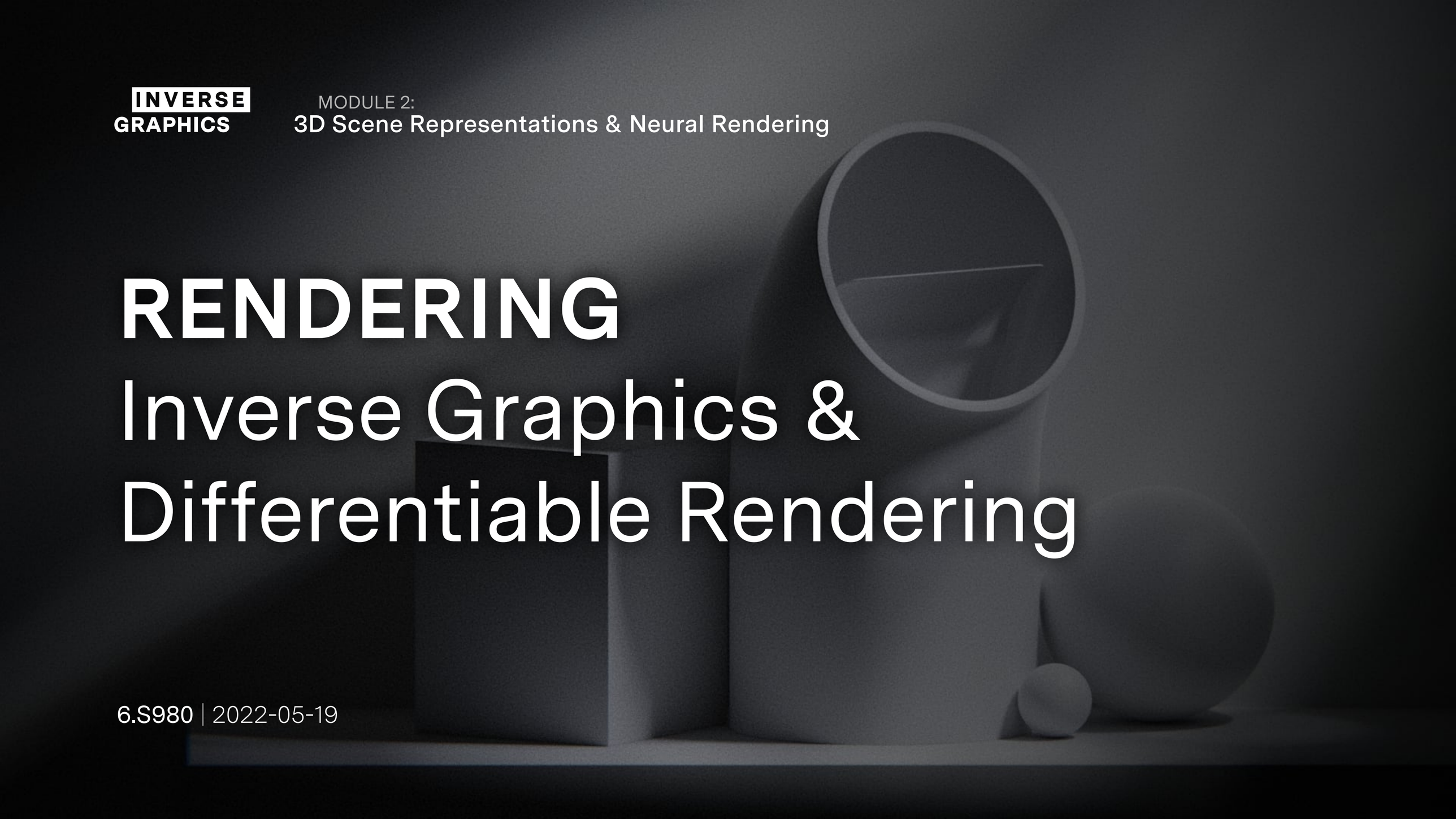


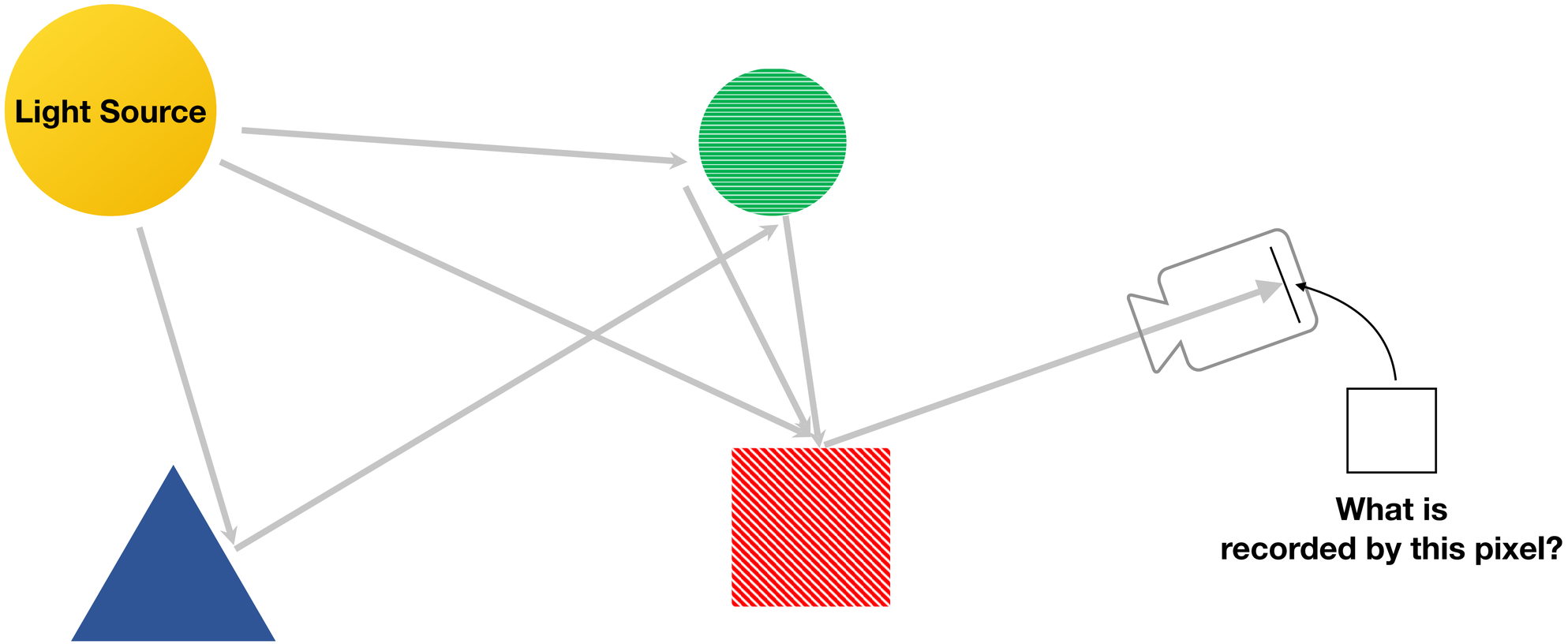
INVERSE GRAPHICS

MODULE 2:
3D Scene Representations & Neural Rendering

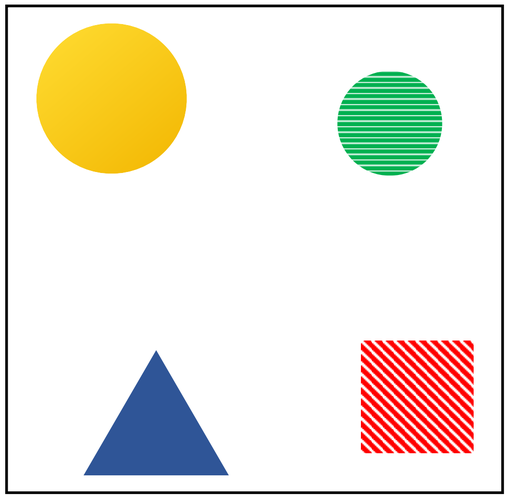
RENDERING
Inverse Graphics & Differentiable Rendering

6.S980 | 2022-05-19

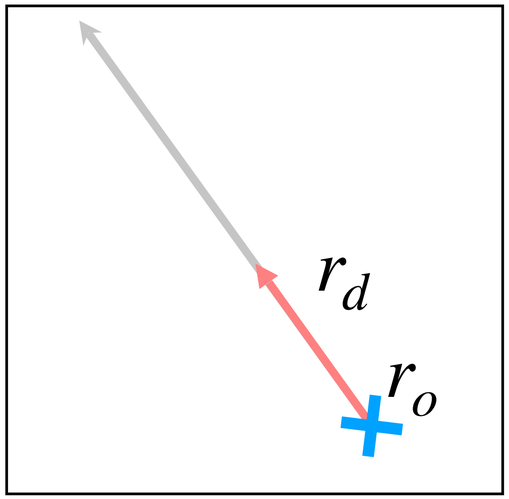
Recap - Light Transport & Cameras



Recap: Function Signature of Rendering



Scene : Light sources, materials, shapes, …



Camera Ray

Radiance of the ray

Recap: The rendering equation

emitted radiance

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

solid angle differential

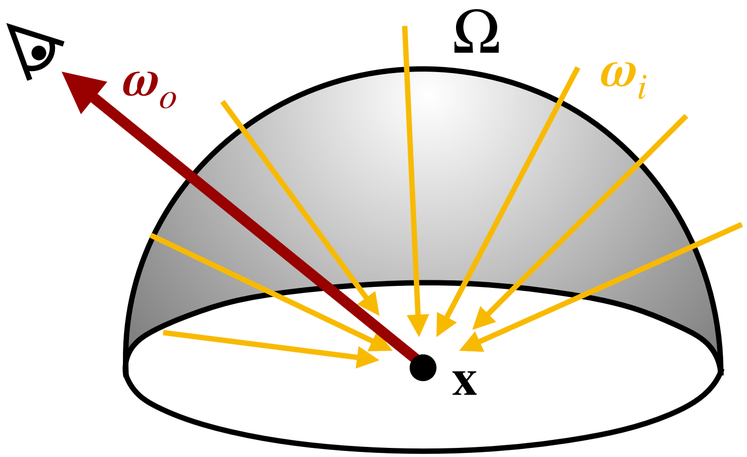
outgoing radiance in $\boldsymbol{\omega}_o$

fraction of incoming light that is reflected into $\boldsymbol{\omega}_o$

incoming radiance

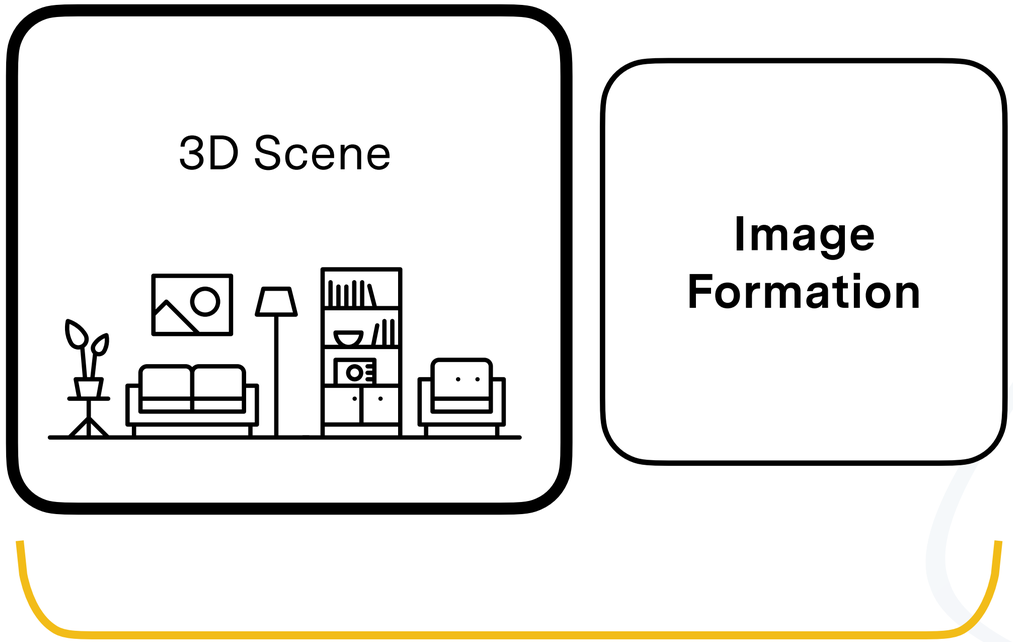
angle weight of photon density

Conservation of energy: $\int_{\boldsymbol{\omega}_i \in \Omega} f(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, (\boldsymbol{\omega}_i \cdot \mathbf{n})\, d\boldsymbol{\omega}_i \leq 1$



Slide Credit: Stanford CS348I, Computer Graphics in the Age of AI, Stanford, C. Karen Liu & Jiajun Wu

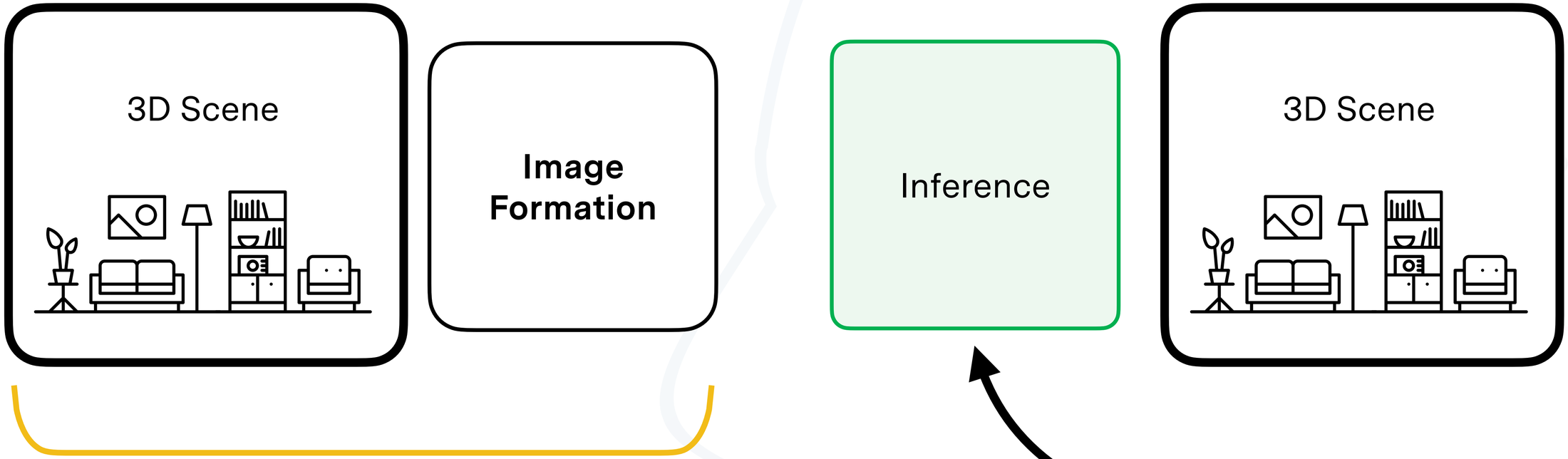
Today: Differentiable Rendering, AKA “Inverse Graphics”



Graphics

“inverse graphics”…?

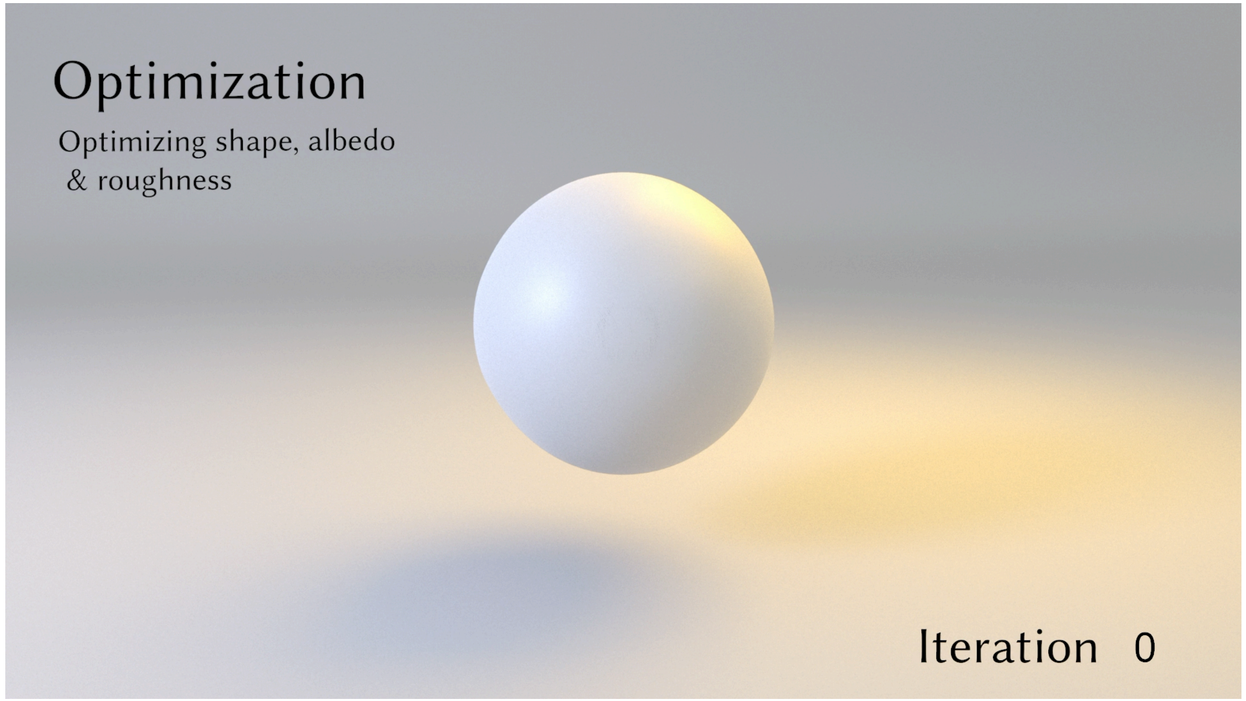
Today: Differentiable Rendering, AKA “Inverse Graphics”



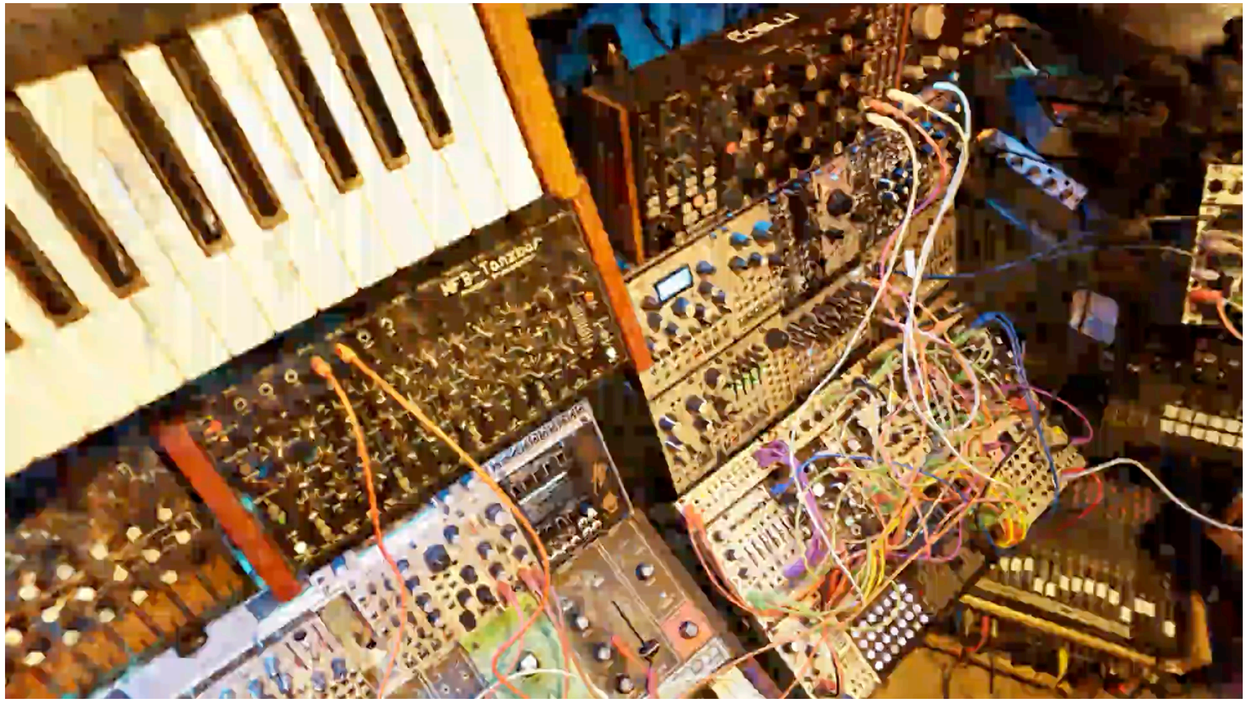
Graphics

“inverse graphics”…?

Differentiable Rendering



Differentiable Signed Distance Function Rendering,
Vicini et al. 2022



InstantNGP, Müller et al. 2022

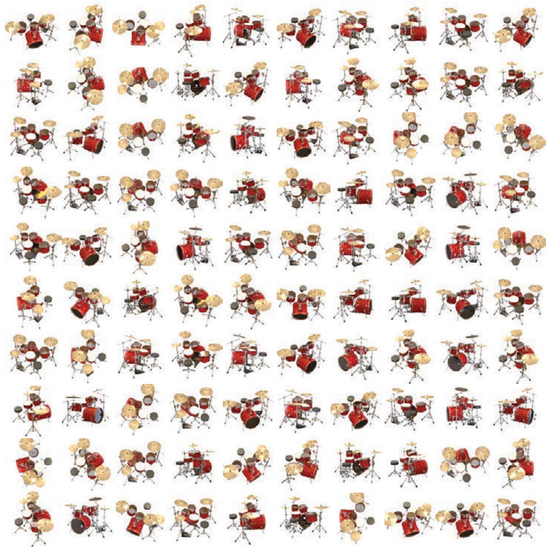
Why?	**Part of our problem relates to inverting the rendering process: Given 2D images, we want to reconstruct 3D scenes. Differentiable rendering is one way of exactly inverting the rendering process.**
What you'll learn.	**The structure of differentiable renderers, pros and cons and assumptions of different algorithms.**

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

3D Scene

Camera Poses

Render



Images

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

Images

Reconstruct

3D Scene

Camera Poses

How to get camera poses?

Images

Reconstruct

3D Scene

Camera Poses

Bundle Adjustment (Multi-View Geometry Lecture)!

Bundle adjustment minimizes the reprojection error of all observations:

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

Images

Camera Poses

Can assume we know the camera poses.

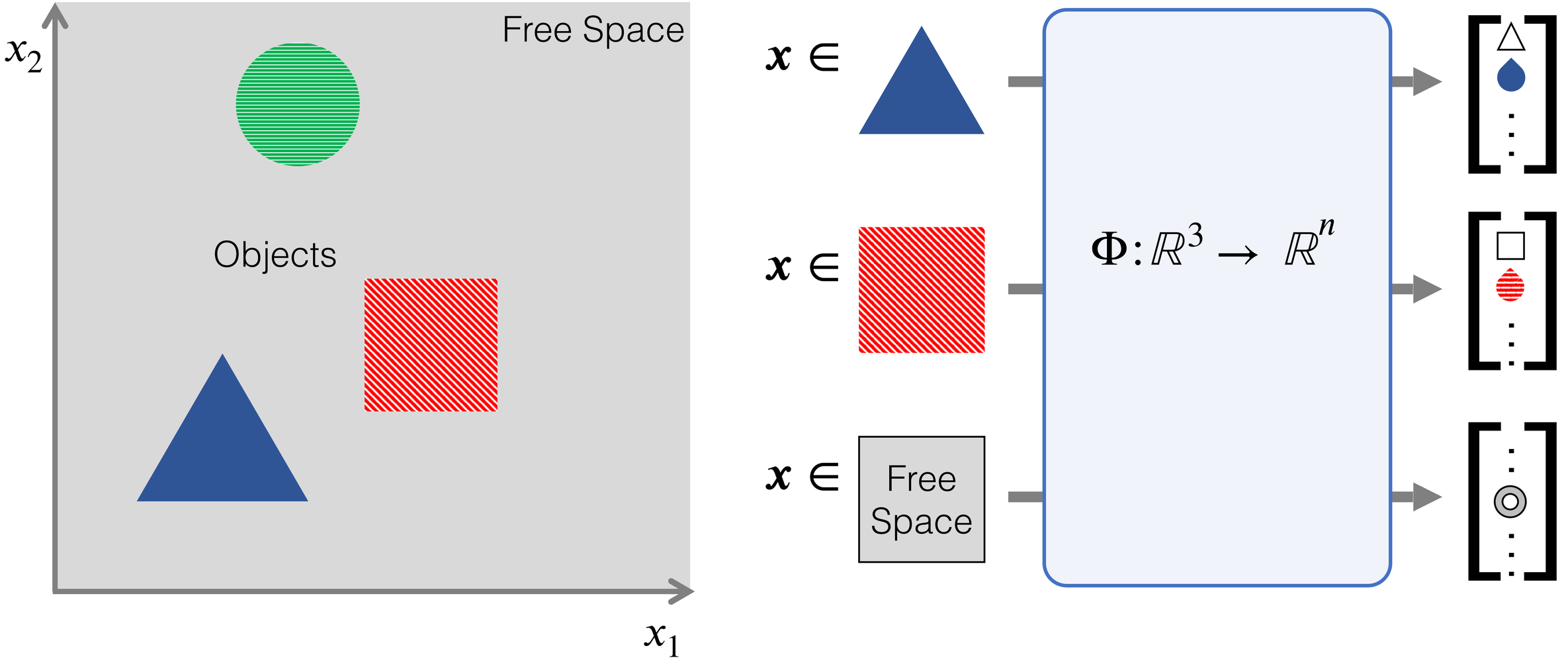
Images

Camera Poses

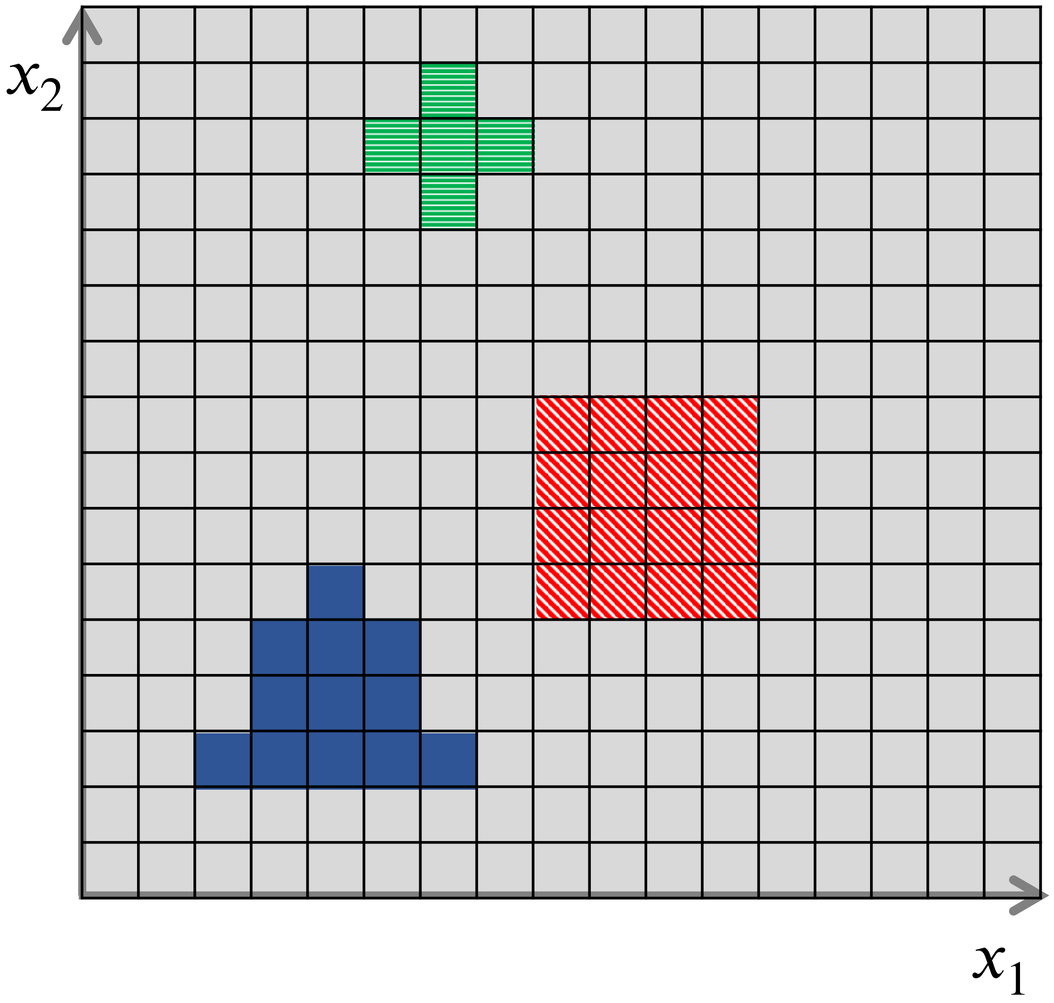
Reconstruct

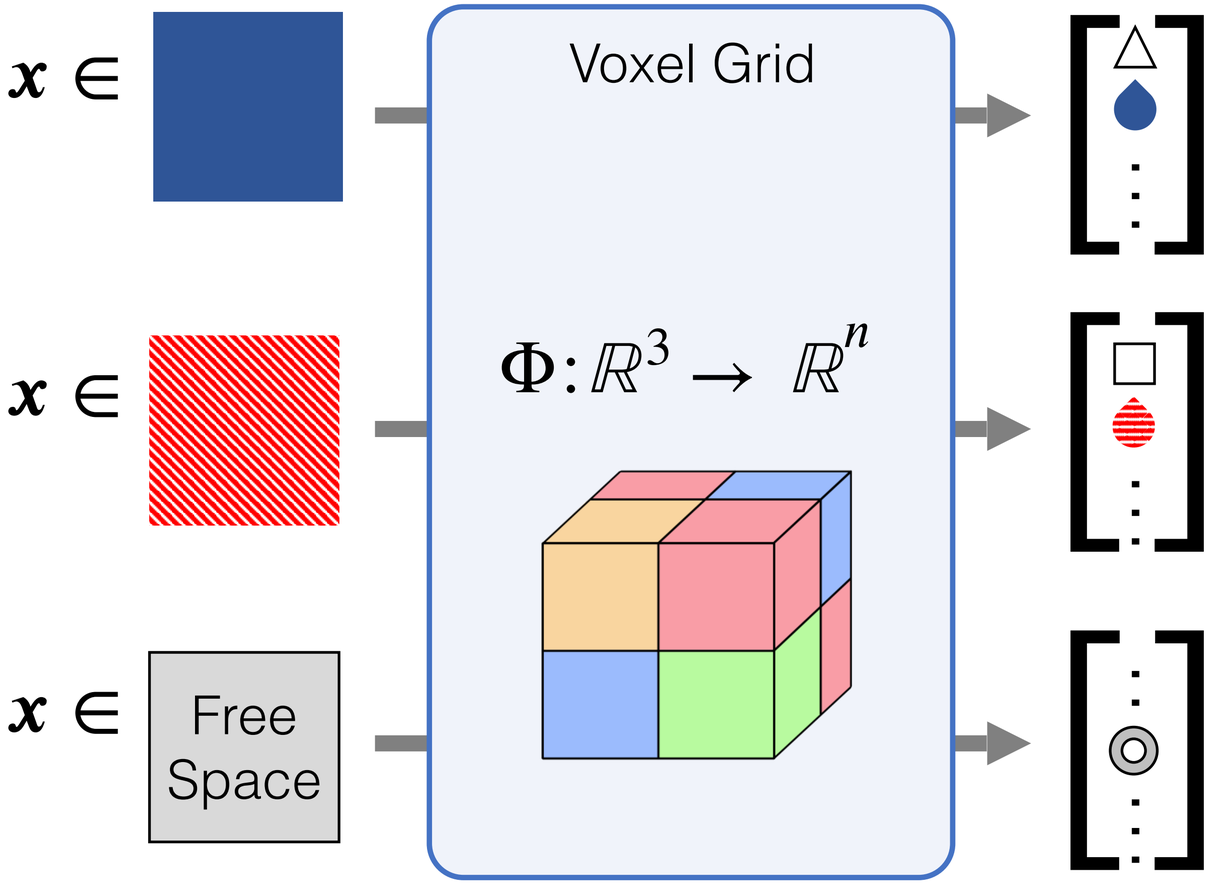
3D Scene

Starting Point: 3D Scene Representation

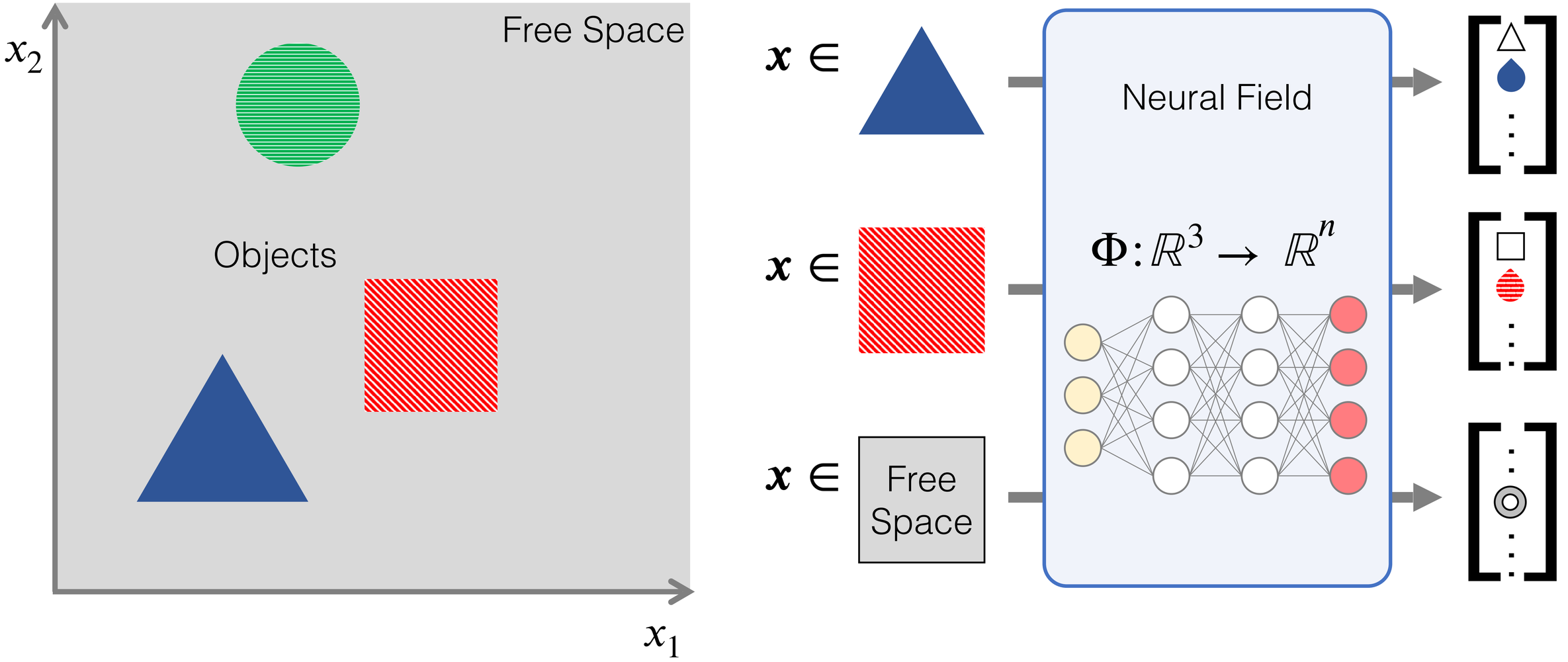


Starting Point: 3D Scene Representation





Starting Point: 3D Scene Representation



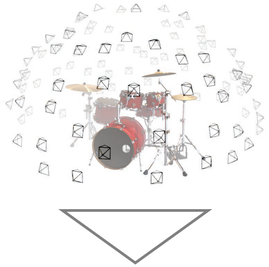
Differentiable Rendering

Scene Representation

Renderer

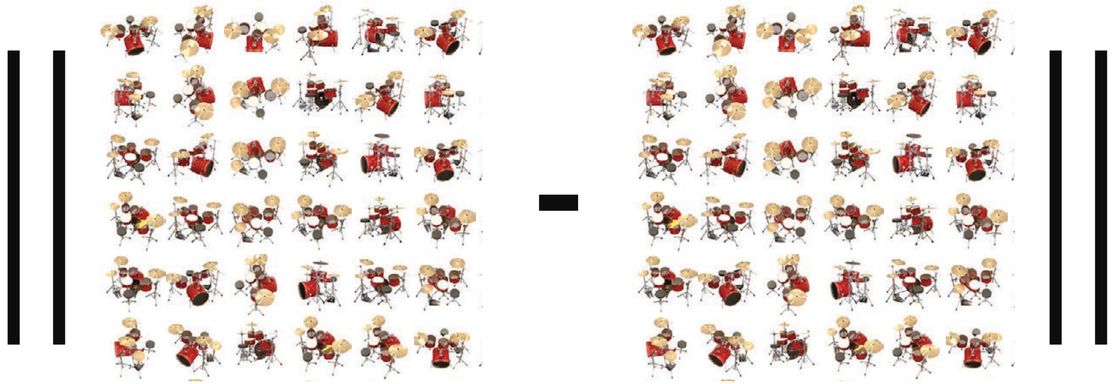
Rendered Images

Differentiable Rendering



Scene Representation

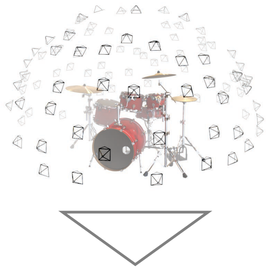
Renderer



Rendered Images

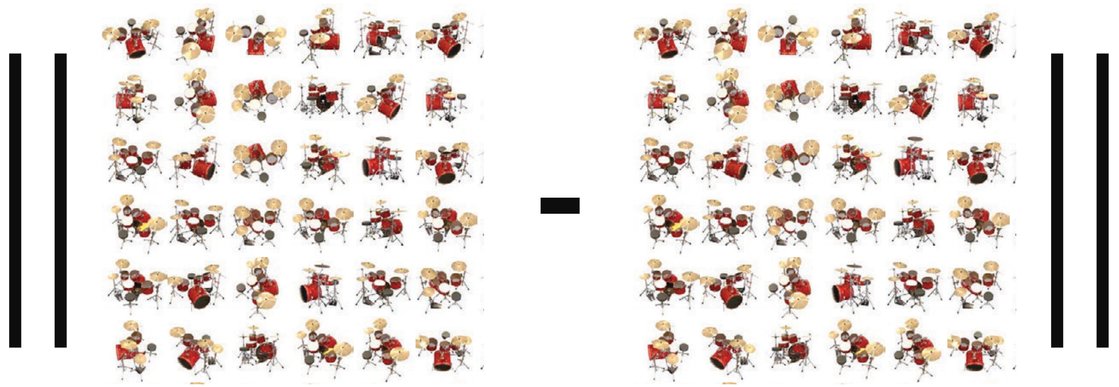
GT Images

Differentiable Rendering



Scene Representation

Differentiable Renderer

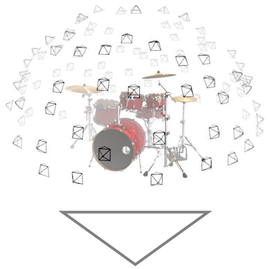


Rendered Images

GT Images

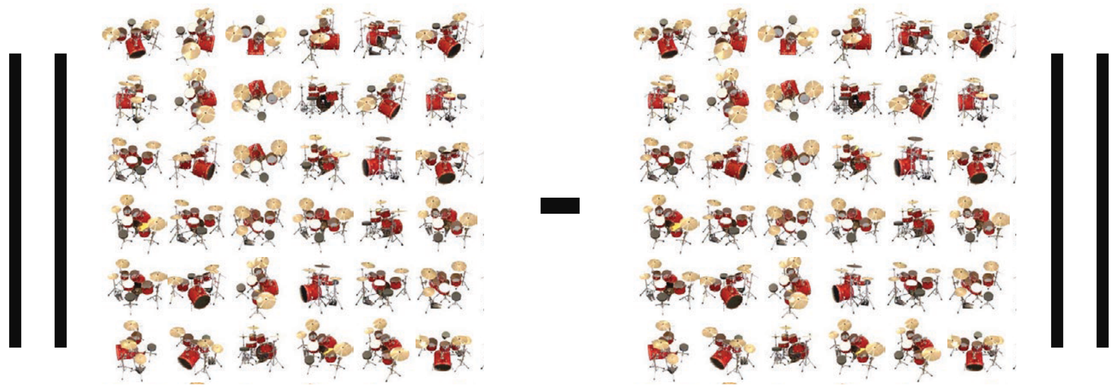
Optimization via SGD

Differentiable Rendering



Scene Representation

Differentiable Renderer



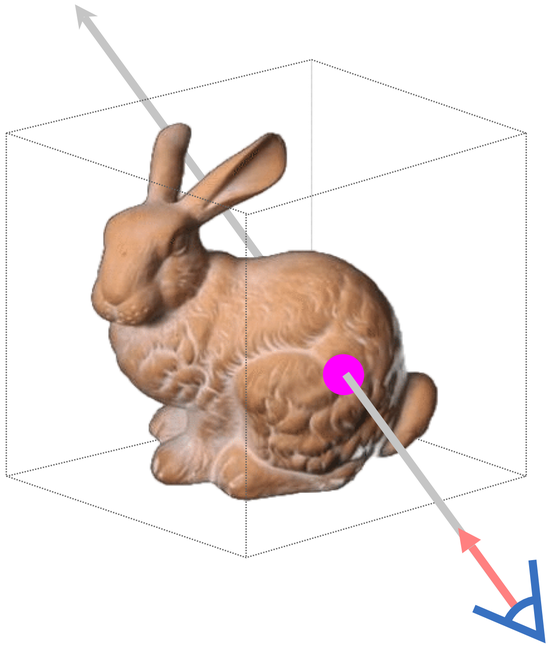
Rendered Images

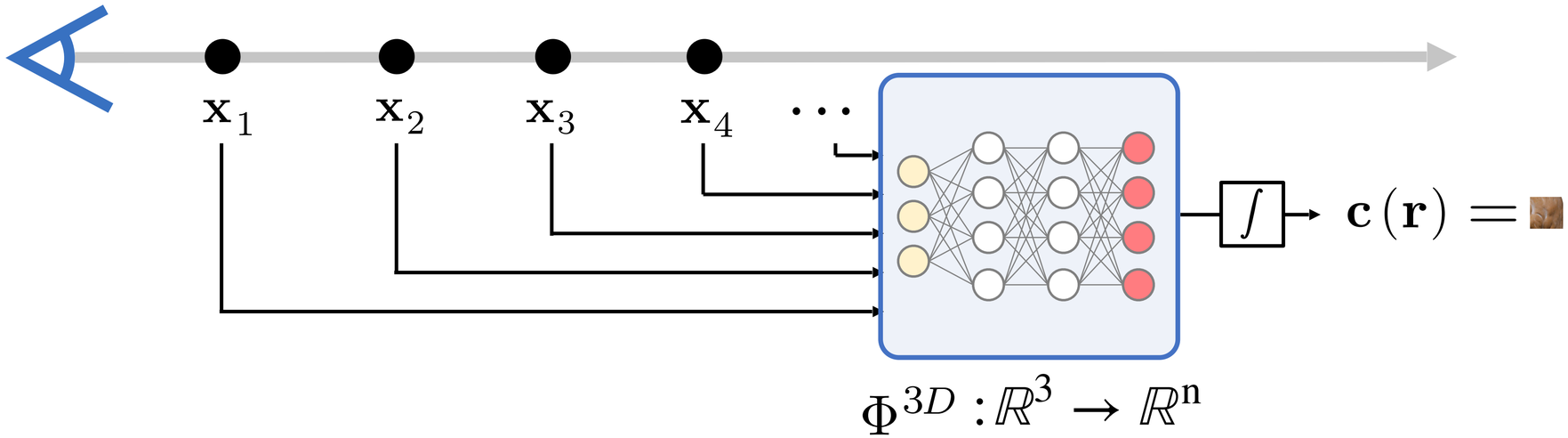
GT Images

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

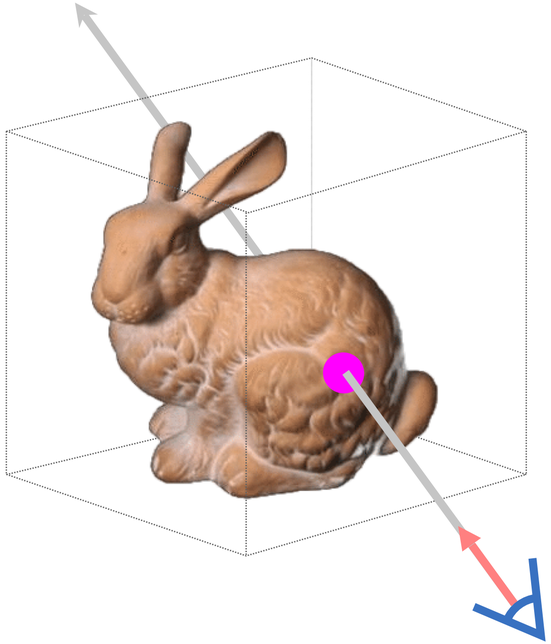
Questions?

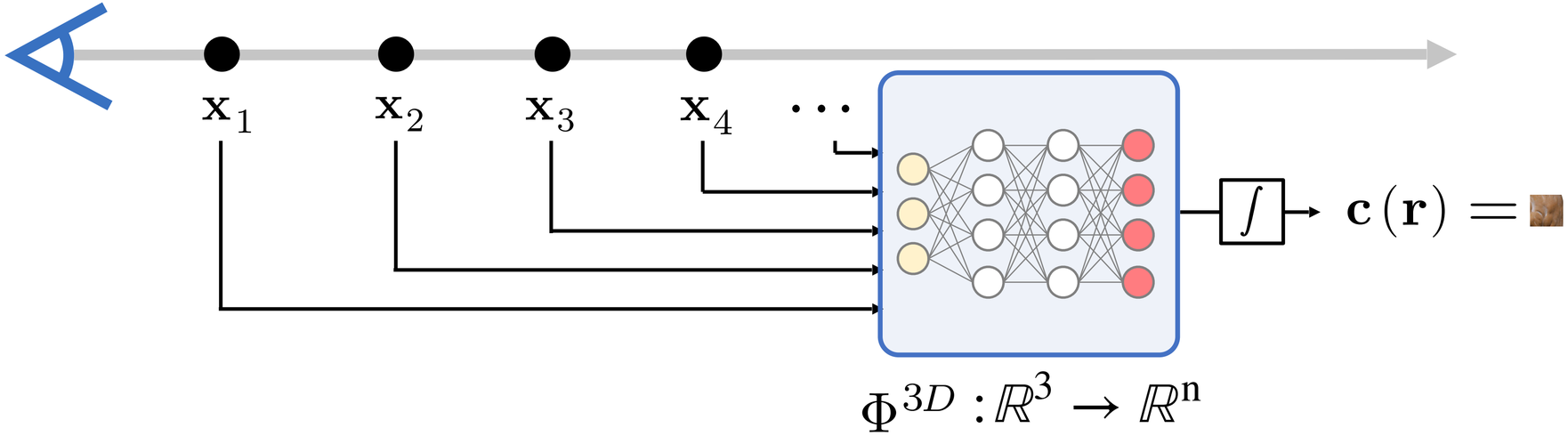
General structure of Neural Renderers for 3D Fields





General structure of Neural Renderers for 3D Fields





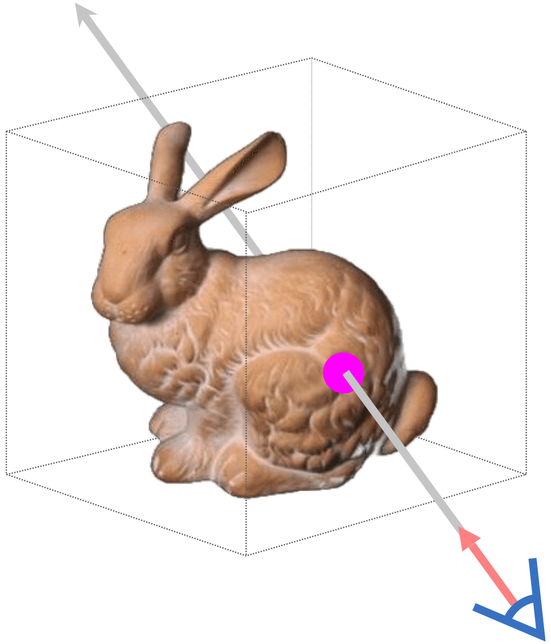
Sphere-Tracing [JC Hart, 1996]

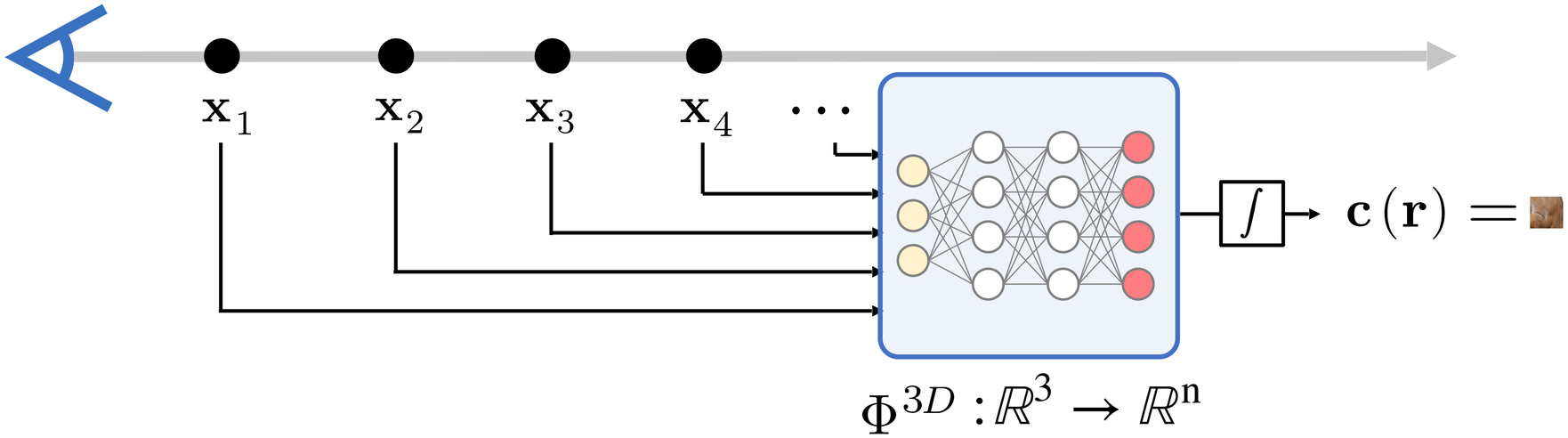
Volumetric

Hybrid implicit-volumetric

Learned aggregation

General structure of Neural Renderers for 3D Fields





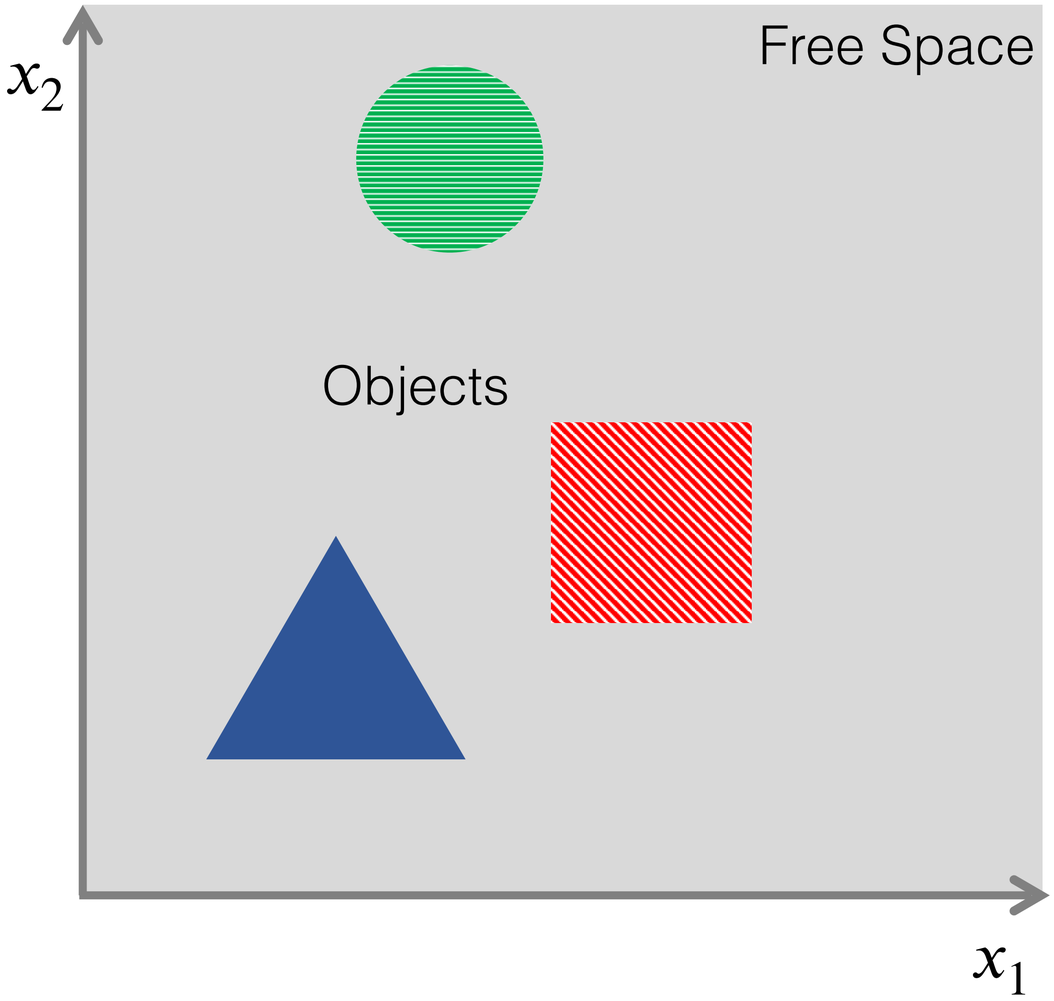
Sphere-Tracing
[JC Hart, 1996]

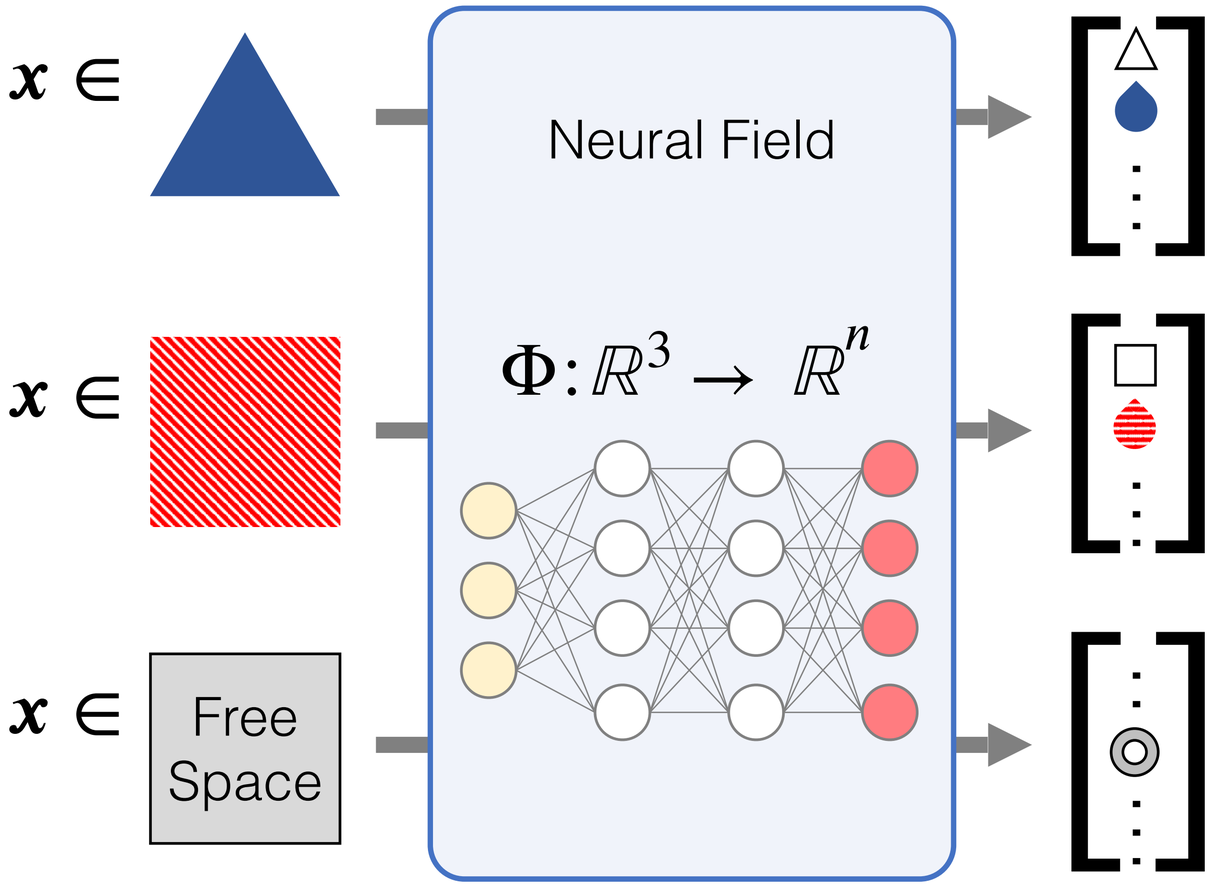
Volumetric

Hybrid implicit-volumetric

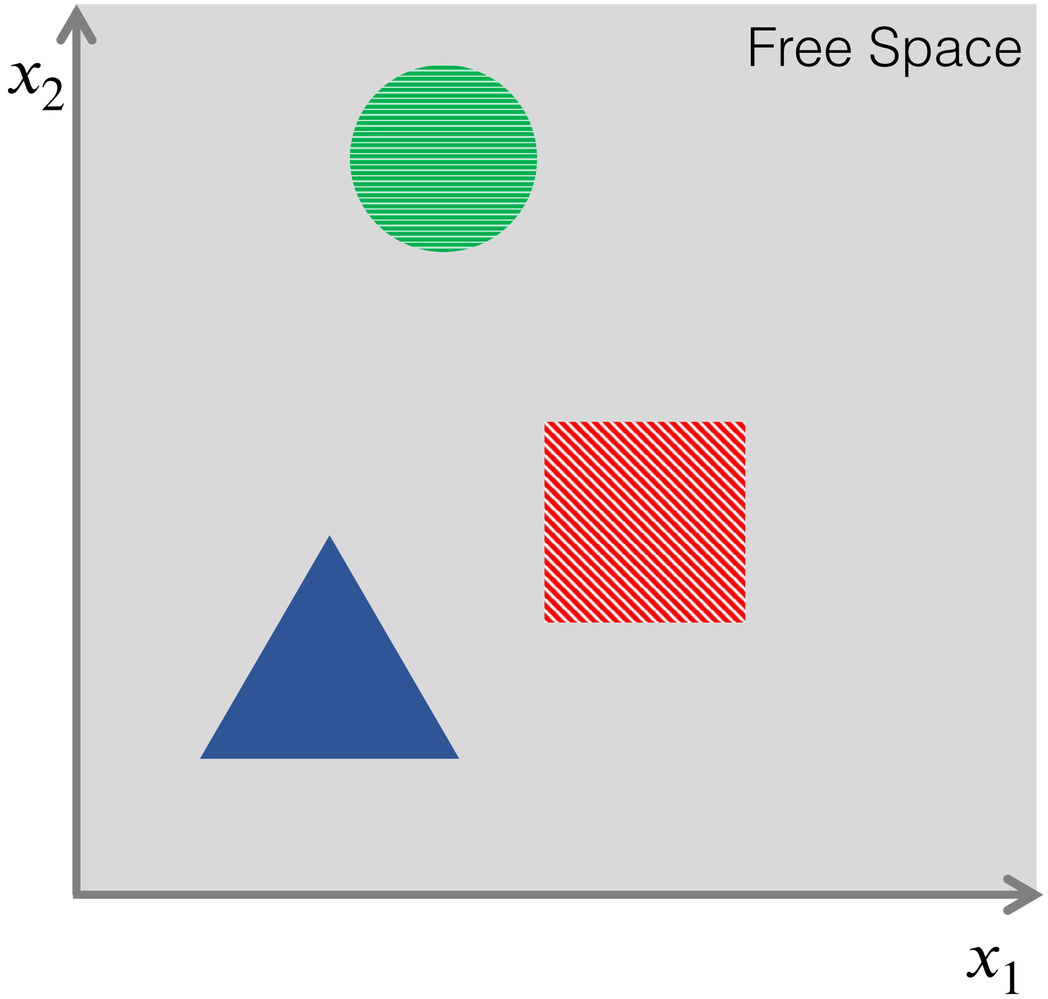
Learned aggregation

Scene Representation Networks (Sitzmann et al. 2019)





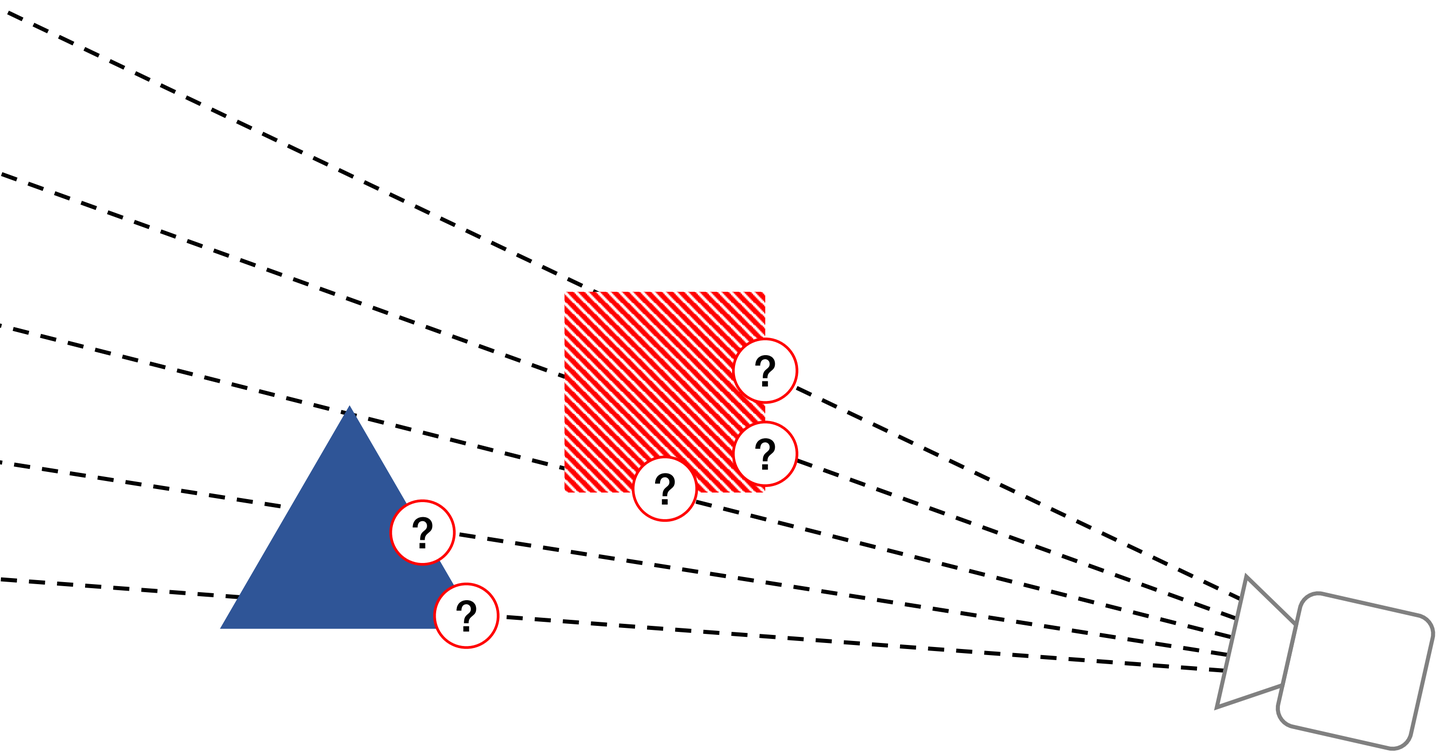
Sphere Tracing



Sphere Tracing

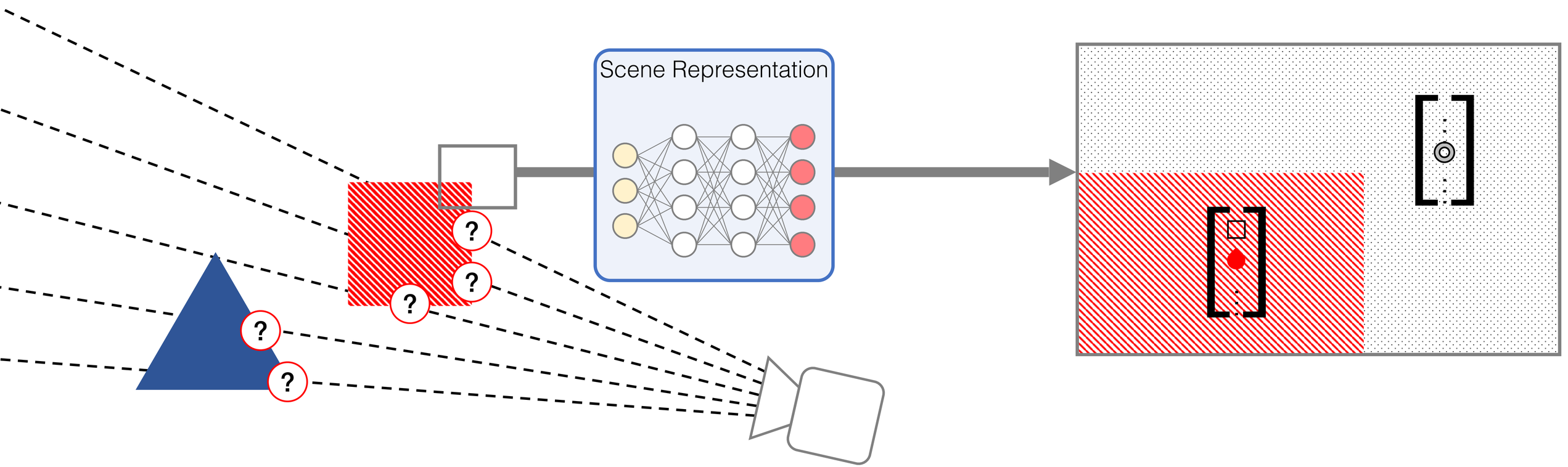
Sphere Tracing Step 1: Intersection Testing.

- How to parameterize scene geometry?



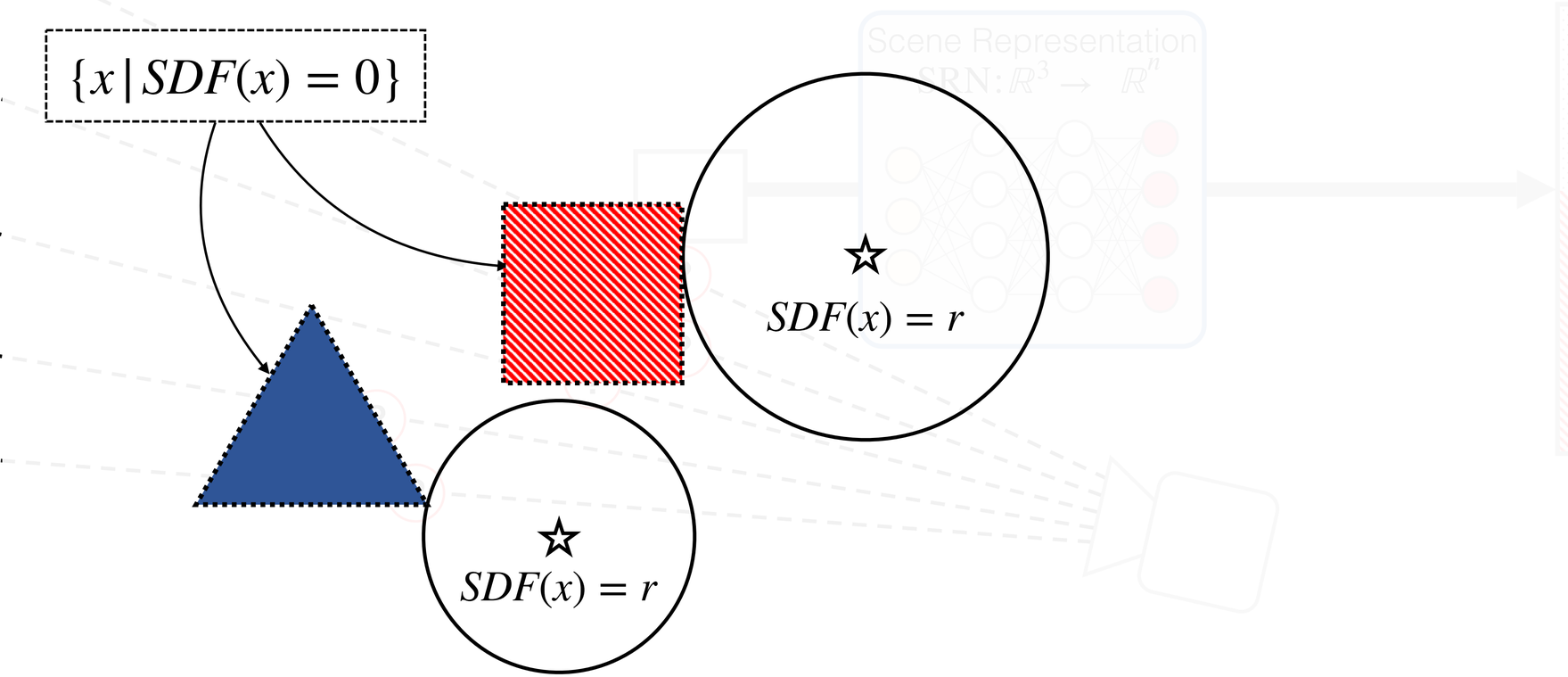
Sphere Tracing Step 1: Intersection Testing.

- How to parameterize scene geometry?



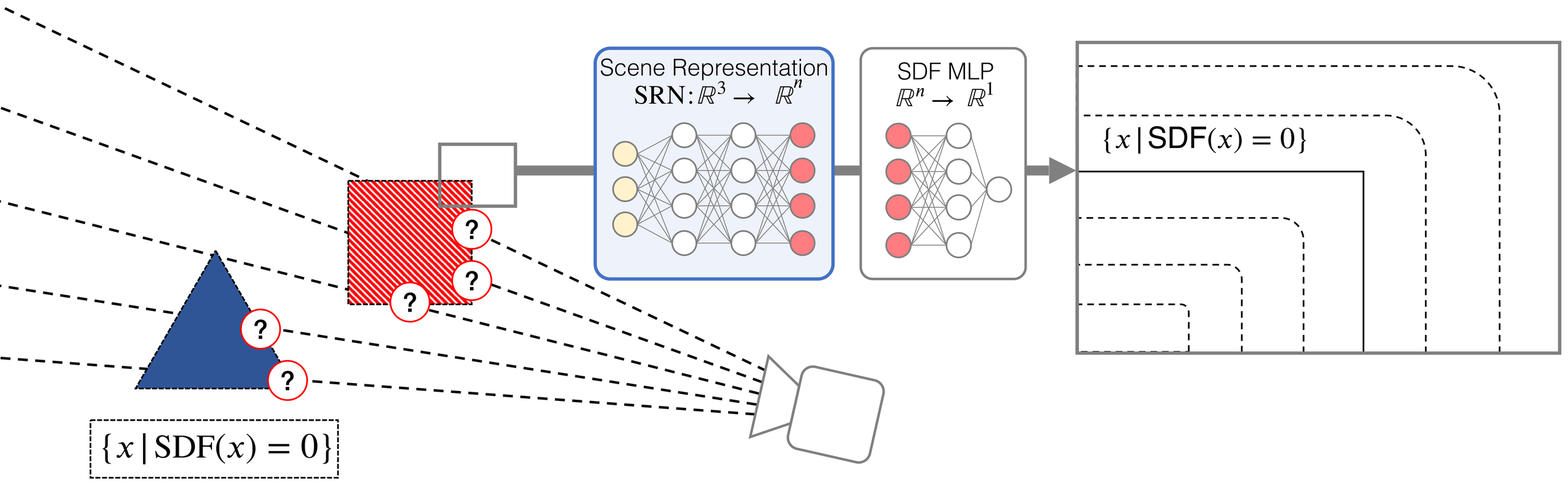
Neural Renderer Step 1: Intersection Testing.

- How to parameterize scene geometry?



Sphere Tracing Step 1: Intersection Testing.

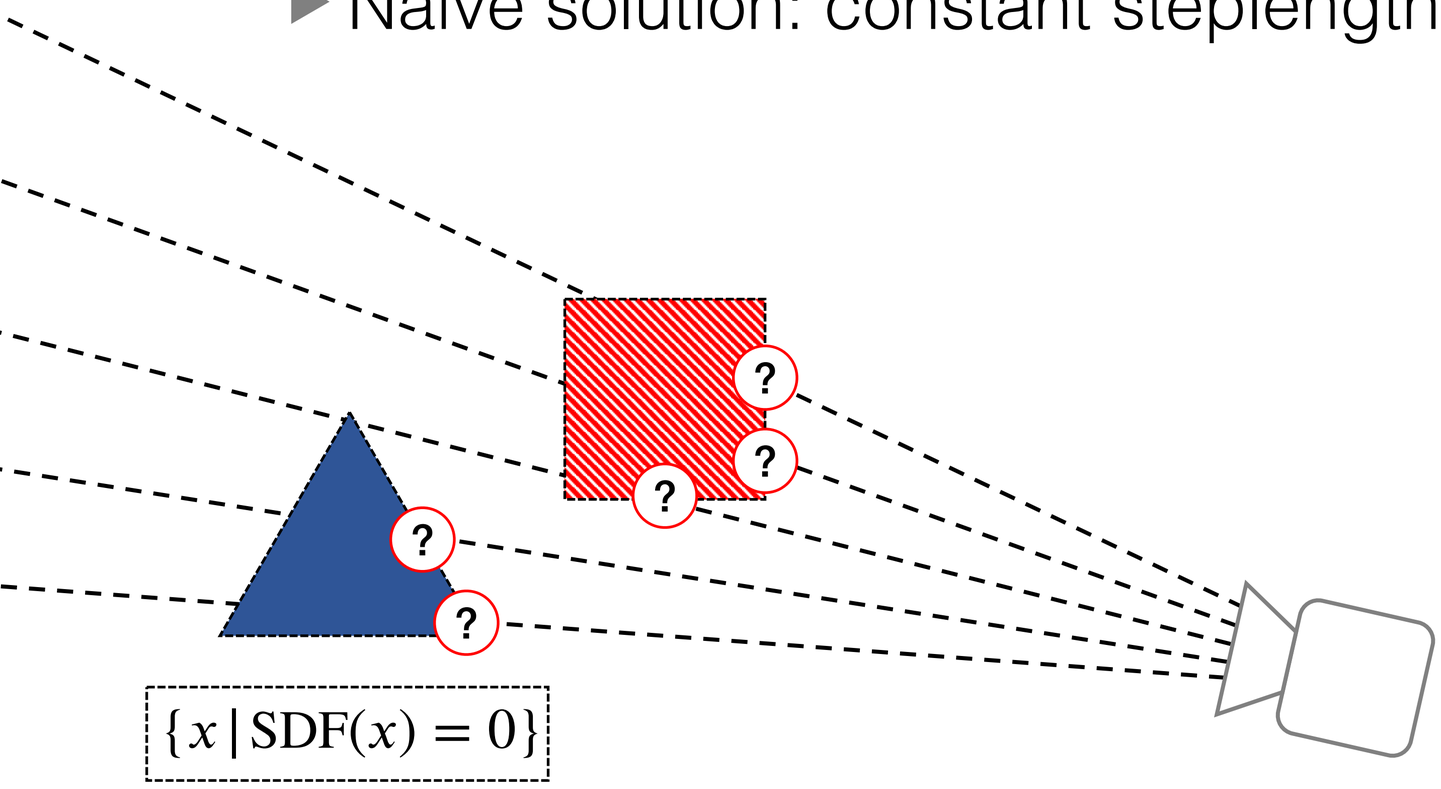
- How to parameterize scene geometry?
- Signed Distance Function maps point to distance to closest surface.



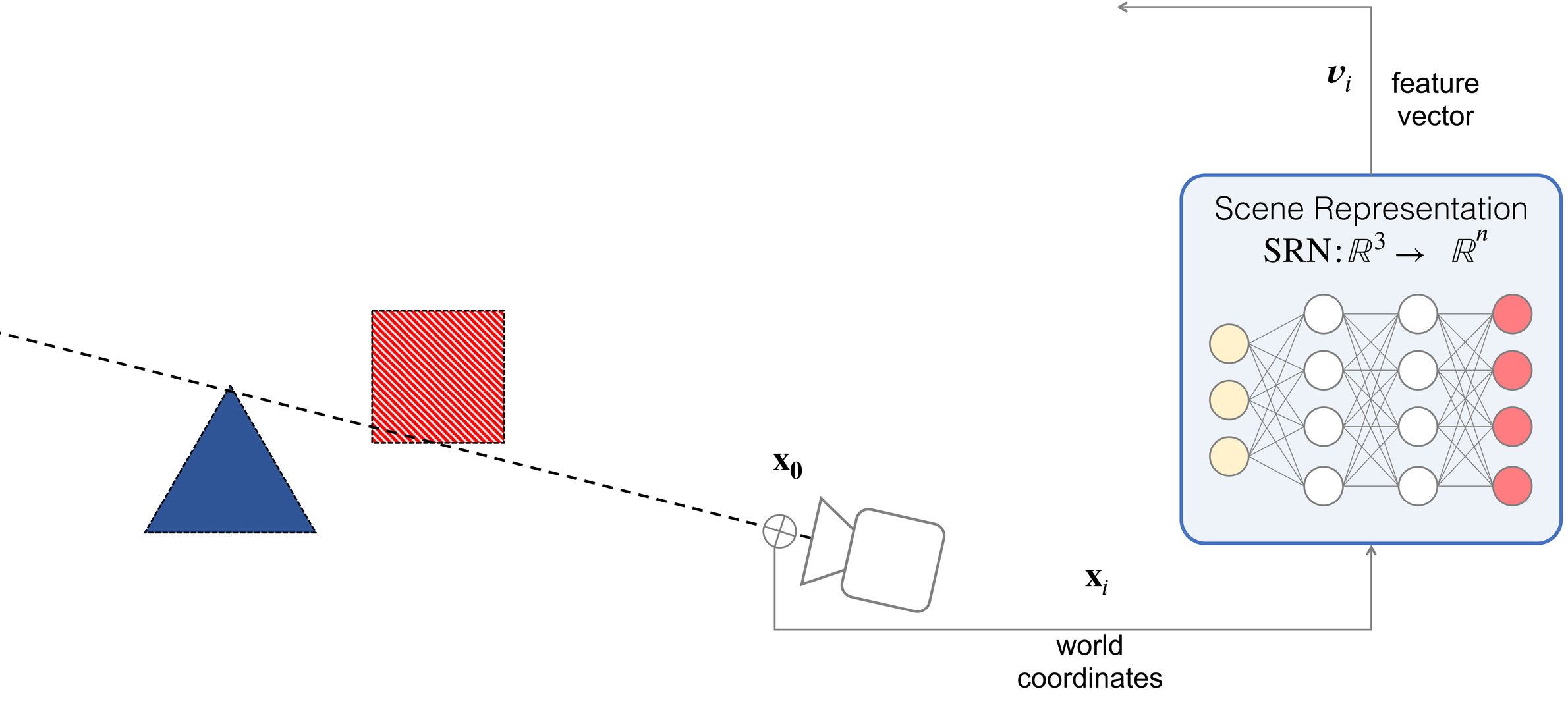
DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation, Mescheder et al.

Sphere Tracing Step 1: Intersection Testing.

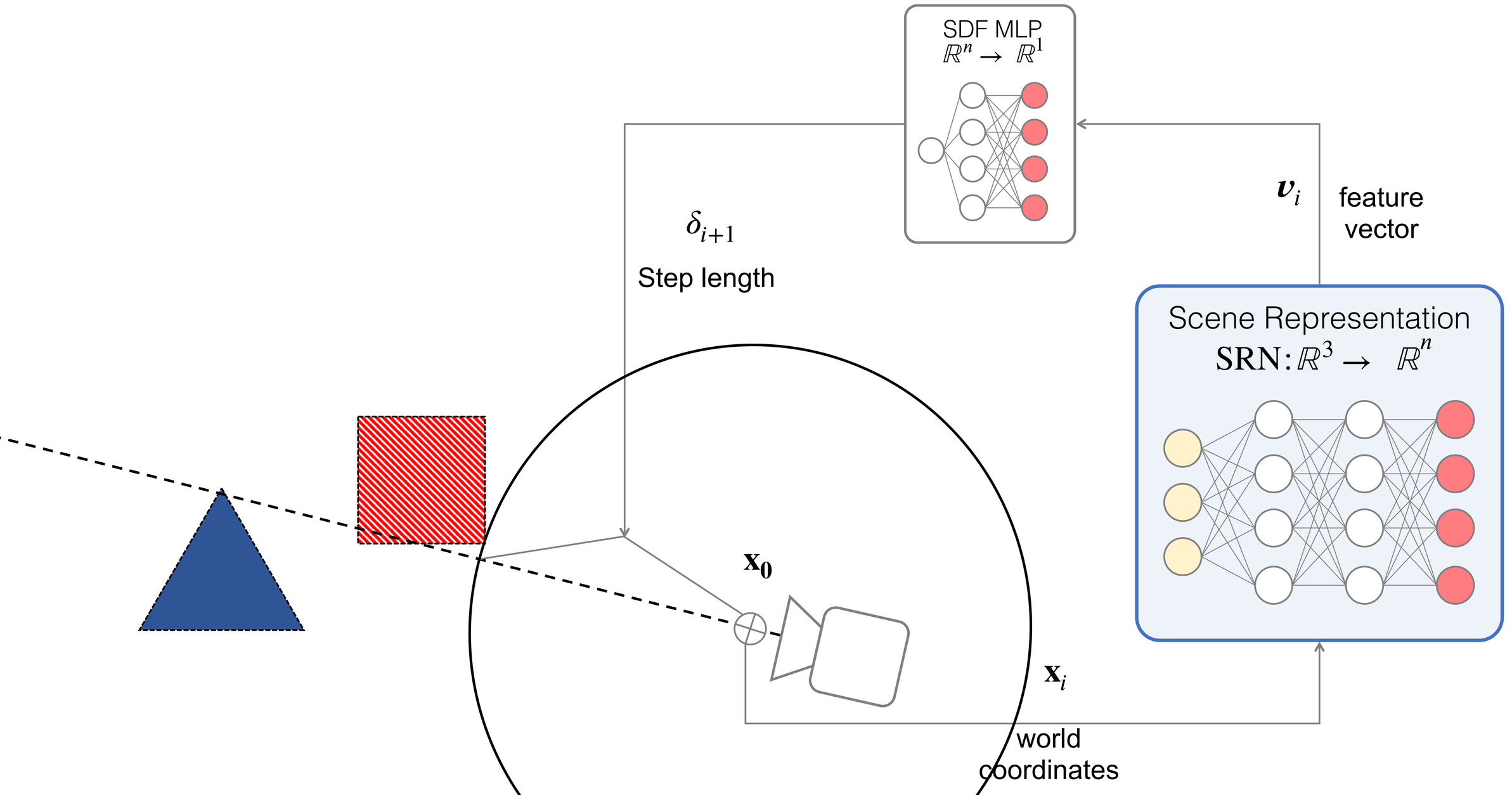
- How to find intersections?
- March along ray until we find zero-levelset.
- Naïve solution: constant steplength. Too expensive!



Sphere Tracing Step 1: Intersection Testing.

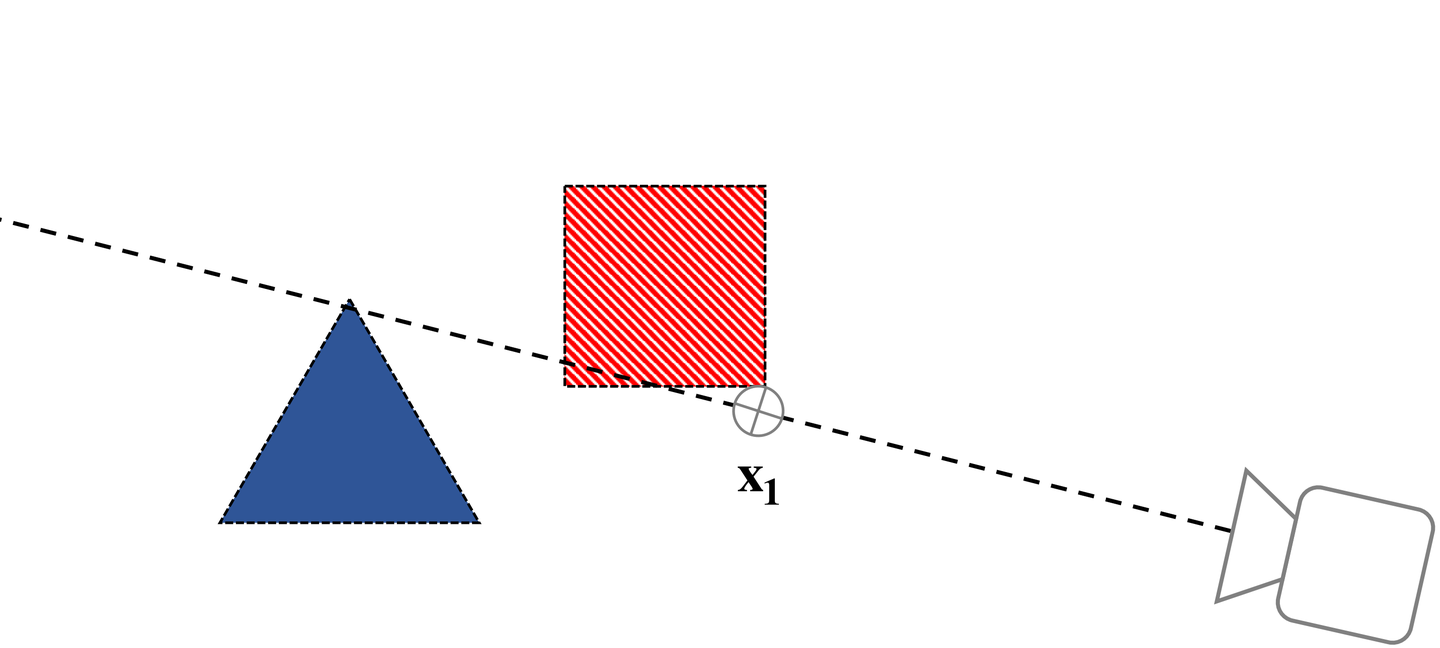


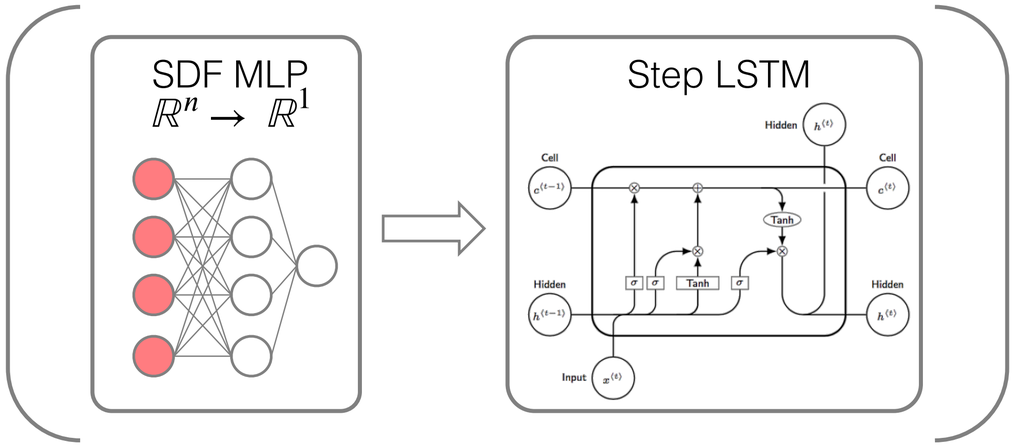
Sphere Tracing Step 1: Intersection Testing.



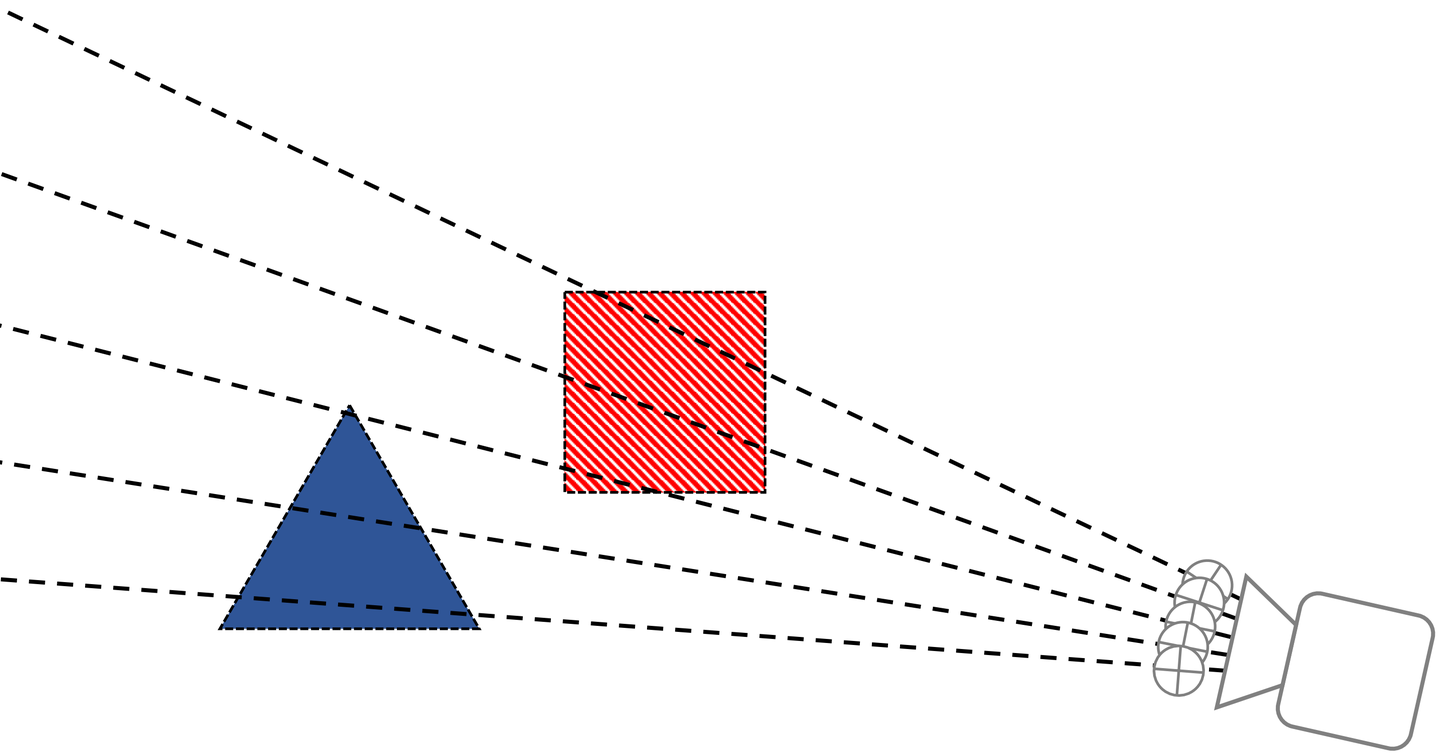
Sphere Tracing Step 1: Intersection Testing.

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

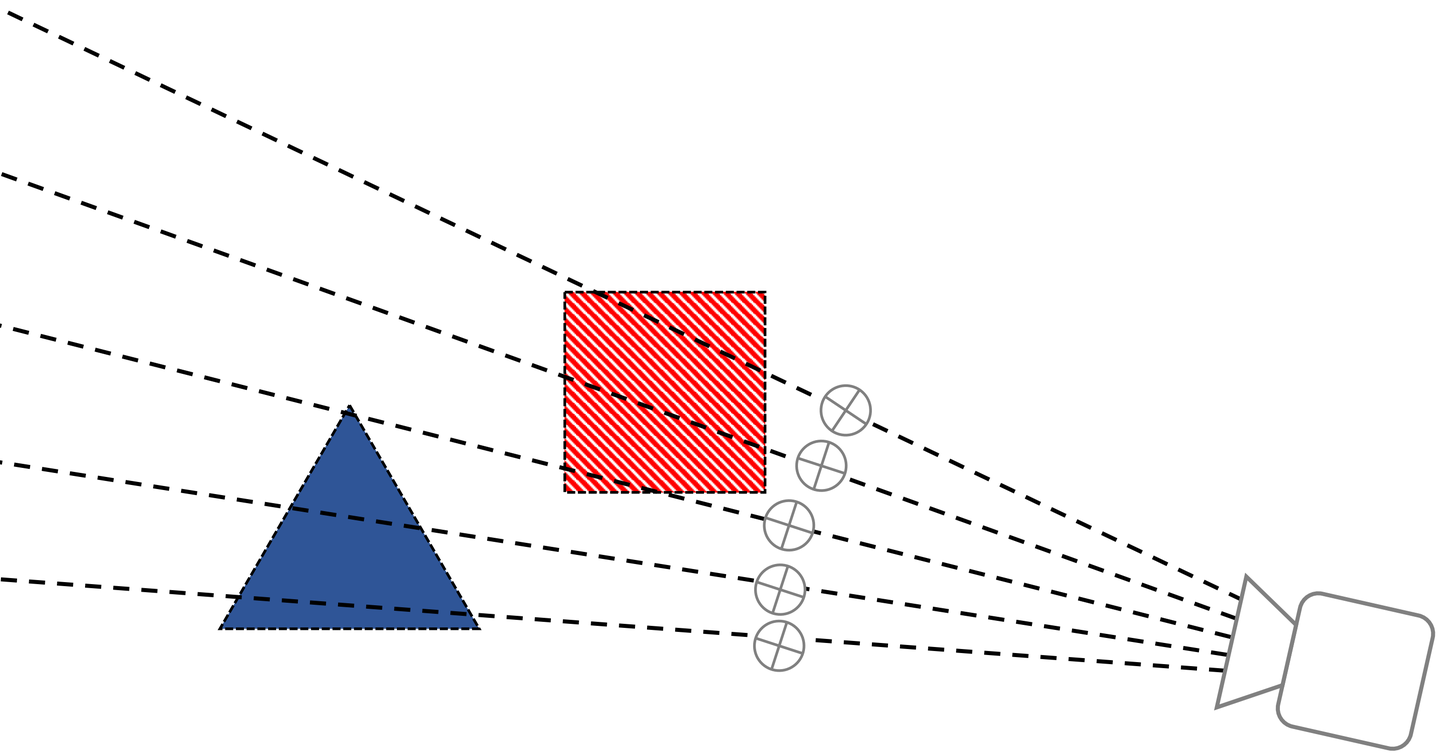




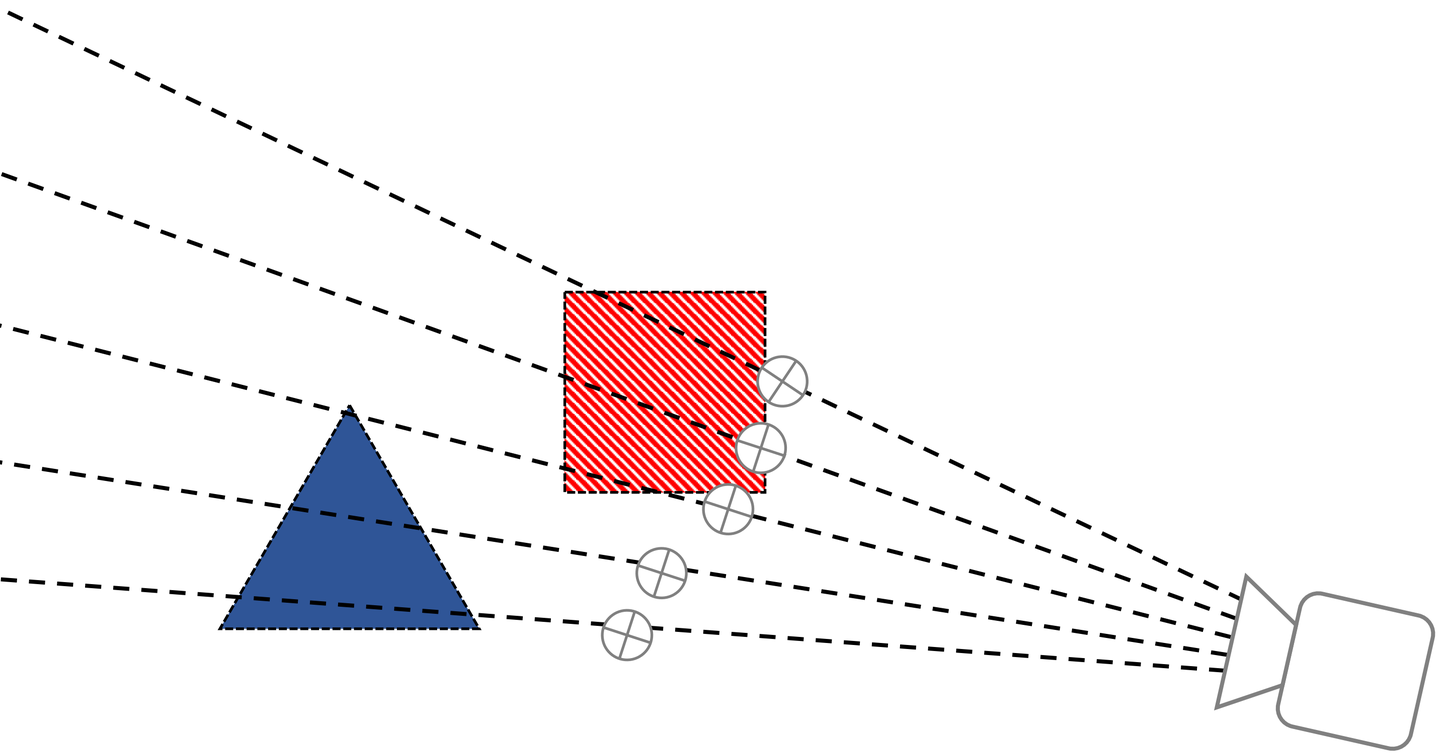
Sphere Tracing Step 1: Intersection Testing.



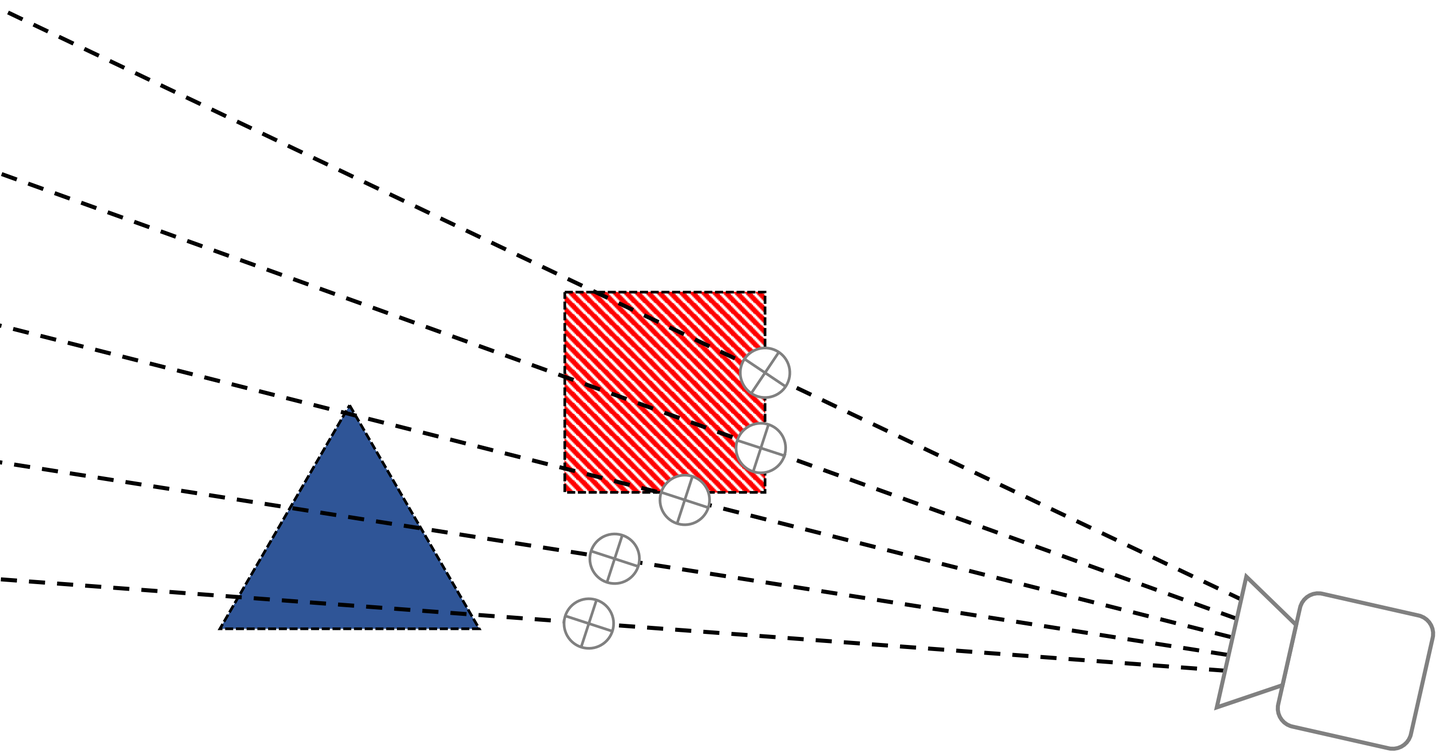
Sphere Tracing Step 1: Intersection Testing.



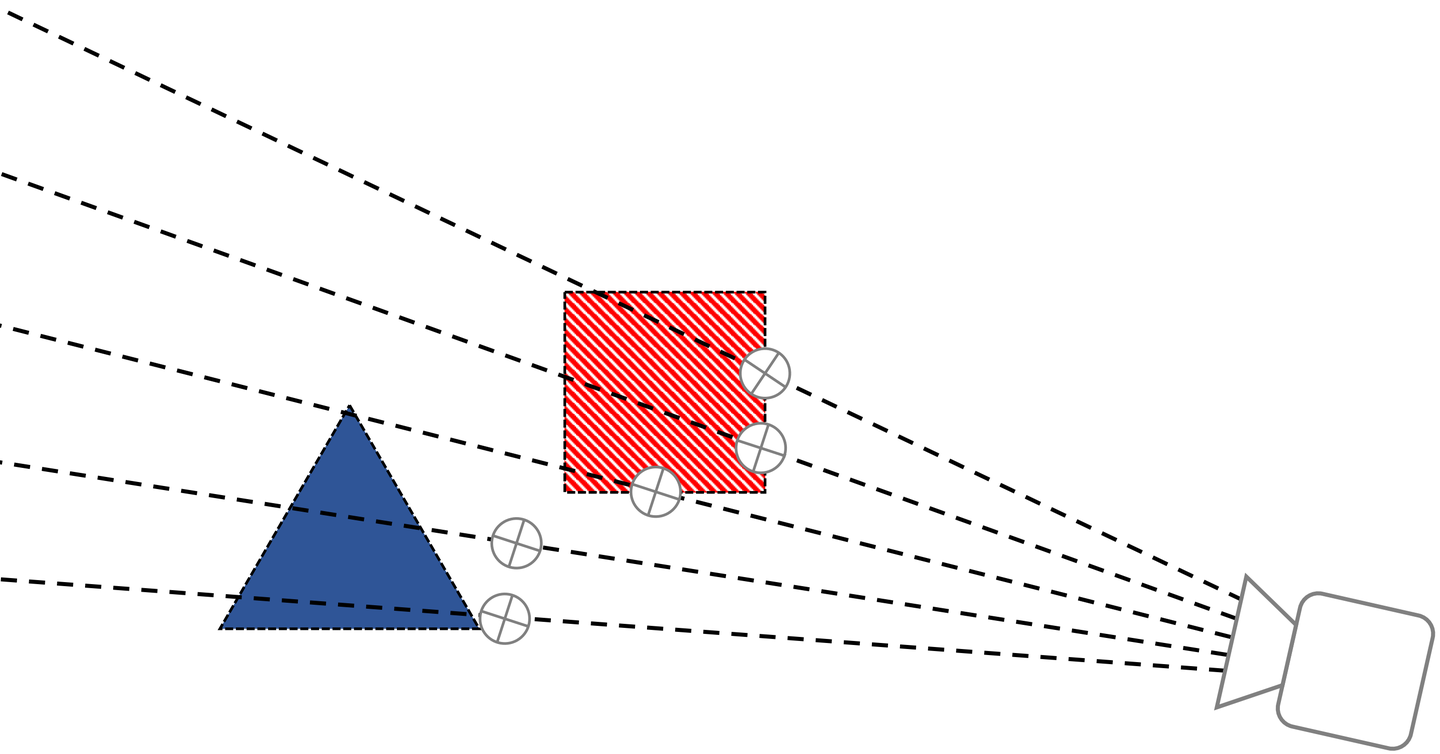
Sphere Tracing Step 1: Intersection Testing.



Sphere Tracing Step 1: Intersection Testing.

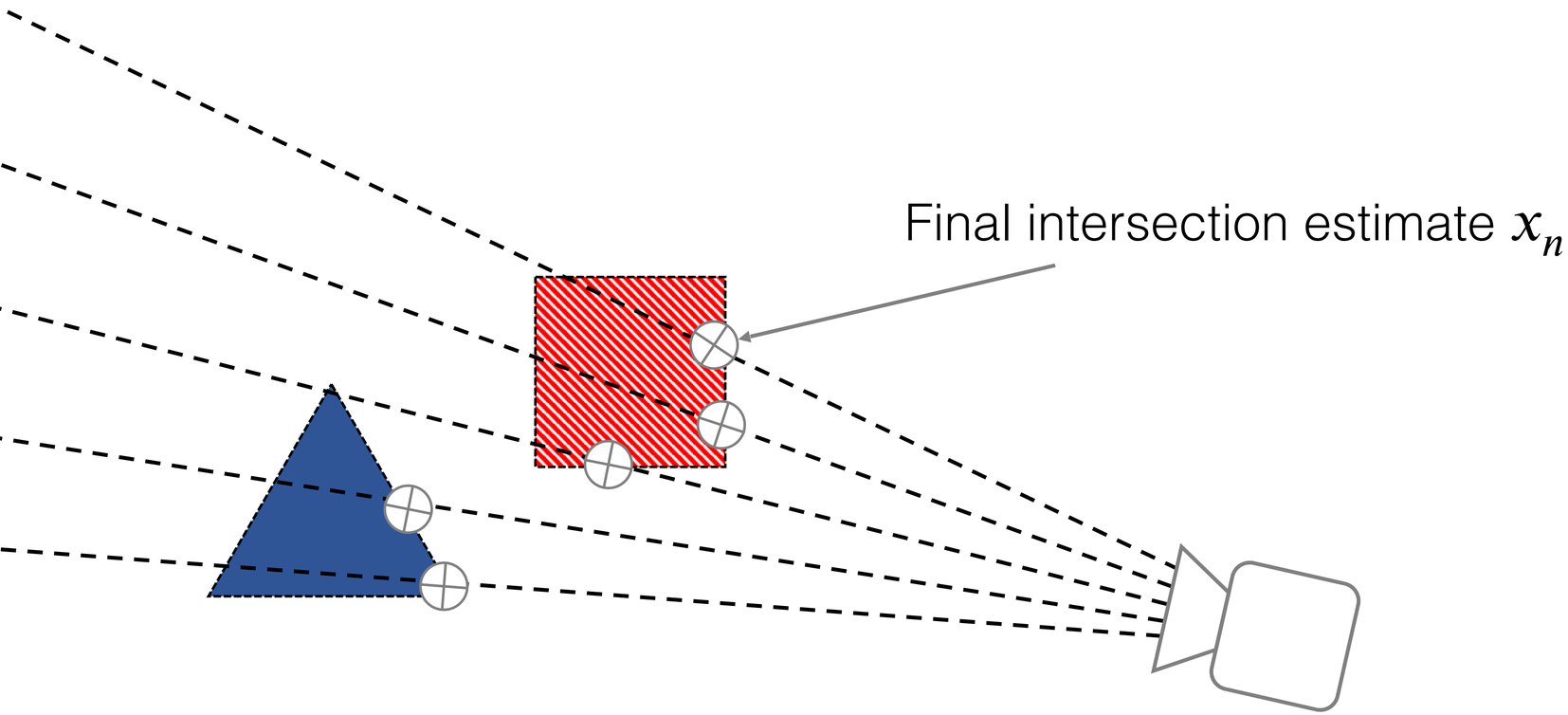


Sphere Tracing Step 1: Intersection Testing.

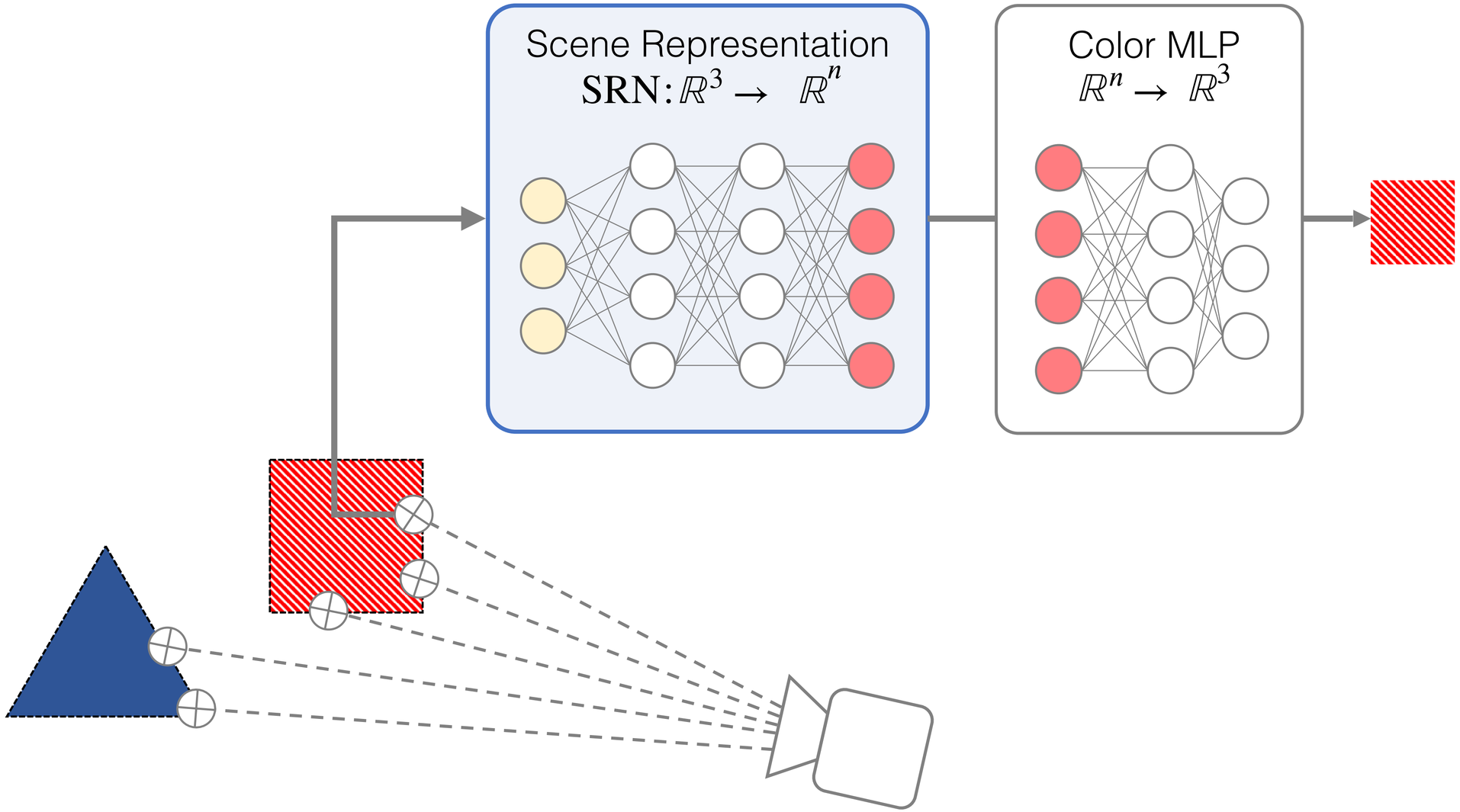


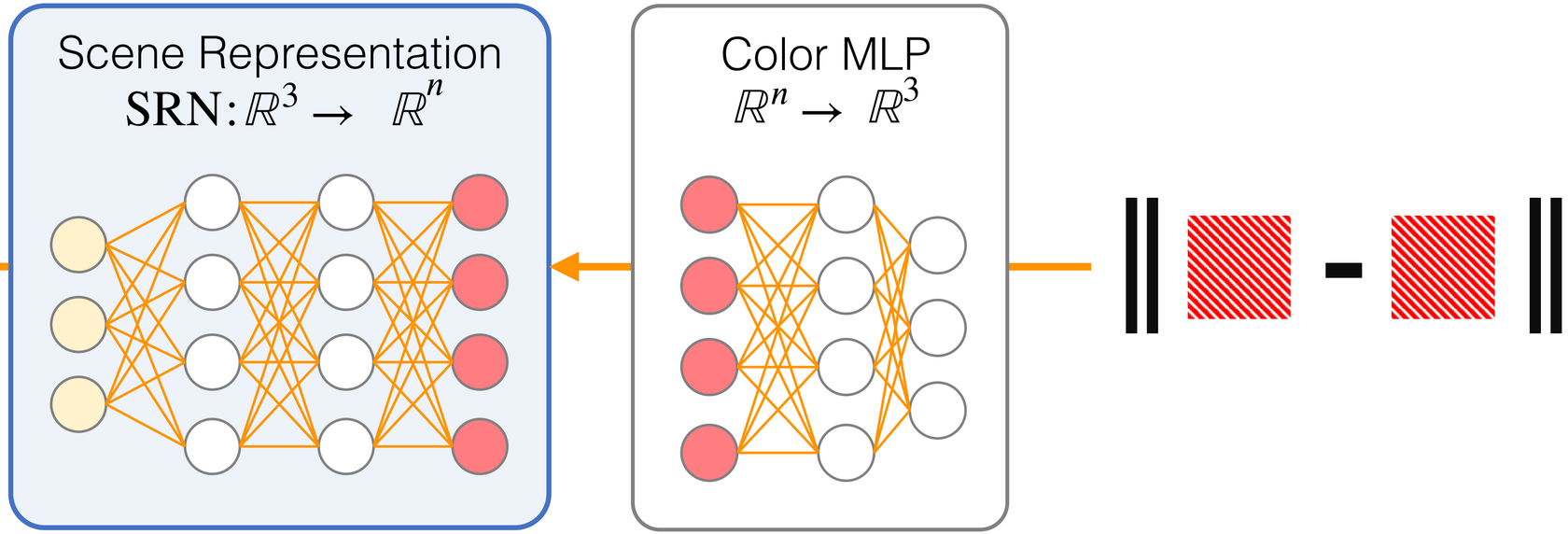
Sphere Tracing Step 1: Intersection Testing.

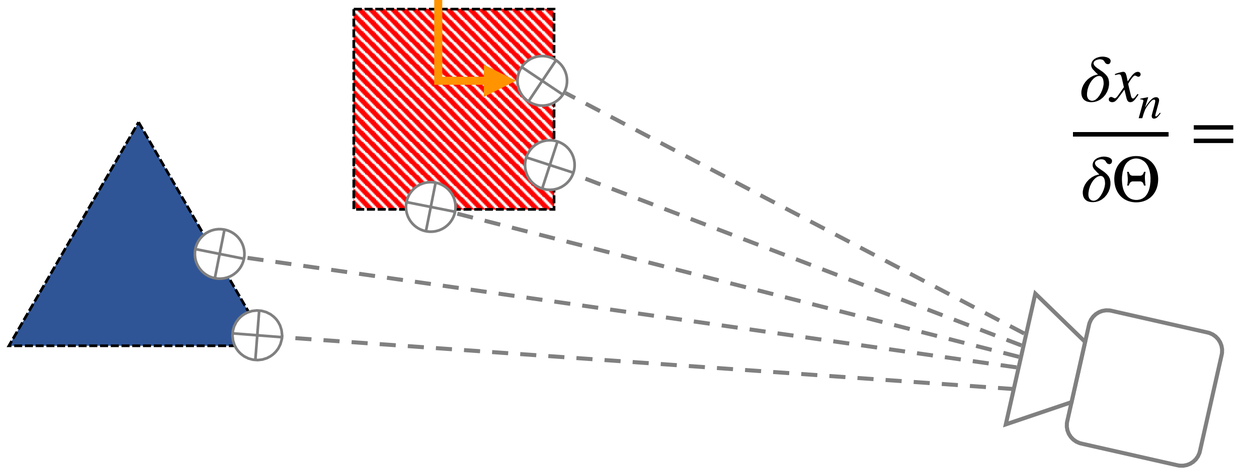
- Length of each ray yields the depth map!



Neural Renderer Step 2: Color Generation







$$\frac{\delta x_n}{\delta \Theta} = \frac{\delta x_n - 1}{\delta \Theta} + \frac{SDF(SRN(x_{n-1}))}{\delta \Theta} = \cdots$$

- End-to-end differentiable!
- But: Not obvious how these gradients manifest zero-levelsets.

Questions?

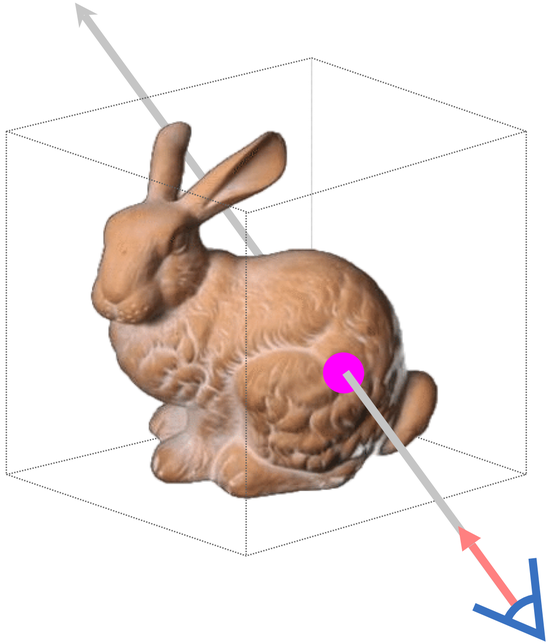
Further reading on computing gradients of sphere-tracing based renderers

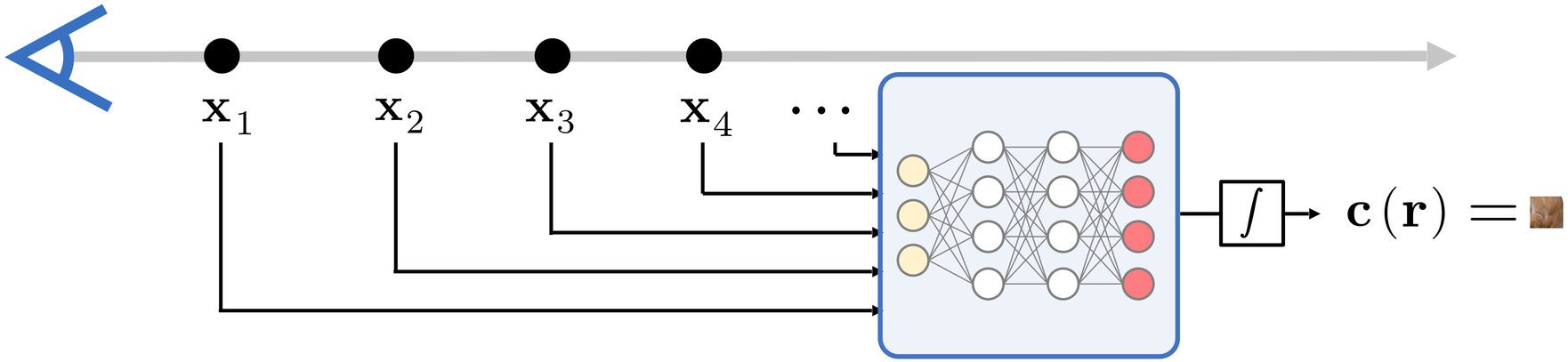
Differentiable Rendering of Neural SDFs through Reparameterization
Bangaru et al. 2022

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

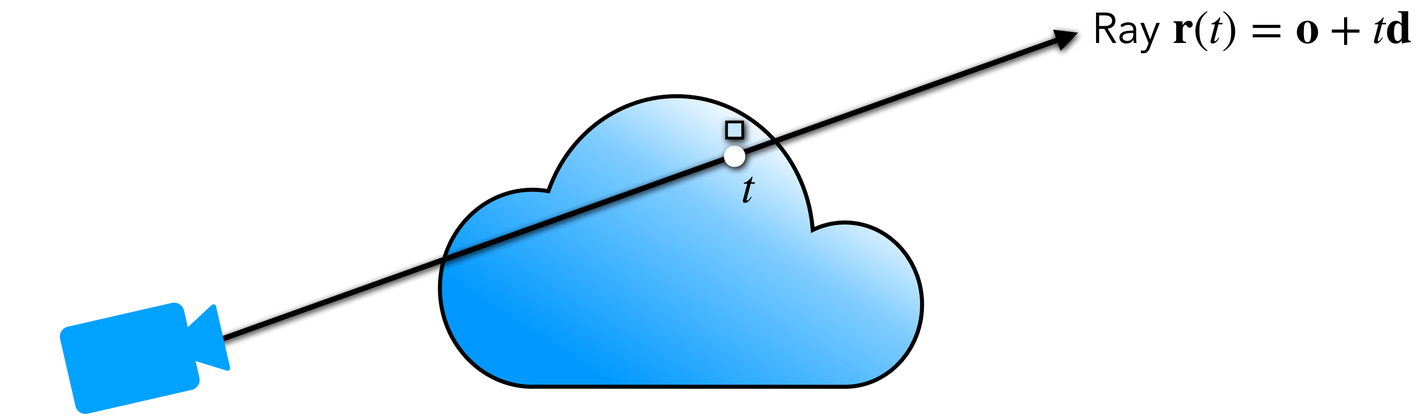
Scene is a cloud of tiny colored particles

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

Neural Radiance Fields, Mildenhall et al. 2021

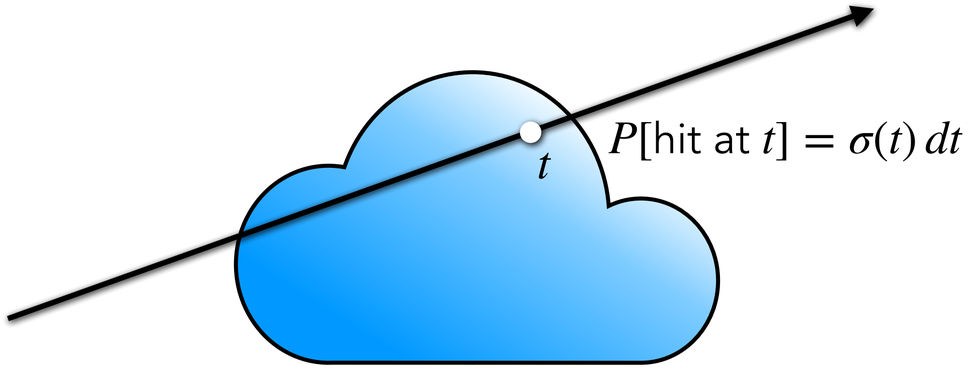
Slide credit: Ben Mildenhall

Volume Rendering



If a ray traveling through the scene hits a particle at t, we return its color $\mathbf{c}(t)$

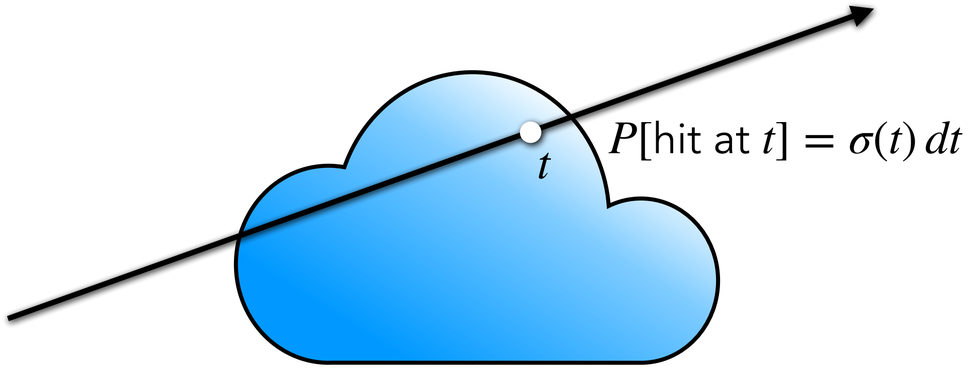
Volume Rendering



Probability that ray stops in a small interval around t is $\sigma(t)\,dt$.

σ is known as the **Volume Density.**

Volume Rendering



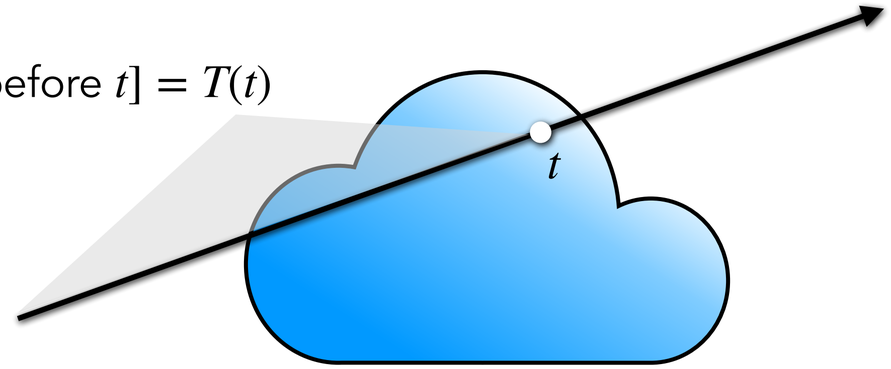
Probability that ray stops in a small interval around t is $\sigma(t)\, dt$.
σ is known as the **Volume Density.**

Our field thus has the function signature:

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

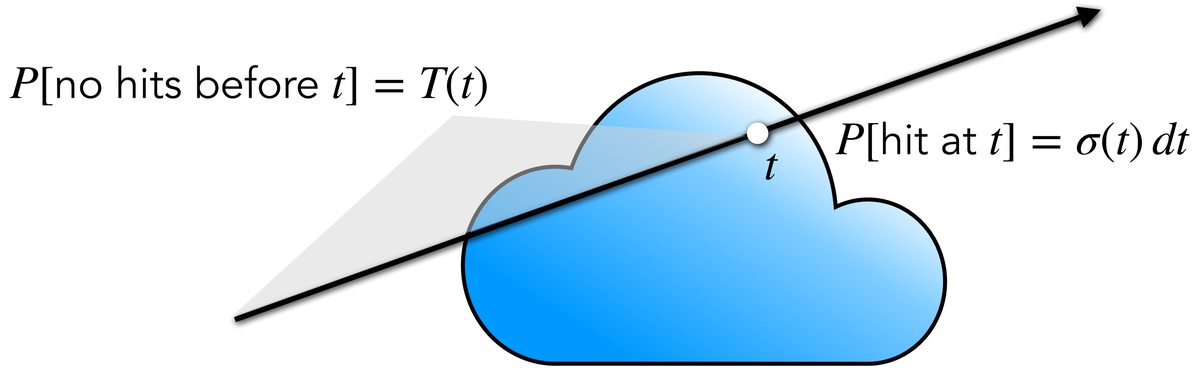
Volume Rendering

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



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.**

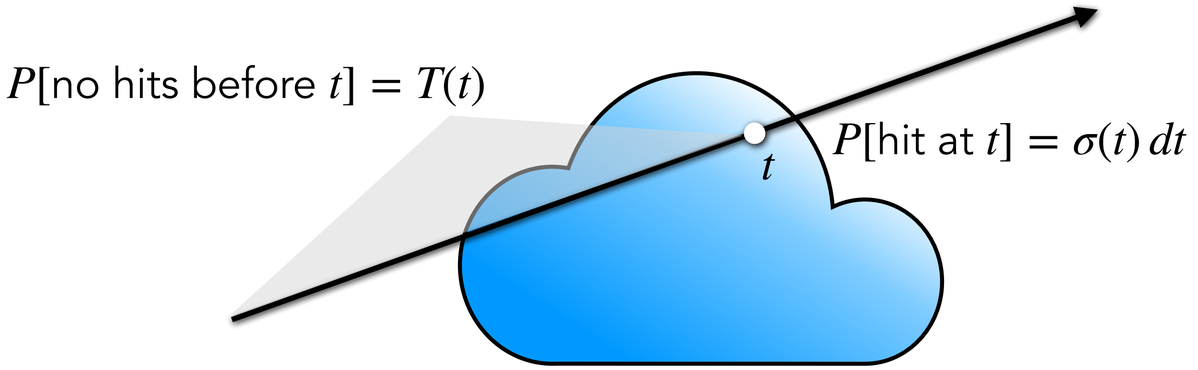
Volume Rendering



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

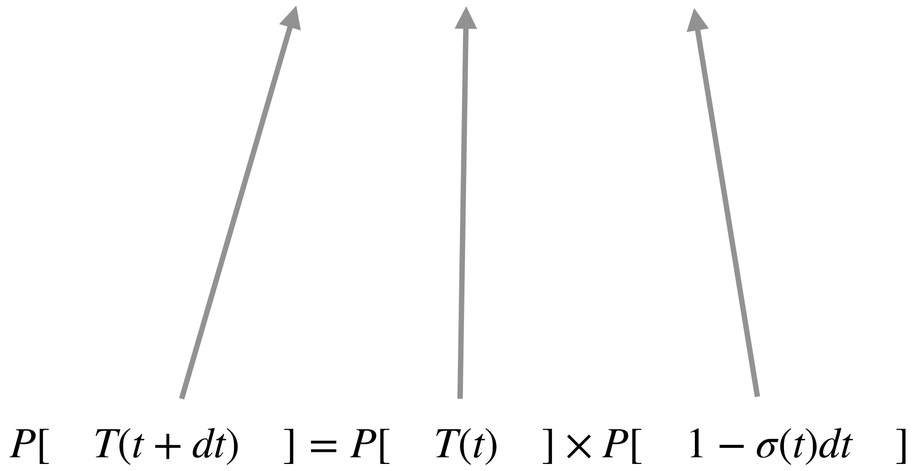


σ and T are related via

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

Volume Rendering

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



Volume Rendering

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

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

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

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

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

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

Neural Radiance Fields, Mildenhall et al. 2021

Volume Rendering

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

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

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

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

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

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

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

Volume Rendering

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

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

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

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

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

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

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

Neural Radiance Fields, Mildenhall et al. 2021

Volume Rendering

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

Neural Radiance Fields, Mildenhall et al. 2021

Volume Rendering

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

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

Neural Radiance Fields, Mildenhall et al. 2021

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 .

Neural Radiance Fields, Mildenhall et al. 2021

Questions?

Volume Rendering

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

Volume Rendering

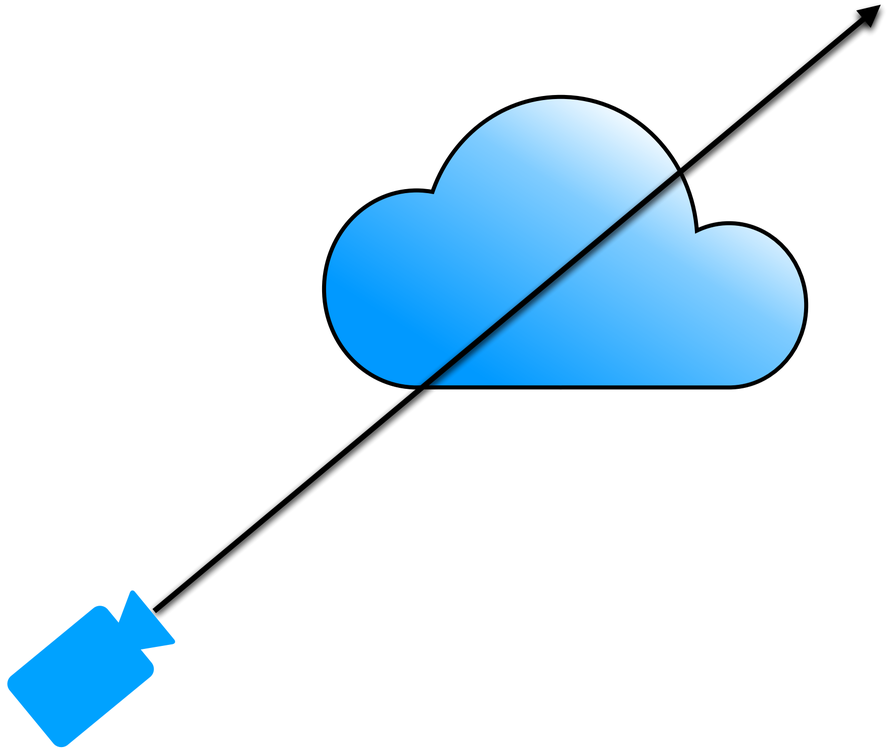
Then $T'(t) = T(t)\sigma(t)$ is probability that ray stops *exactly* at t .

So the expected color returned by the ray will be

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

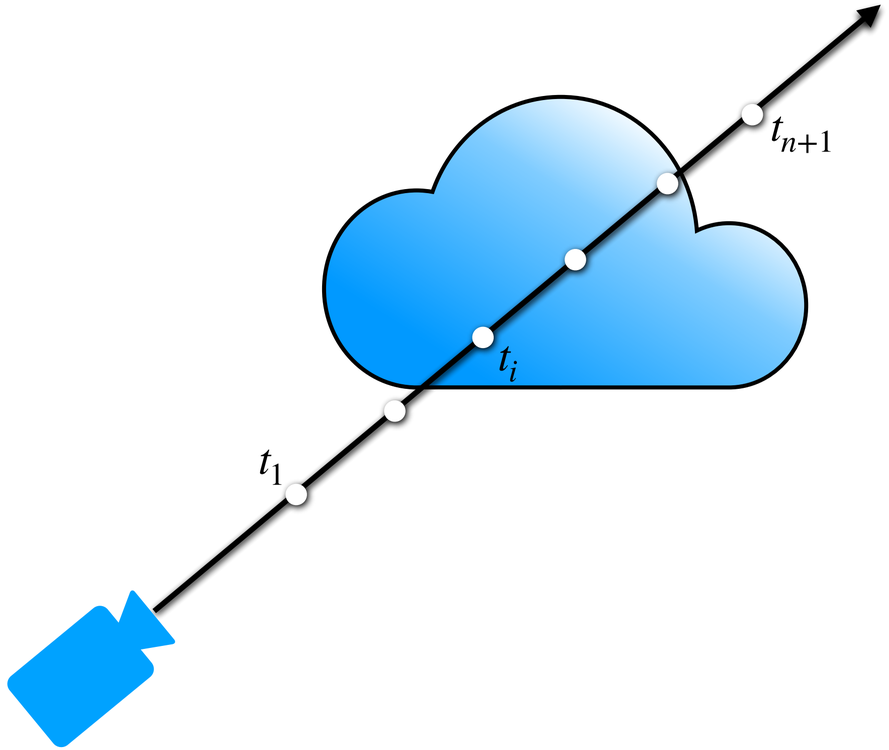
Note the nested integral!

Approximating the nested integral



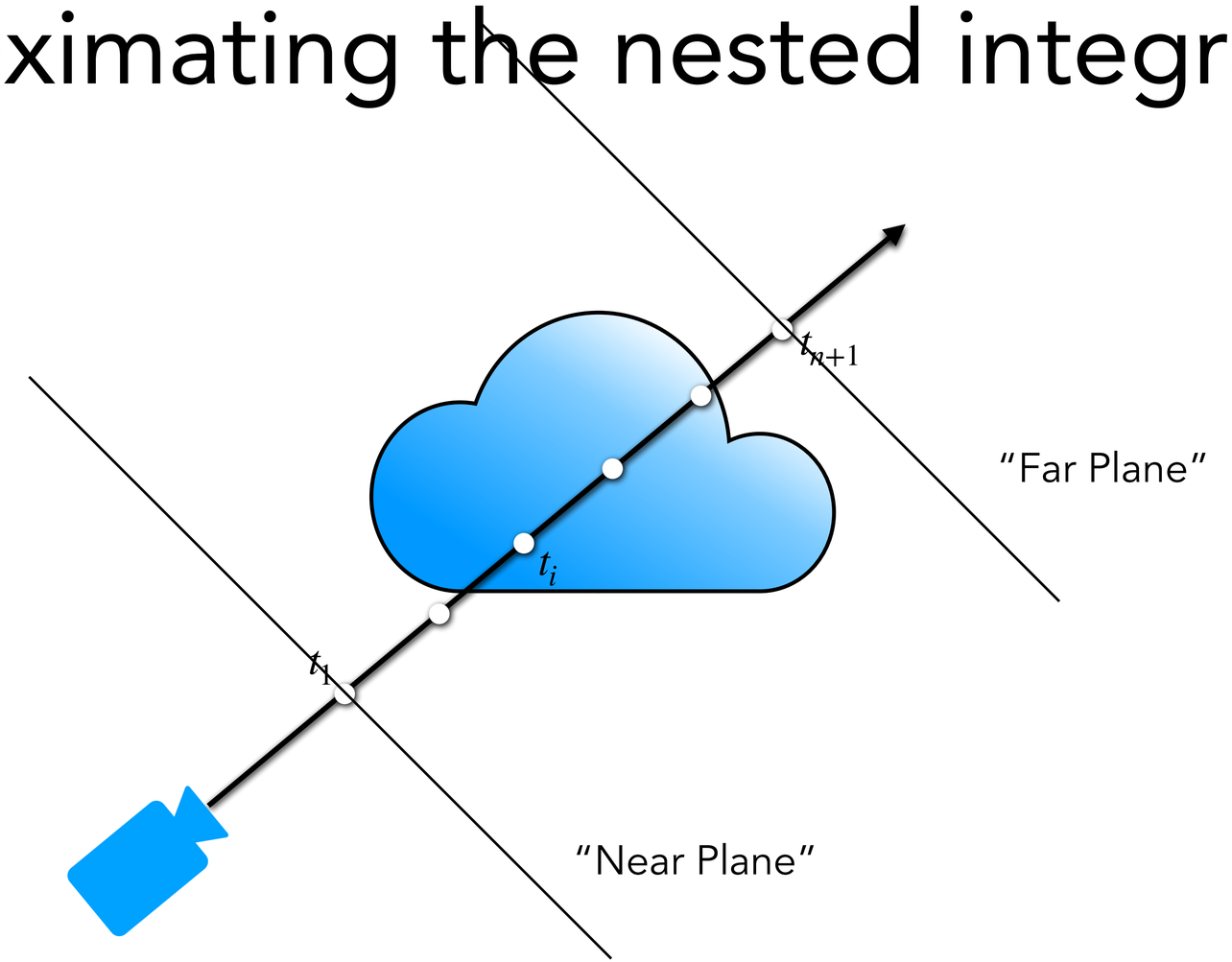
We use quadrature to approximate the nested integral,

Approximating the nested integral



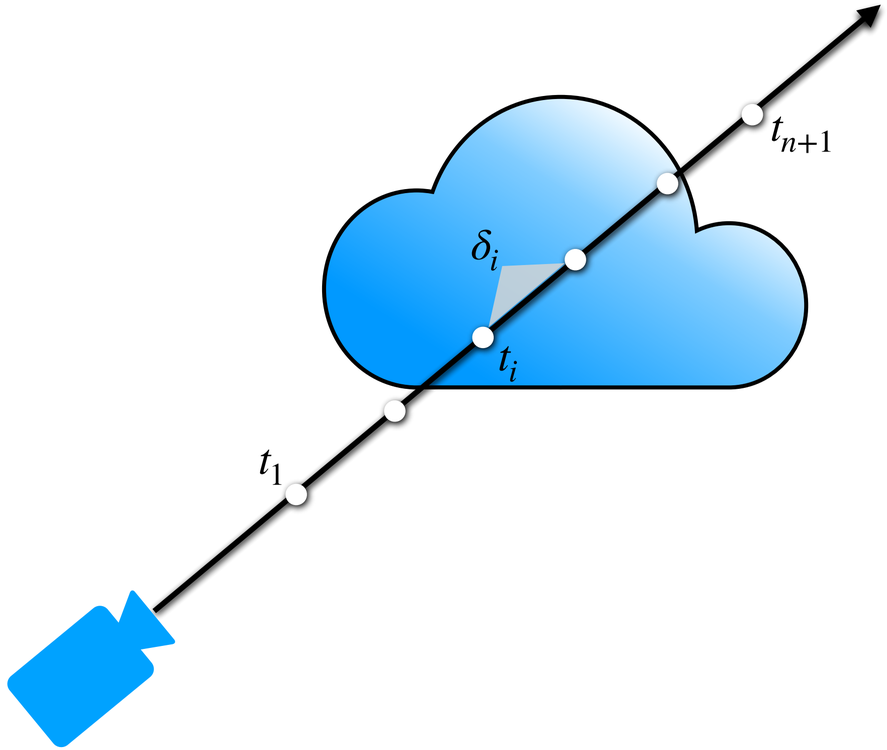
We use quadrature to approximate the nested integral,
splitting the ray up into n segments with endpoints $\{t_1, t_2, \ldots, t_{n+1}\}$

Approximating the nested integral



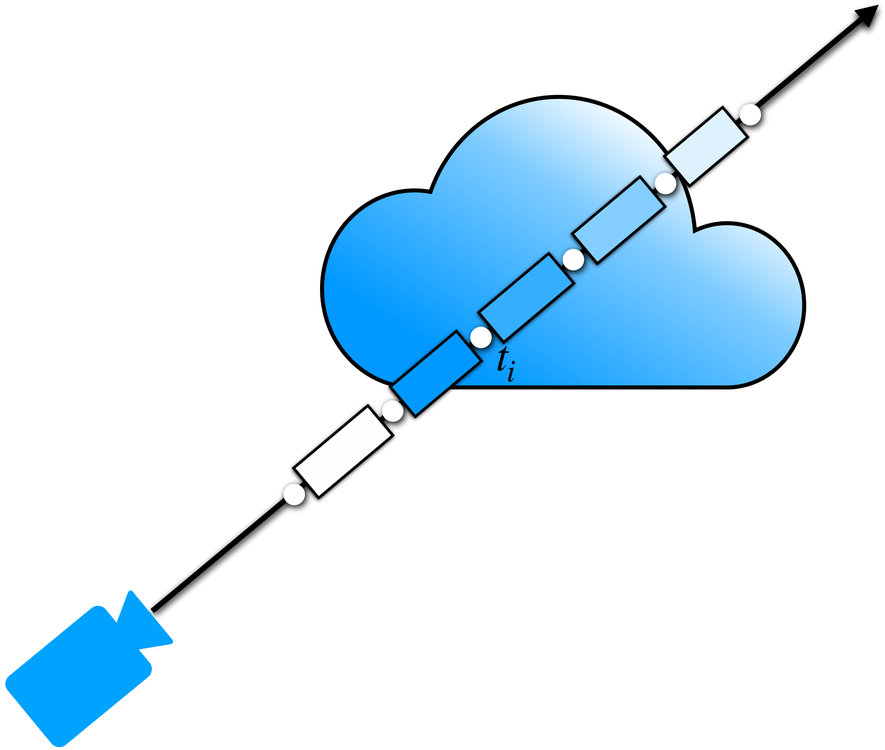
We use quadrature to approximate the nested integral,
splitting the ray up into n segments with endpoints $\{t_1, t_2, \ldots, t_{n+1}\}$

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

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

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

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

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

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

"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?"

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

Neural Radiance Fields, Mildenhall et al. 2021

Slide credit: Ben Mildenhall

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

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

Summary: volume rendering integral estimate

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

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

colors

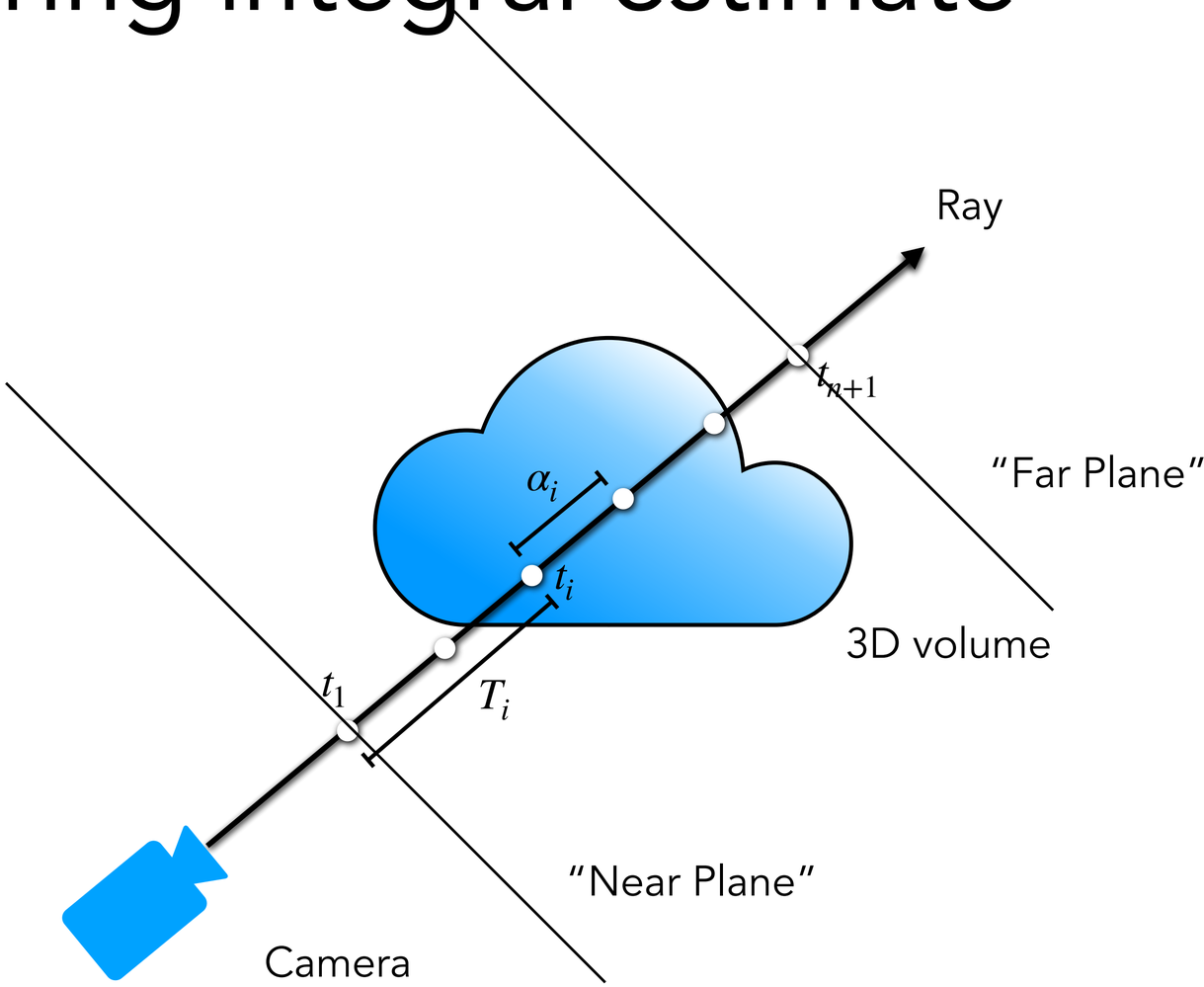
weights

How much light is blocked earlier along ray:

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

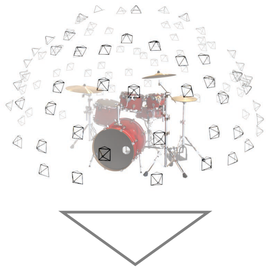
How much light is contributed by ray segment *i*:

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



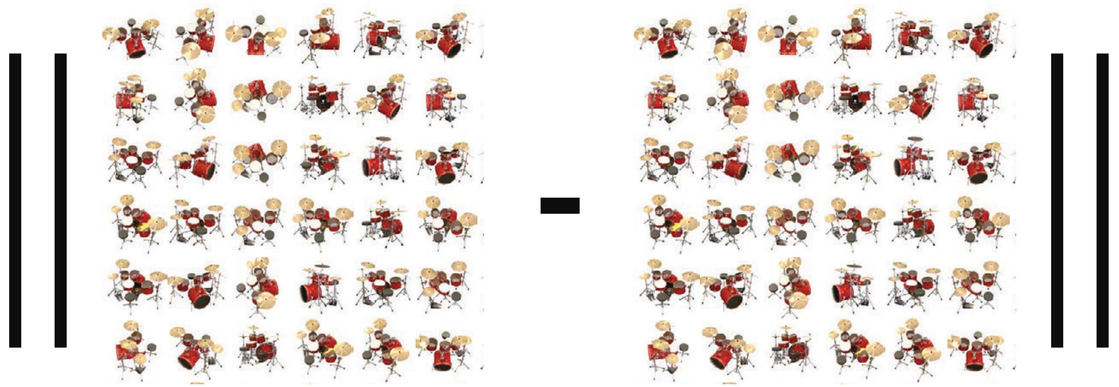
Questions?

Differentiable Renderer is done!



Scene Representation

Differentiable Renderer

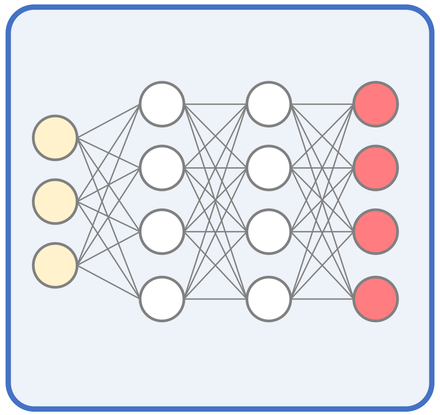


Rendered Images

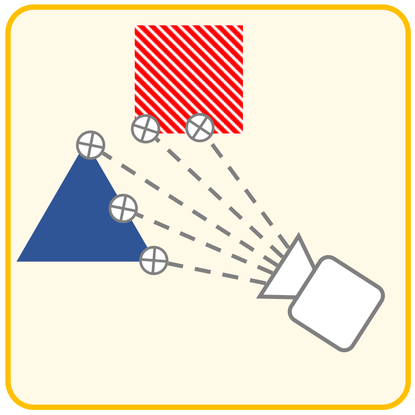
GT Images

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.

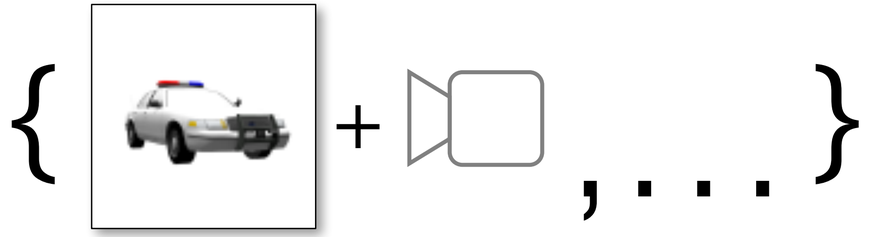


Scene Representation



Neural Renderer

Optimization



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

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

Novel View Synthesis

Normal map RGB

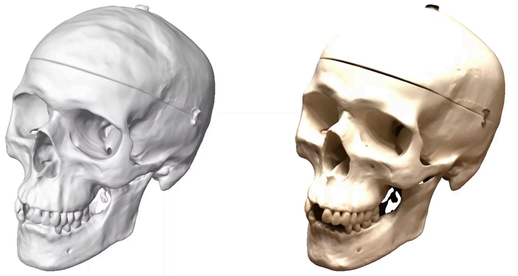
Questions?

The explosion of neural implicit representations in inverse graphics

Scene Representation Networks
Sitzmann et al. 2019
First differentiable renderer for implicit representations.
Focused on single-image reconstruction.

2019

2020

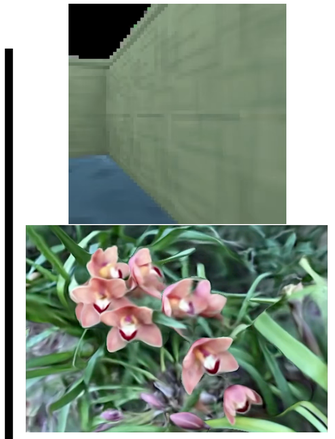


IDR
Yariv et al. 2020
Improved ray-marcher.
View dependence.
Focused on single-scene reconstruction

NeRF
Mildenhall et al. 2020
Alternative diff. renderer.
High-frequency detail.
View dependence.
Focused on single-scene reconstruction.

PixelNeRF
Yu et al. 2020
Forward-pass inference via convnet.
Focused on single-image reconstruction.

The explosion of neural implicit representations in inverse graphics

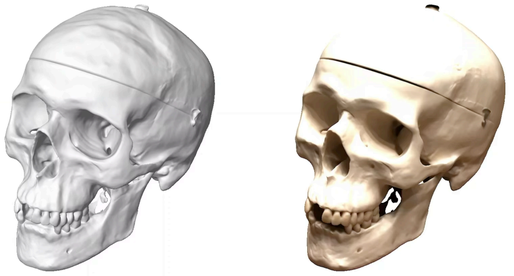


2019

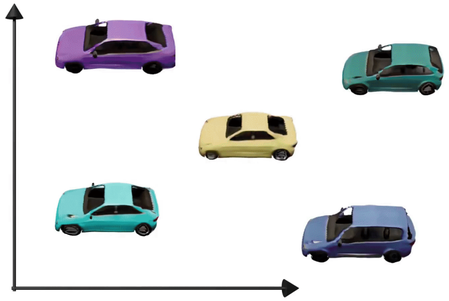
2020

Sitzmann et al. 2020

Yariv et al. 2020



Schwarz et al. 2020



Mildenhall et al. 2020

Chan et al. 2020

Yu et al. 2020

Lin et al. 2020

3D Inverse Graphics Taxonomy

Field Parameterization

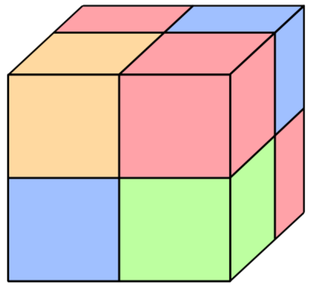
Geometry & Appearance Model

Forward Map

Inference Method

DeepVoxels (CVPR 2018)

Field Parameterization



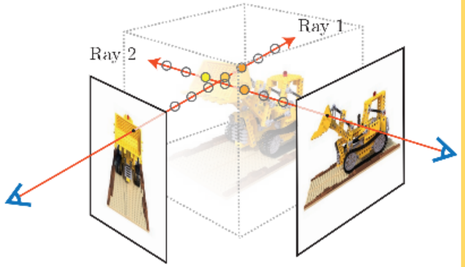
Geometry & Appearance Model

$$\Phi(\mathbf{x}) = \mathbf{f}$$

$$\mathbf{f} \in \mathbb{R}^n$$

Feature-based Representation

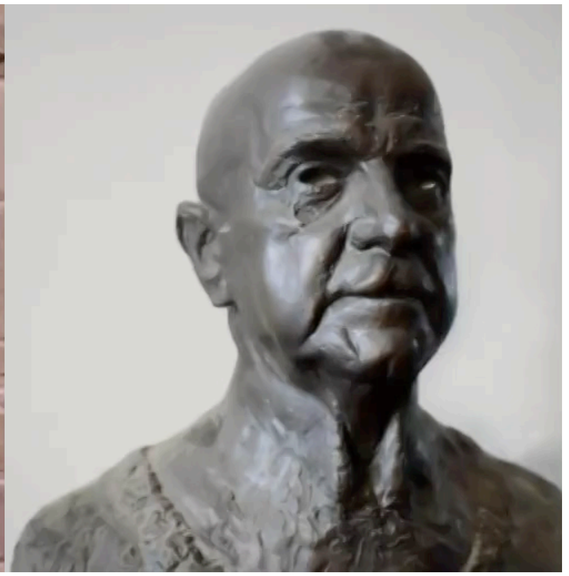
Forward Map



Volumetric Sampling, Softmax Attention

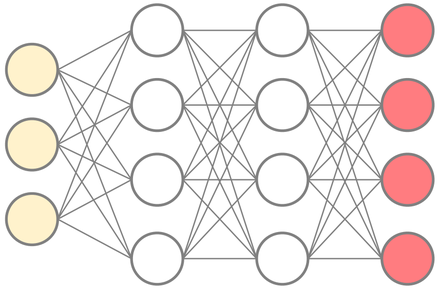
Inference Model

Per-Scene Optimization

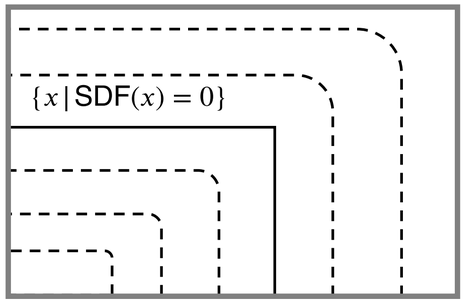


- Reasonable quality overfitting
- Fast rendering
- Noisy geometry
- Requires Many Images

Scene Representation Networks (NeurIPS 2019)



Field Parameterization



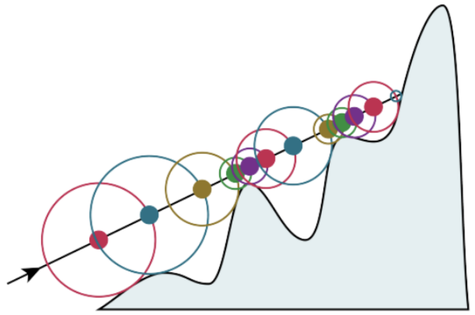
SDF

$$x \mapsto C$$

View-independent Radiance

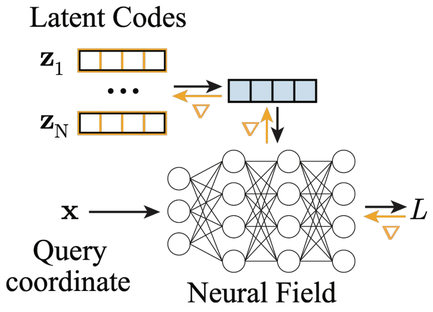
Geometry & Appearance Model

LSTM-based sphere tracing



Forward Map

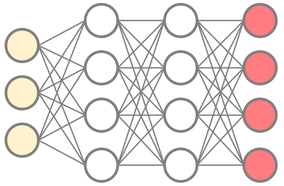
Auto-Decoding or Per-Scene Optimization



Inference Model

Scene Representation Networks (NeurIPS 2019)

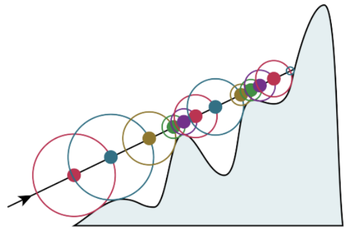
Field Parameterization



Geometry & Appearance Model

SDF
+
View-independent Radiance

Forward Map



Volume Rendering

Inference Model

Auto-Decoding
OR
Per-Scene Optimization

Overfitting

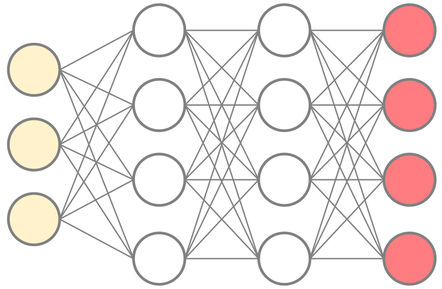
Single Image

- 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)



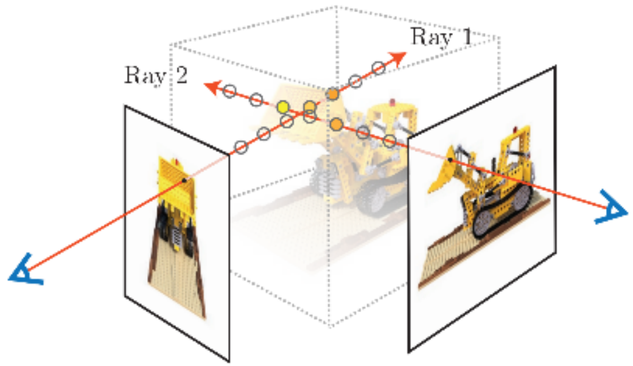
Field Parameterization

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

View-Dependent Density + Radiance

Geometry & Appearance Model

Volume Rendering



Forward Map

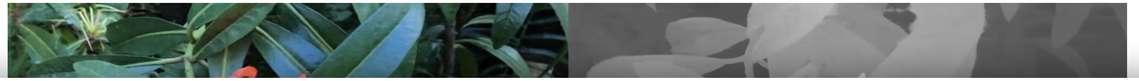
Per-Scene Optimization

Inference Model

NeRF (Mildenhall et al. ECCV 2020)

Field

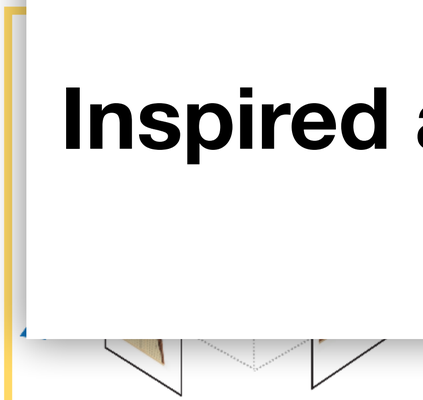
Geometry &



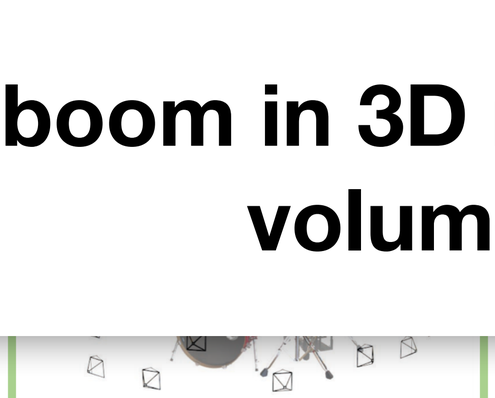
Great out-of-the-box performance
Great looking results for single-scene reconstruction
Ben donated slides ;)

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

ic!



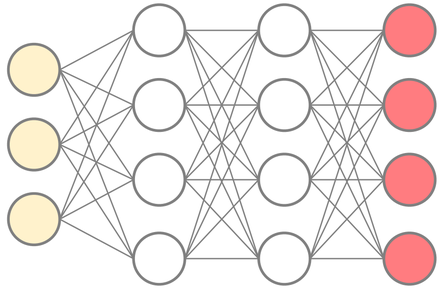
Volume Rendering



Per-Scene Optimization

- Noisy or Cloudy Geometry
- Requires Many Images

NeRF (Mildenhall et al. ECCV 2020)



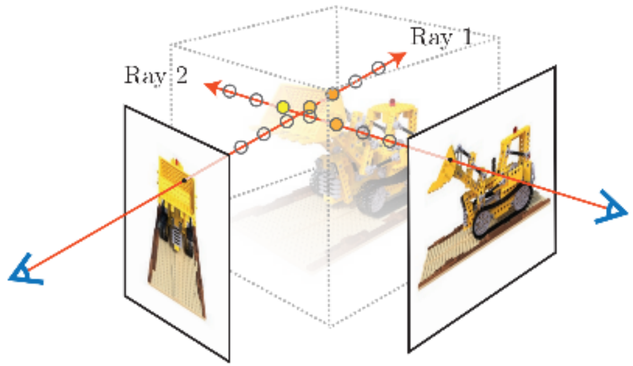
Field Parameterization

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

View-Dependent Density + Radiance

Geometry & Appearance Model

Volume Rendering

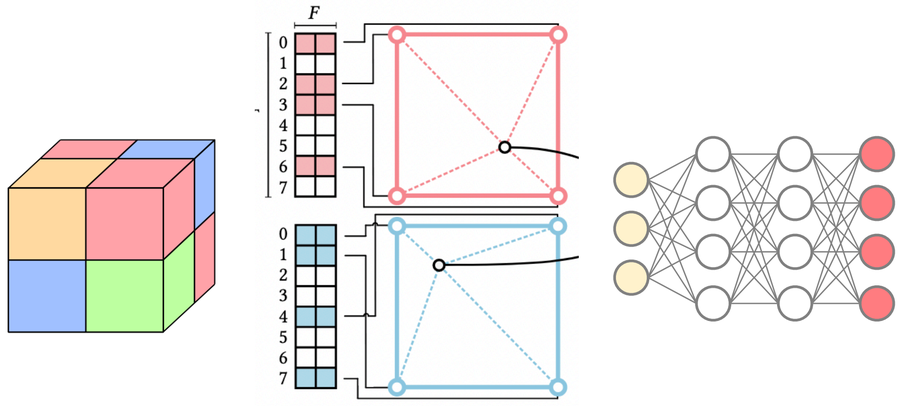


Forward Map

Per-Scene Optimization

Inference Model

Faster Radiance Fields w/ Hybrid Data Structures



Field Parameterization

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

View-Dependent Density + Radiance

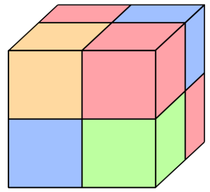
Geometry & Appearance Model

Volume Rendering

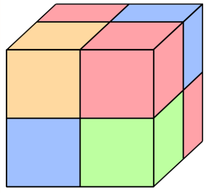
Forward Map

Per-Scene Optimization

Inference Model



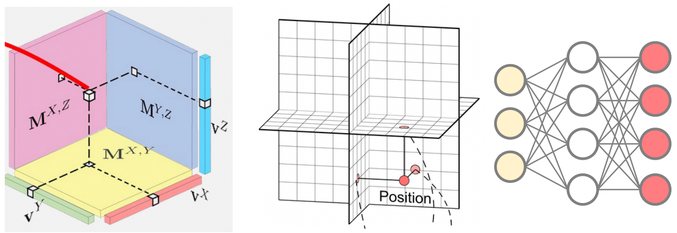
Plenoxels: Radiance Fields …
[Yu et al. 2022]
Direct Voxel Grid Optimization
[Sun et al. 2021]



Voxel Grid

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

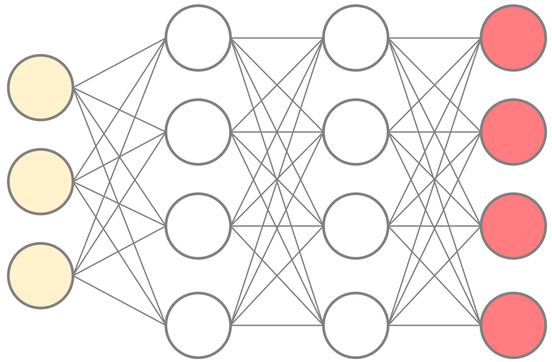
Point Cloud + MLP



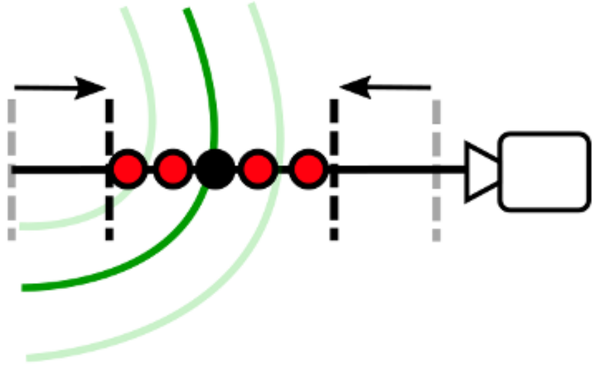
Tensor Factorization / Triplane + MLP

Efficient Geometry-aware 3D…
[Chan et al. 2022]
TensorRF: Tensor Radiance Fields
[Chen & Xu et al. 2022]

Hybrid Volumetric / Level Set Representations

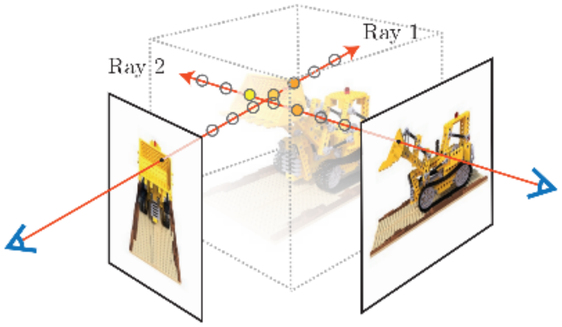


Field Parameterization



Geometry & Appearance Model

Volume Rendering



Forward Map

Per-Scene Optimization

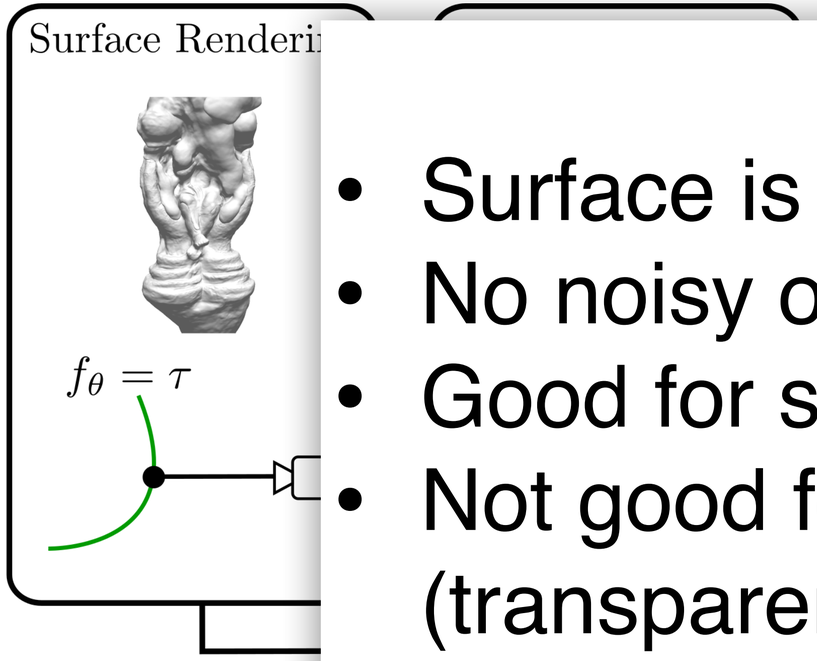
Inference Model

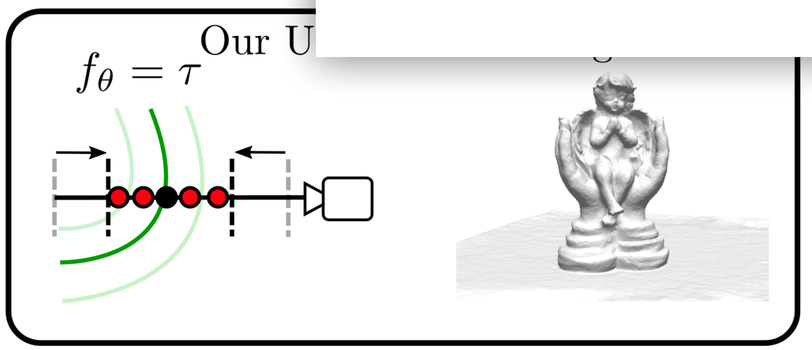
Hybrid Volumetric / Level Set Representations

UNISURF
[Oechsle et al., ICCV 2021]

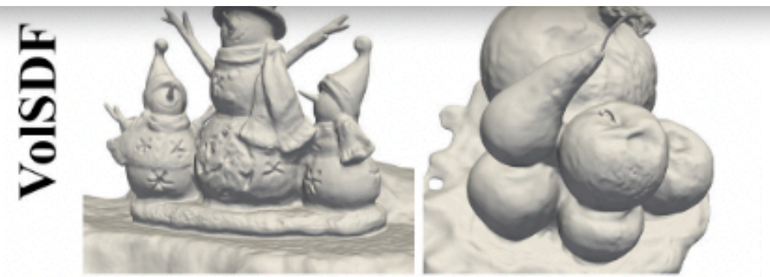
VolSDF
[Yariv et al., NeurIPS 2021]

NeuS
[Wang et al., NeurIPS 2021]

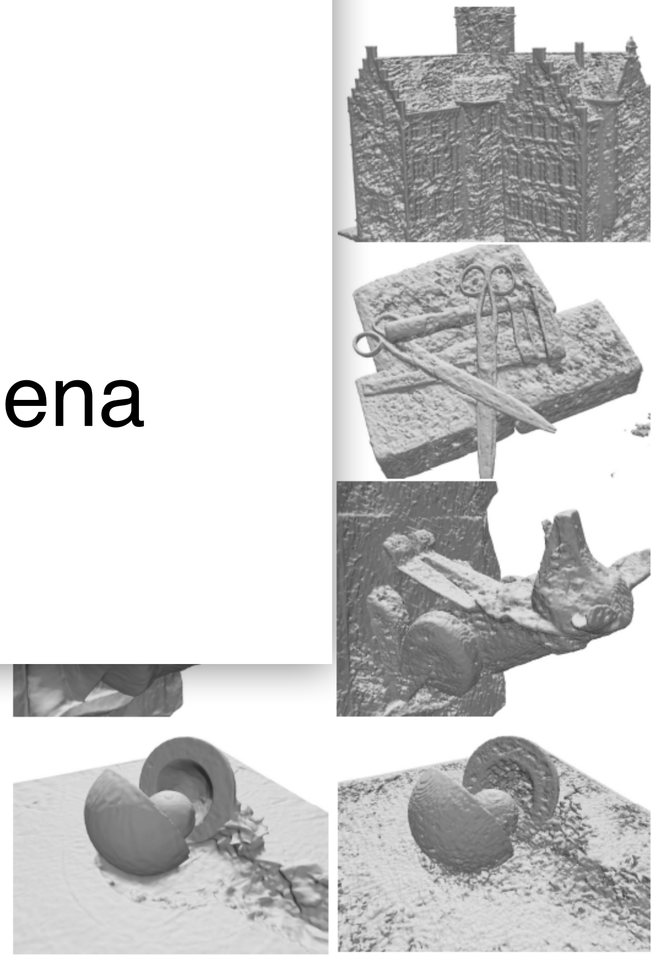




- Surface is well defined
- No noisy or cloudy geometry
- Good for solid objects
- Not good for complex scene phenomena (transparency)



NeRF



arXiv:2111.11426v4 [cs.CV] 5 Apr 2022

EUROGRAPHICS 2022
D. Meneveaux and G. Patanè
(Guest Editors)

Volume 41 (2022), Number 2
STAR – State of The Art Report

Neural Fields in Visual Computing and Beyond

Yiheng Xie[1,2] Towaki Takikawa[3,4] Shunsuke Saito[5] Or Litany[4] Shiqin Yan[1] Numair Khan[1] Federico Tombari[6,7]
James Tompkin[1] Vincent Sitzmann[8†] Srinath Sridhar[1†]

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

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

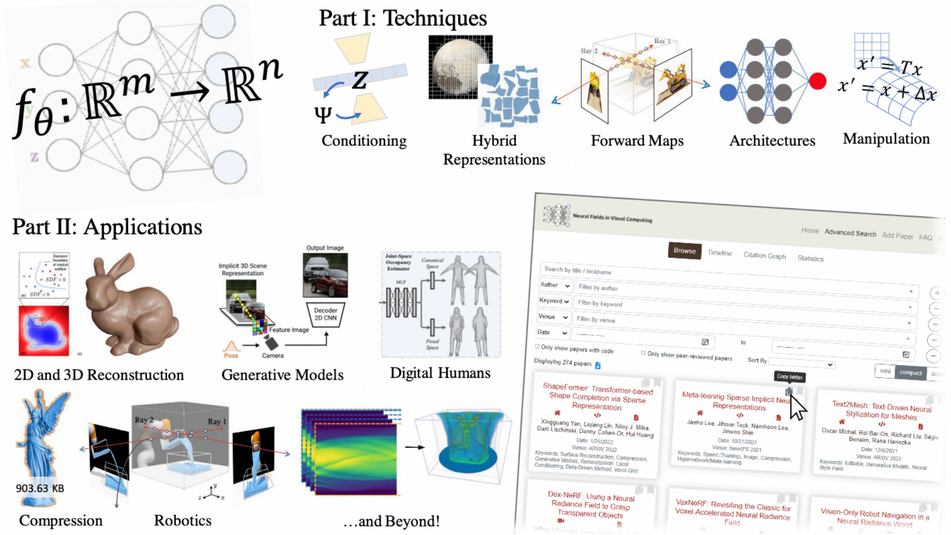


Figure 1: **Contribution of this report.** Following a survey of over 250 papers, we provide a review of (**Part I**) techniques in neural fields such as prior learning and conditioning, representations, forward maps, architectures, and manipulation, and of (**Part II**) applications in visual computing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

How far does this get us?

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

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

$$L_o(\mathbf{x}, \boldsymbol{\omega}_o, \lambda) = L_e(\mathbf{x}, \lambda)$$

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

How far does this get us?

$$\sigma = \sigma(\mathbf{x})$$
$$\mathbf{c} = \mathbf{c}(\mathbf{x}, \omega_o)$$

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

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

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

How far does this get us?

$$\sigma = \sigma(\mathbf{x})$$
$$\mathbf{c} = \mathbf{c}(\mathbf{x}, \omega_o)$$

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

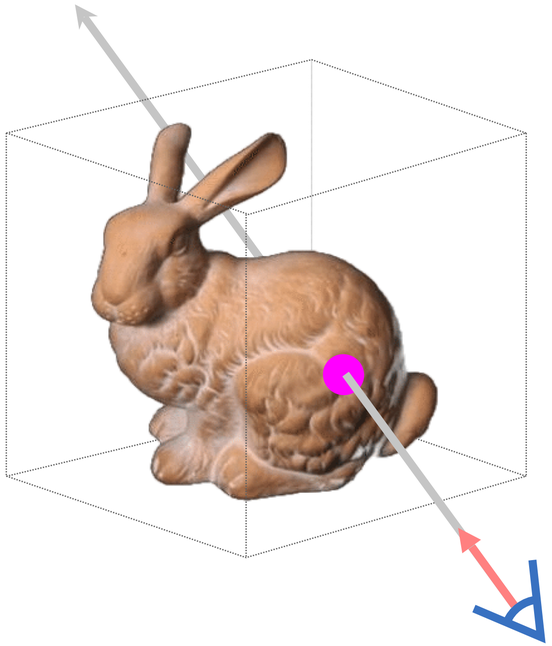
$$L_o(\mathbf{x}, \boldsymbol{\omega}_o, \lambda) = L_e(\mathbf{x}, \omega_o, \lambda)$$

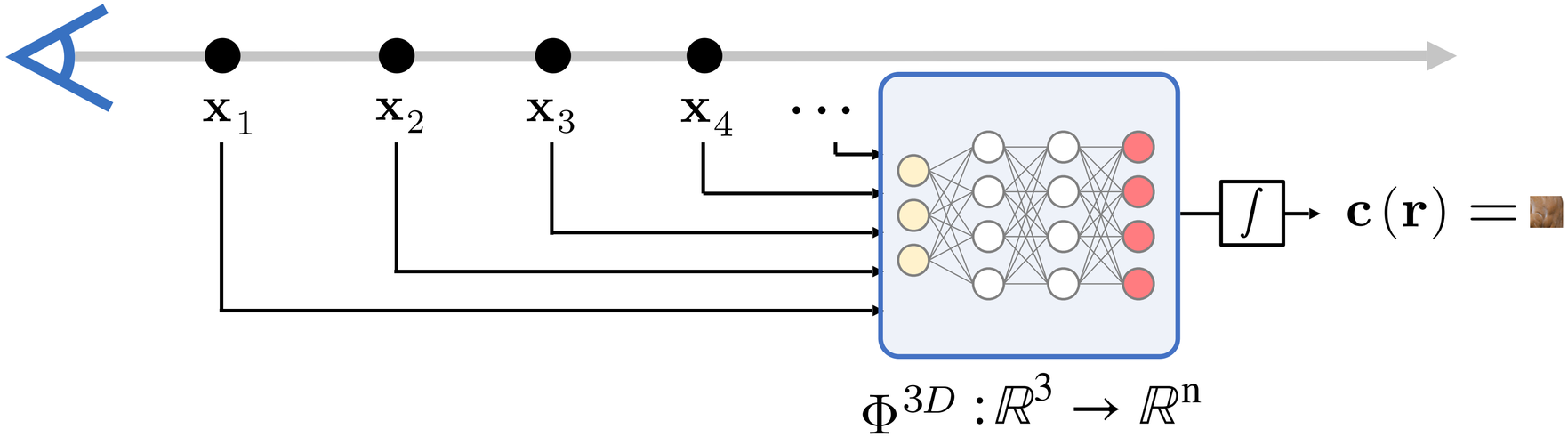
Can now approximate everything, but materials etc. are not enforced.

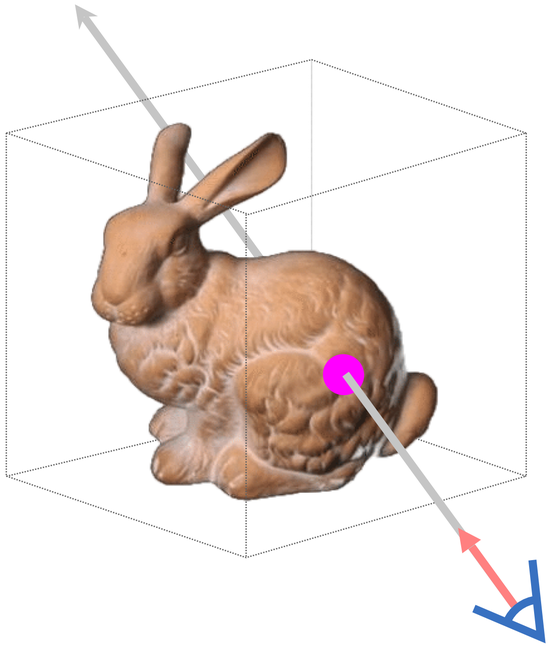
Questions?

Direction-dependent color

General structure of Neural Renderers for 3D-structured Representations

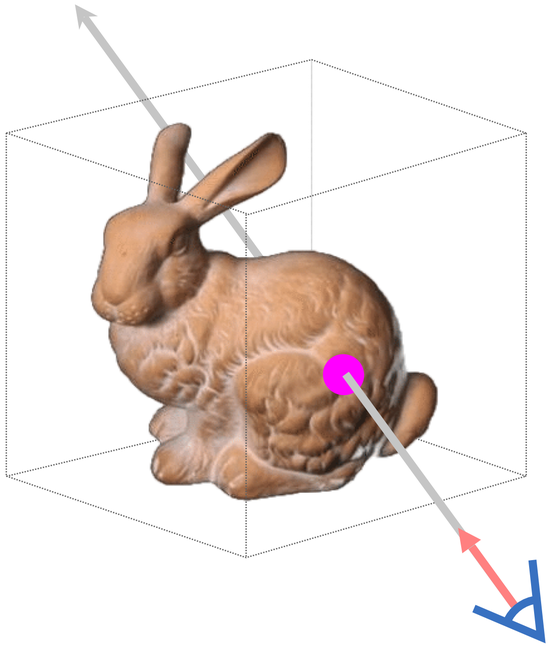






Light Field

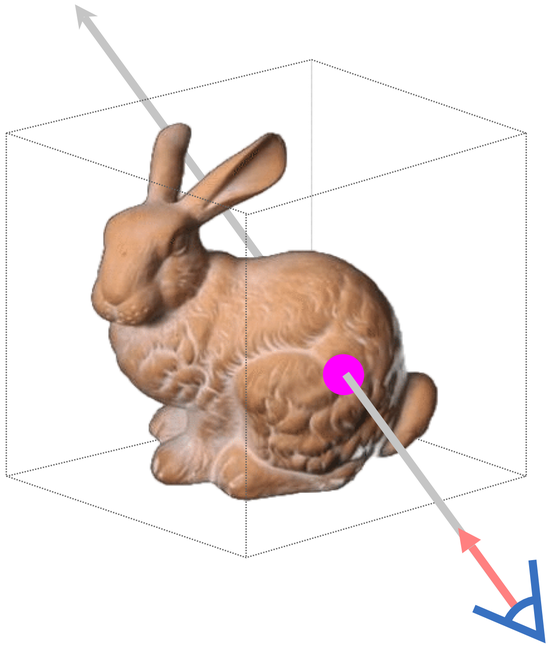
[Adelson et al. 1991, Levoy et al. 1996, Gortler et al. 1996]

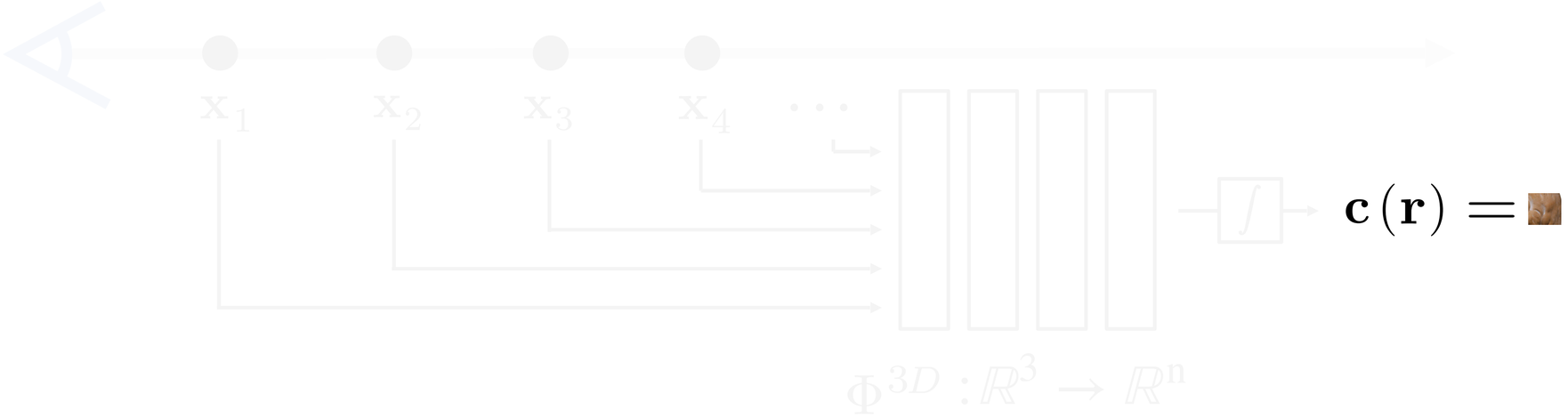


Light Field Networks

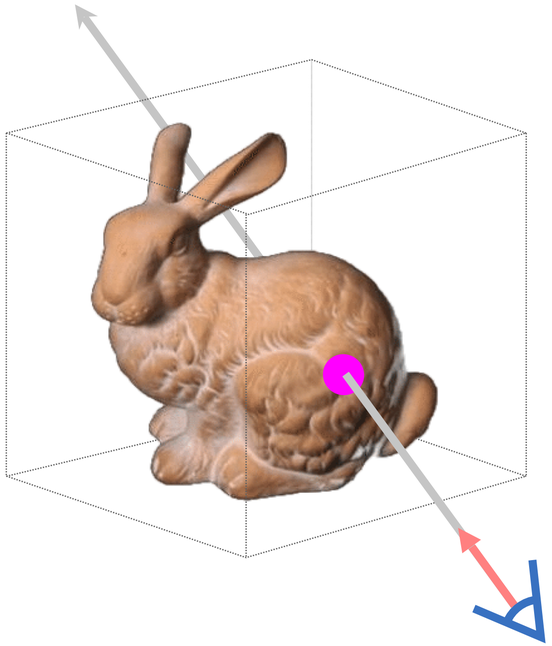
$\mathbf{c}(\mathbf{r}) =$

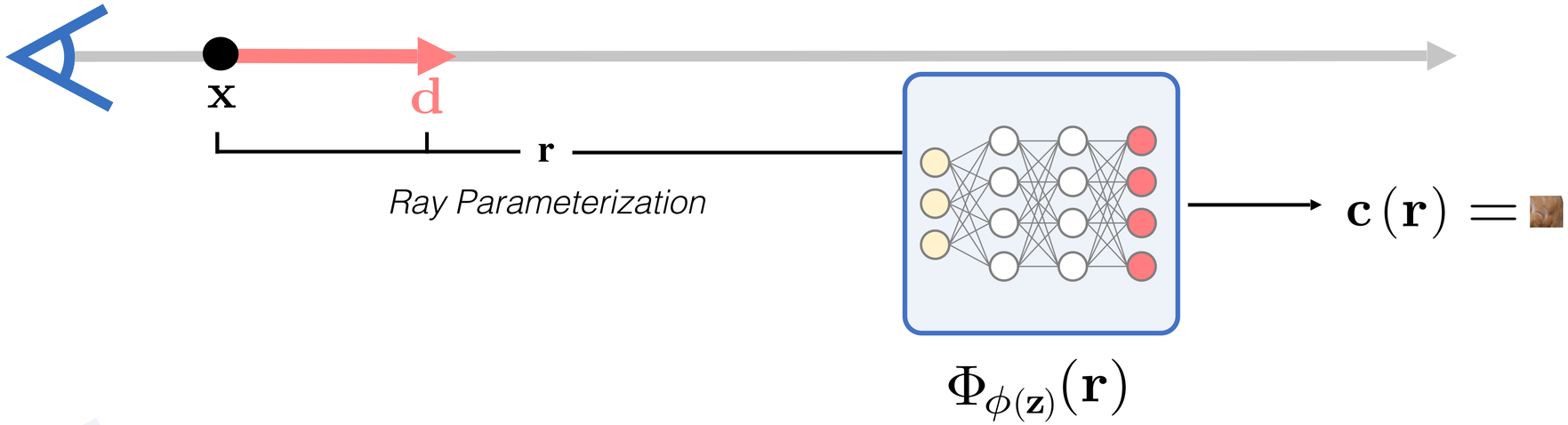
Light Field Networks





Light Field Networks





An Alternative Scene Representation

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

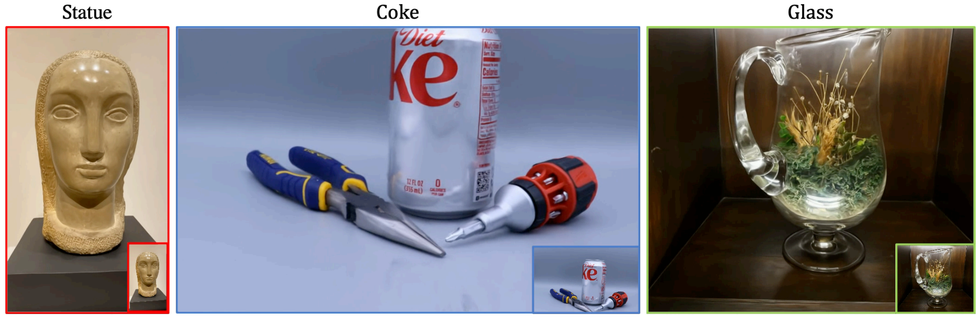
Context Views

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

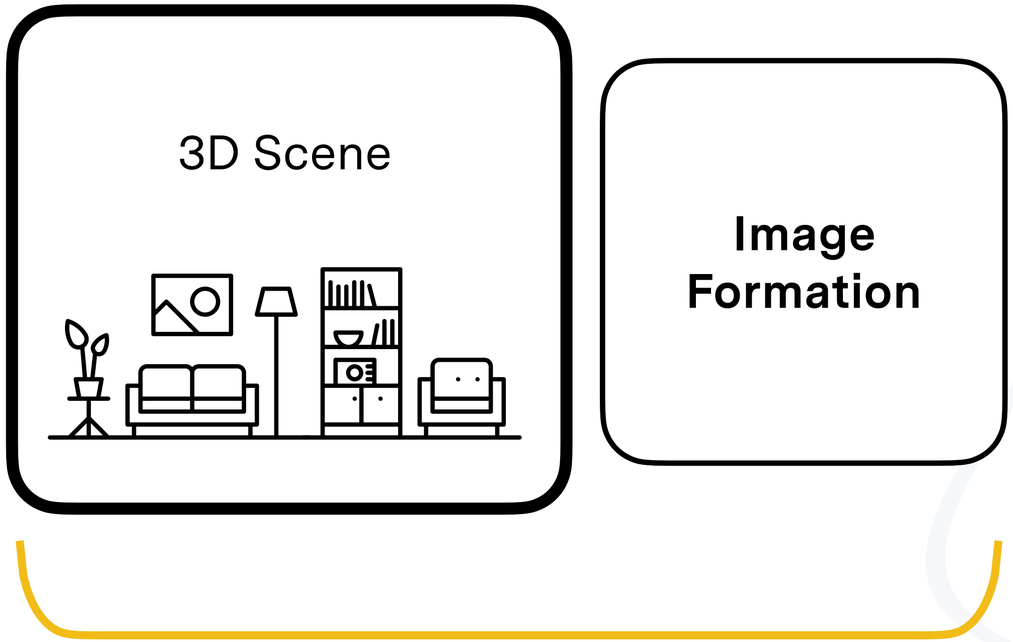
Machine Learning

Multi-View Consistency Prior

Light Field Networks, Sitzmann et al. 2021

Scene Representation Transformer, Sajjadi et al. 2021

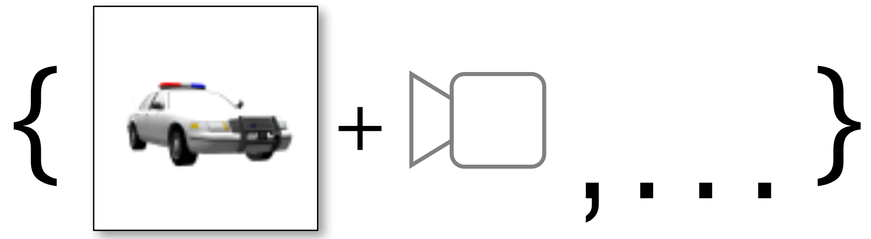
Today: Differentiable Rendering, AKA “Inverse Graphics”

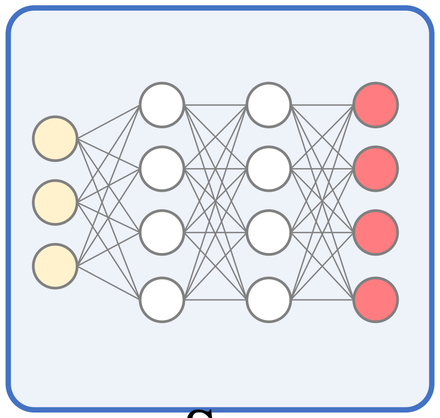


Graphics

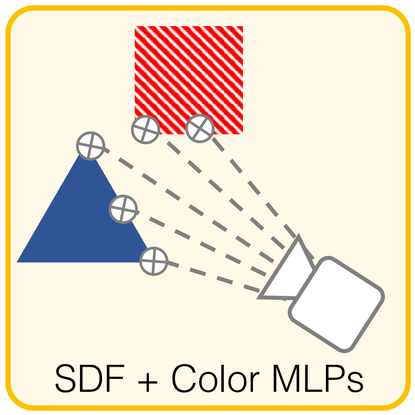
“inverse graphics”…?

Overfitting case: Inference = Fitting via Gradient Descent





Srn_{ϕ}



Render_{θ}

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

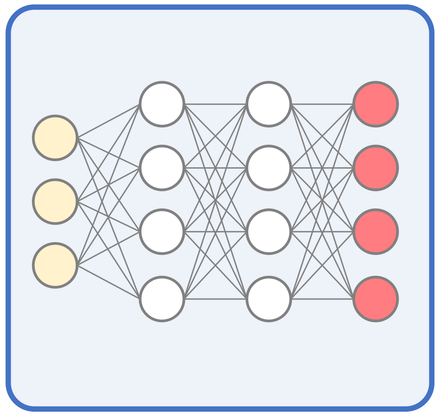
Optimization

Novel View Synthesis

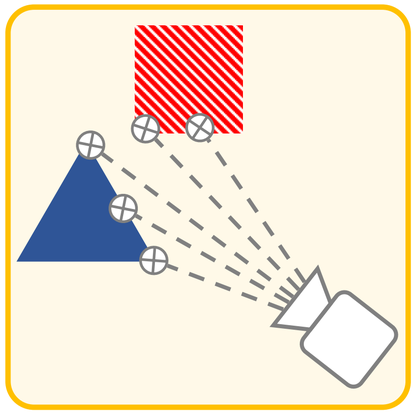
Normal map RGB

Sitzmann et al: Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations, NeurIPS 2020.

What if observations don't constrain scene representation?



Srn_{ϕ}



Render_{θ}

$+$

$(\mathcal{I}, \xi)$

No 3D inform.

$$\underset{\phi,\theta}{\text{argmin}} \left\| \text{Render}_{\theta}(\text{Srn}_{\phi}, \xi) - \mathcal{I} \right\|$$

Input

Normal map

RGB

GT

Today: Differentiable Rendering, AKA “Inverse Graphics”