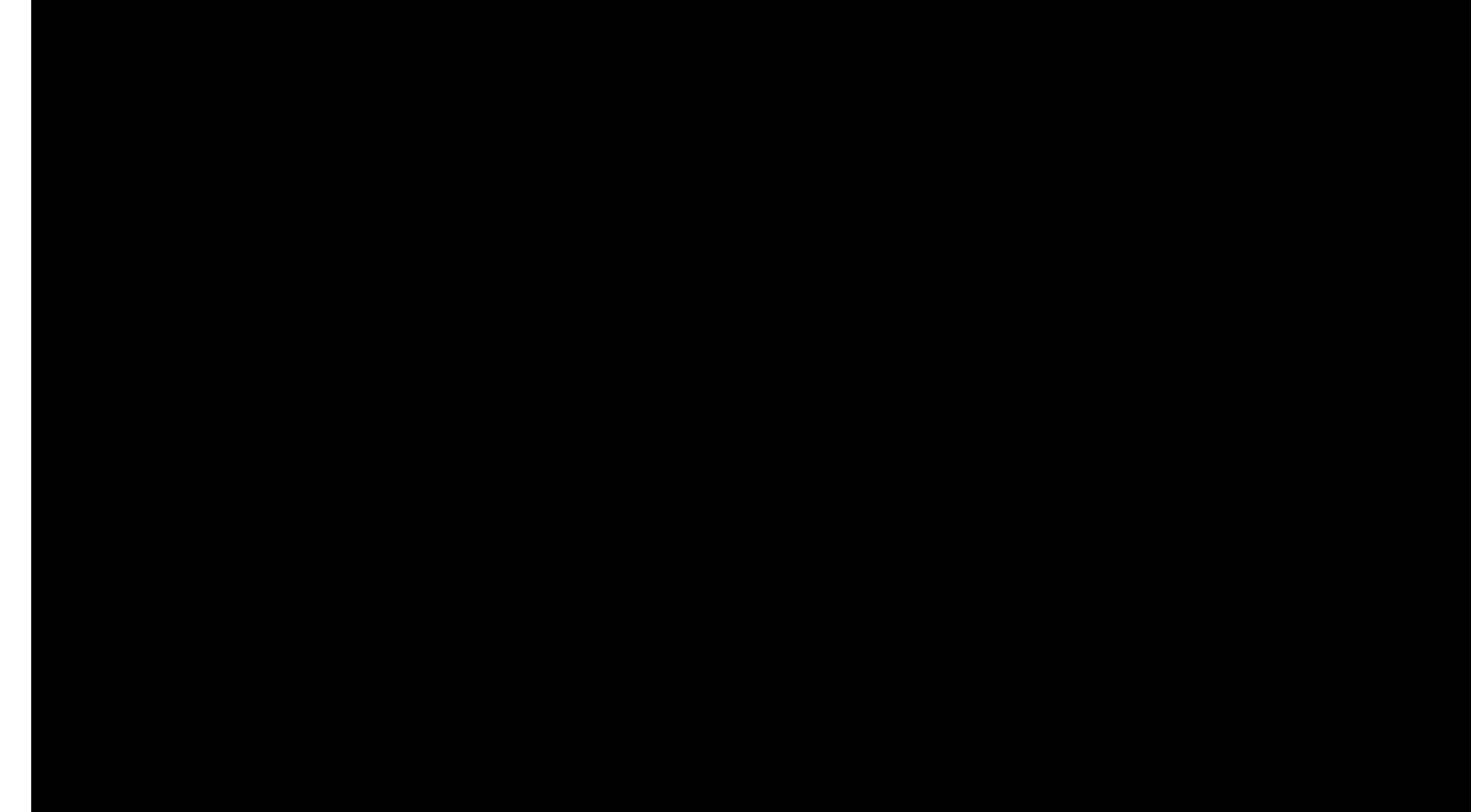
NeuLF: Efficient Novel View Synthesis with Neural 4D Light Field,
Li et al. 2022



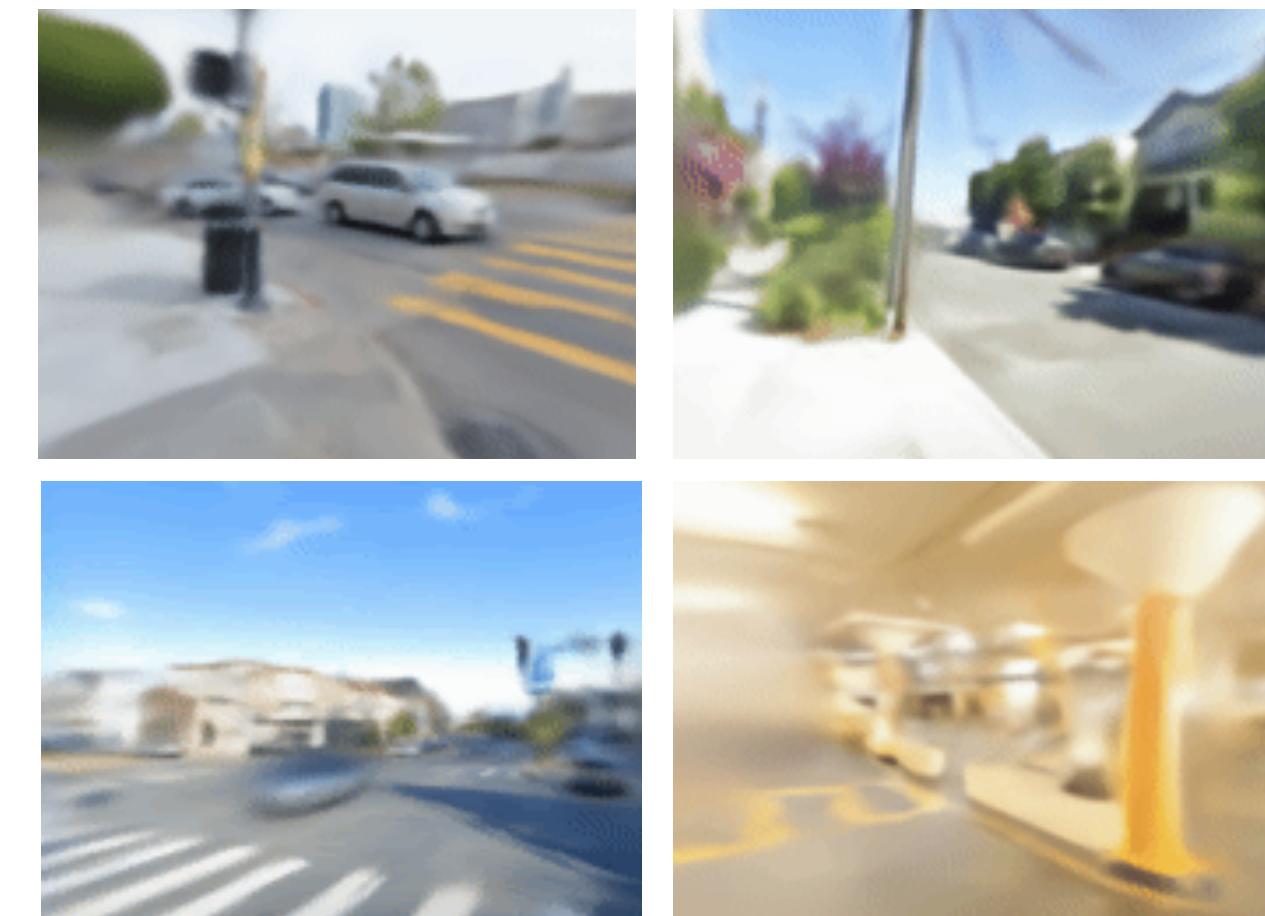
Learning Neural Light Fields with Ray-Space Embedding
Networks, Attal et al. 2022

Machine Learning

Multi-View Consistency Prior

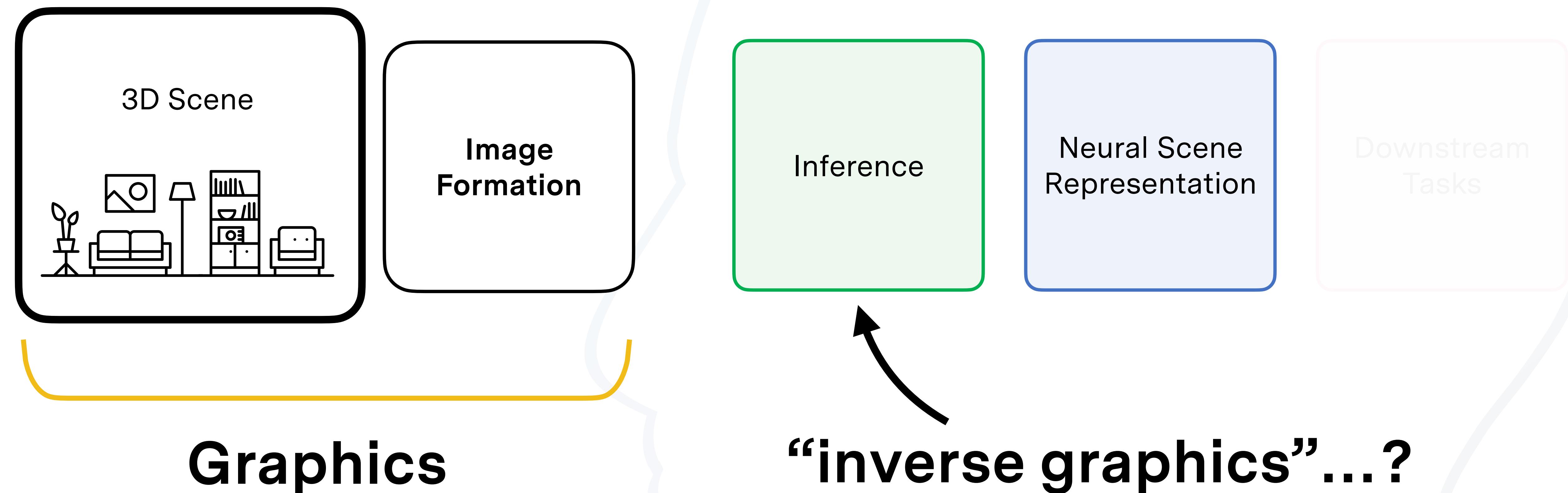


Light Field Networks, Sitzmann et al. 2021

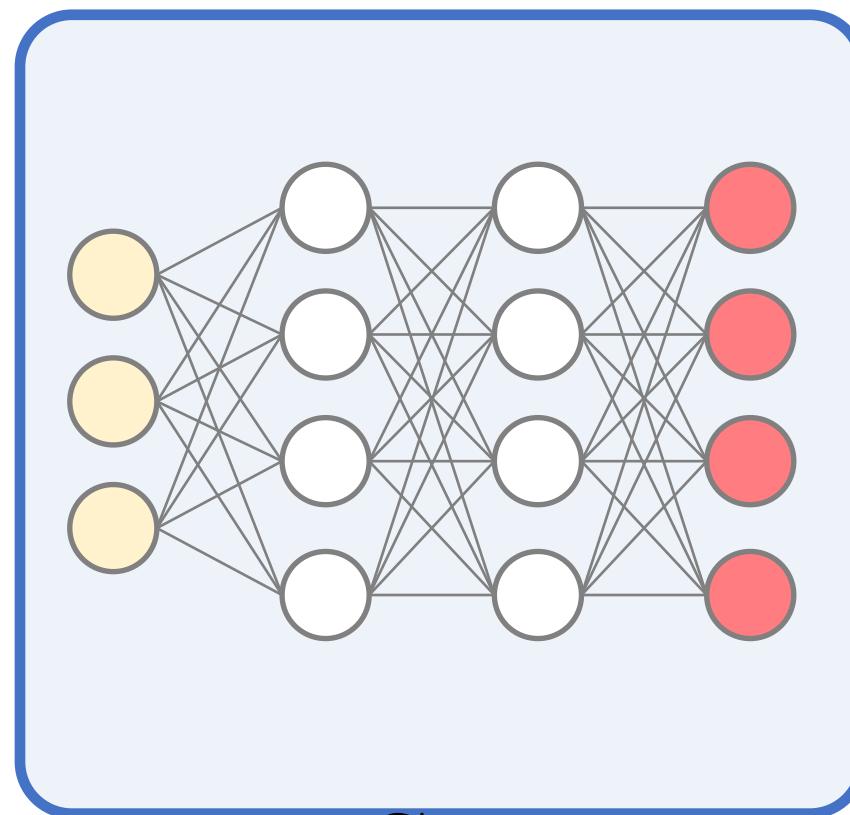
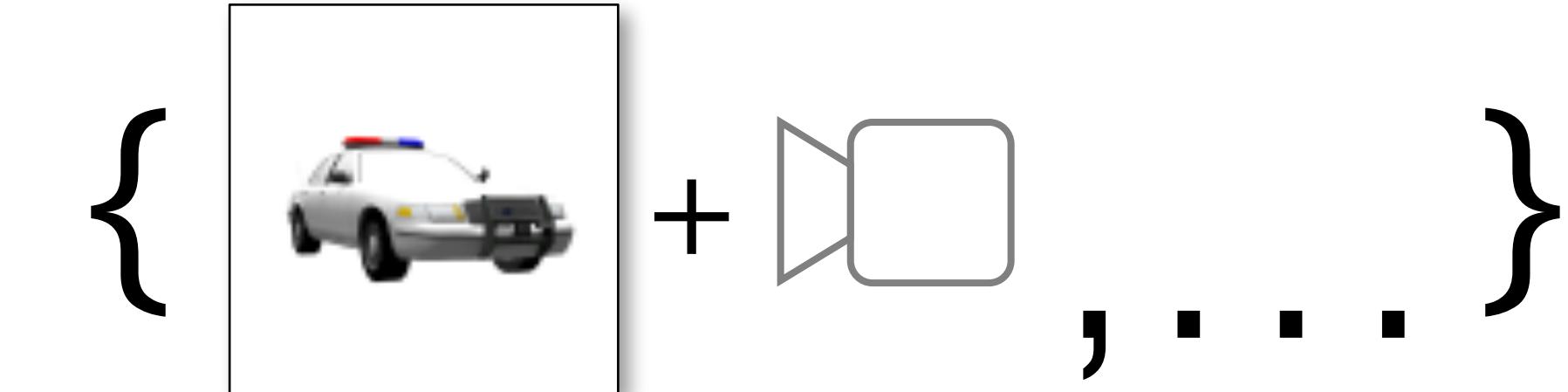


Scene Representation Transformer, Sajjadi et al. 2021

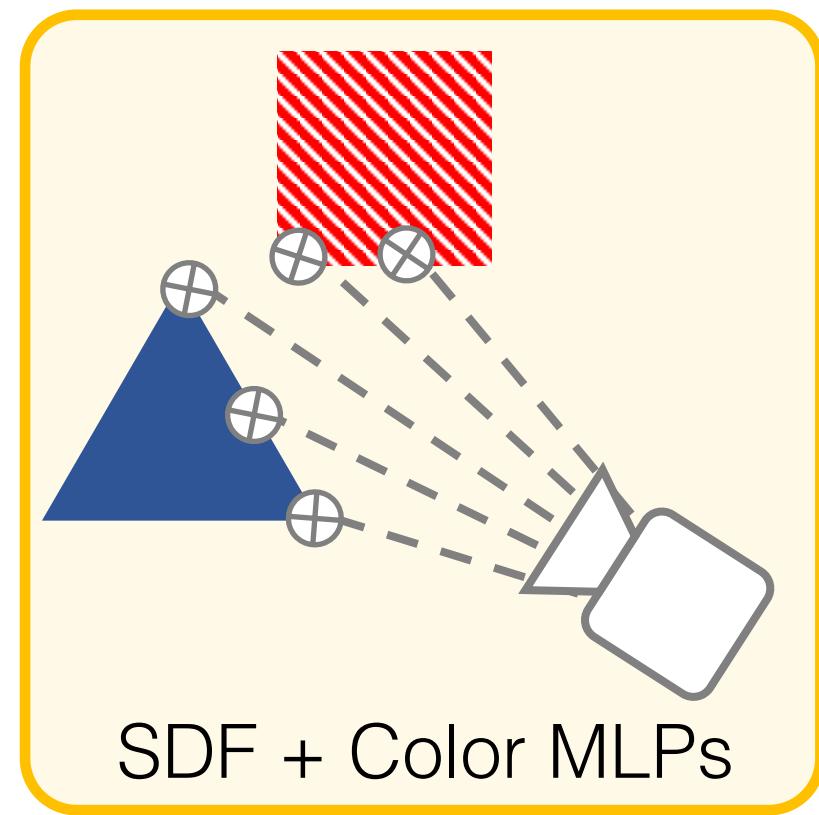
Today: Differentiable Rendering, AKA “Inverse Graphics”



Overfitting case: Inference = Fitting via Gradient Descent Optimization



Srn_ϕ

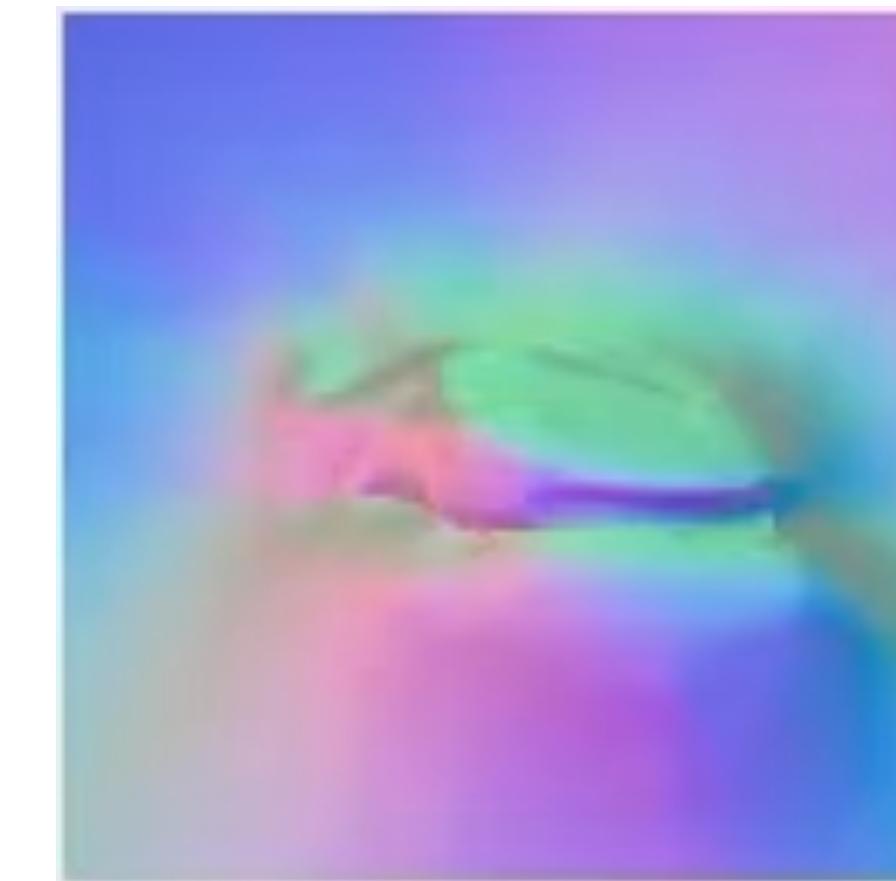


Render $_\theta$

$$\min \left\| \text{Render}_\theta(\text{Srn}_\phi, \xi_i) - \mathcal{I}_i \right\|$$



Novel View Synthesis

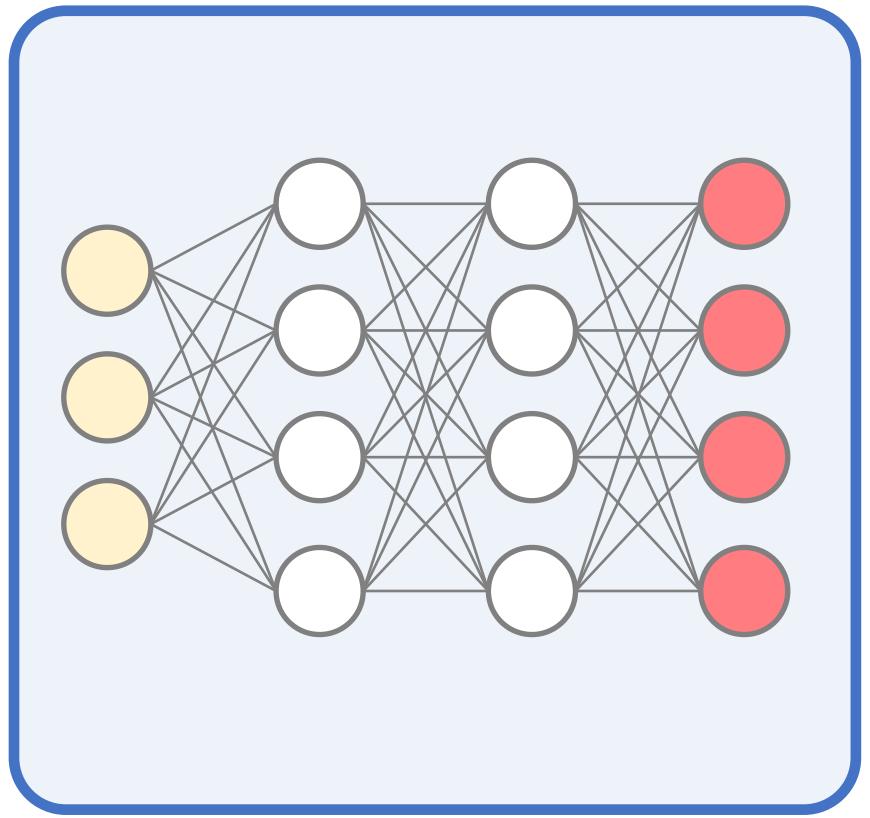


Normal map

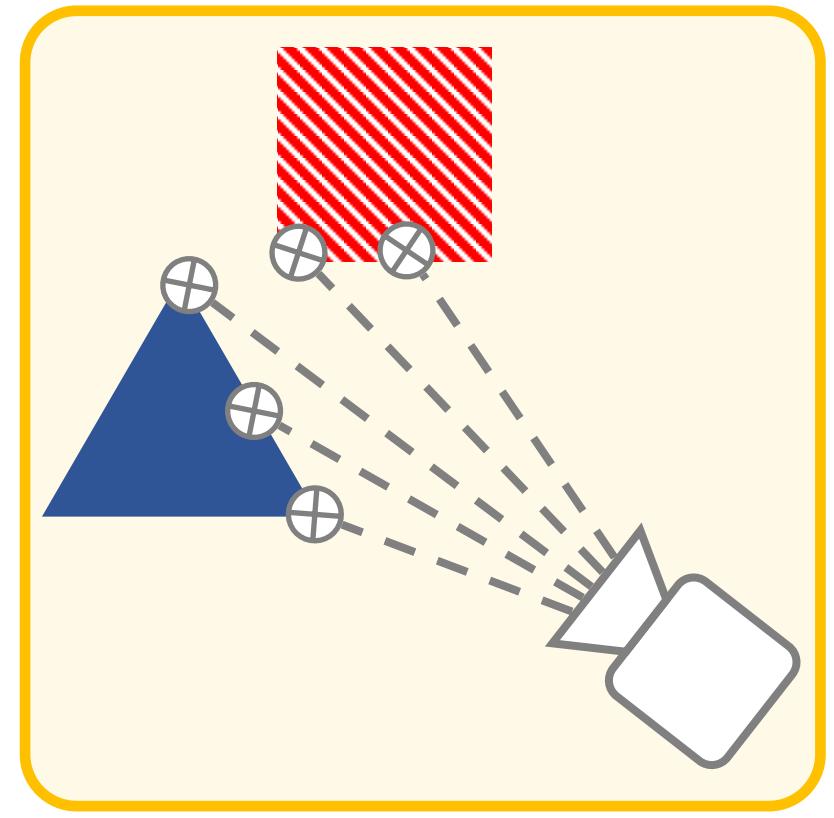


RGB

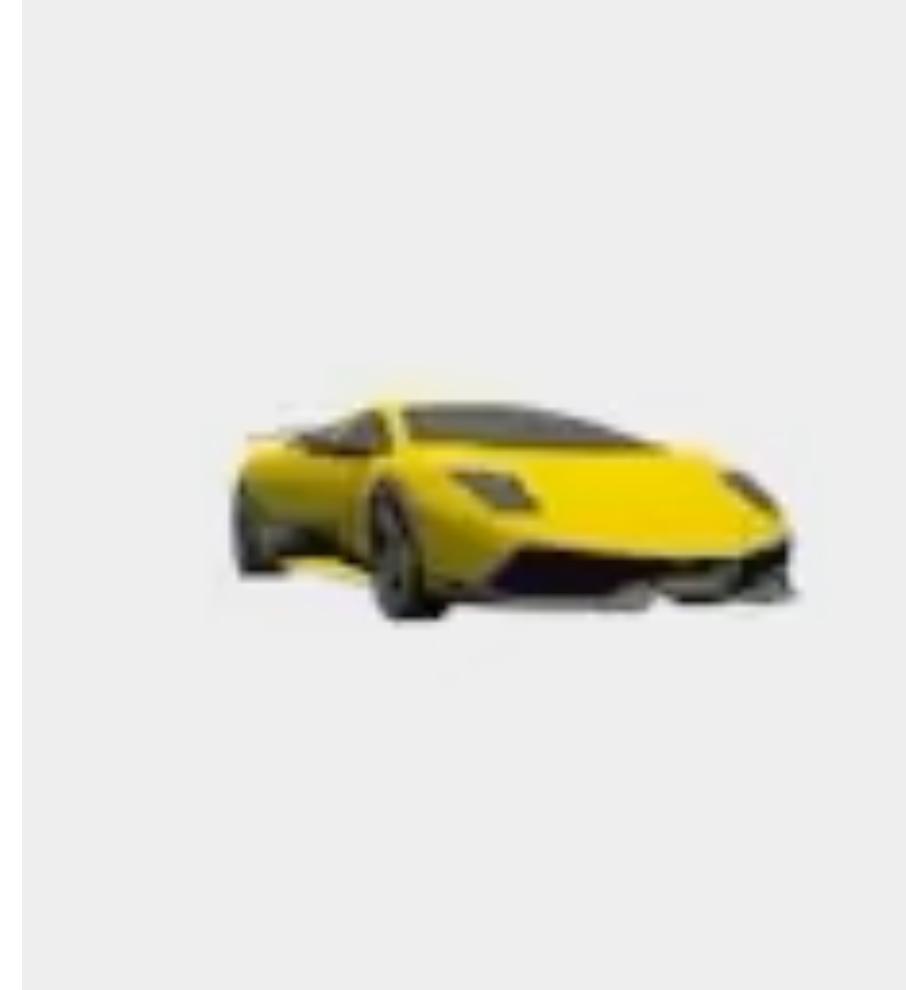
What if observations don't constrain scene representation?



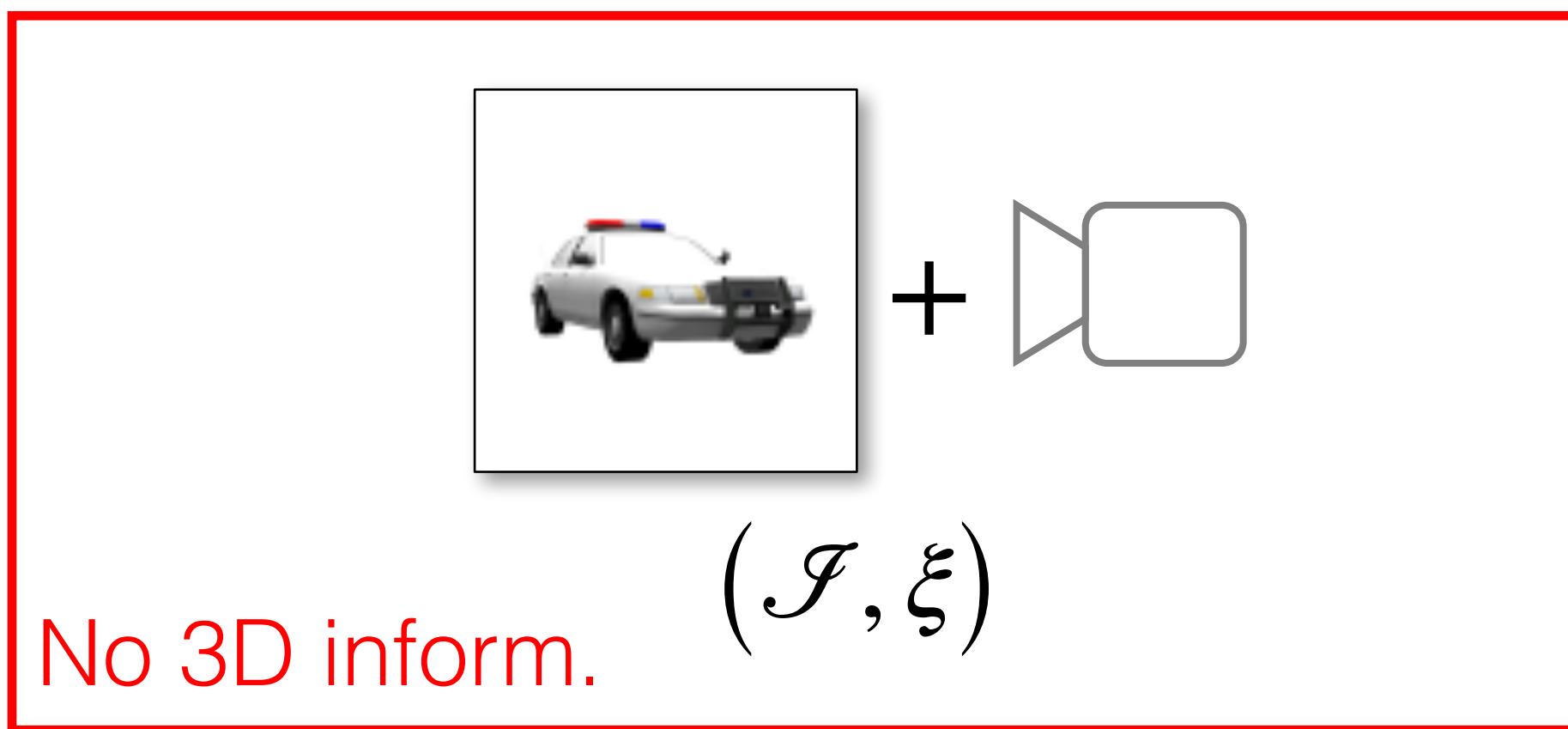
Srn_ϕ



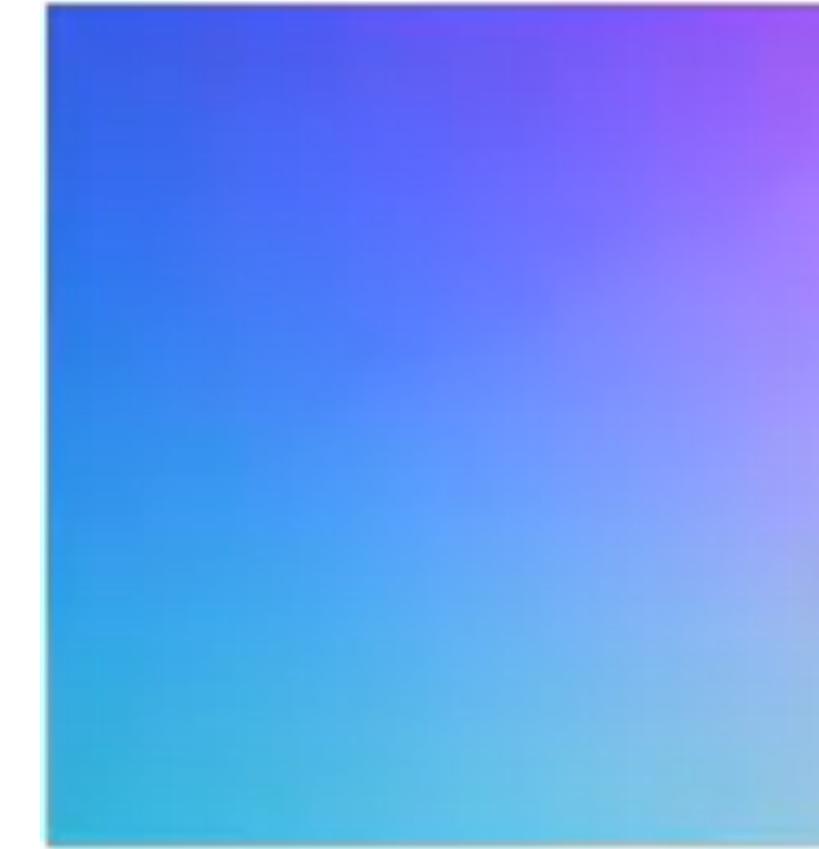
Render_θ



Input



$$\operatorname{argmin}_{\phi, \theta} \| \text{Render}_\theta(\text{Srn}_\phi, \xi) - J \|$$



Normal map

RGB



GT

Today: Differentiable Rendering, AKA “Inverse Graphics”

