Lecture 18
Implicit neural representations
Neural rendering

February 12nd 2024

Meirav Galun

Last time

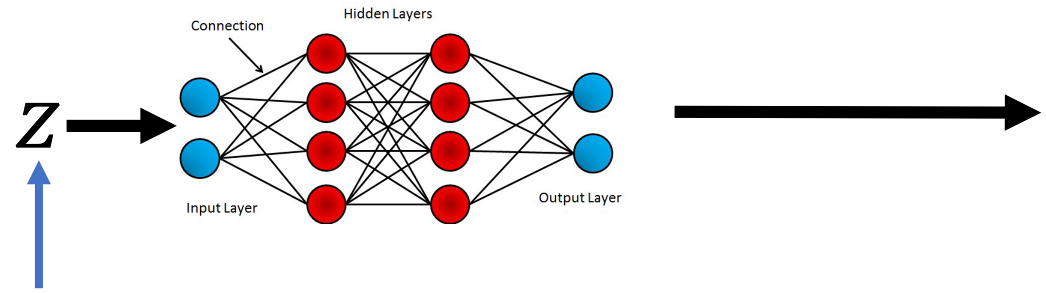
Deep image (implicit) prior for inverse problems

- Constructing an <u>implicit prior</u> by neural network

$$\min_{x} \|d(x) - \hat{x}\|$$

s.t. x is an output of CNN

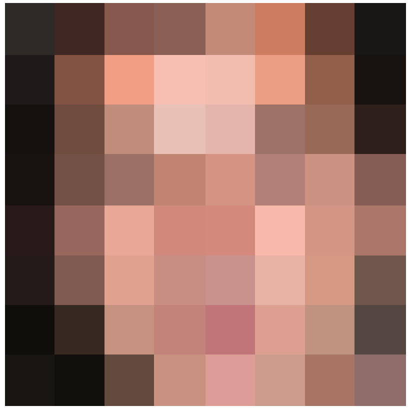
The network weights parametrize the restored image



x

$\rightarrow \|d(x) - \hat{x}\|$

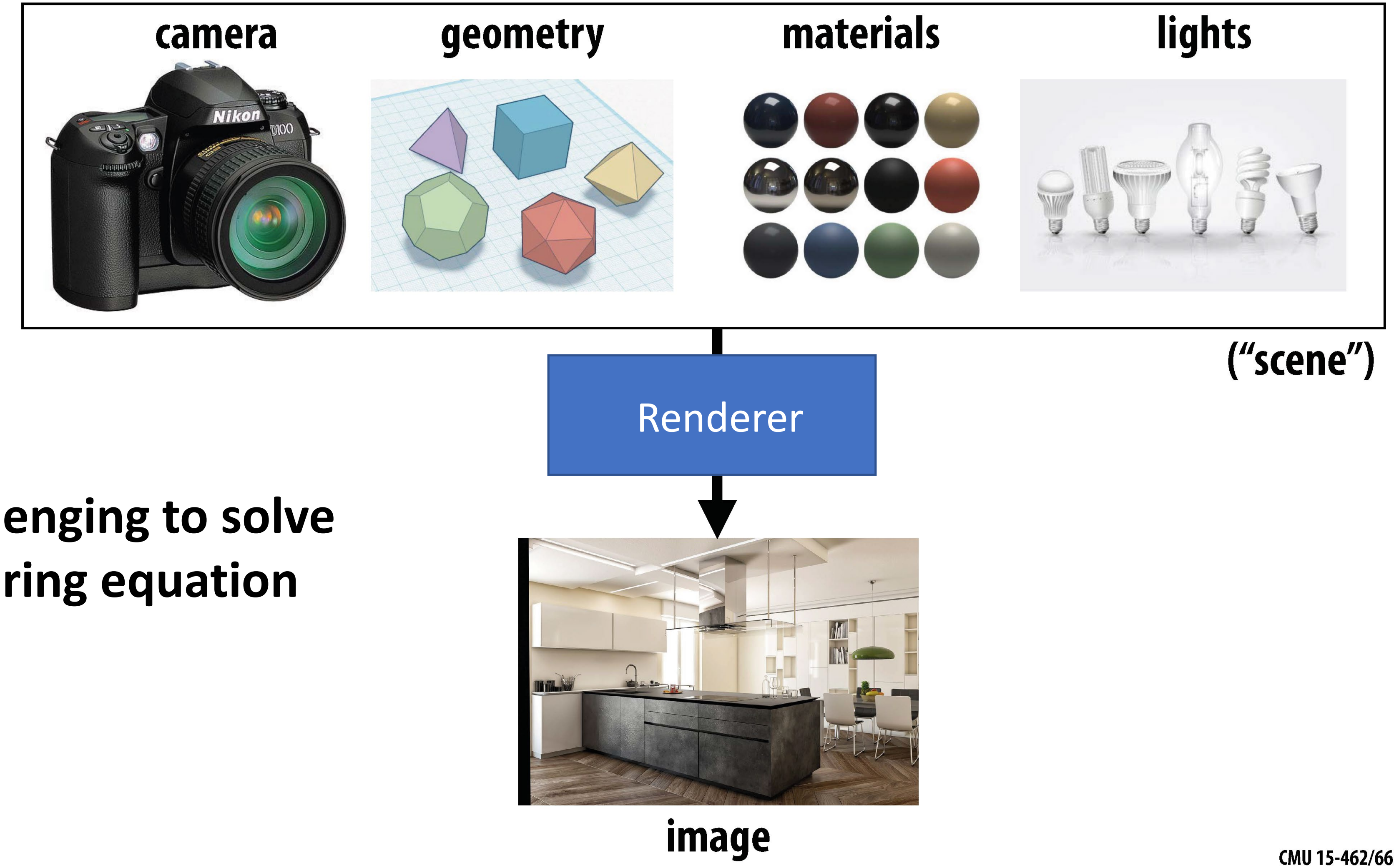
$\hat{x}$



Ulyanov, D., Vedaldi, A., & Lempitsky, V., Deep image prior, CVPR, 2018

Last time computer graphics and rendering

The process of generating a photorealistic image from a 3D model



Very challenging to solve the rendering equation

CMU 15-462/662

Last time Rendering equation (global illumination)

James Kajiya, 1986

- Computing reflection equation requires knowing the incoming radiance from surfaces

- But determining incoming radiance requires knowing reflected radiance from surfaces

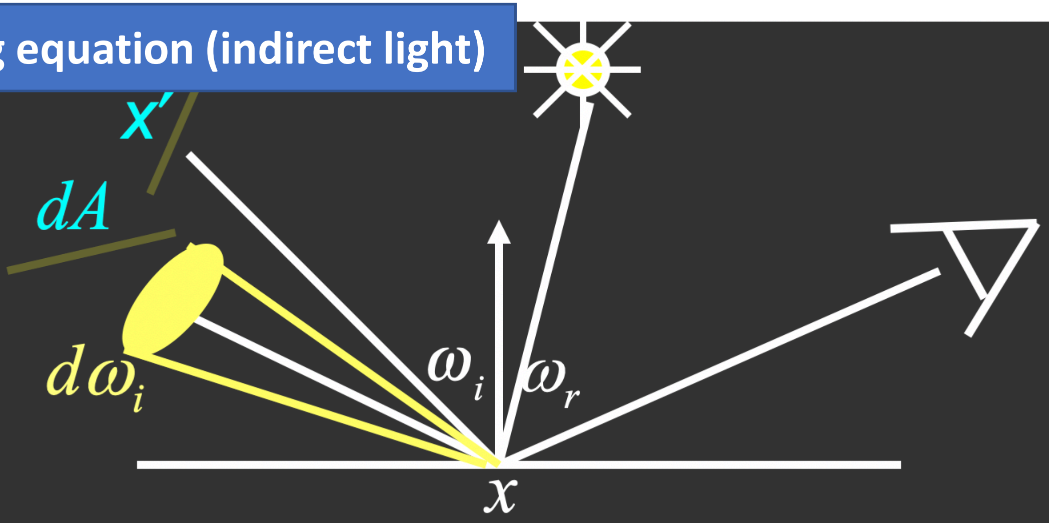
- So we have to compute another integral, we have exactly the same equation

- Rendering equation is recursive

Reflection equation (direct light)

$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_{\Omega} L_i(x,\omega_i) f(x,\omega_i,\omega_r) \cos\theta_i d\omega_i$$

Rendering equation (indirect light)



$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_{\Omega} L_r(x',-\omega_i) f(x,\omega_i,\omega_r) \cos\theta_i d\omega_i$$

Reflected Light (Output Image)	Emission	Reflected Light	BRDF	Cosine of Incident angle
UNKNOWN	KNOWN	UNKNOWN	KNOWN	KNOWN

Slide by Lior Yariv

Material covered today

- Neural rendering
  (Deep-based computer graphics)

Deep image or video generation approaches that enable explicit or implicit control of scene properties such as illumination, camera parameters, pose, geometry, appearance and semantic structure

State of the Art on Neural Rendering, A. Tewari et al., 2020

Material covered today

- Implicit neural scene representations

  A network can parametrize

  - Geometry
  - 3D volumes
  - Continuous functions

Why not utilizing geometry of a scene by an explicit representation?
Why is it less beneficial to employ explicit representation in neural nets?

Based on

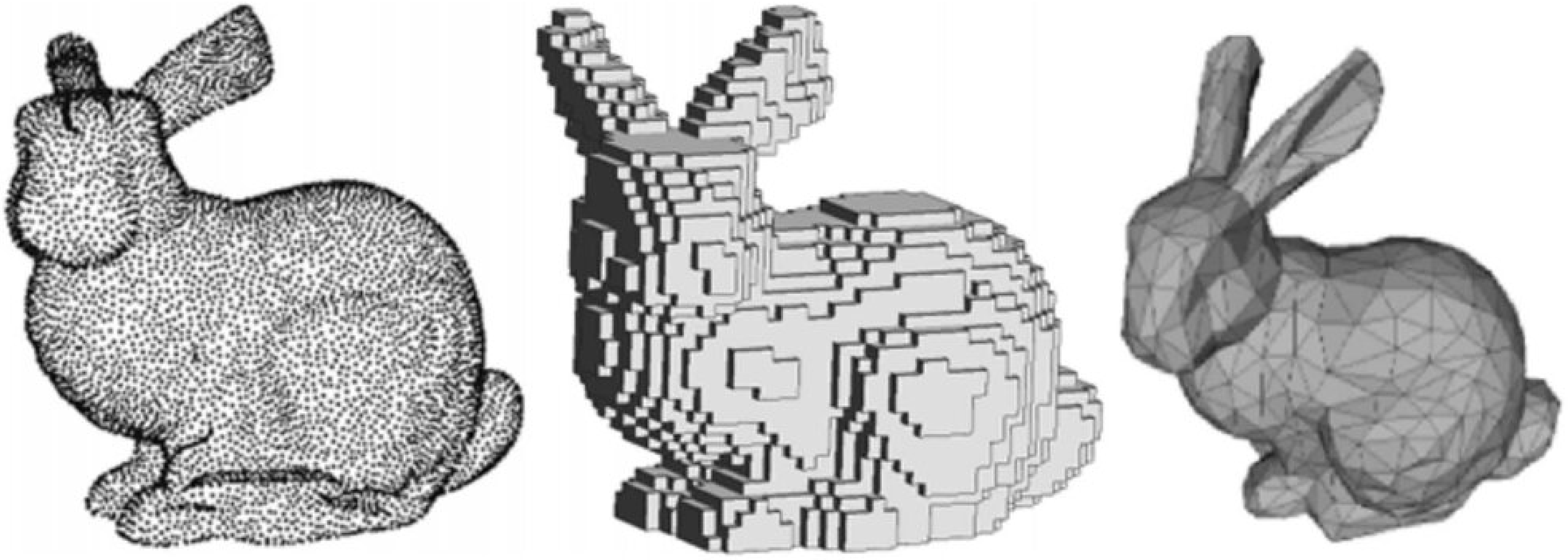
1. The ECCV 2022 Tutorial Neural Volumetric Rendering for Computer Vision
2. In particular, slides by Matt Tancik and Ben Mildenhall

Geometry

Scene representation

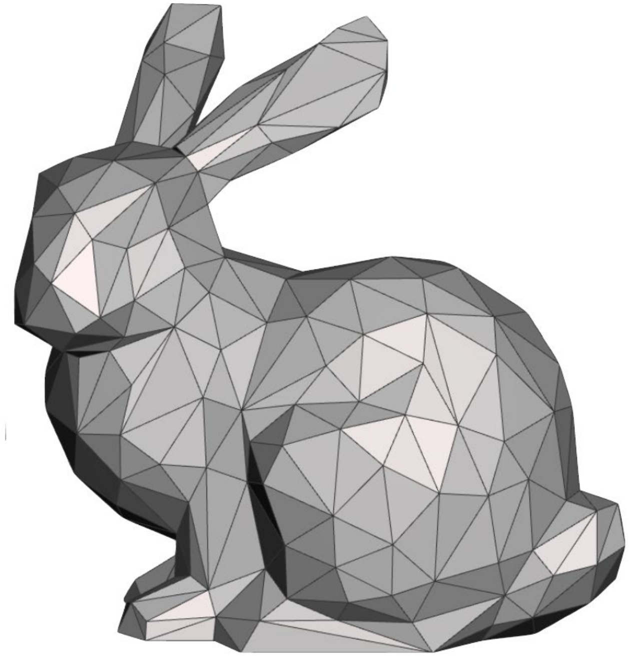
Explicit (discretization of the object geometry)

- triangle (polygon) mesh
- voxels
- point cloud



Geometry

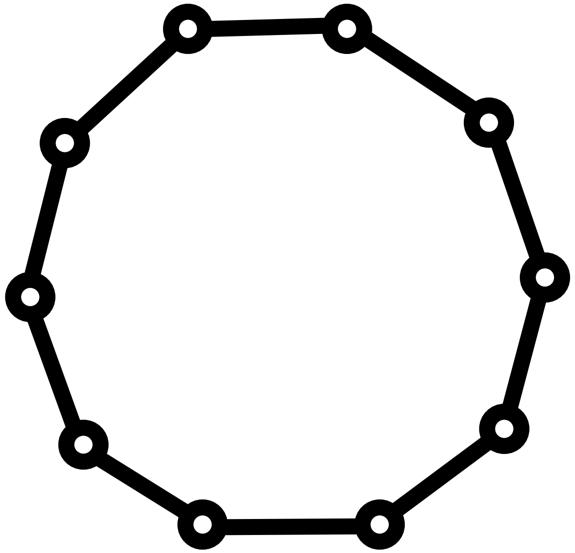
Mesh representation



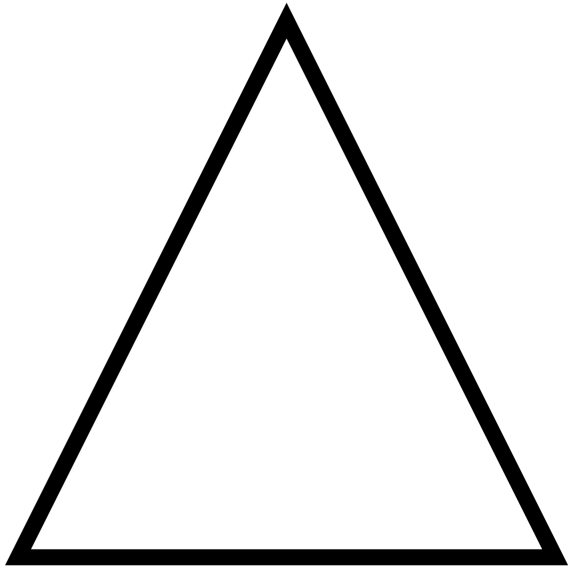
Slides on explicit geometry
by Matt Tancik

Task: represent target geometry by a triangular mesh

Gradient Based Optimization



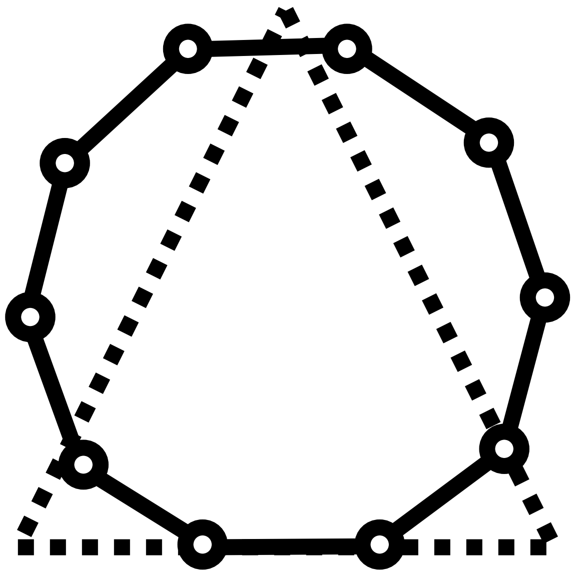
Initial Geometry



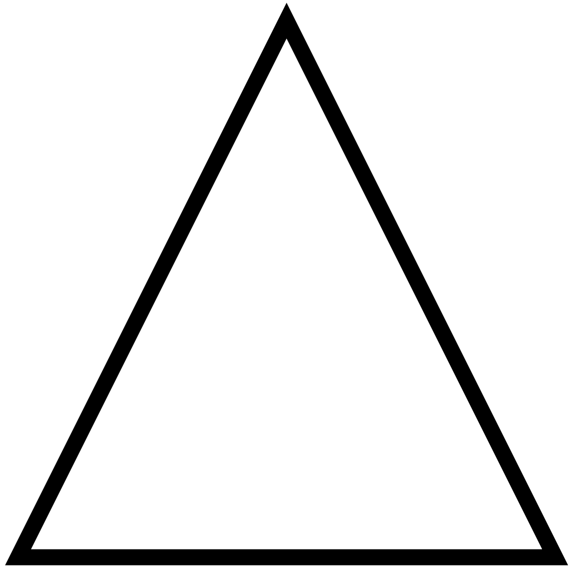
Target Geometry

Task: represent target geometry by a triangular mesh

Gradient Based Optimization

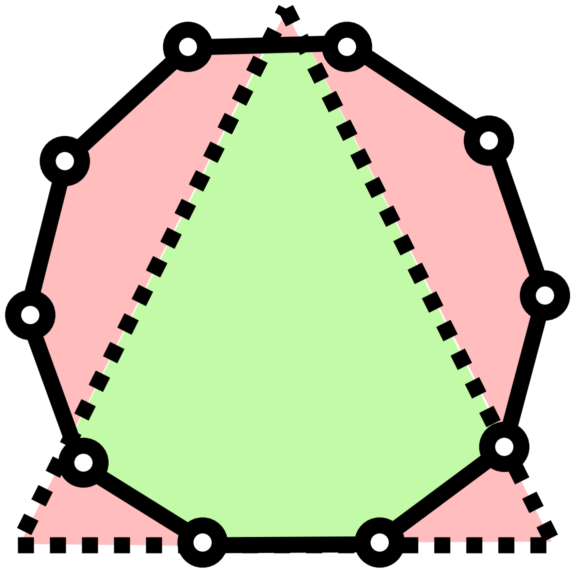


Initial Geometry

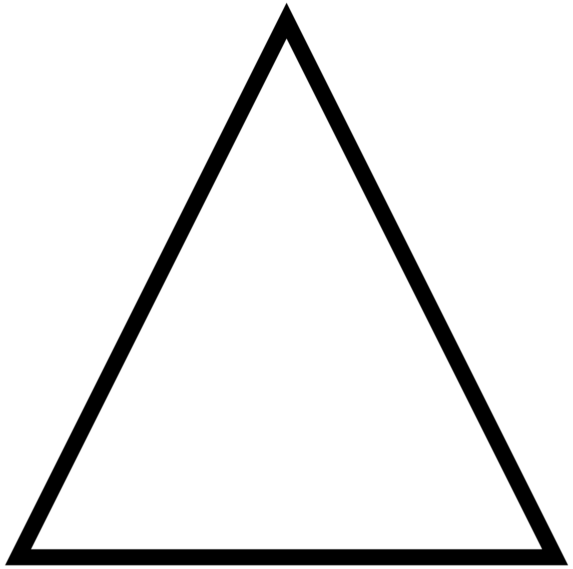


Target Geometry

Task: represent target geometry by a triangular mesh
Gradient Based Optimization



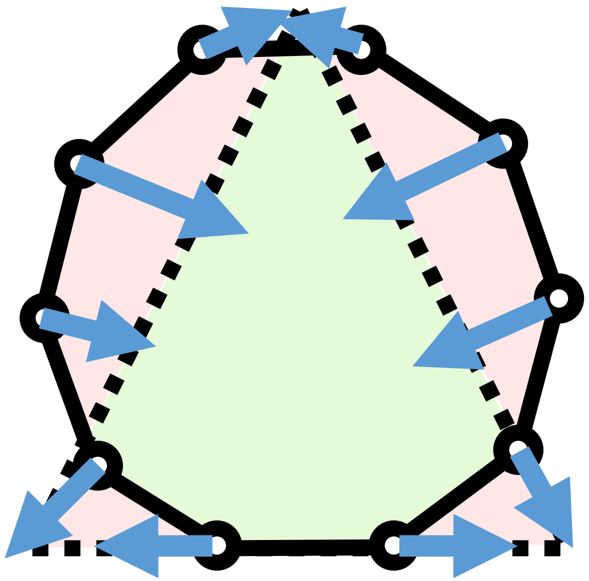
Compute Gradients



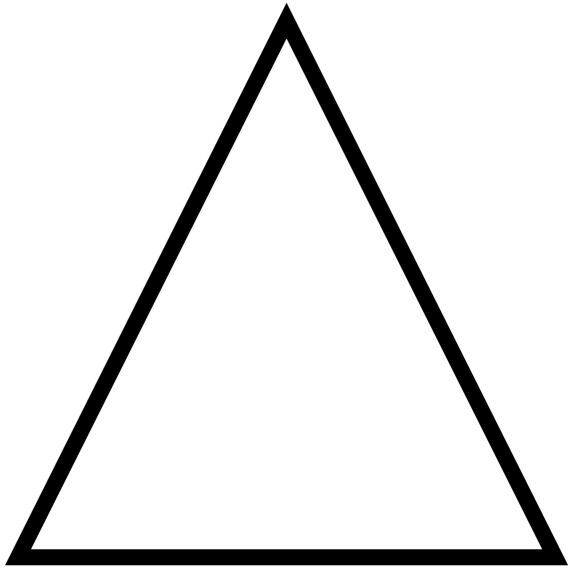
Target Geometry

Task: represent target geometry by a triangular mesh

Gradient Based Optimization

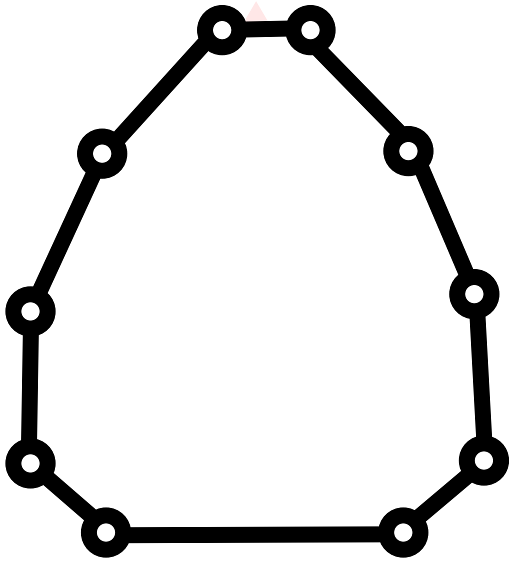


Compute Gradients

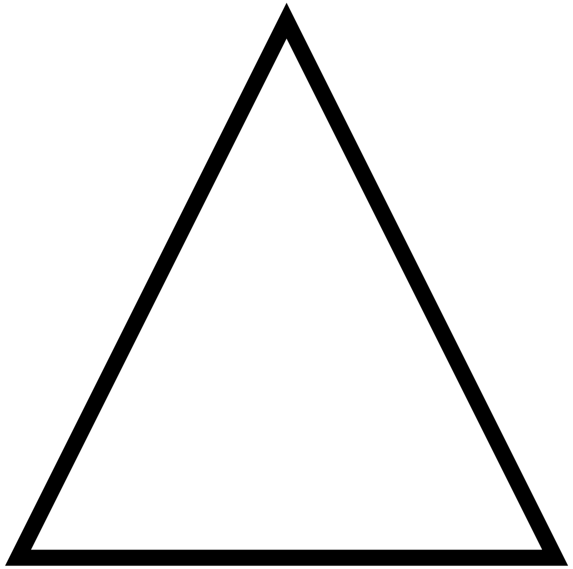


Target Geometry

Task: represent target geometry by a triangular mesh
Gradient Based Optimization

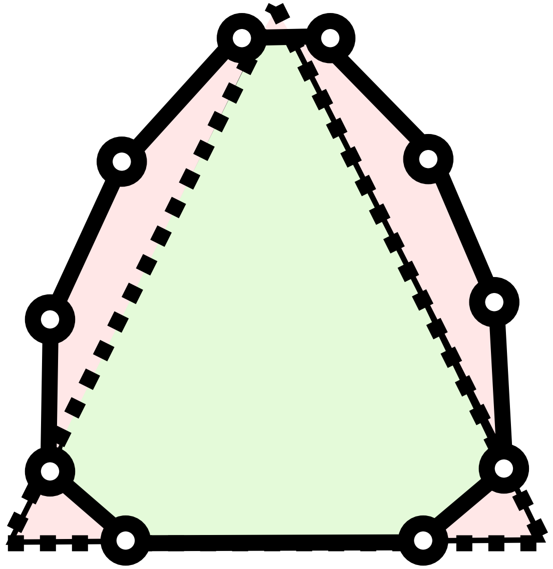


Update positions

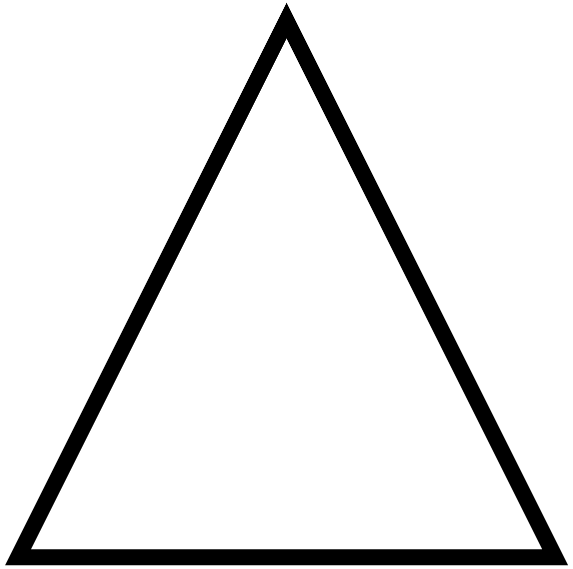


Target Geometry

Task: represent target geometry by a triangular mesh
Gradient Based Optimization

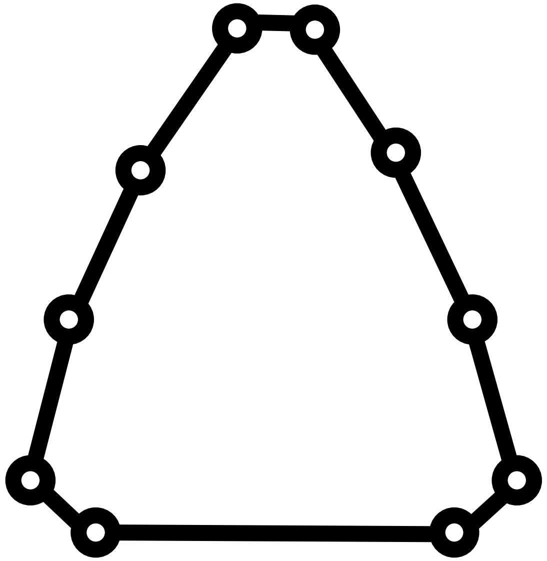


Compute New Error

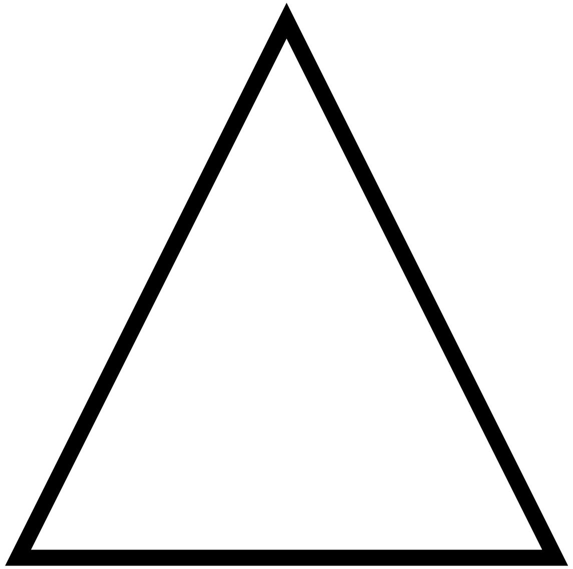


Target Geometry

Task: represent target geometry by a triangular mesh
Gradient Based Optimization

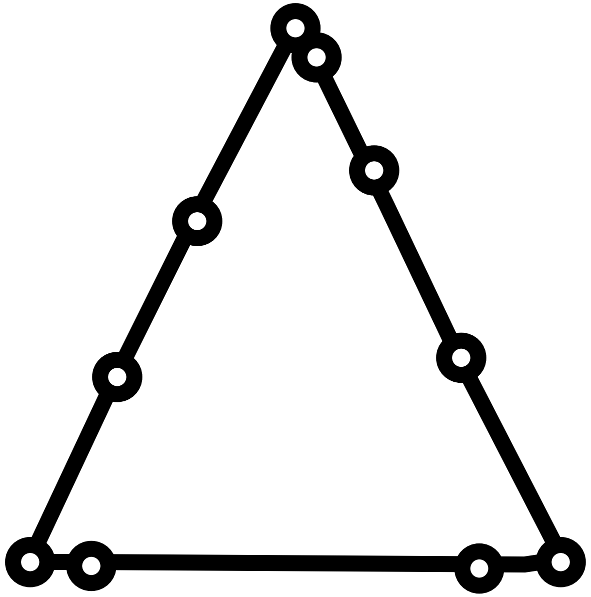


Repeat

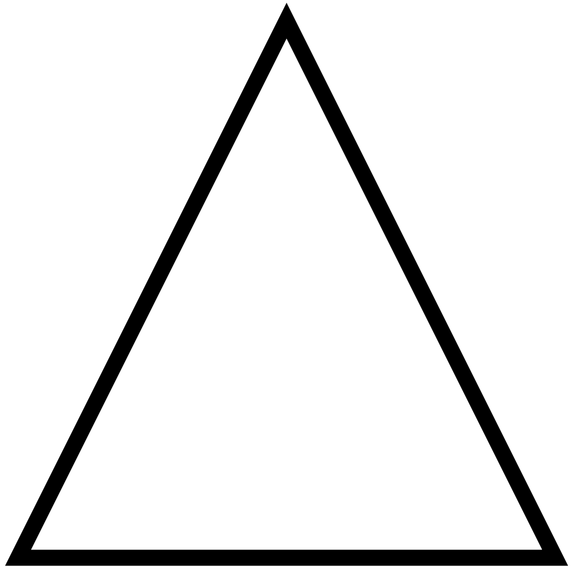


Target Geometry

Task: represent target geometry by a triangular mesh
Gradient Based Optimization



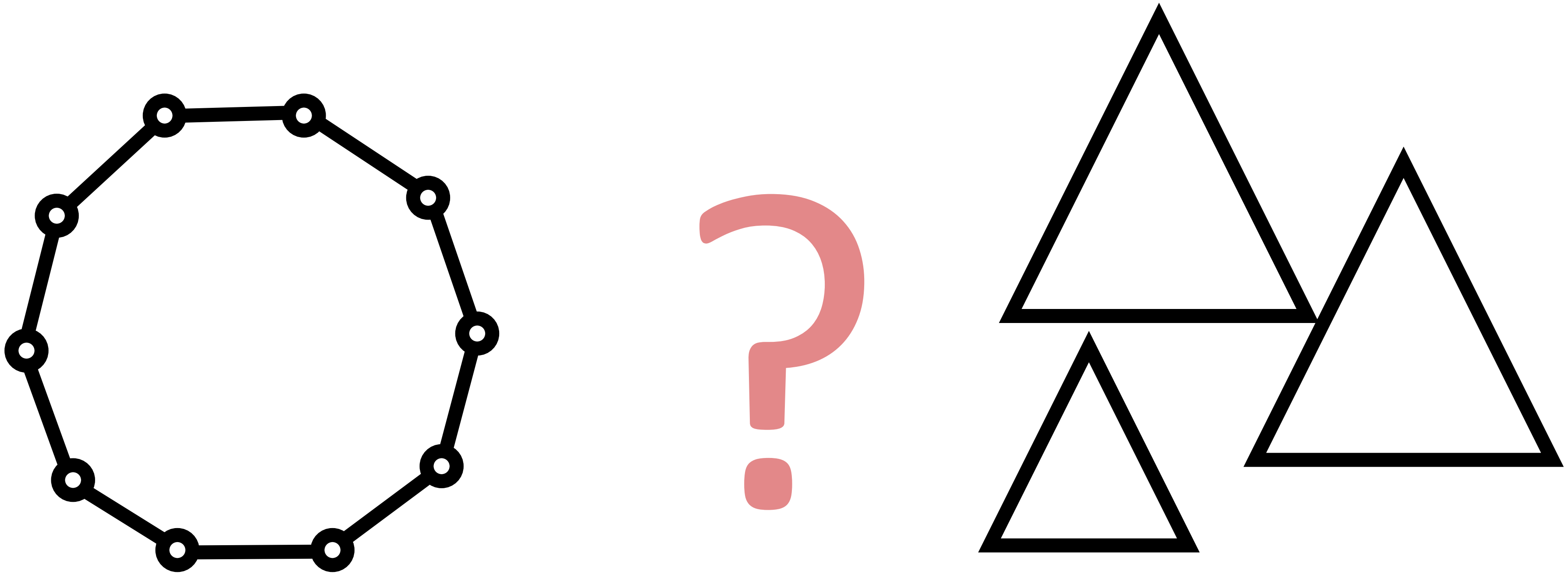
Repeat



Target Geometry

Task: represent target geometry by a triangular mesh

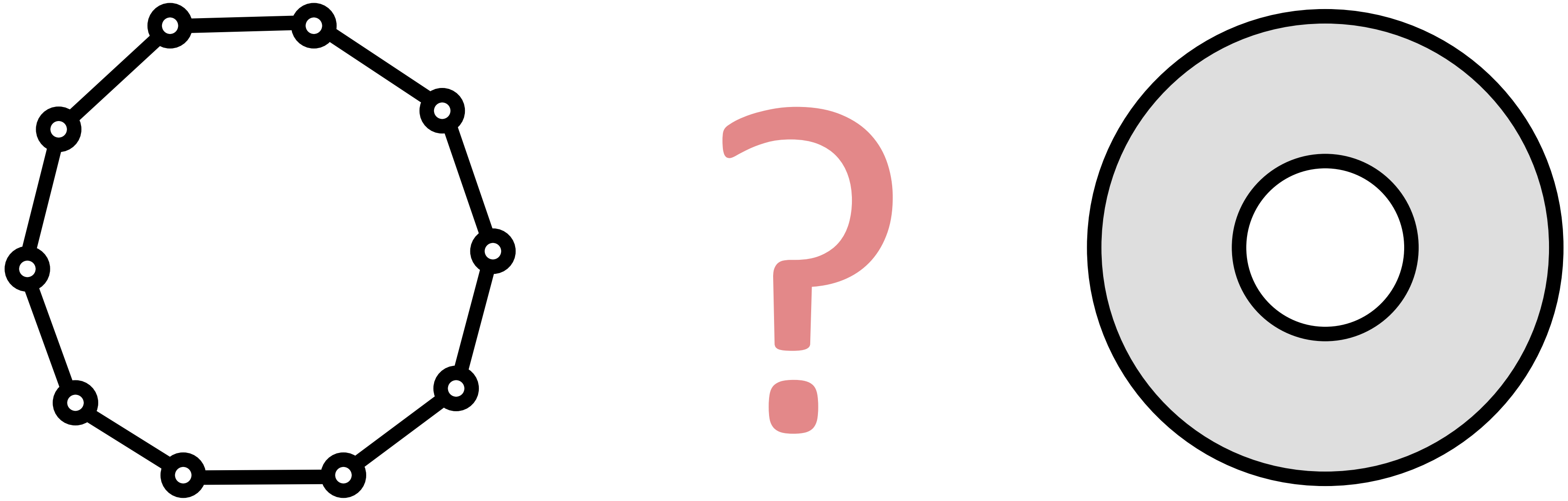
Gradient Based Optimization



Initial Geometry

Target Geometry

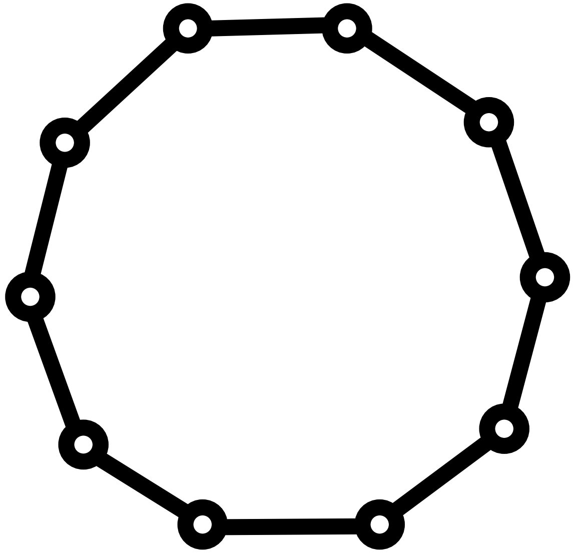
Gradient Based Optimization



Initial Geometry

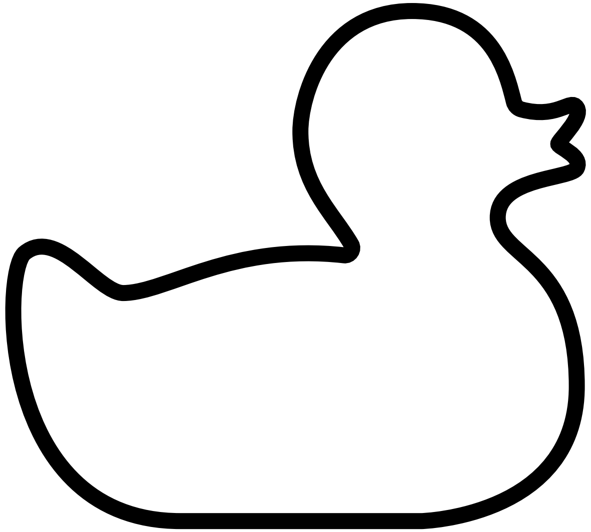
Target Geometry

Task: represent target geometry by a triangular mesh
Gradient Based Optimization



Initial Geometry

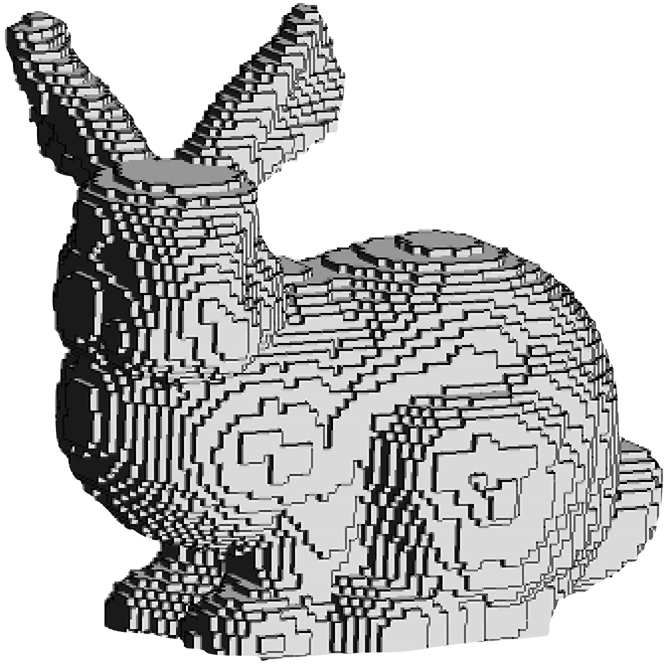
?



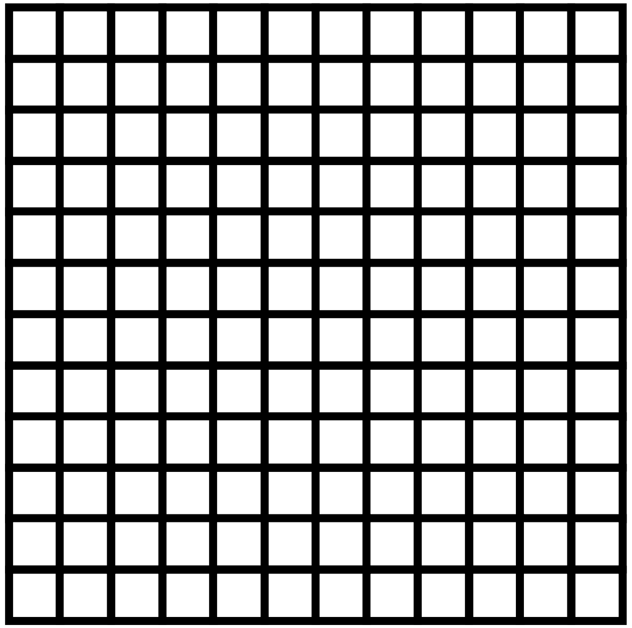
Target Geometry

Geometry

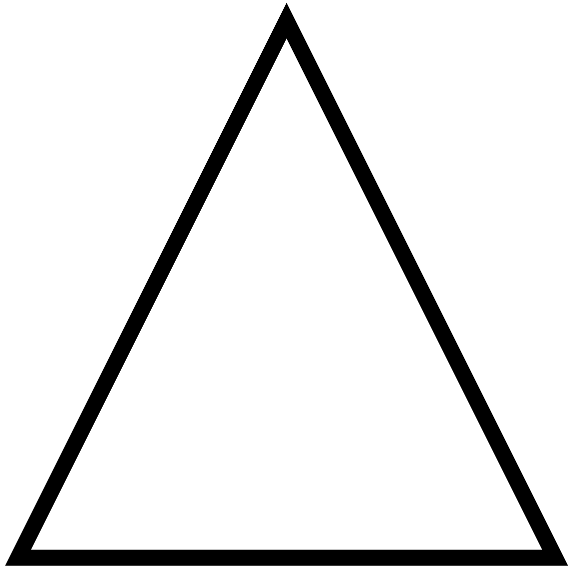
Voxel representation



Task: represent target geometry by voxels
Gradient Based Optimization

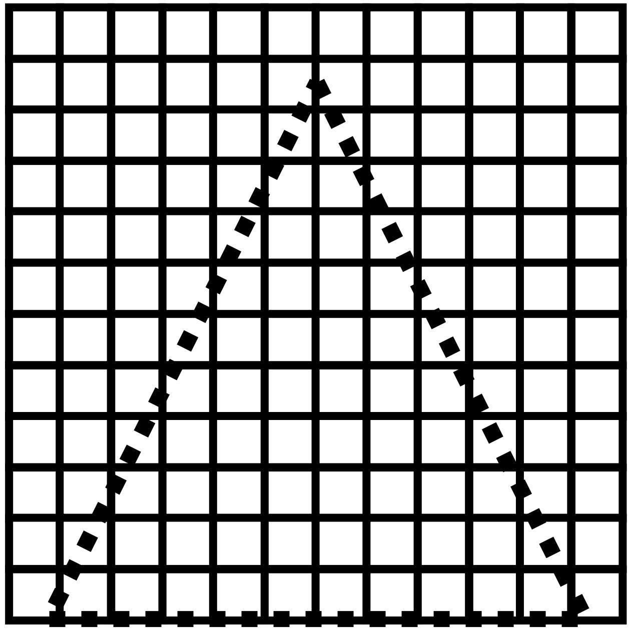


Initialized Grid

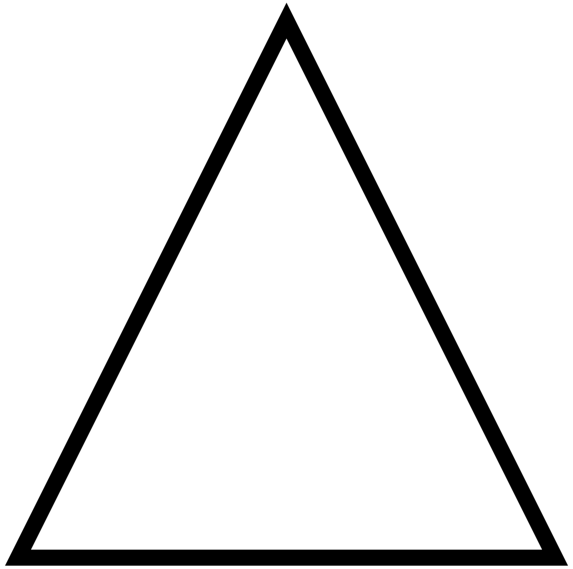


Target Geometry

Task: represent target geometry by voxels
Gradient Based Optimization

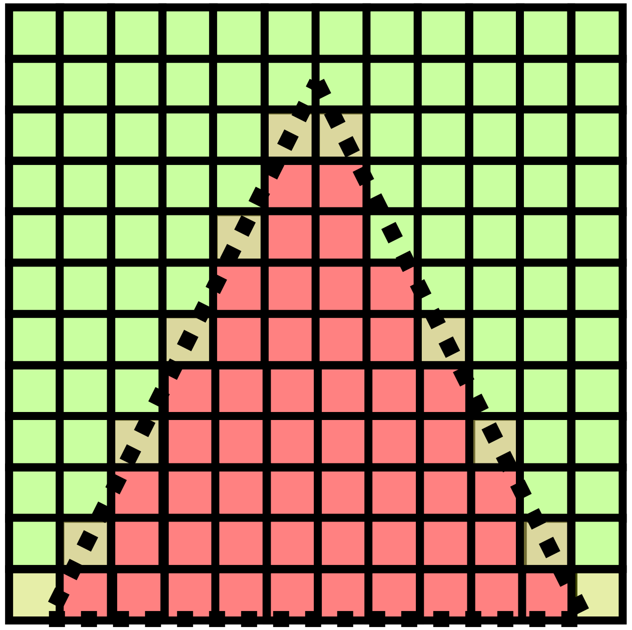


Initialized Grid

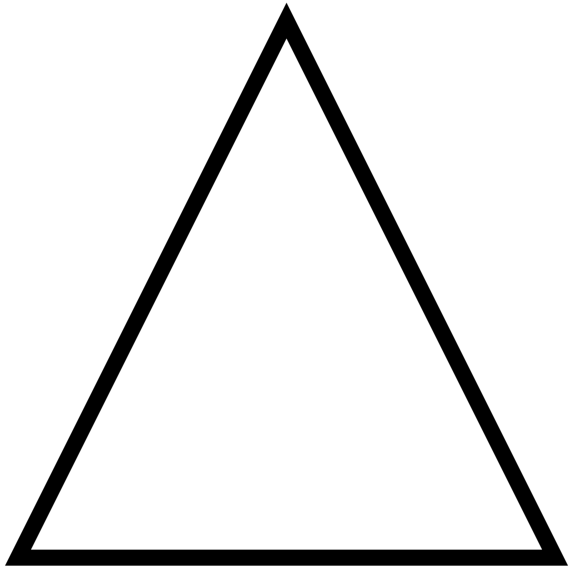


Target Geometry

Task: represent target geometry by voxels
Gradient Based Optimization

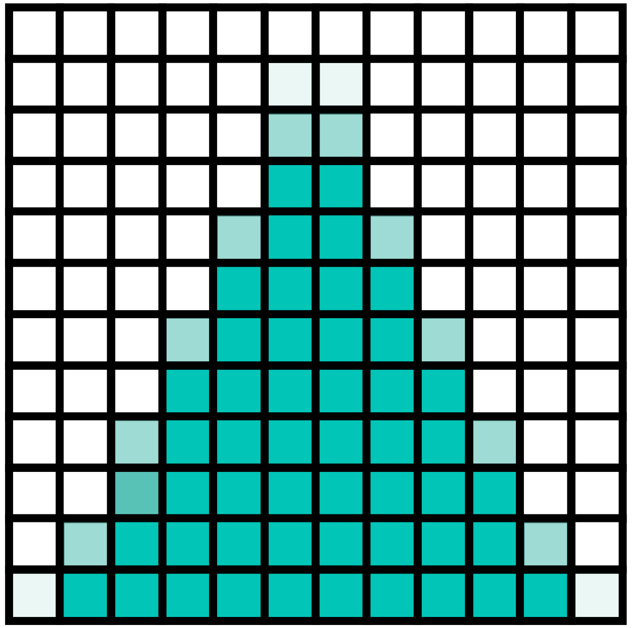


Loss

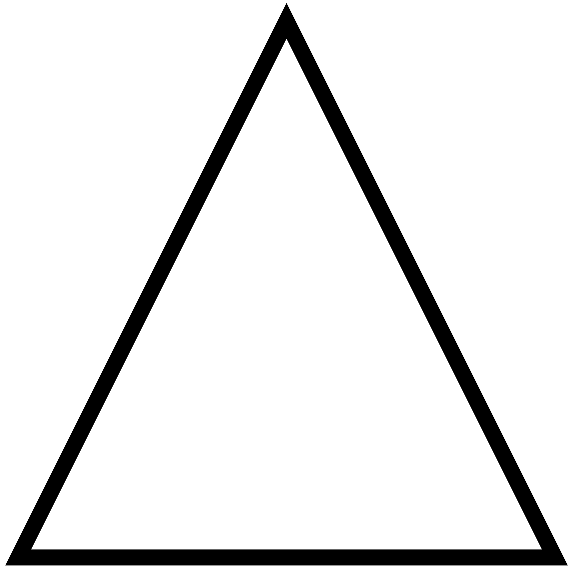


Target Geometry

Task: represent target geometry by voxels
Gradient Based Optimization

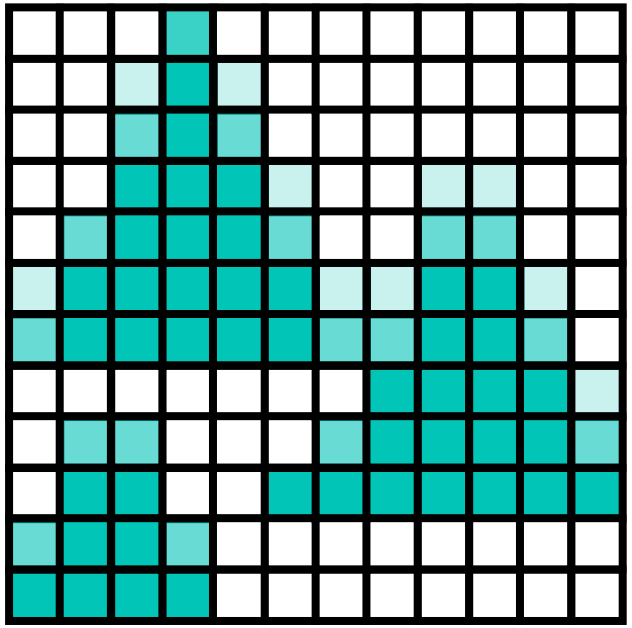


Repeat

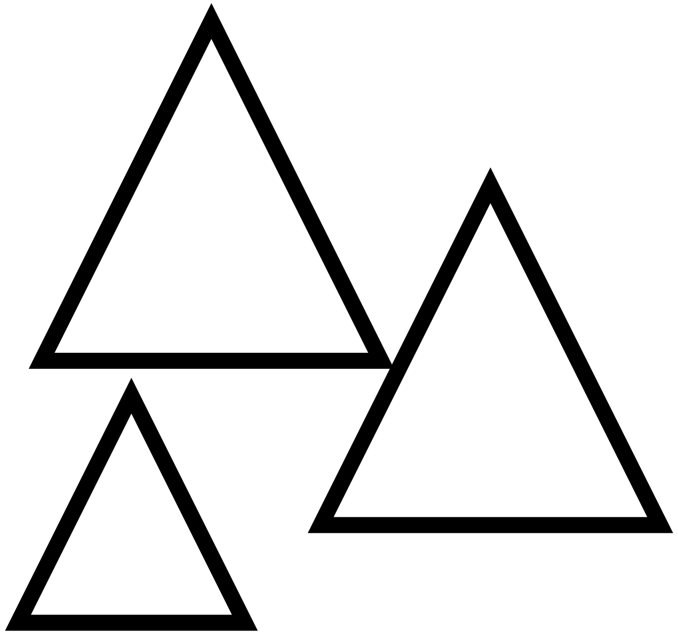


Target Geometry

Task: represent target geometry by voxels
Gradient Based Optimization

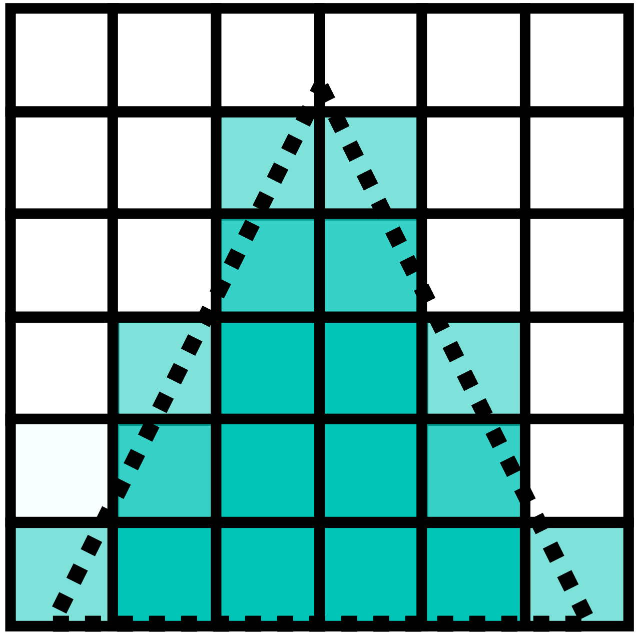


Reconstruction

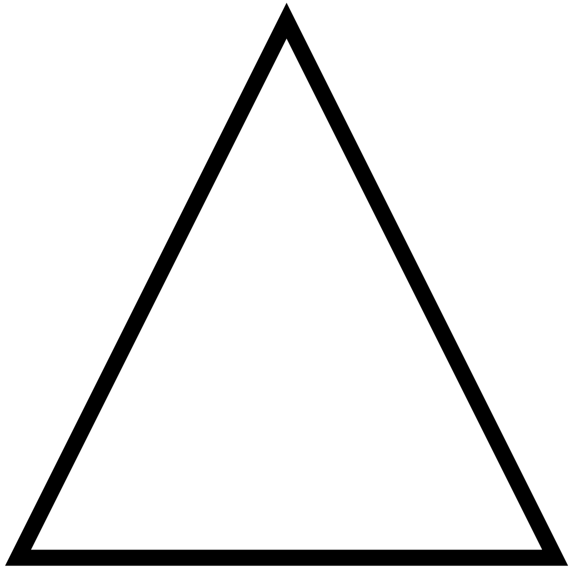


Target Geometry

Task: represent target geometry by voxels
Gradient Based Optimization



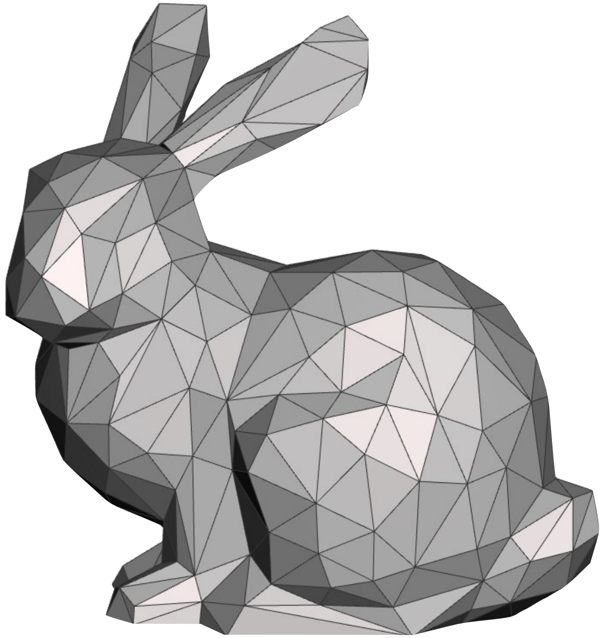
Reconstruction



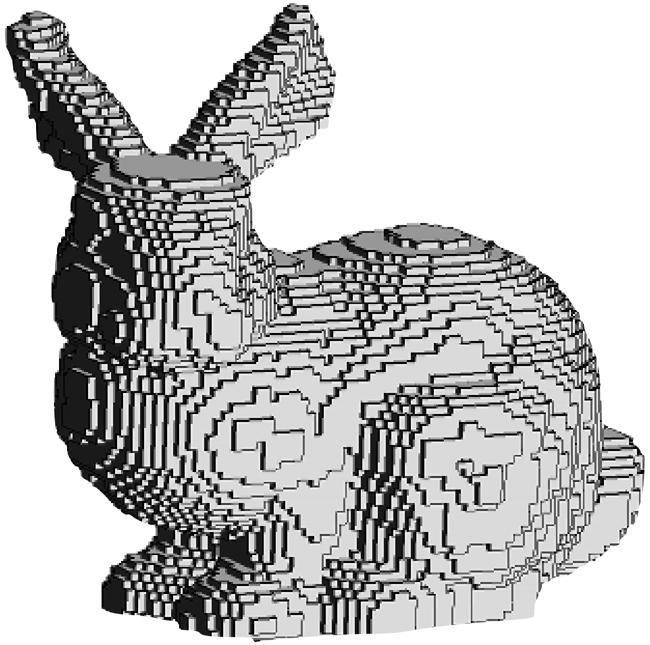
Target Geometry

Geometry Representations

✓ Implicit (continuous) representations



Mesh Representation



Voxel Representation

Small memory footprint

Hard to optimize

Easy to optimize

Large memory footprint

Geometry

Implicit Scene representation

How to represent a general shape with implicit functions?

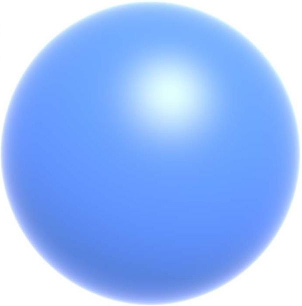
Implicit representations

- algebraic surfaces
- complicated to tailor algebraic expressions, that will fit general shapes

More expressive implicit representations

- level set $f: R^3 \to R,\ f(x, y, z) = 0$
- signed distance function

- **Examples:**

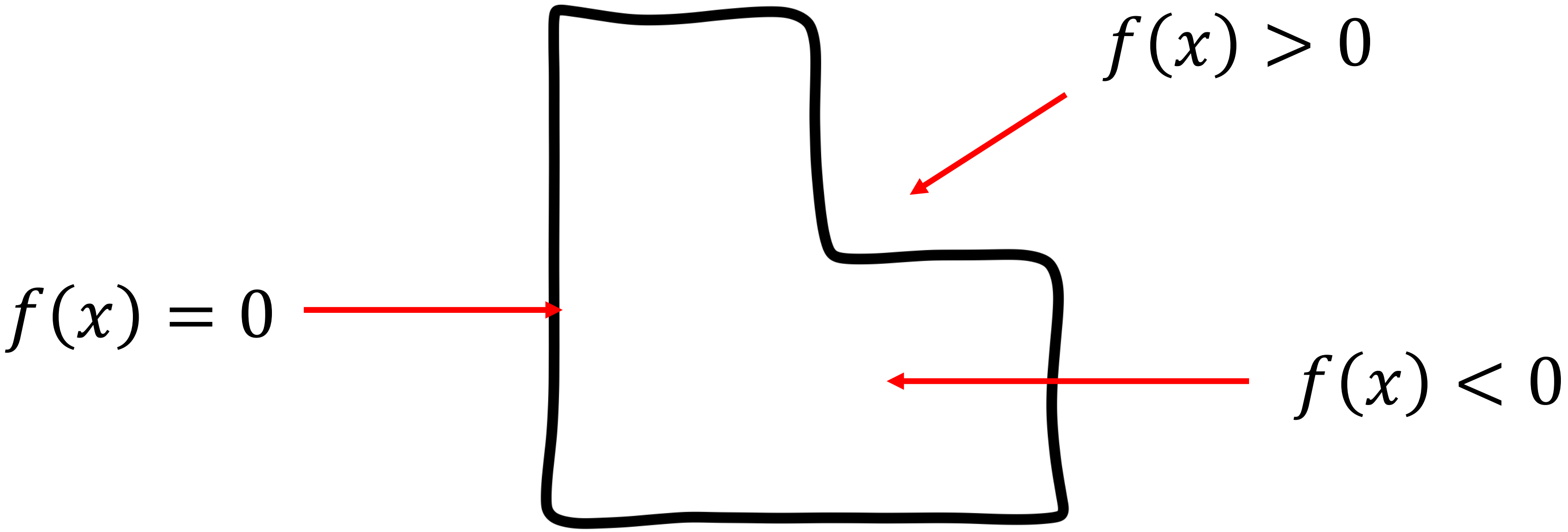


$x^2 + y^2 + z^2 = 1$

$(R - \sqrt{x^2 + y^2})^2 + z^2 = r^2$

Geometry

Implicit shape representation

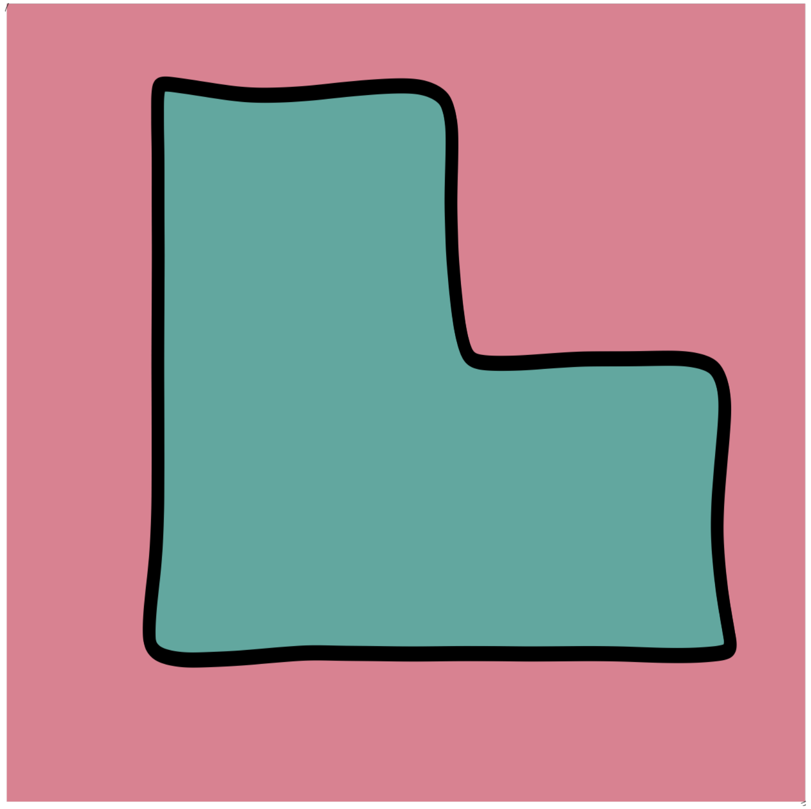


Surface represented implicitly

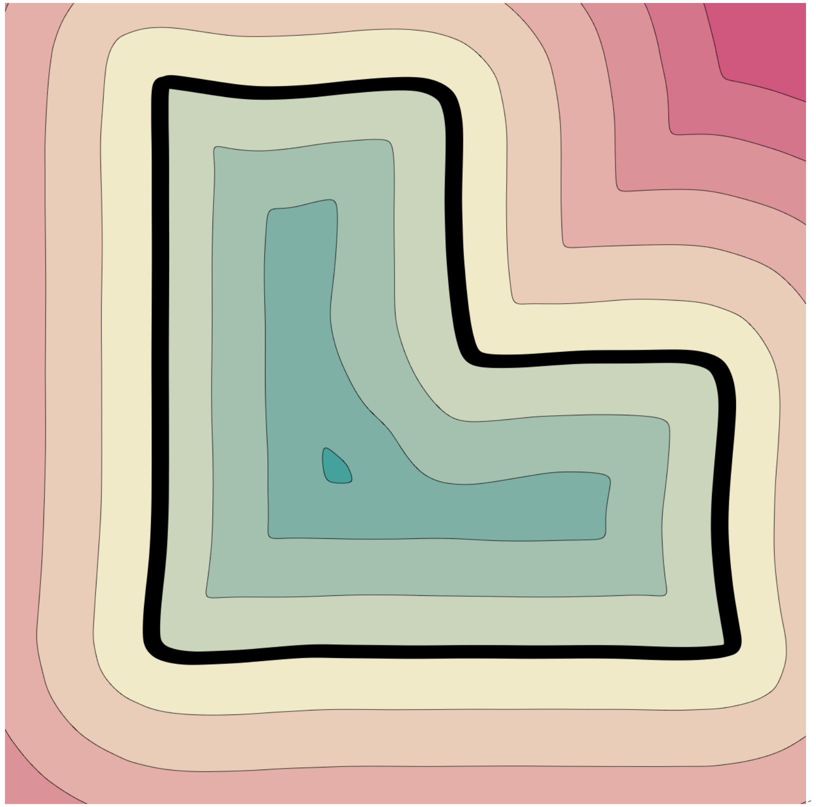
$$s = \{x \in \mathbb{R}^3 | f(x) = 0\}$$

Geometry

Implicit shape representation



Indicator / occupancy



Signed Distance Function (SDF)

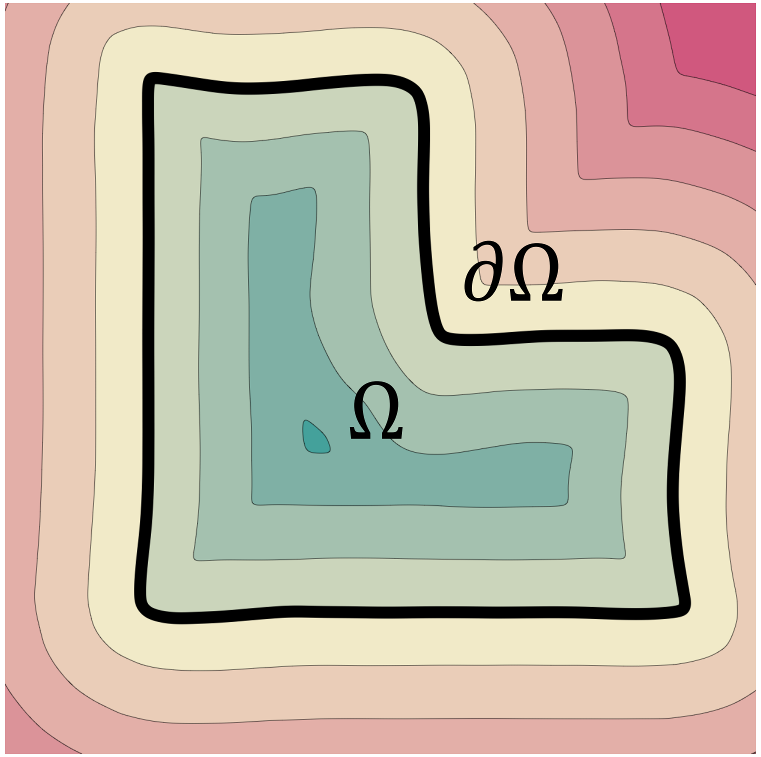
Geometry

Implicit shape representation

Eikonal equation

$$\|\nabla f(\boldsymbol{x})\| = 1, \boldsymbol{x} \in \Omega$$

$$f(\boldsymbol{x}) = 0, \boldsymbol{x} \in \partial\Omega$$



Signed distance function (SDF)

Implicit representation

Properties

- continuous representation
- can represent arbitrary topology at arbitrary resolution
- not limited by excessive memory requirements
- geometric quantities, e.g., normals
- blend well with deep learning techniques

How?

Implicit neural representations

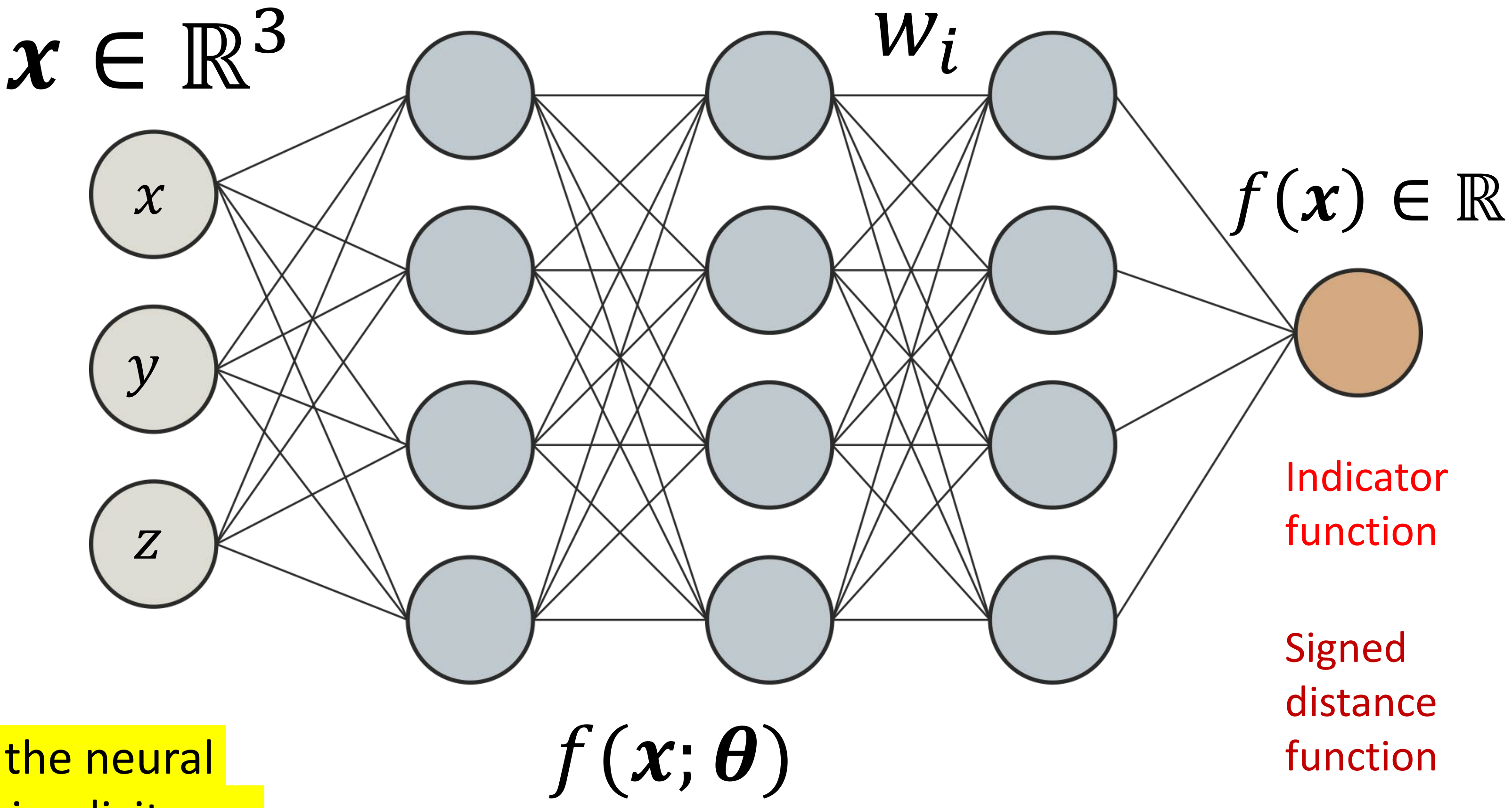
[Park et al. 2019, Chen & Zhang 2019, Mescheder et al. 2019, Atzmon et al. 2019]

Theorem (Universality).

Any watertight piecewise linear surface can be exactly represented as the neural level set S of MLP with ReLU activations.

$$S = \{x | f(x; \theta) = 0\}$$

After training, the obtained weights in the neural net actually represent the shape, in an implicit way.



$\boldsymbol{x} \in \mathbb{R}^3$
x
y
z
w_i
$f(\boldsymbol{x}) \in \mathbb{R}$
$f(\boldsymbol{x}; \boldsymbol{\theta})$

Indicator function

Signed distance function

Controlling neural level sets, Atzmon et al., 2019

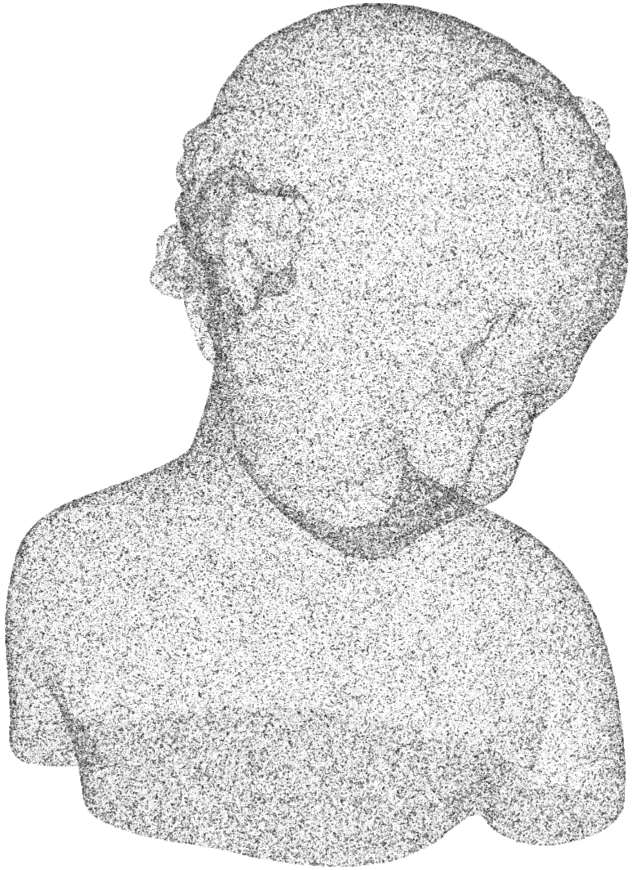
How to learn implicit neural representations?

Surface represented implicitly

$$S_\theta = \{\boldsymbol{x} | f(\boldsymbol{x}; \theta) = 0\}$$

How to learn implicit neural representations?

- Full 3D supervision
- Raw data (weak supervision)



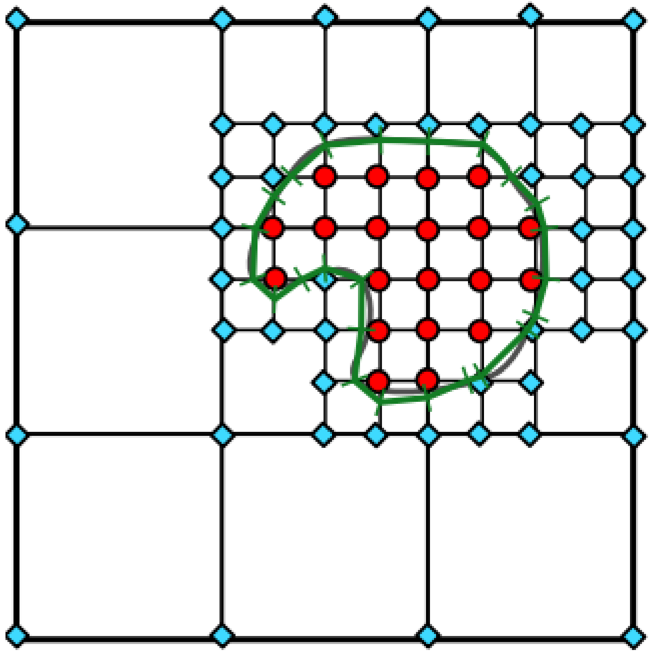
Learning implicit representation
Full 3D supervision

- Representing the 3D geometry as the decision boundary of a classifier that *learns* to separate the object's inside from its outside

- After *training* the weights of the neural net represent the surface

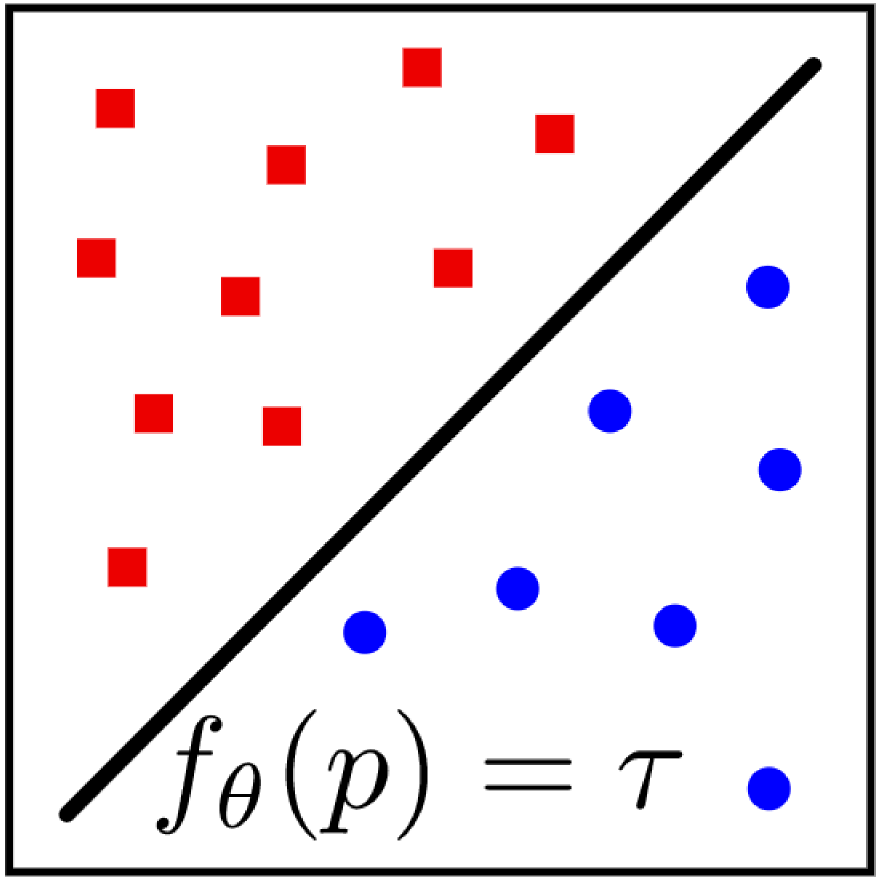
- This yields a continuous implicit surface representation

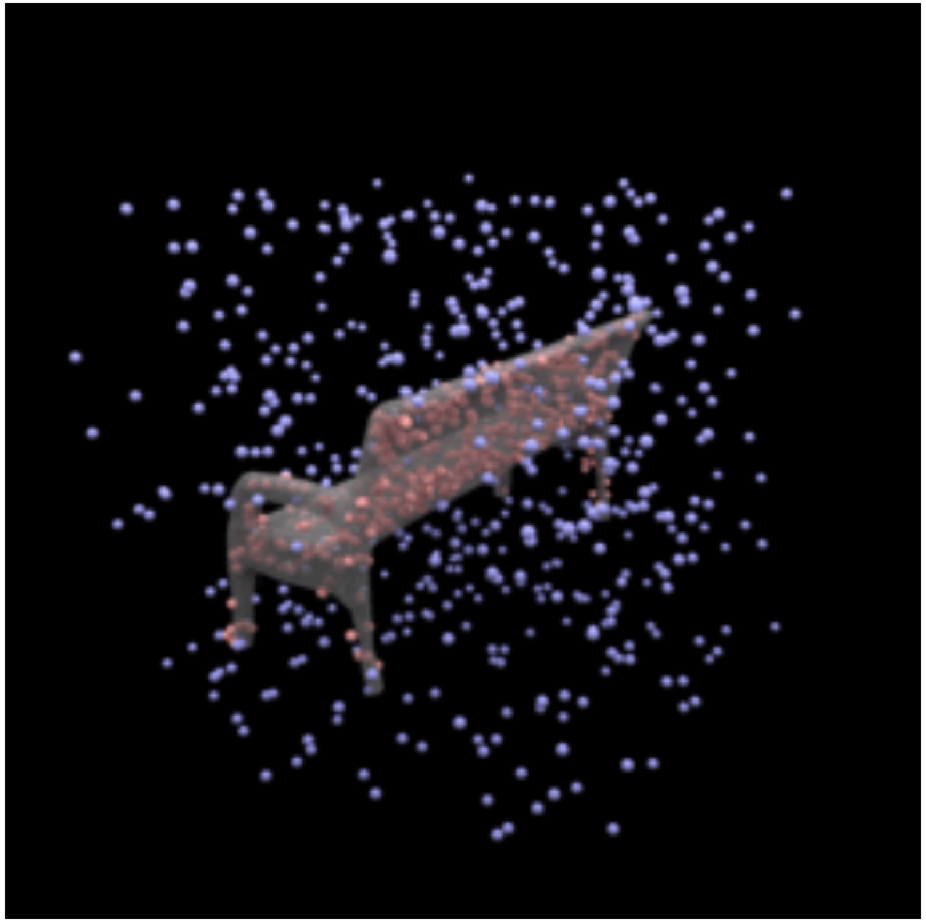
- *At inference*, queries of 3D points, allows to construct watertight meshes, by using marching cubes algorithm


marching cubes

Occupancy Networks: Learning 3D Reconstruction in Function Space, Mescheder et al, 2019

Learning implicit representation
Full 3D supervision





Occupancy Networks, Mescheder et al, 2019

Learning implicit representation
Full 3D supervision

Occupancy Networks, Mescheder et al, 2019

Learning implicit representation
Full 3D supervision

Occupancy network.

Learning non-linear function

$$f_\theta : \mathbb{R}^3 \rightarrow [0,1]$$

Input: $\boldsymbol{p} \in \mathbb{R}^3$
Output: probability of occupancy

The decision boundary, $f_\theta(\boldsymbol{p}) = \tau,\ (\tau = 0.5)$, represents the surface of the reconstructed shape

Occupancy Networks, Mescheder et al, 2019

Learning implicit representation
Full 3D supervision

(Recap)

Occupancy networks, Mescheder et al., 2019

- Full 3D supervision of the occupancy function is needed
- After *training* the weights of the neural net represent the surface
- *At inference*, queries of 3D points, allows to construct watertight meshes, by using marching cubes algorithm
- **Caveat.** Full 3D supervision is complicated and expensive

Learning implicit representation

By weak supervision, from the raw data

Point clouds

- given an input point cloud $\chi = \{x_i\}_{i \in I} \subset \mathbb{R}^3$
- our goal is to compute θ
- $f(x;\theta)$ is approximately the signed distance function to a plausible surface $\mathcal{M}$ defined by χ
- without any additional supervised data preparation

How?

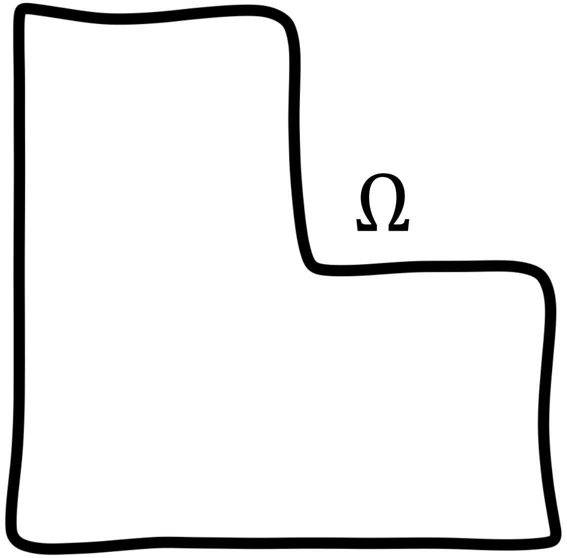
Learning implicit representation

By weak supervision

Eikonal PDE

$$\|\nabla f(\boldsymbol{x})\| = 1$$

$$f(\boldsymbol{x}) = 0, \boldsymbol{x} \in \Omega$$



Implicit geometric regularization **(IGR)** by Gropp, Yariv, Haim, Atzmon and Lipman 2020

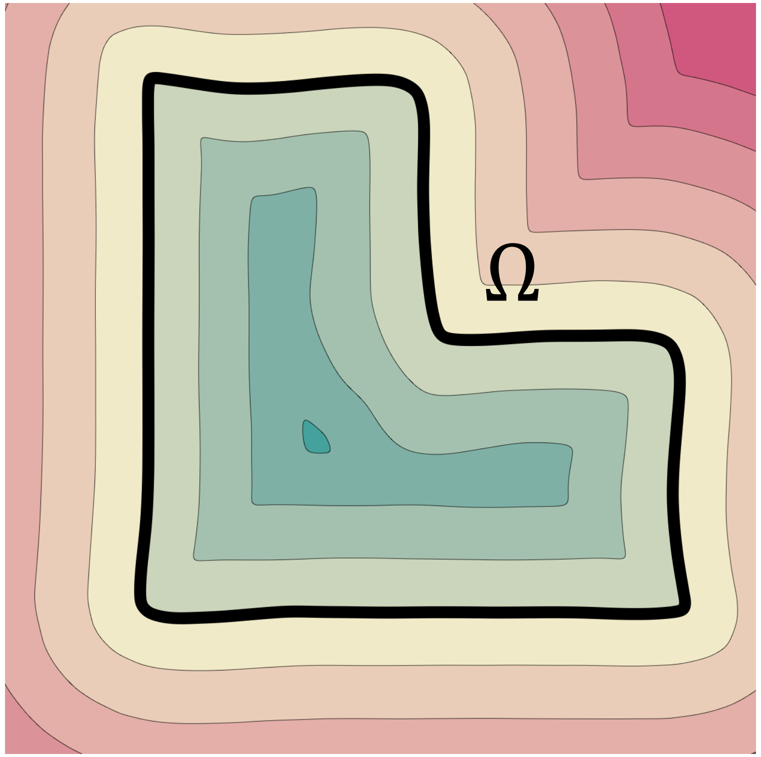
Learning implicit representation

By weak supervision

Eikonal PDE

$$\|\nabla f(\boldsymbol{x})\| = 1$$

$$f(\boldsymbol{x}) = 0, \boldsymbol{x} \in \Omega$$



Signed distance function (SDF)

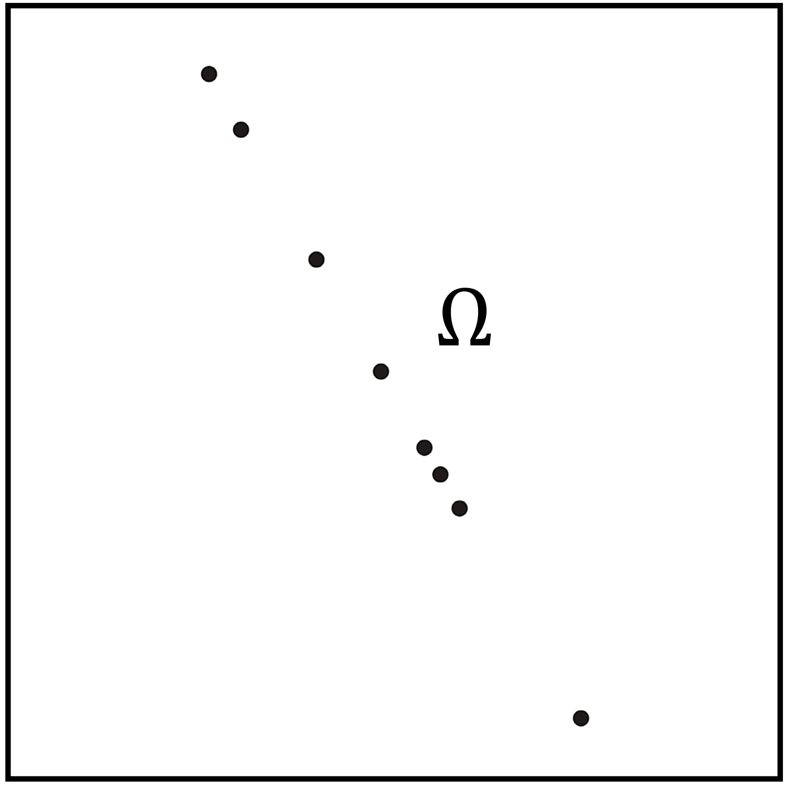
Learning implicit representation

By weak supervision

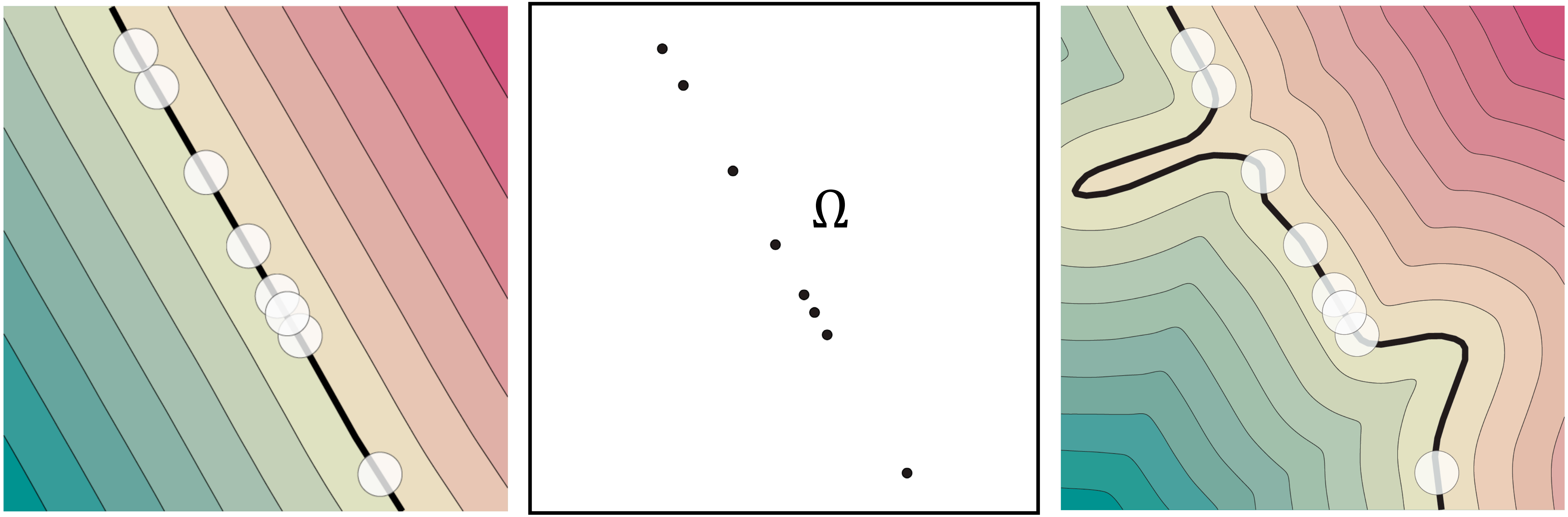
Eikonal PDE

$$\|\nabla f(\boldsymbol{x})\| = 1$$

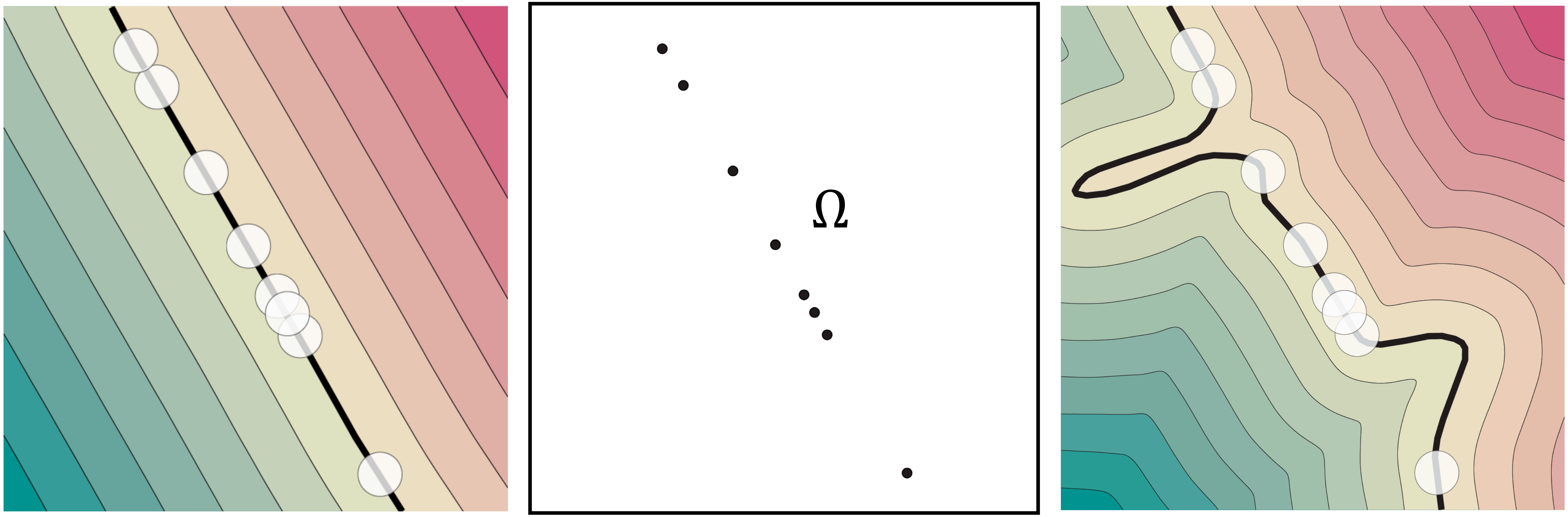
$$f(\boldsymbol{x}) = 0, \boldsymbol{x} \in \Omega$$



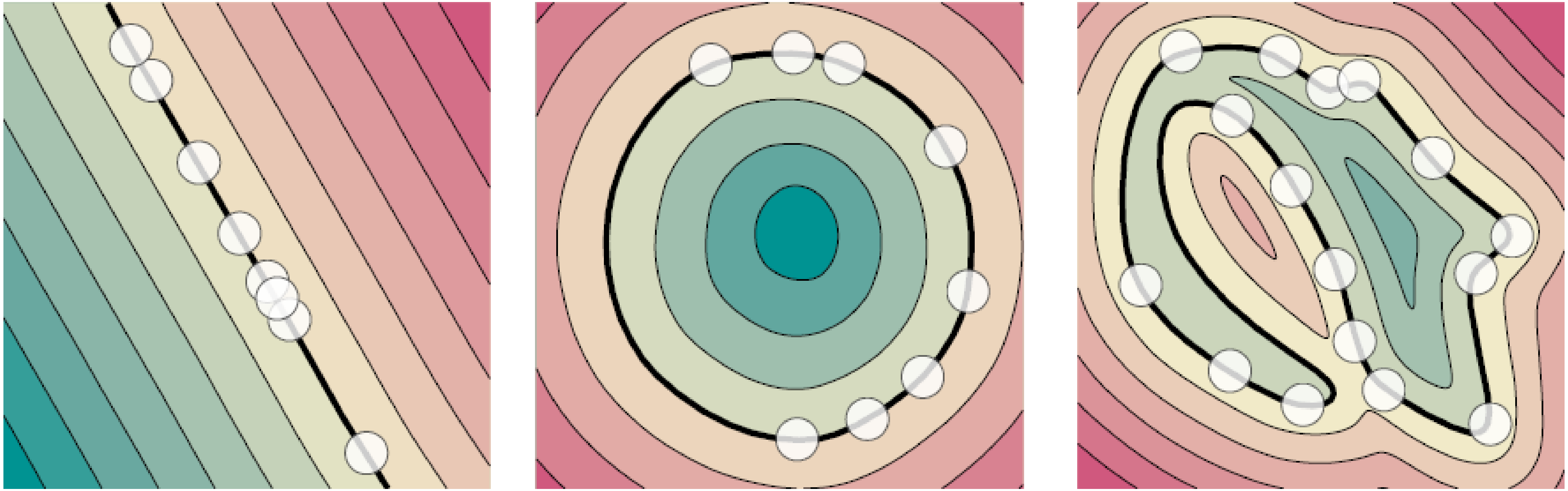
Implicit geometric regularization **(IGR)** by Gropp, Yariv, Haim, Atzmon and Lipman 2020


Ω

$$\text{loss}(\theta) = \sum_{i\in I} \underbrace{|f(x_i;\theta)|^2}_{\text{vanish}} + \underbrace{\lambda \mathbb{E}_x(\|\nabla_x f(x;\theta)\| - 1)^2}_{\text{Eikonal}}$$

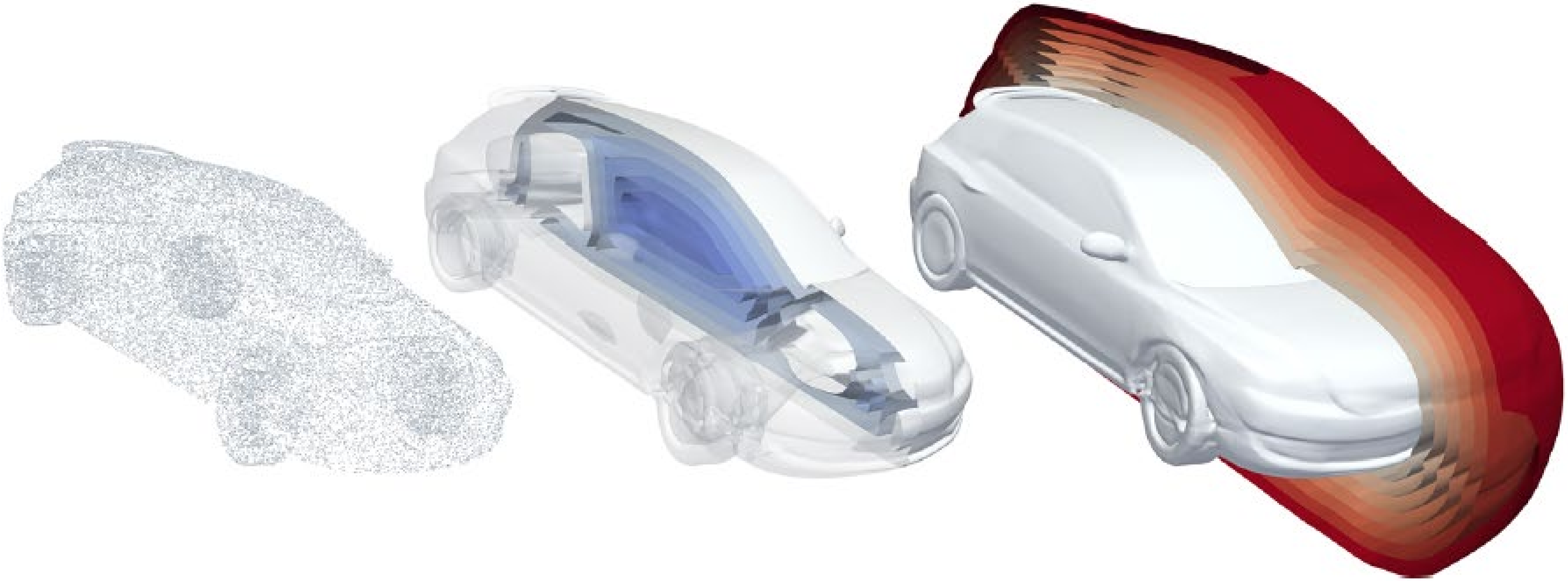


$$\text{loss}(\theta) = \sum_{i \in I} \underbrace{|f(x_i;\theta)|^2}_{\text{vanish}} + \underbrace{\lambda \mathbb{E}_x (\|\nabla_x f(x;\theta)\| - 1)^2}_{\text{Eikonal}}$$



Weak supervision

Implicit geometric regularization **(IGR)**, Gropp et al., 2020



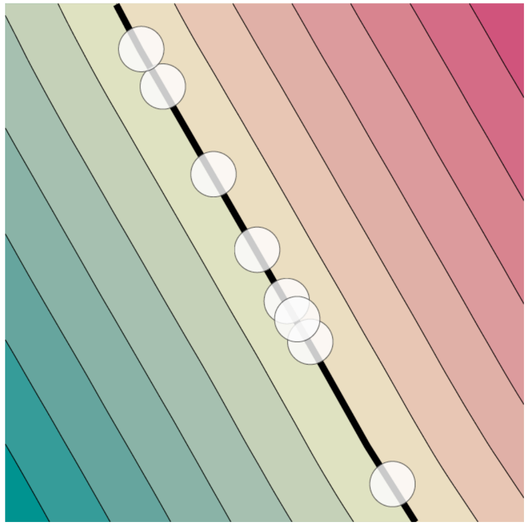
Weak supervision

Implicit geometric regularization **(IGR)**, Gropp et al., 2020

Inductive bias

Theorem (Convergence and linear reproduction)
Gradient descent of the linear model with random initialization converges with probability 1 to the reproducing plane

$$\mathrm{loss}(\theta) = \sum_{i \in I} \left(w^T x_i\right)^2 + \lambda\left(\|w\|^2 - 1\right)^2$$

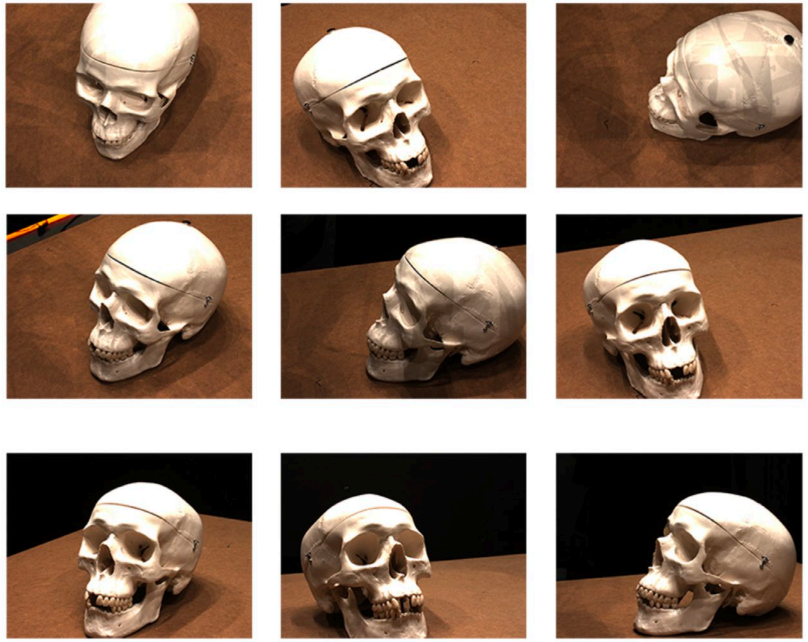


Learning implicit neural representation

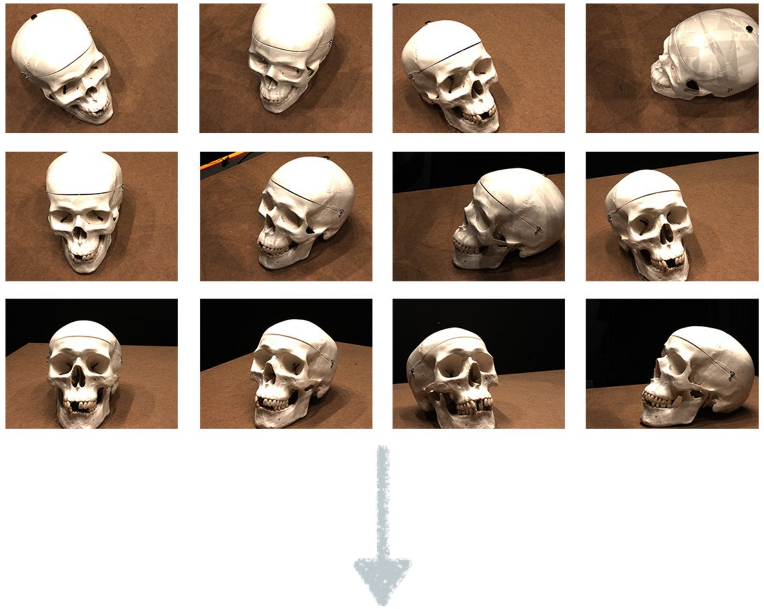
By weak supervision, from the raw data

Point clouds

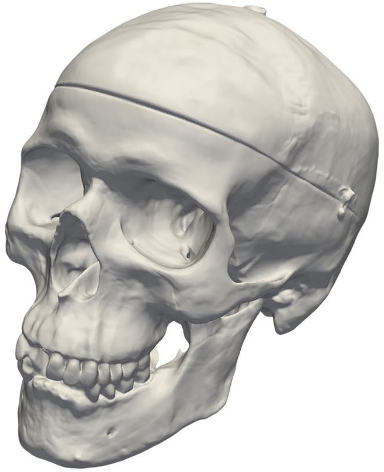
Images



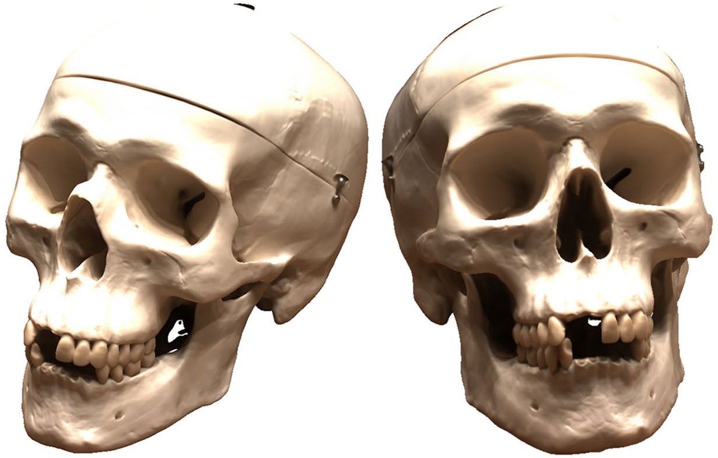
Neural rendering



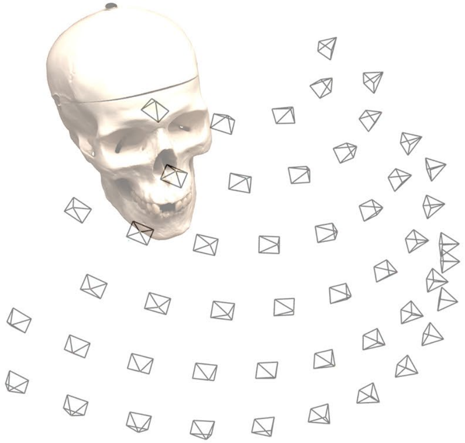
Geometry reconstruction



Render new views



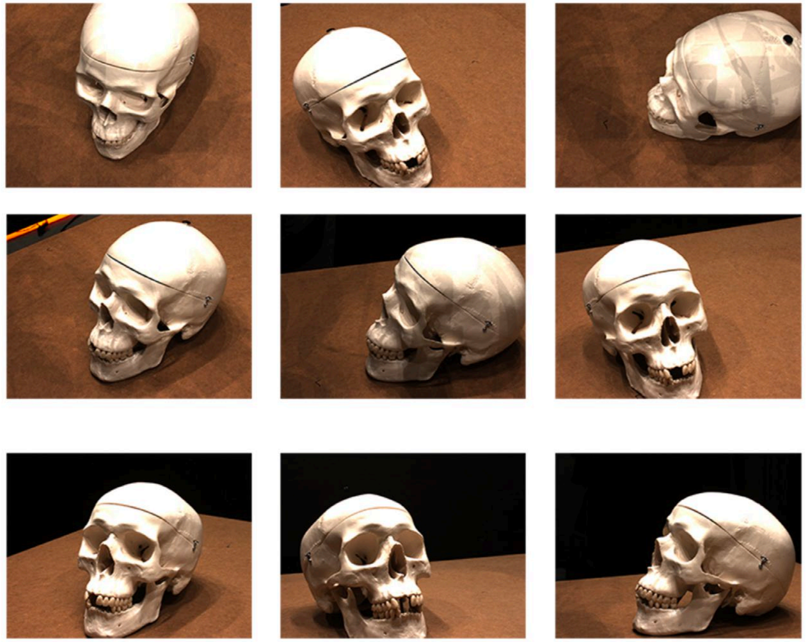
Cameras



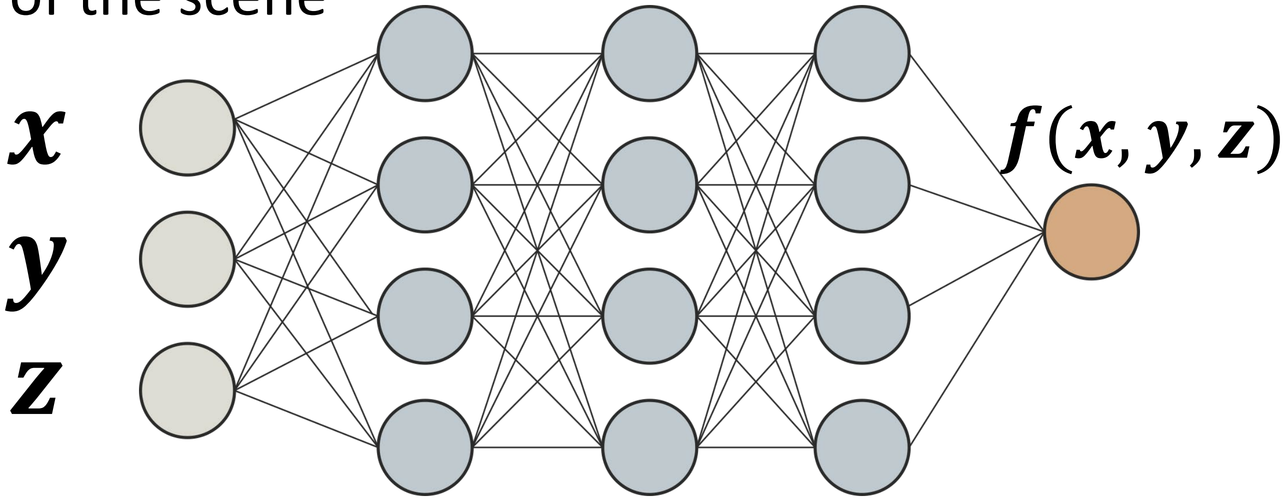
Slide by Lior Yariv

Neural rendering

- Learning from raw data (weak supervision)



- Building (implicit) neural representation of the scene



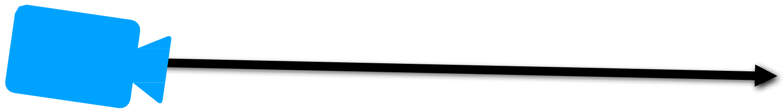
Neural Volumetric Rendering

Slides on volume rendering formulation
by Ben Mildenhall

Neural Volumetric Rendering

*c*omputing color along

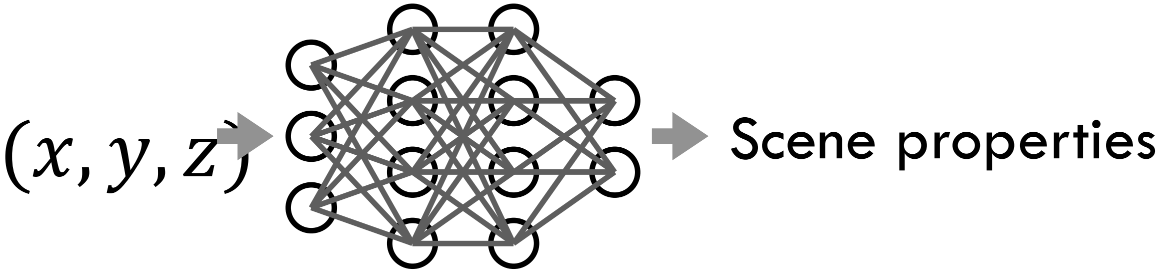
rays through 3D space



What color is this pixel?

Neural Volumetric Rendering

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

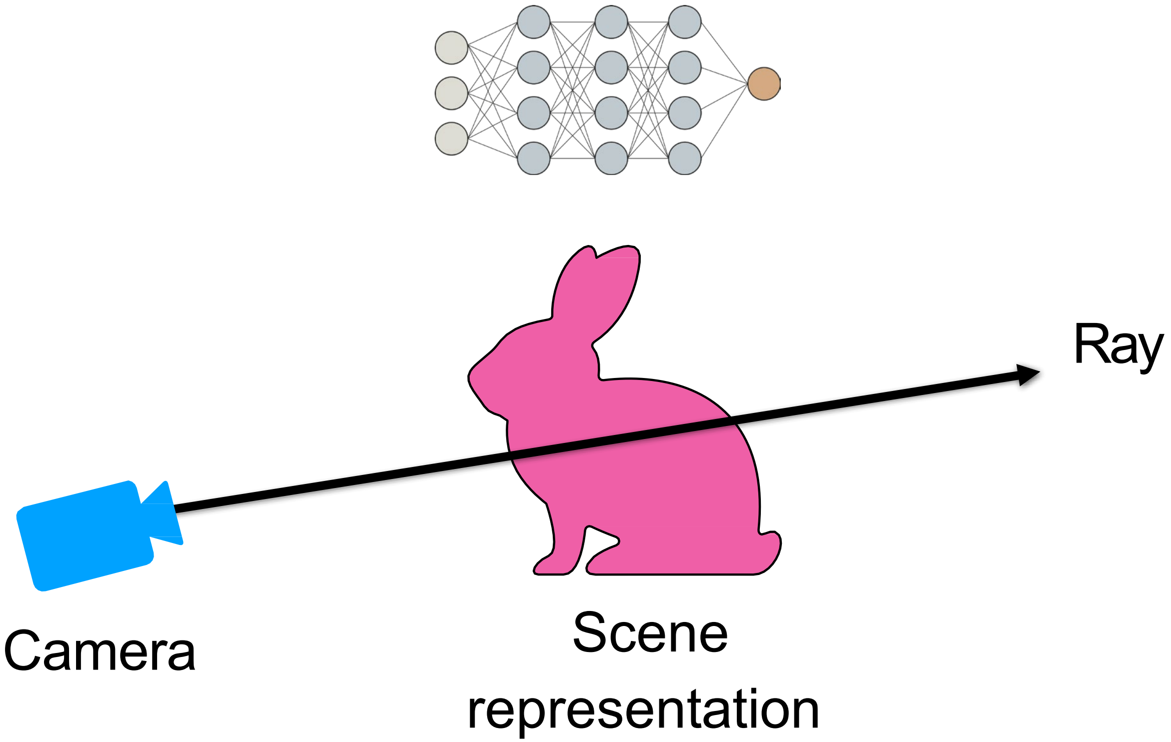


(x, y, z) → → Scene properties

Neural Volumetric Rendering

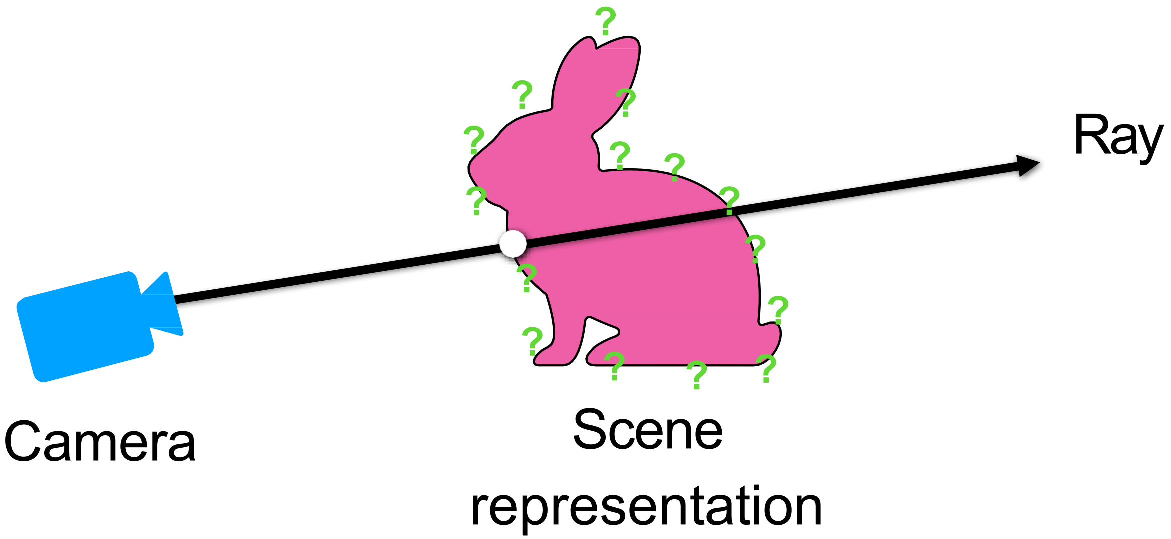
continuous, differentiable
rendering model without
concrete ray/surface intersections

Neural rendering



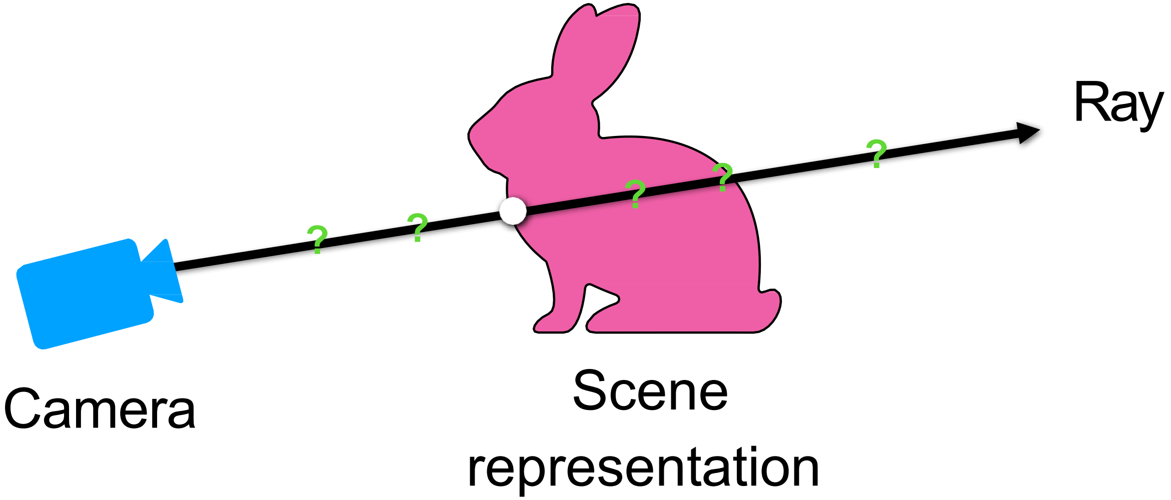
Want to know how ray interacts with scene

Neural rendering - surface vs. volume rendering



Surface rendering — loop over geometry, check for ray hits

Neural rendering - surface vs. volume rendering



Volume rendering — loop over ray points, query geometry

Neural Volumetric Rendering

NeRF

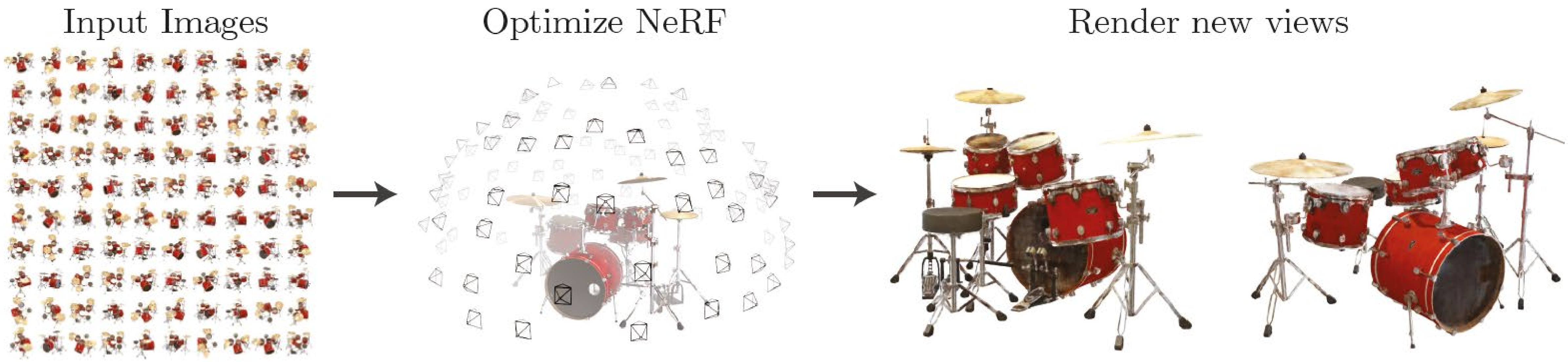
Representing Scenes as **Ne**ural **R**adiance **F**ields for View Synthesis

By Mildenhall, Srinivasan, Tancik, Barron, Ramamoorth and Ng, 2020

NeRF

Representing Scenes as Neural Radiance Fields for View Synthesis

By Mildenhall, Srinivasan, Tancik, Barron, Ramamoorth and Ng, 2020



A NeRf stores a volumetric *scene representation as the weights of an MLP*, trained on many images with known pose

Slides on NeRF are based on slides of Yoni Kasten and Dolev Ofri

NeRF

Inference

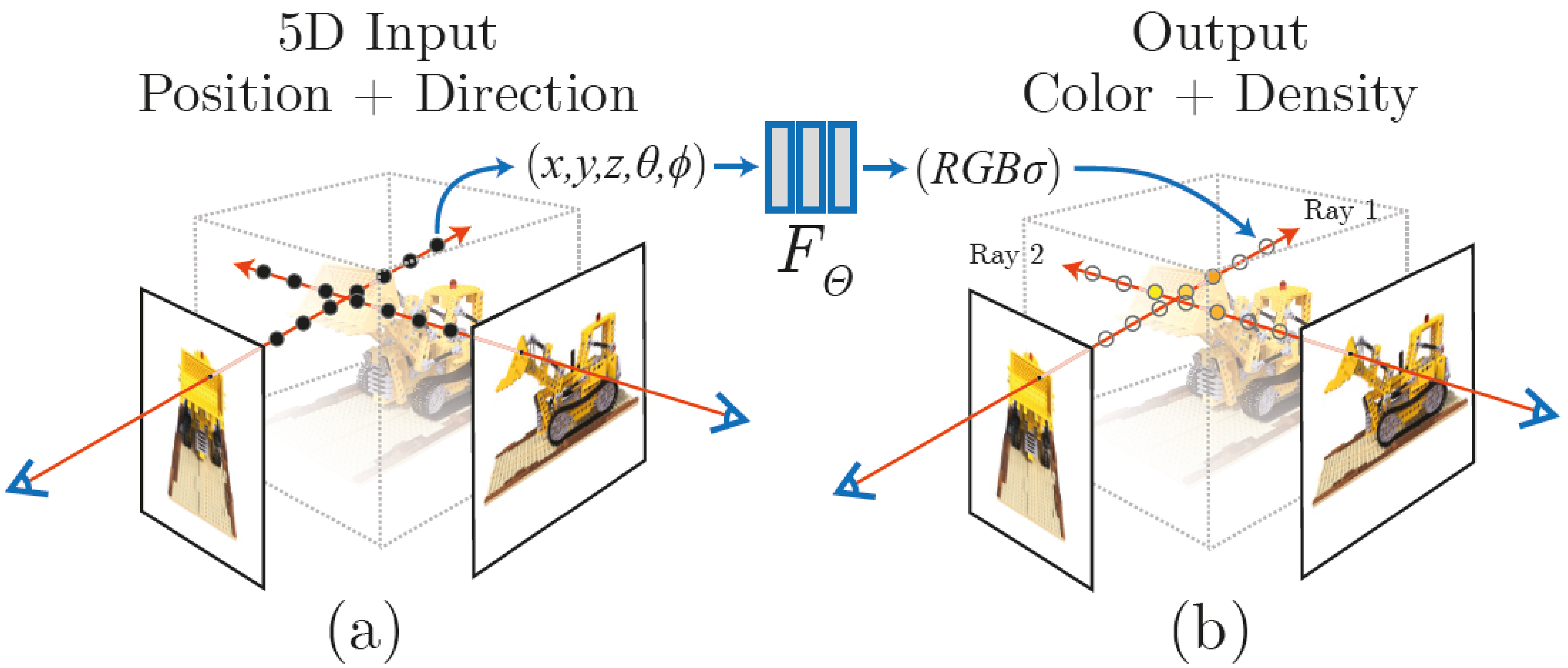
The scene is represented by MLP

Input: spatial location (x, y, z) and viewing direction (θ, ϕ)

Output: volume density (opacity), radiance emitted at direction (θ, ϕ) at point (x, y, z)

NeRF

Inference: *render new photorealistic images from the learned scene*



New views are rendered by integrating the density and color at regular intervals along each viewing ray (volume rendering)

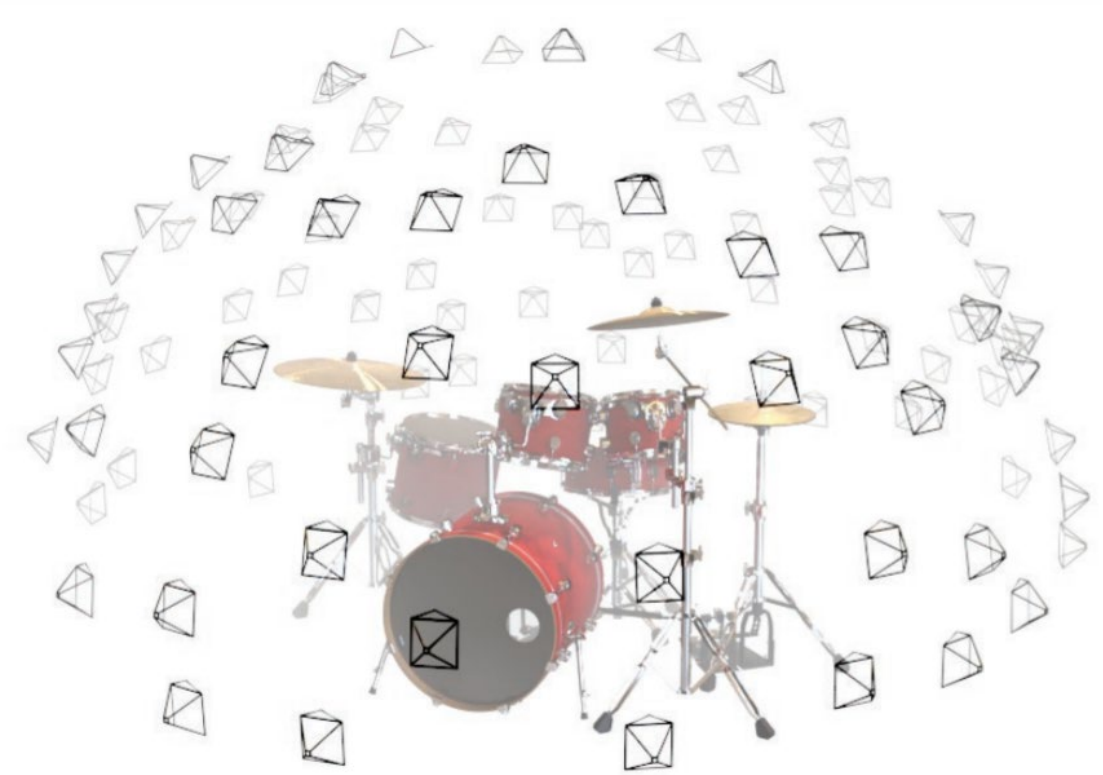
NeRF

Training

Objective: reconstruct all training views by volume rendering

Multiview Images of a single scene

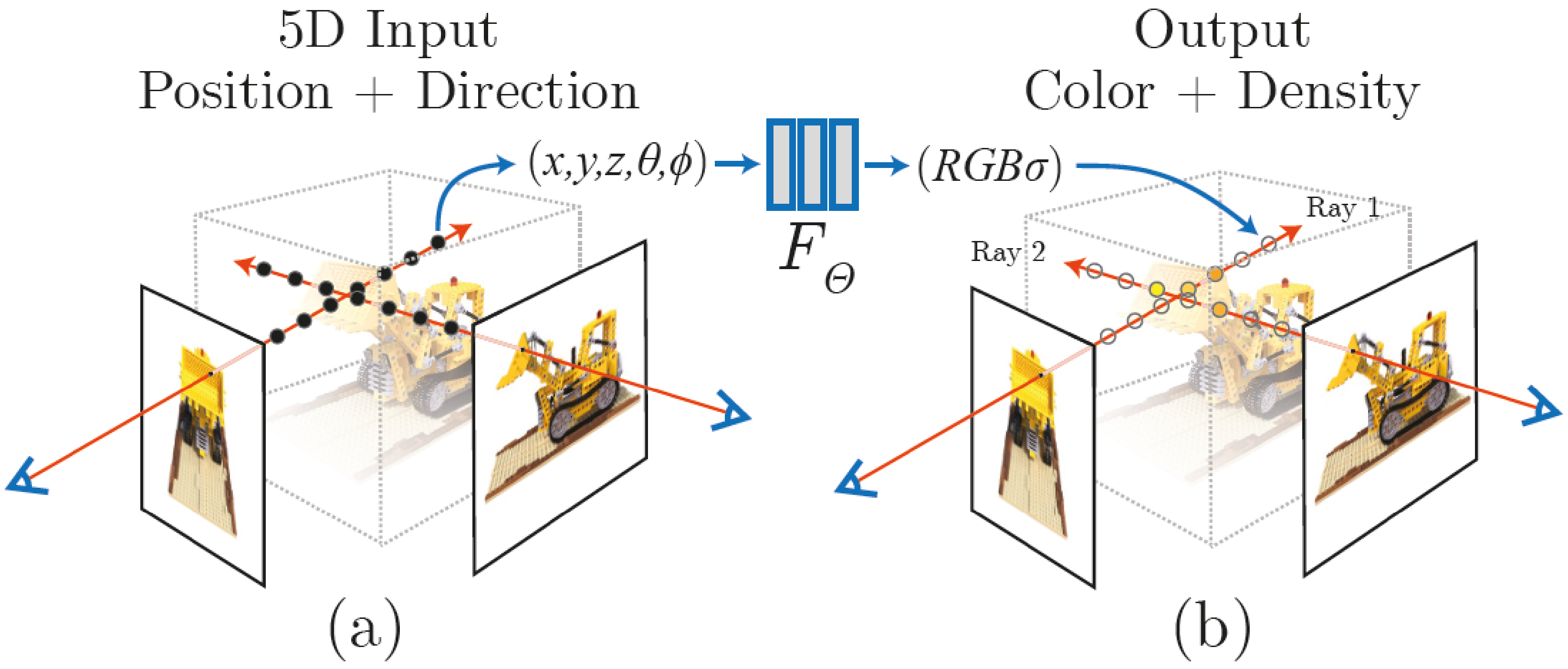
Camera poses



NeRF

Training

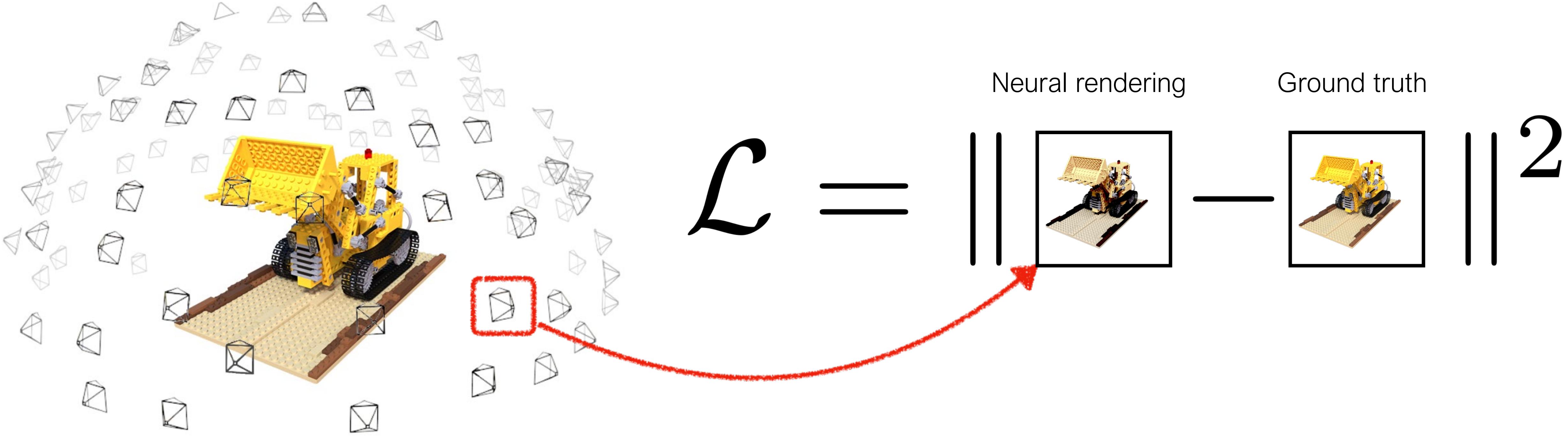
reconstruct all training views by differentiable volume rendering



NeRF

Training Loss

Simulate the rendering of a learned neural scene representation in a differentiable way, and minimize:



NeRF

Neural volume rendering

Neural volume rendering refers to methods that generate images by tracing a ray into the scene and taking an integral over the length of the ray

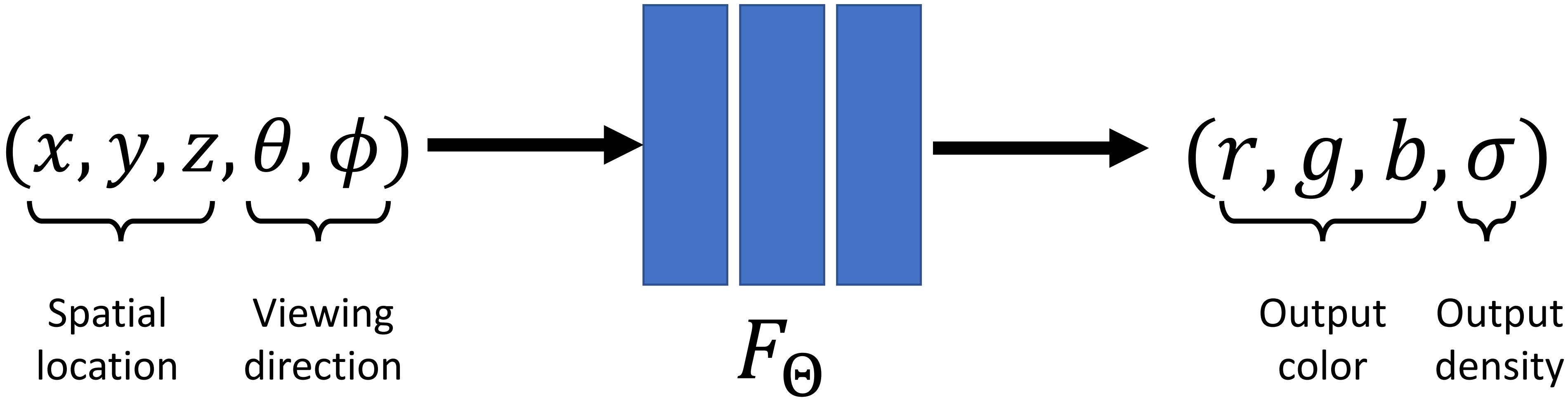
A neural network (MLP) encodes a function from the 3D coordinates on the ray to quantities like density and color, which are integrated to yield an image

Two key properties:

- Integration over the ray
- Coordinate-based scene representation

NeRF

Scene representation



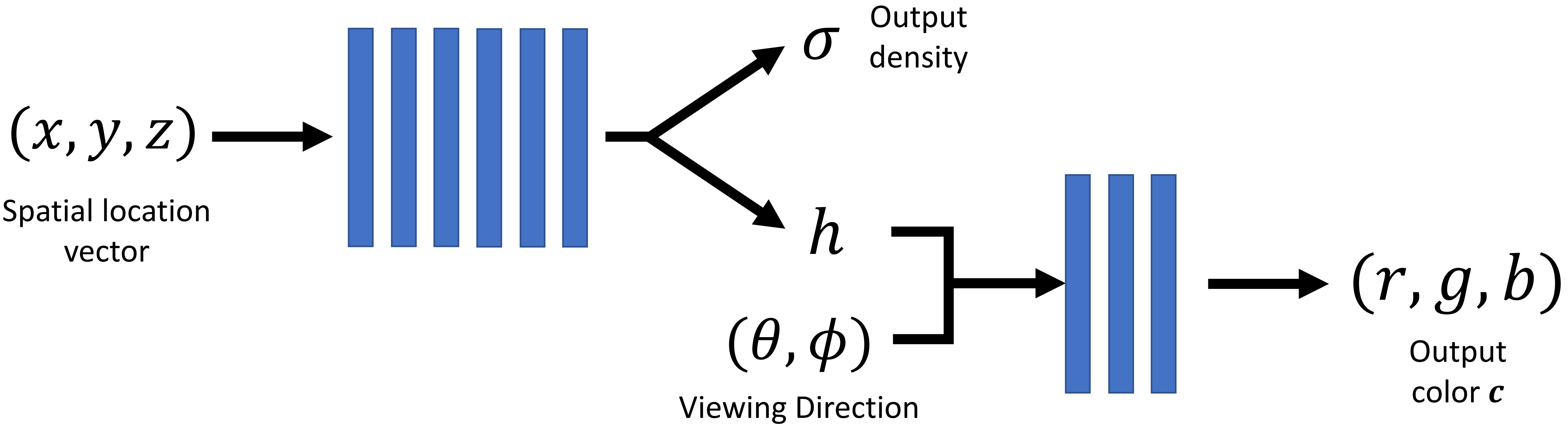
$$(\underbrace{x, y, z}_{\text{Spatial location}}, \underbrace{\theta, \phi}_{\text{Viewing direction}}) \longrightarrow F_{\Theta} \longrightarrow (\underbrace{r, g, b}_{\text{Output color}}, \underbrace{\sigma}_{\text{Output density}})$$

Multi-Layered Perceptron (MLP)
9 layers
256 channels

Slide credit: Jon Barron’s talk

NeRF

Scene representation

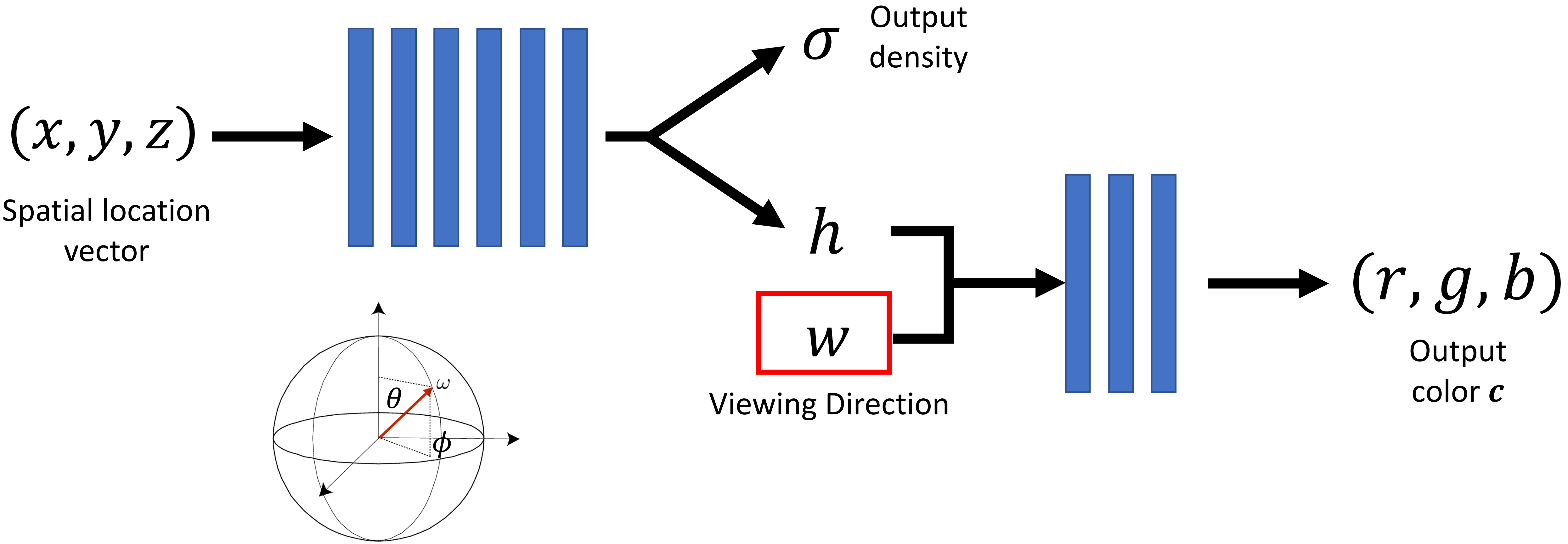


σ (spatial location)

$\boldsymbol{c}$ (spatial location, viewing direction)

NeRF

Scene representation



(x, y, z)

Spatial location vector

σ Output density

h

w

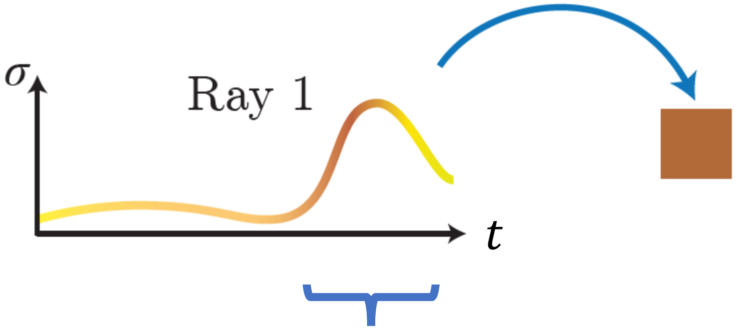
Viewing Direction

(r, g, b)

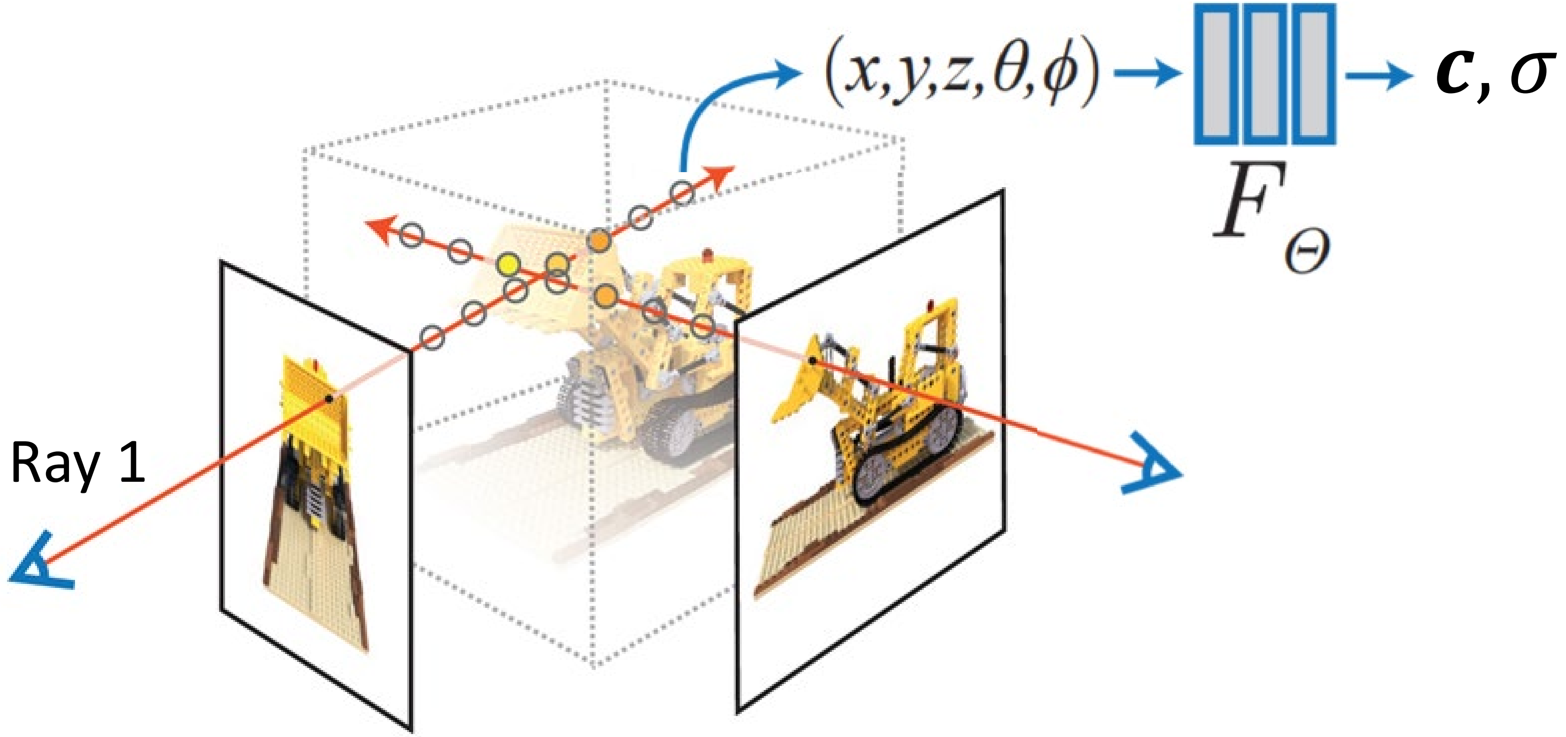
Output color $\boldsymbol{c}$

NeRF

Volume rendering



First hit of the ray

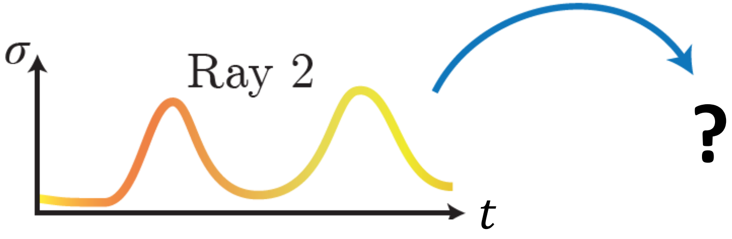


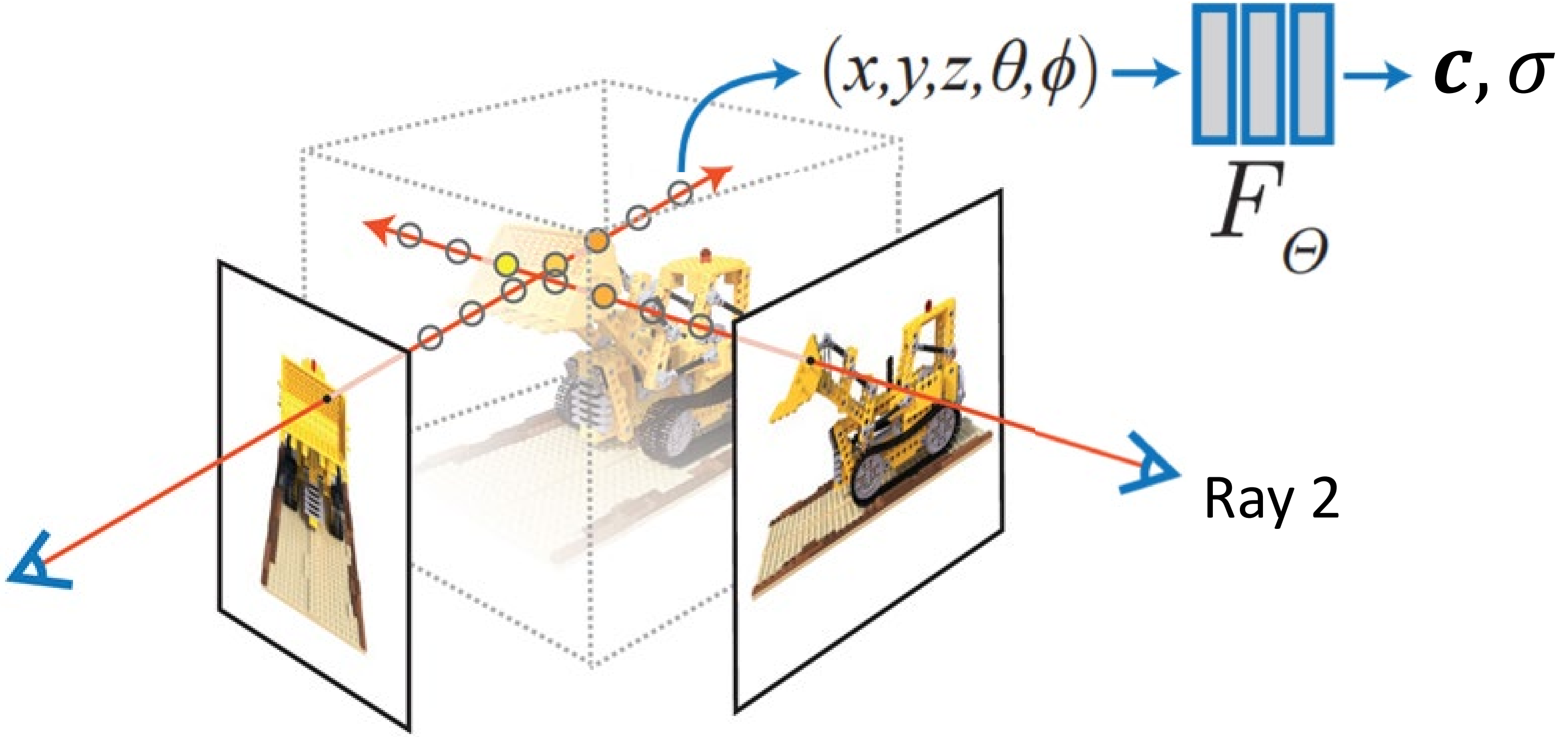
$\boldsymbol{r}(t)$ – camera ray $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$

σ – volume density

NeRF

Volume rendering





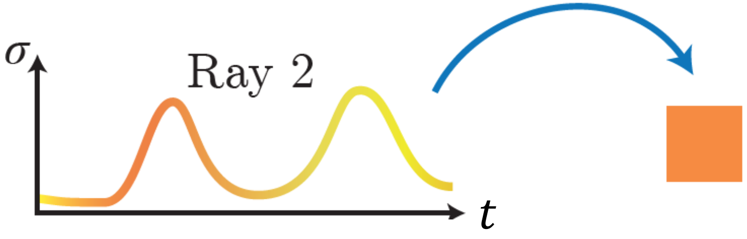
$\boldsymbol{r}(t)$ – camera ray $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$

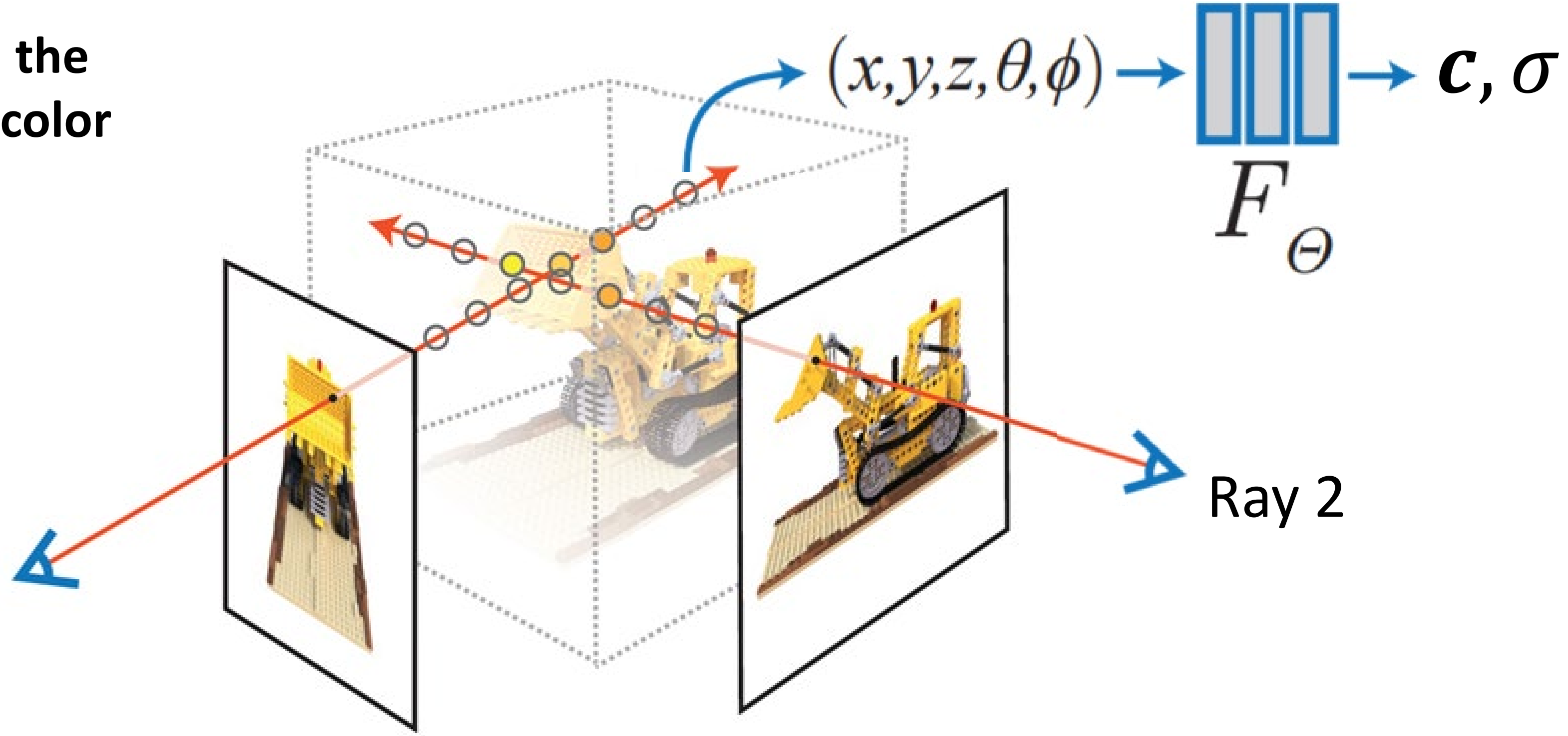
σ – volume density

NeRF

Volume rendering

We need to combine the density and the visibility in order to get the required color



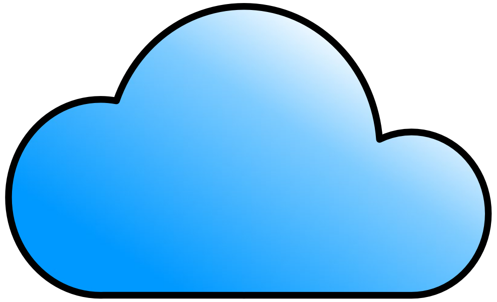


$\boldsymbol{r}(t)$ – camera ray $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$

σ – volume density

NeRF

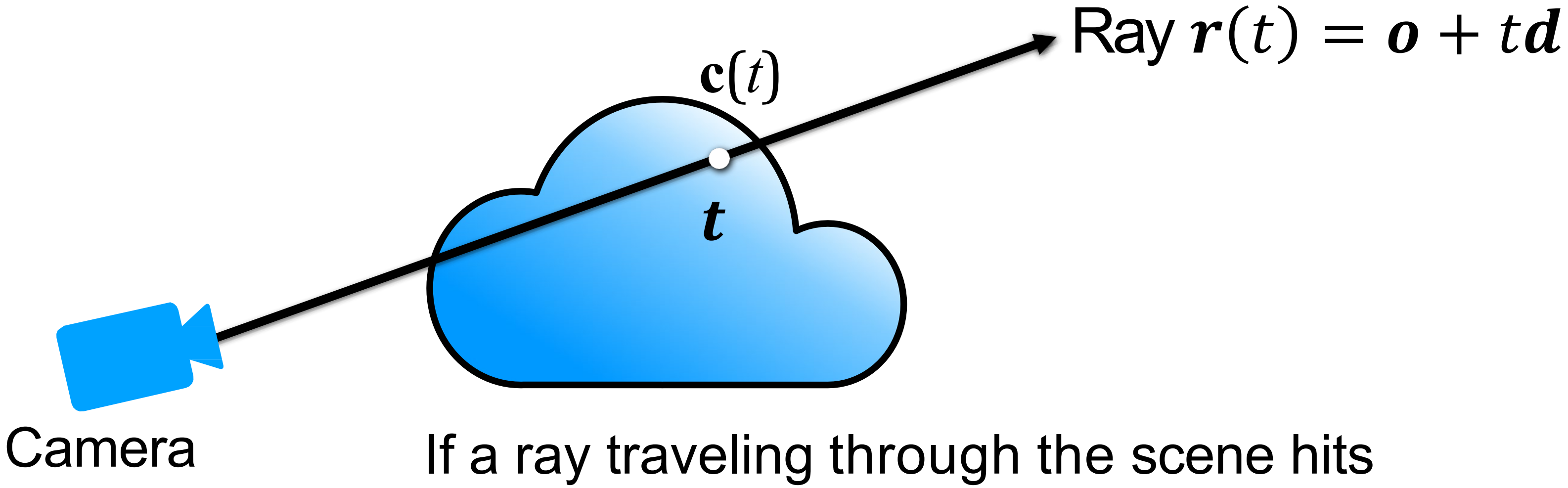
Volume rendering formulation



Scene is a cloud of tiny colored particles

NeRF

Volume rendering formulation

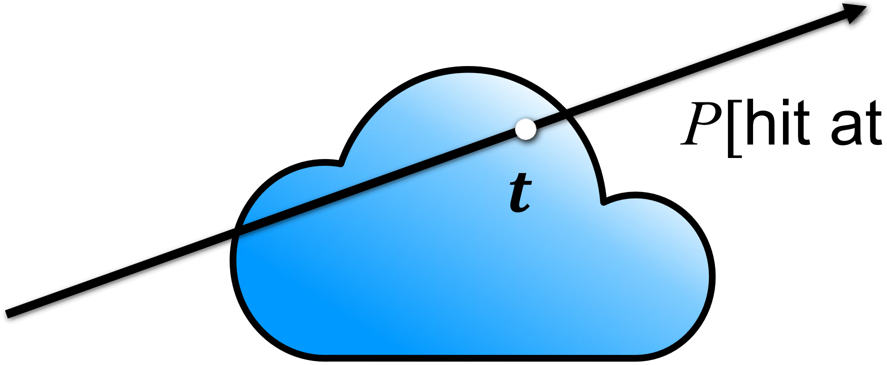


If a ray traveling through the scene hits a particle at distance t along the ray, we return its color $\mathbf{c}(t)$

NeRF

Volume rendering formulation

What does it mean for a ray to "hit" the volume?



$P[\text{hit at } t] = \sigma(t)\,dt$

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

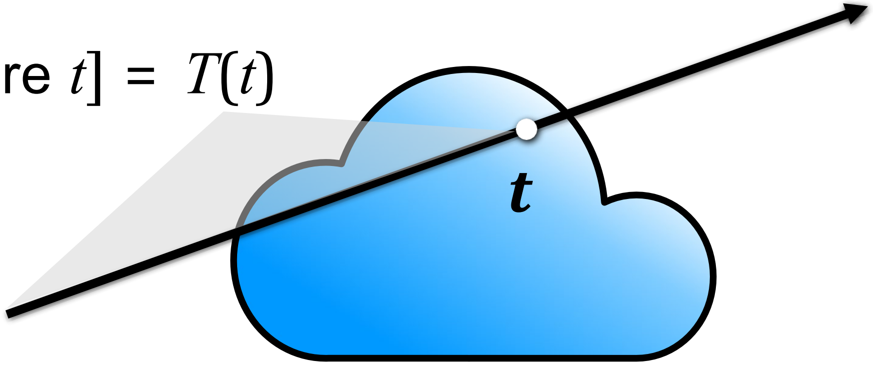
σ is called the "volume density"

NeRF

Volume rendering formulation

Probabilistic interpretation

P[no hits before t] = $T(t)$



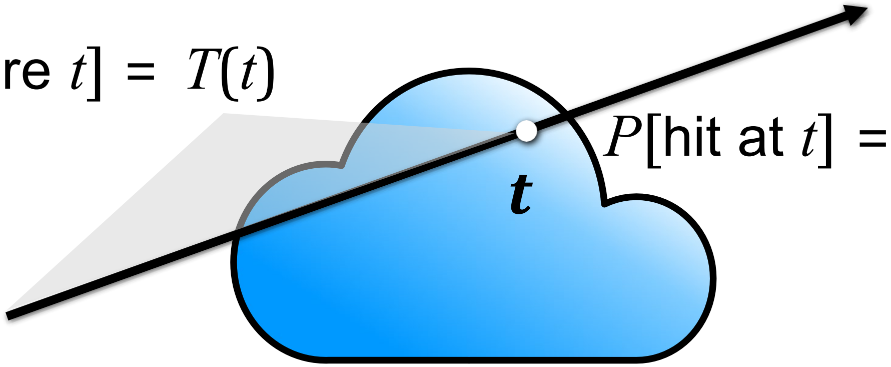
To determine if t is the *first* hit along the ray, need to know $T(t)$: the probability that the ray makes it through the volume up to t.

$T(t)$ is called "transmittance"

NeRF

Volume rendering formulation

Probabilistic interpretation



P[no hits before t] = $T(t)$

P[hit at t] = $\sigma(t)\,dt$

The product of these probabilities tells us

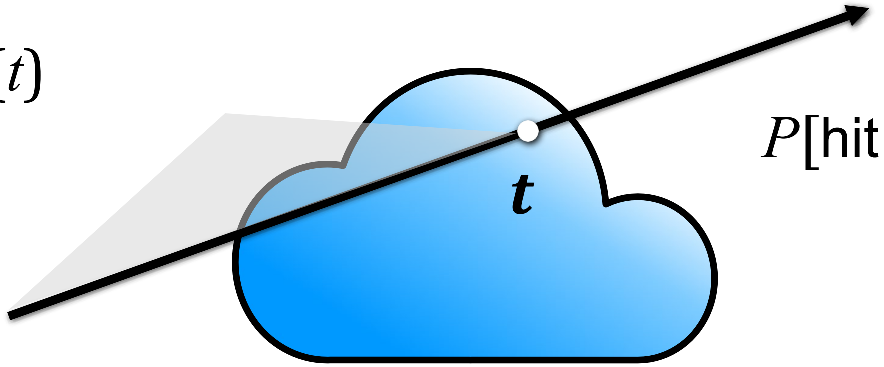
P[first hit at t] = P[no hit before t] × P[hit at t] = $T(t)\sigma(t)dt$

NeRF

Volume rendering formulation

Calculating T given σ

P[no hits before t] = $T(t)$



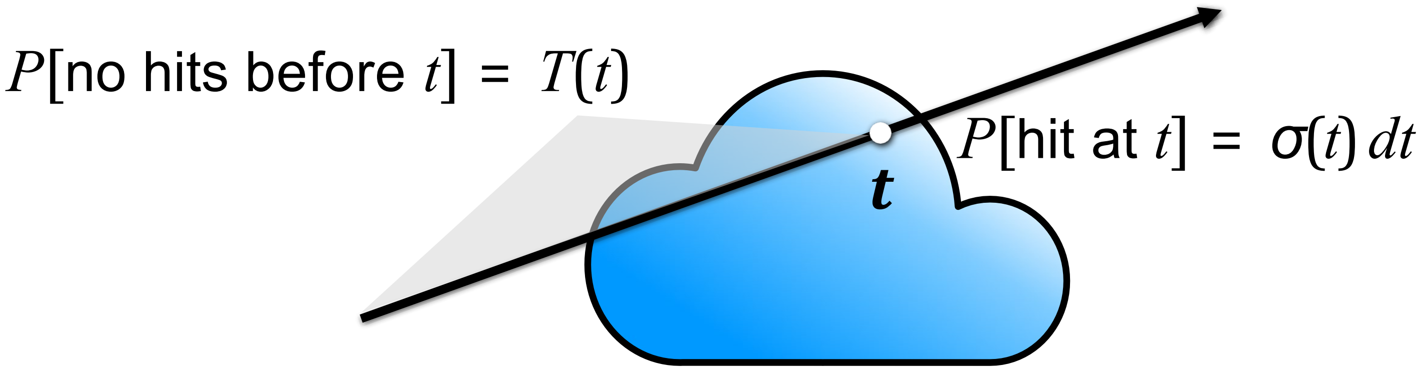
P[hit at t] = $\sigma(t)\,dt$

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

NeRF

Volume rendering formulation

Probabilistic interpretation



P[no hits before t] = $T(t)$
P[hit at t] = $\sigma(t)\,dt$
$\boldsymbol{t}$

Finally, we can write the probability that a ray terminates at $\boldsymbol{t}$ as a function of only the density σ

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

$$= T(t)\sigma(t)dt$$

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

NeRF

Volume rendering formulation

Expected value of color along ray

This means the expected color returned by the ray will be

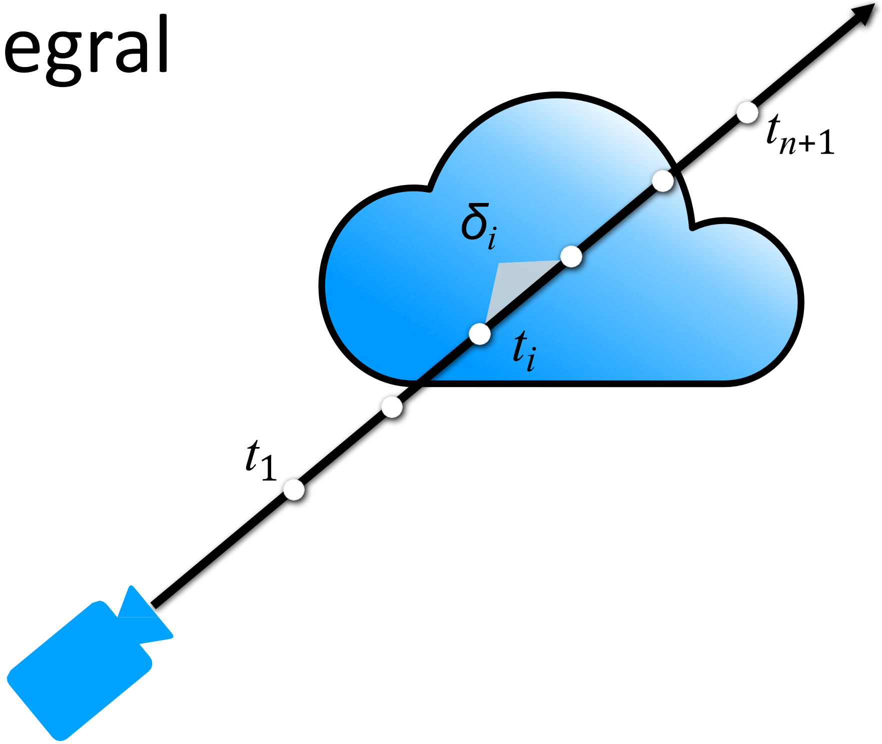
$$\int_{t_s}^{t_e} T(t)\sigma(t)c(t)dt$$

Note the nested integral!

NeRF

Volume rendering formulation

Approximating the integral



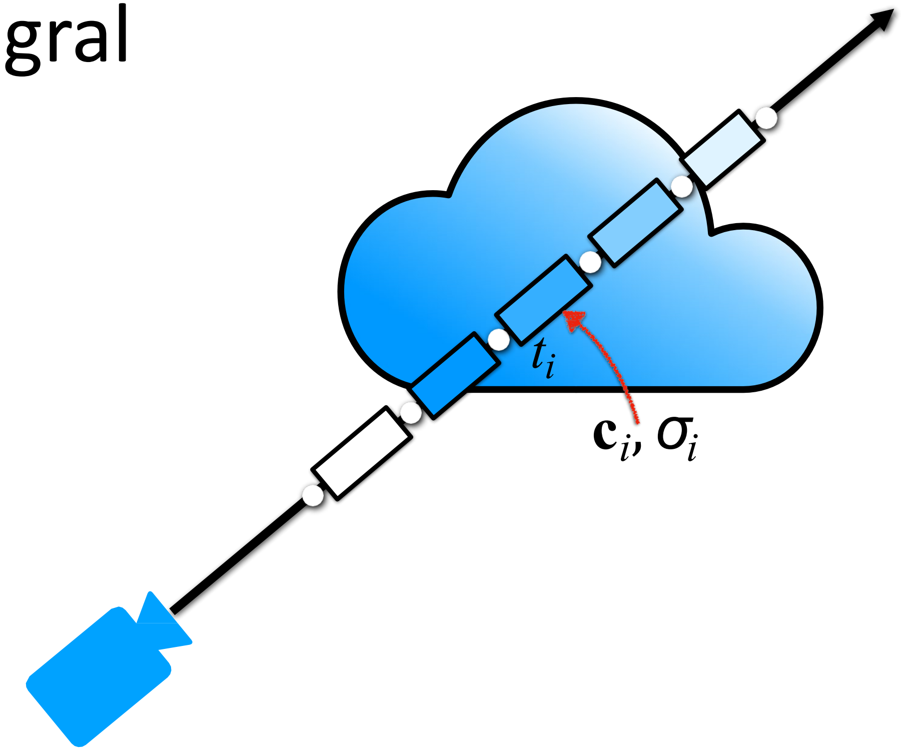
Approximate the nested integral,
splitting the ray up into n segments with endpoints $\{t_1, t_2, \ldots, t_{n+1}\}$
with lengths $\delta_i = t_{i+1} - t_i$

NeRF

Volume rendering formulation

Approximating the integral

$$\int_{t_s}^{t_e} T(t)\sigma(t)c(t)dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i c_i dt$$



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

NeRF

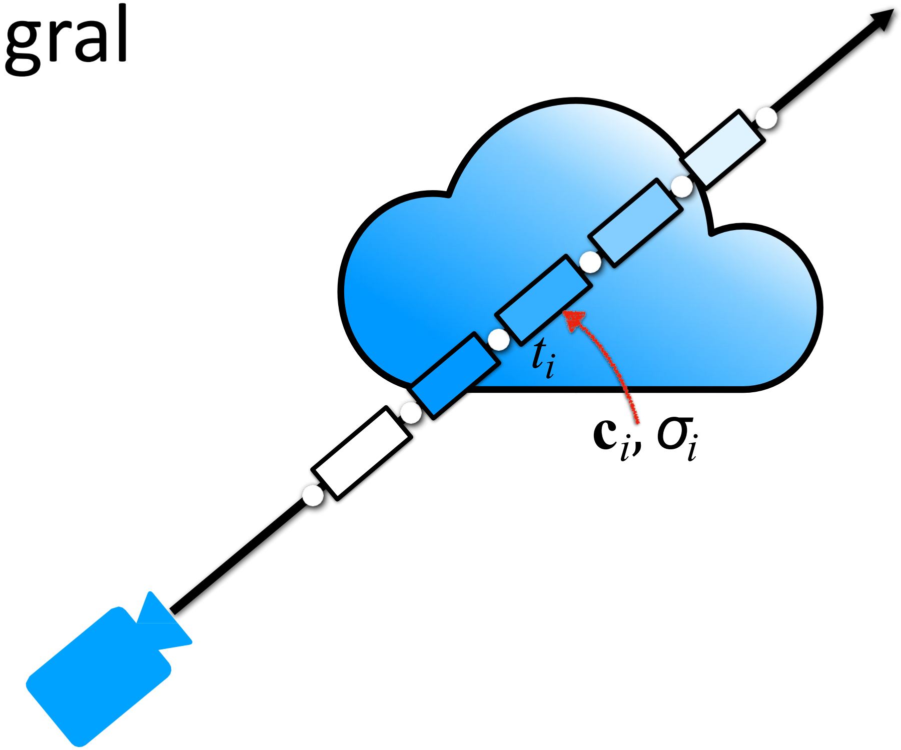
Volume rendering formulation

Approximating the integral

$$\int_{t_s}^{t_e} T(t)\sigma(t)c(t)dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} \boxed{T(t)}\sigma_i c_i dt$$

<u>Caveat:</u> piecewise constant density and color **do not** imply constant transmittance $T(t)$!

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



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

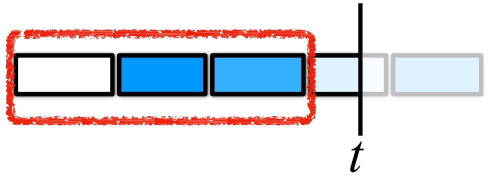
NeRF

Volume rendering formulation

Evaluating T for piecewise constant density σ

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

We need to evaluate at continuous t values
that can lie *partway through* an interval



NeRF

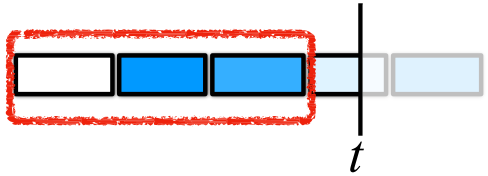
Volume rendering formulation

Evaluating T for piecewise constant density σ

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 light is blocked by all previous segments?"



NeRF

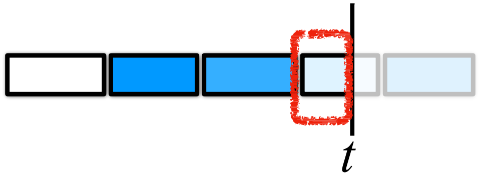
Volume rendering formulation

Evaluating T for piecewise constant density σ

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

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

“How much light is blocked partway through the current segment?”



NeRF

Volume rendering formulation

Approximating the integral

$$\int T(t)\sigma(t)c(t)dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i c_i dt = \sum_{i=1}^{n} T_i \sigma_i c_i \int_{t_i}^{t_{i+1}} \exp\big(-\sigma_i(t - t_i)\big)\, dt$$

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

NeRF

Volume rendering formulation

Connection to material opacity

$$= \sum_{i=1}^{n} T_i c_i \underbrace{(1 - \exp(-\sigma_i \delta_i))}_{\text{segment opacity } \alpha_i}$$

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

$$\mathrm{Color}_{\mathrm{ray}} = \sum_{i=1}^{n} T_i \alpha_i c_i$$

NeRF

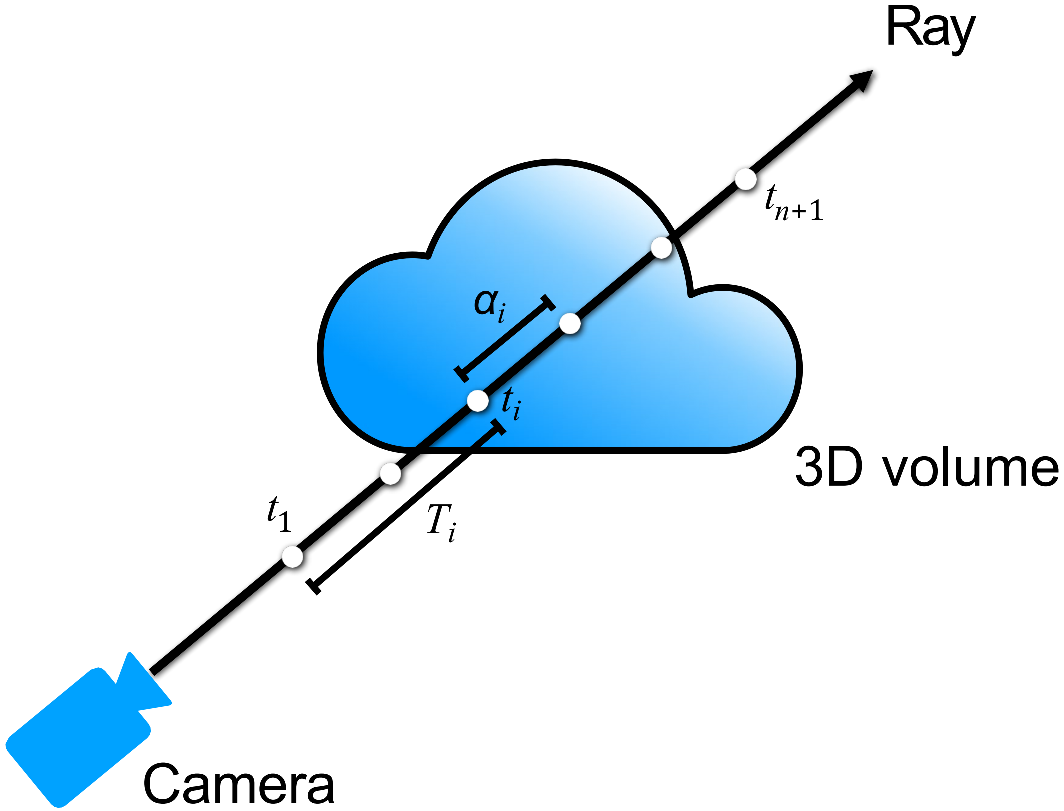
Volume rendering formulation

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

$$C_{\text{ray}} = \sum_{i=1}^{n} \underbrace{T_i \alpha_i}_{\text{Rendering weights}} \underbrace{c_i}_{\text{colors}}$$

Rendering weights

colors



How much light is transmitted earlier along ray:

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

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

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

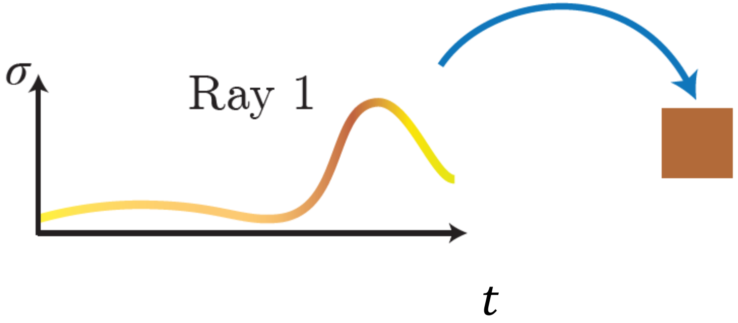
$$\text{Color}_{\text{ray}} = \sum_{i=1}^{n} T_i \alpha_i c_i$$

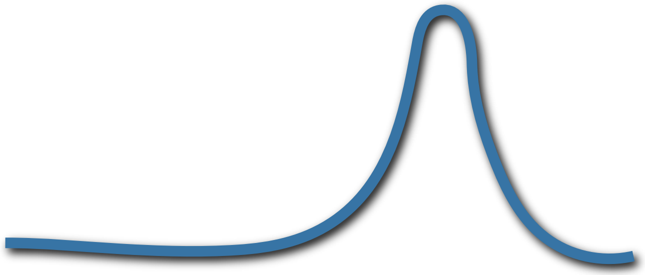
How much light is transmitted earlier along ray:

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

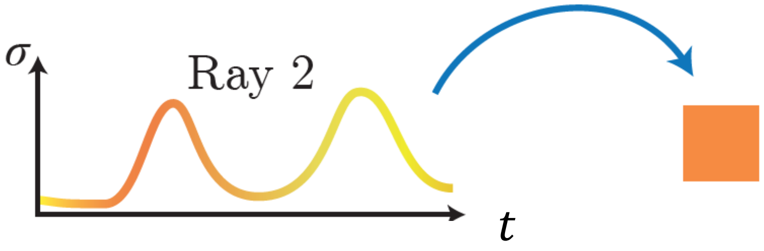
How much light is contributed by ray segment ***i***:

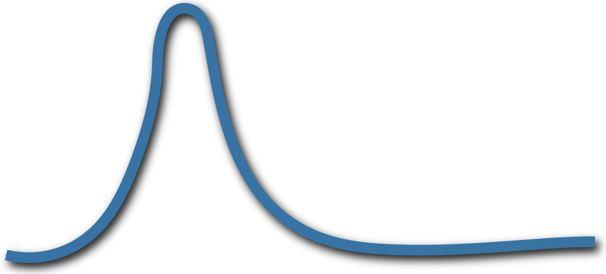
$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$





Rendering weights?





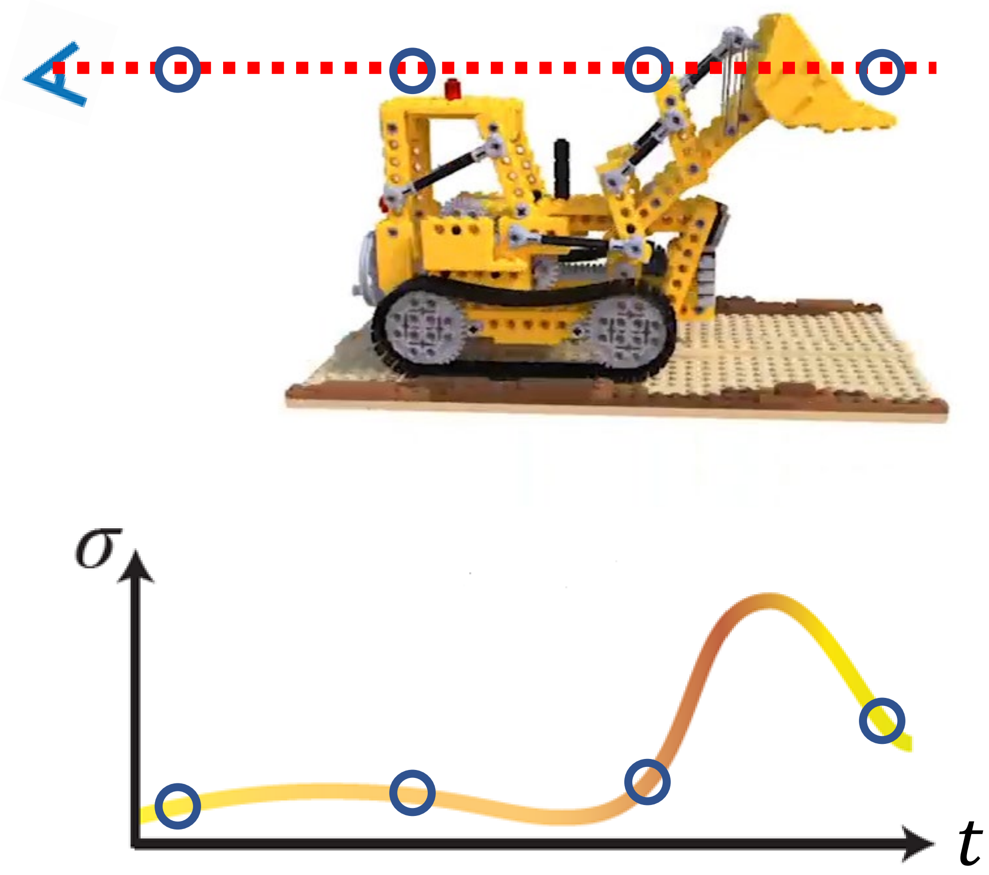
Rendering weights?

NeRF

Sampling along the ray

Sparse uniform sampling

→ Low accuracy

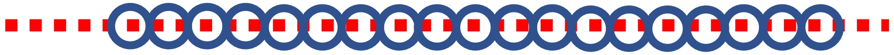


NeRF

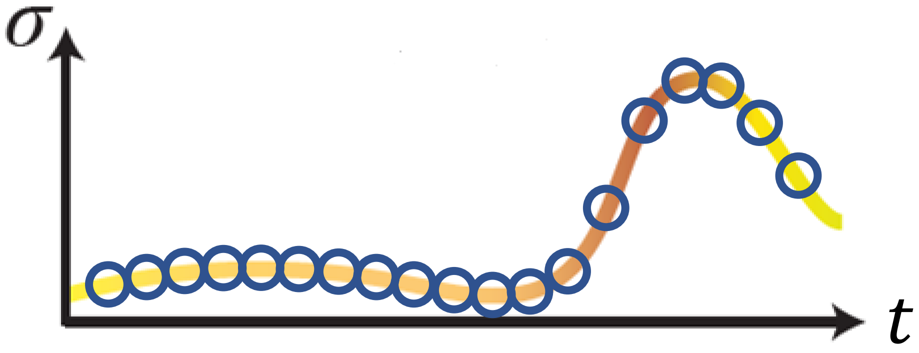
Sampling along the ray

Dense uniform sampling

→ Inefficient



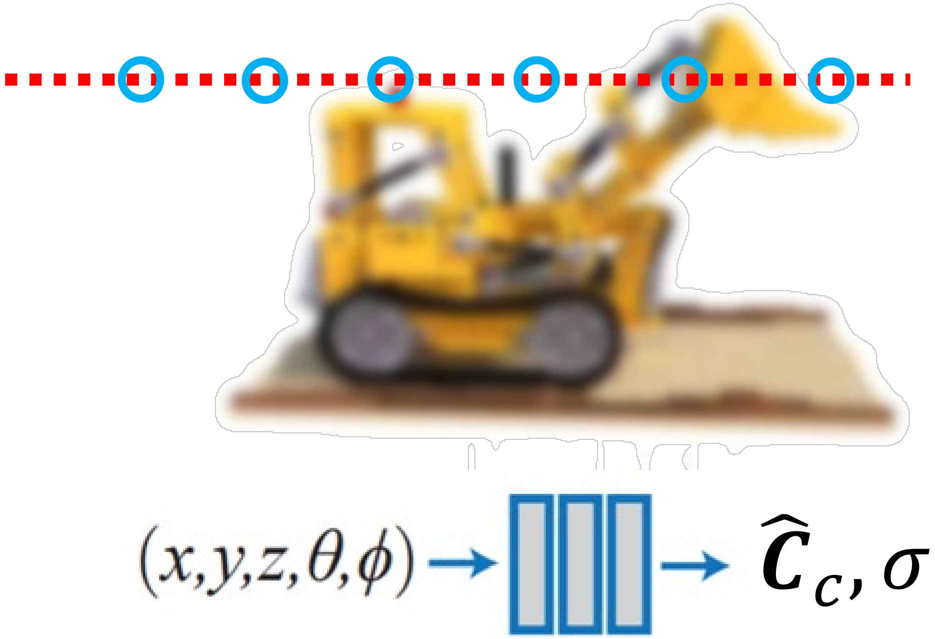
<u>Uniform sampling:</u>
free space and occluded regions that do not contribute to the rendered image are still sampled equally



NeRF

Fine and coarse sampling along the ray

Uniform samples

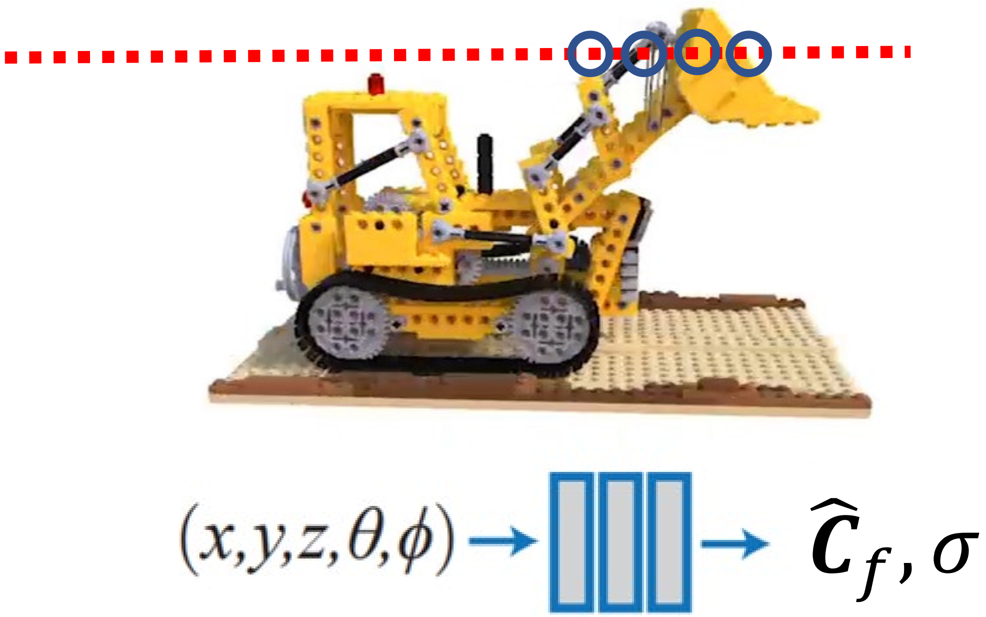


$(x,y,z,\theta,\phi) \rightarrow \boxed{\ } \rightarrow \hat{C}_c, \sigma$

$F_{\Theta c}$

Coarse NeRF

Non-uniform samples



$(x,y,z,\theta,\phi) \rightarrow \boxed{\ } \rightarrow \hat{C}_f, \sigma$

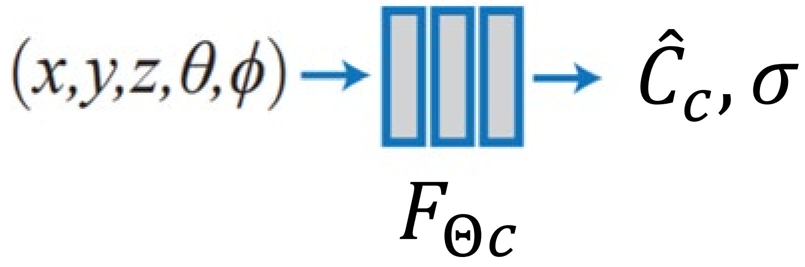
$F_{\Theta f}$

Fine NeRF

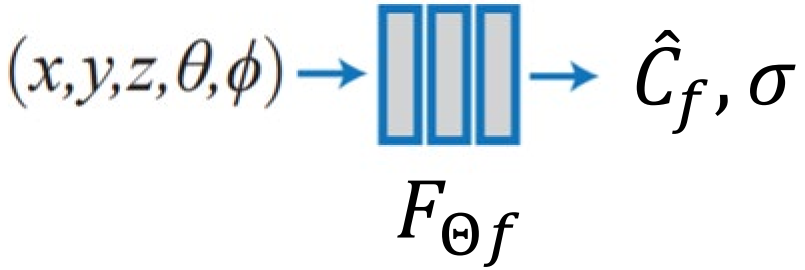
Nerf

Fine and coarse sampling along the ray

Train two networks



Coarse NeRF

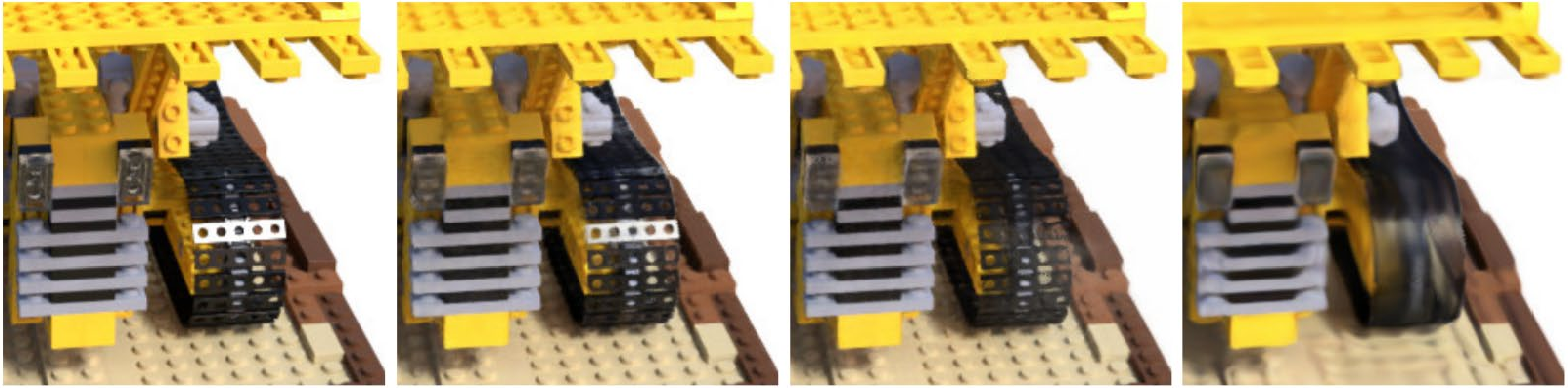


Fine NeRF

$$Loss = \sum_{r \in \mathcal{R}} \left(\left\| \hat{C}_c(r) - C(r) \right\|_2^2 + \left\| \hat{C}_f(r) - C(r) \right\|_2^2 \right)$$

NeRF

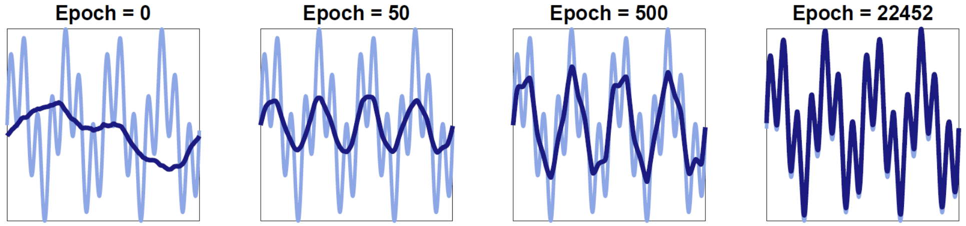
Ablation study



Ground Truth   Complete Model   No View Dependence   No Positional Encoding

NeRF

Positional encoding



Basri et al., NeurIPS 2019

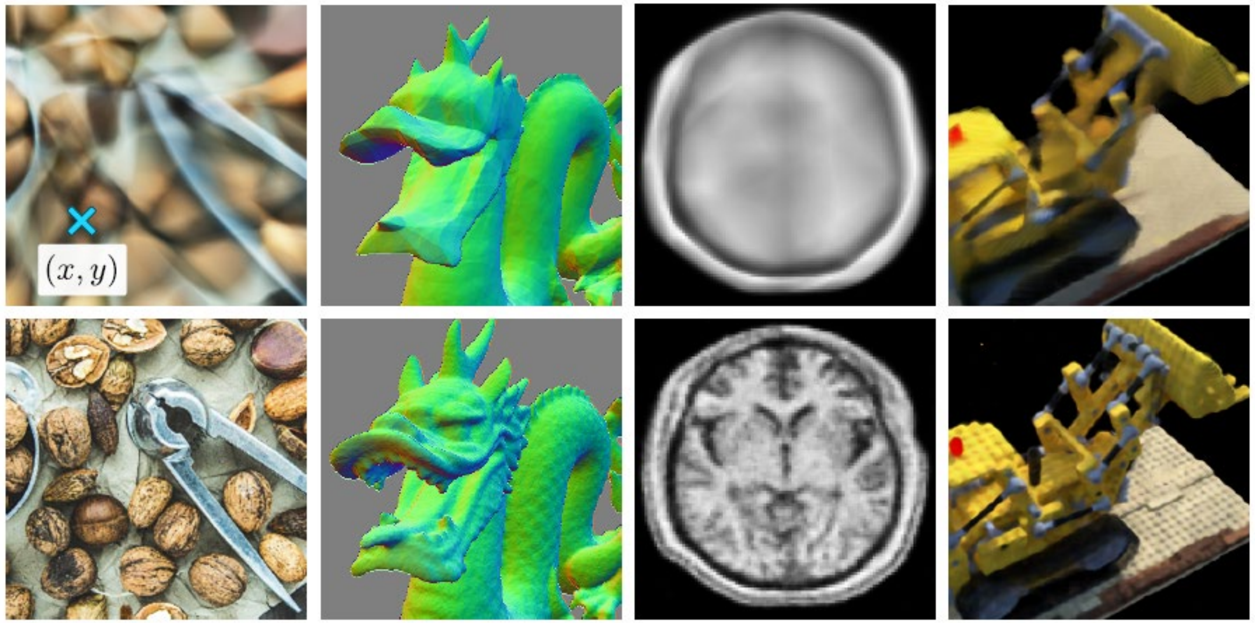
Spectral Bias

FC network fits the lower frequency component of the target function faster than the higher frequencies

Challenge

How to get MLPs converged faster on high-frequency target functions?

Tancik et al., NeurIPS 2020



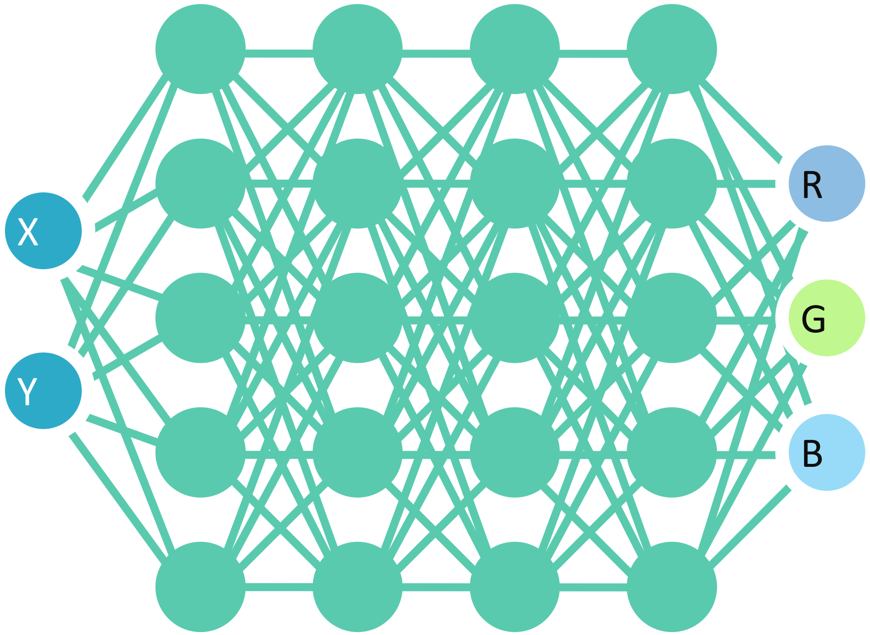
(b) Image regression $(x,y) \rightarrow$ RGB

(c) 3D shape regression $(x,y,z) \rightarrow$ occupancy

(d) MRI reconstruction $(x,y,z) \rightarrow$ density

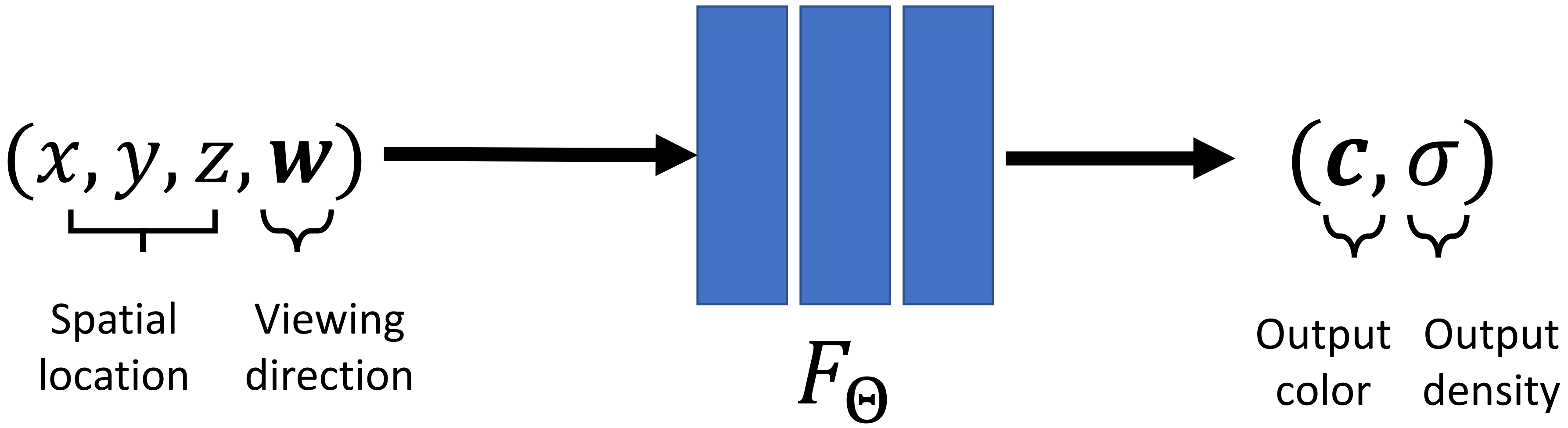
(e) Inverse rendering $(x,y,z) \rightarrow$ RGB, density

Implicit image representation



NeRF

Positional encoding

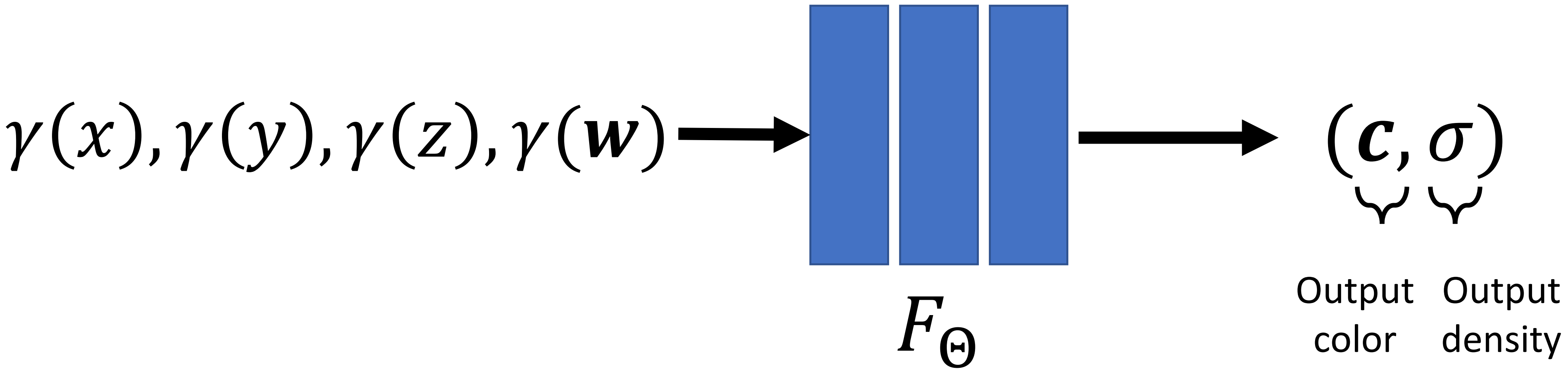


$$(\underbrace{x, y, z}_{\text{Spatial location}}, \underbrace{\boldsymbol{w}}_{\text{Viewing direction}}) \longrightarrow F_\Theta \longrightarrow (\underbrace{\boldsymbol{c}}_{\text{Output color}}, \underbrace{\sigma}_{\text{Output density}})$$

NeRF

Positional encoding

Introducing positional encoding



$$\gamma(x), \gamma(y), \gamma(z), \gamma(\boldsymbol{w}) \longrightarrow F_{\Theta} \longrightarrow (\underbrace{\boldsymbol{c}}_{\text{Output color}}, \underbrace{\sigma}_{\text{Output density}})$$

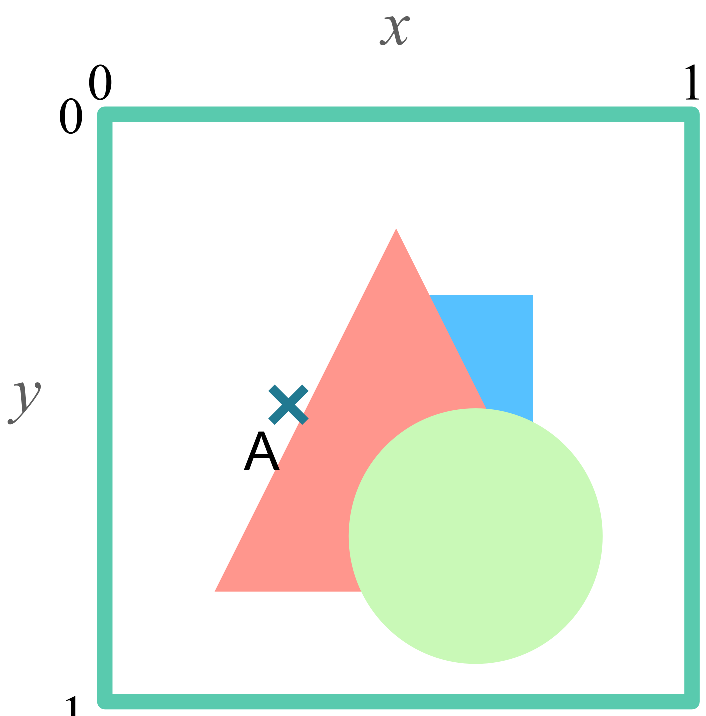
$$^{*}\,\gamma(x) = \left(\sin\left(2^{0}\pi x\right), \cos\left(2^{0}\pi x\right), \ldots, \sin\left(2^{L-1}\pi x\right), \cos\left(2^{L-1}\pi x\right)\right)$$

Why does positional encoding help?

Target Image

Why does positional encoding help?

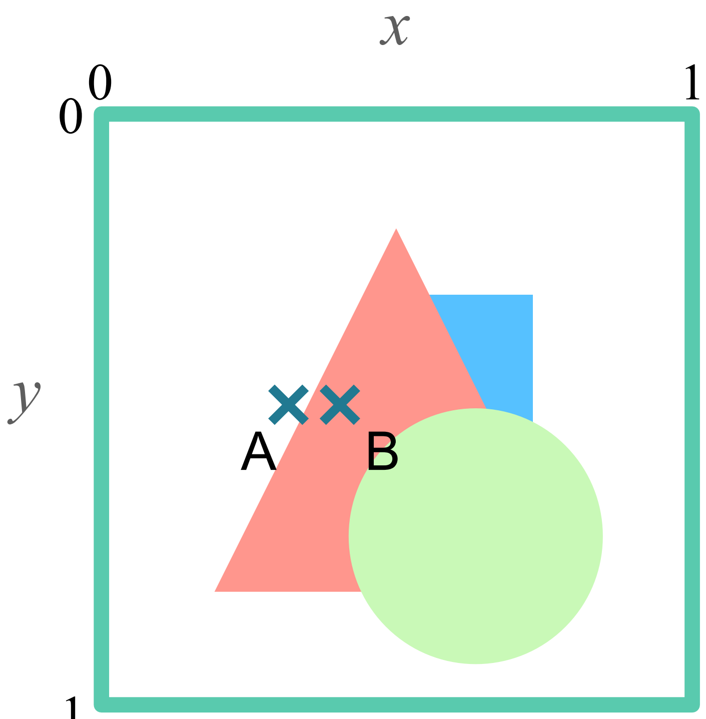
	Input		Target
	x	y	
A	.36	.5	□



Target Image

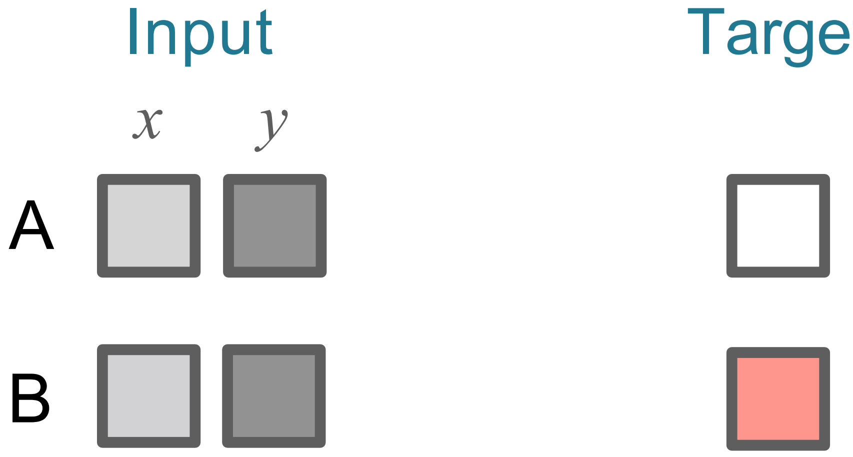
Why does positional encoding help?

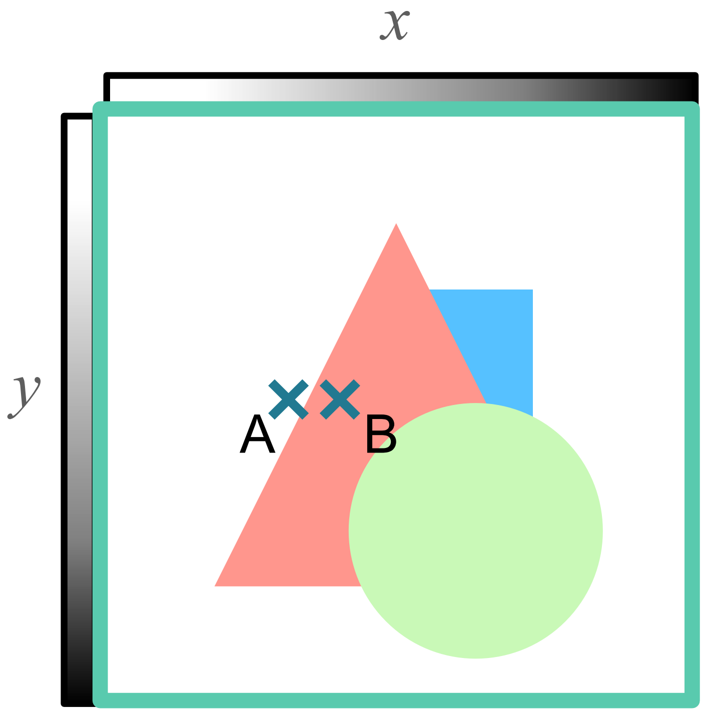
	Input		Target
	x	y	
A	.36	.5	□
B	.38	.5	🟥



Target Image

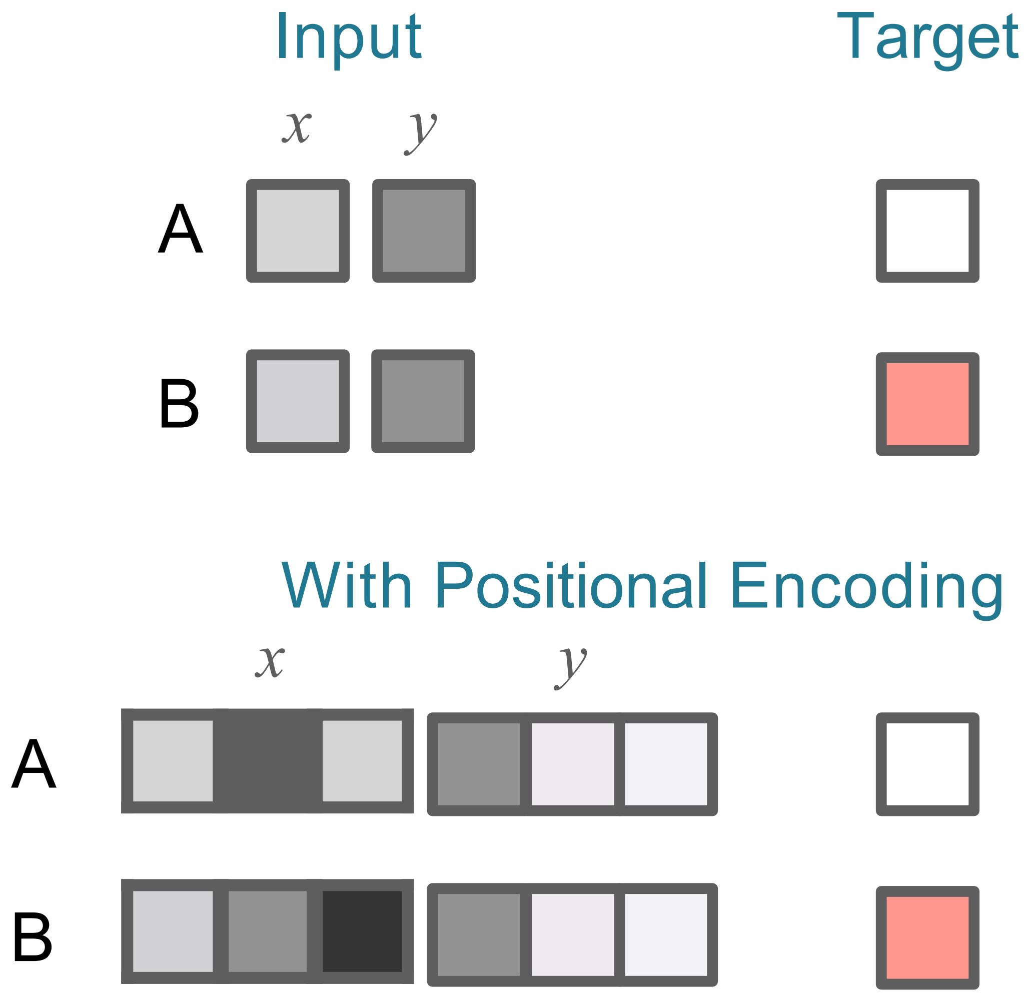
Why does positional encoding help?





Target Image

Why does positional encoding help?



Input

Target

x y

A

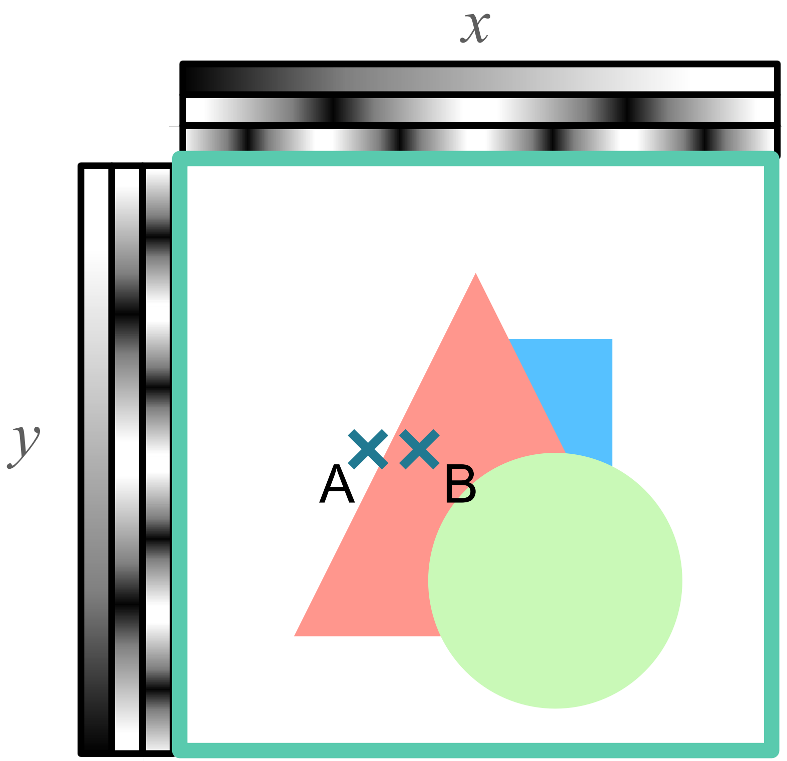
B

With Positional Encoding

x y

A

B



Target Image

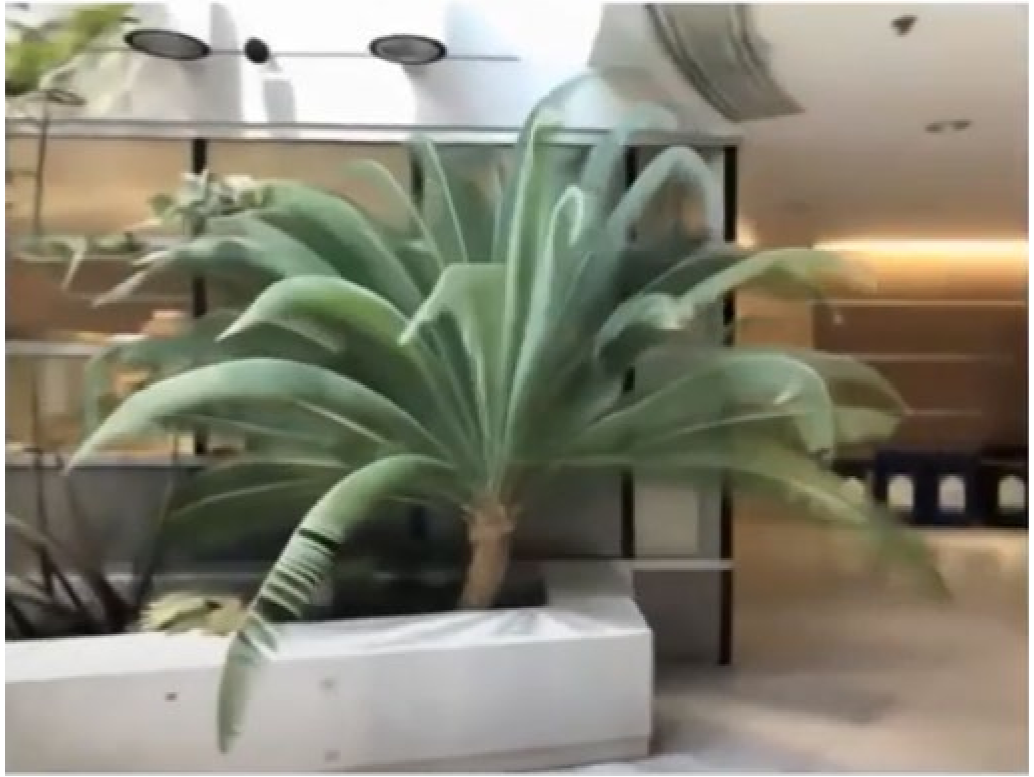
NeRF

Synthetic scenes

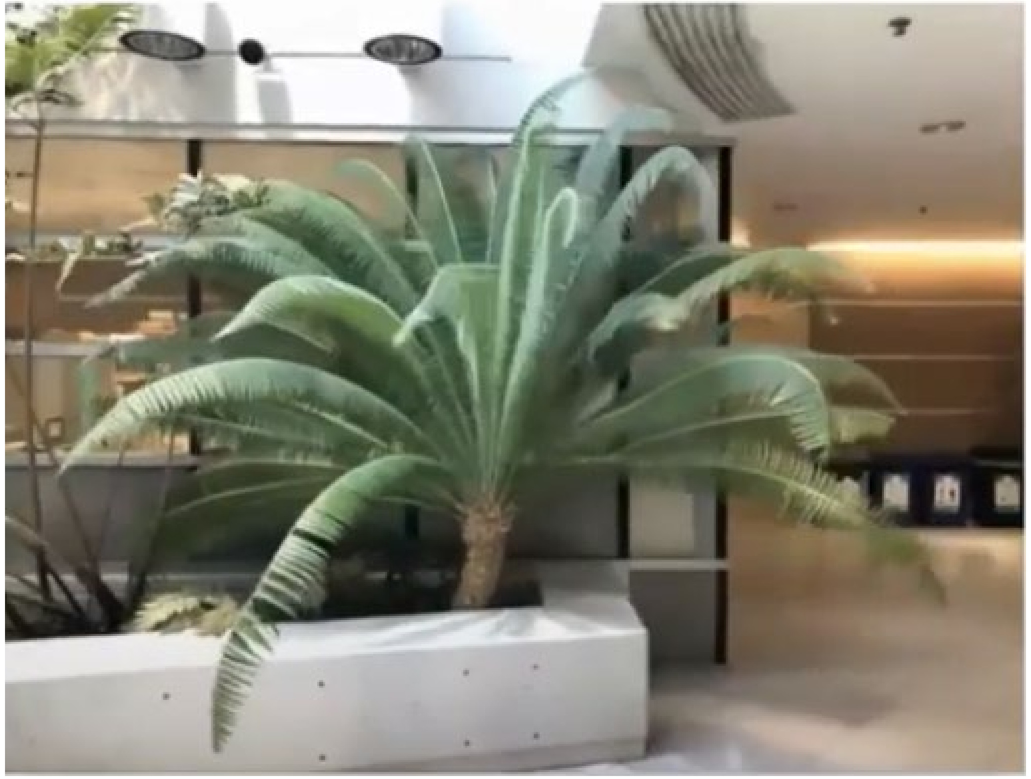
NeRF

Real scenes

Nerf



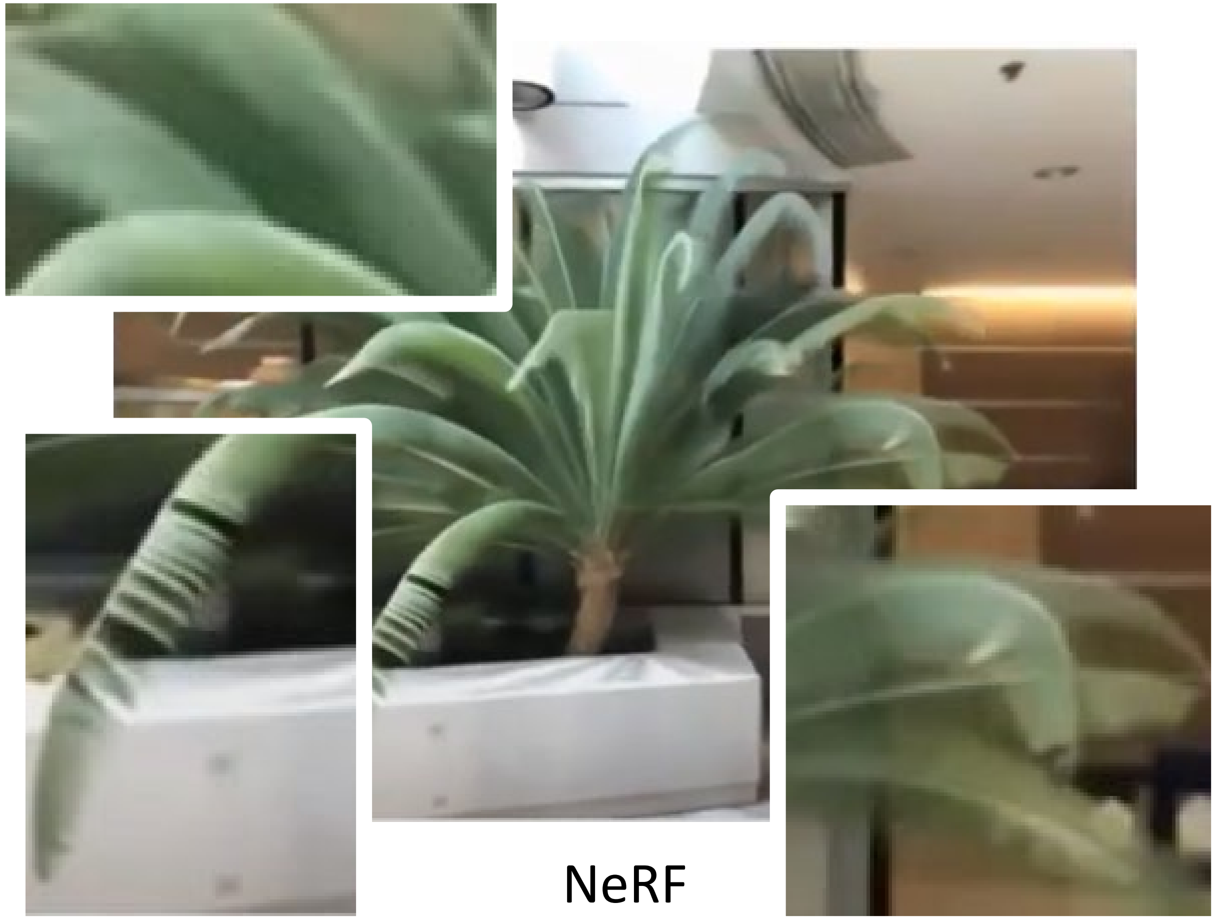
NeRF
No positional encoding



NeRF
With positional encoding

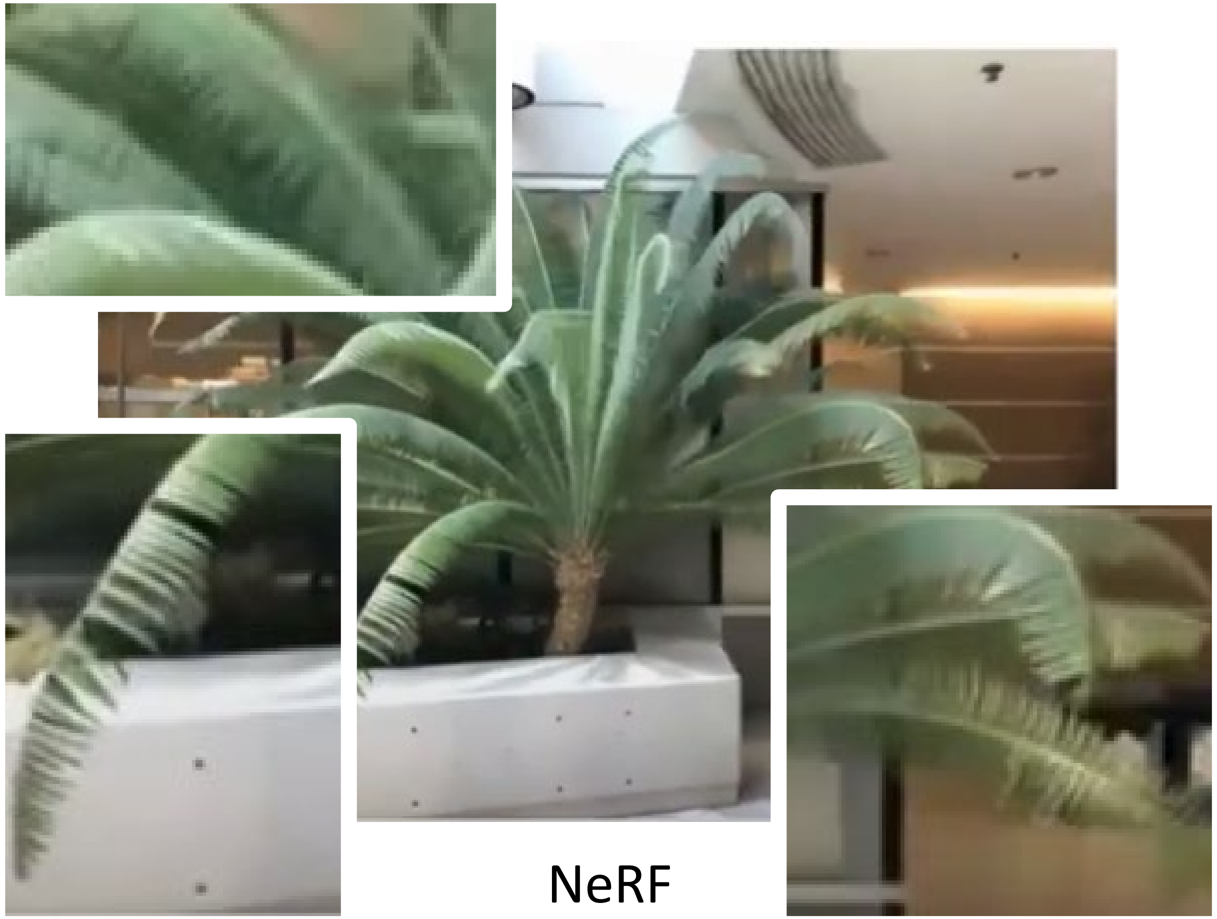
Nerf

Importance of positional encoding



NeRF

No positional encoding



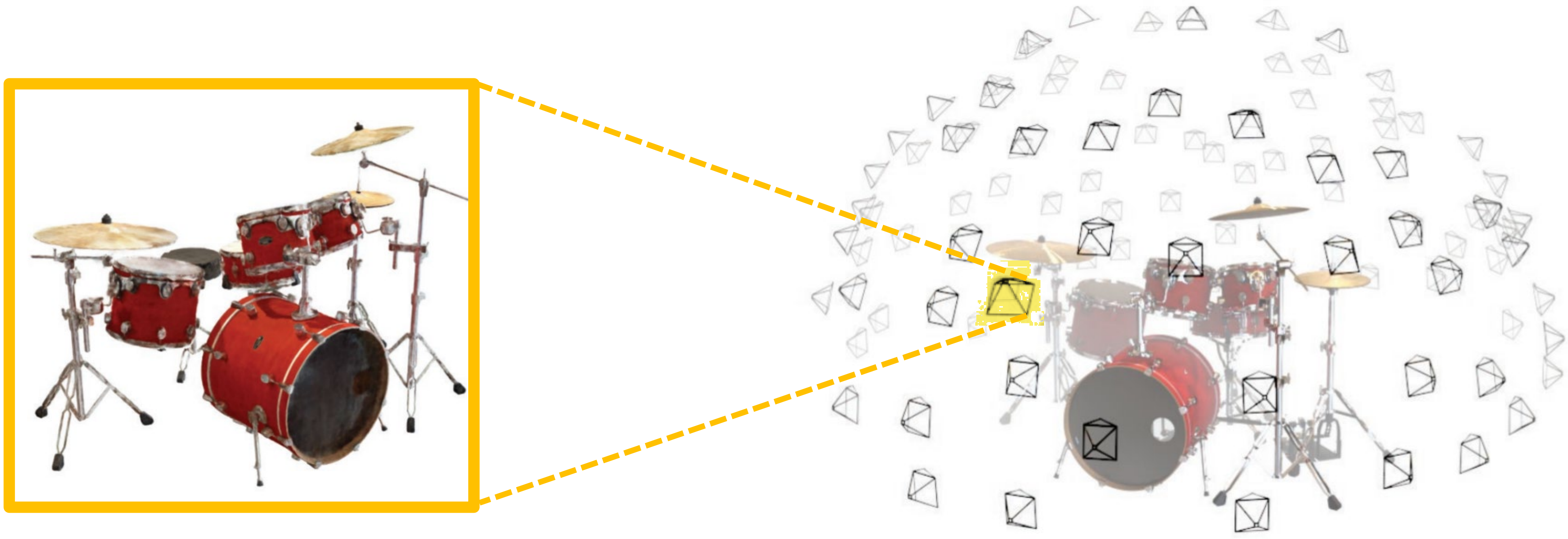
NeRF

With positional encoding

NeRF

Summary

- Novel view synthesis by volume rendering (ray integration)
- Coordinate-base scene representation
- The viewing direction is taken into account
- Encoding the scene in the MLP weights



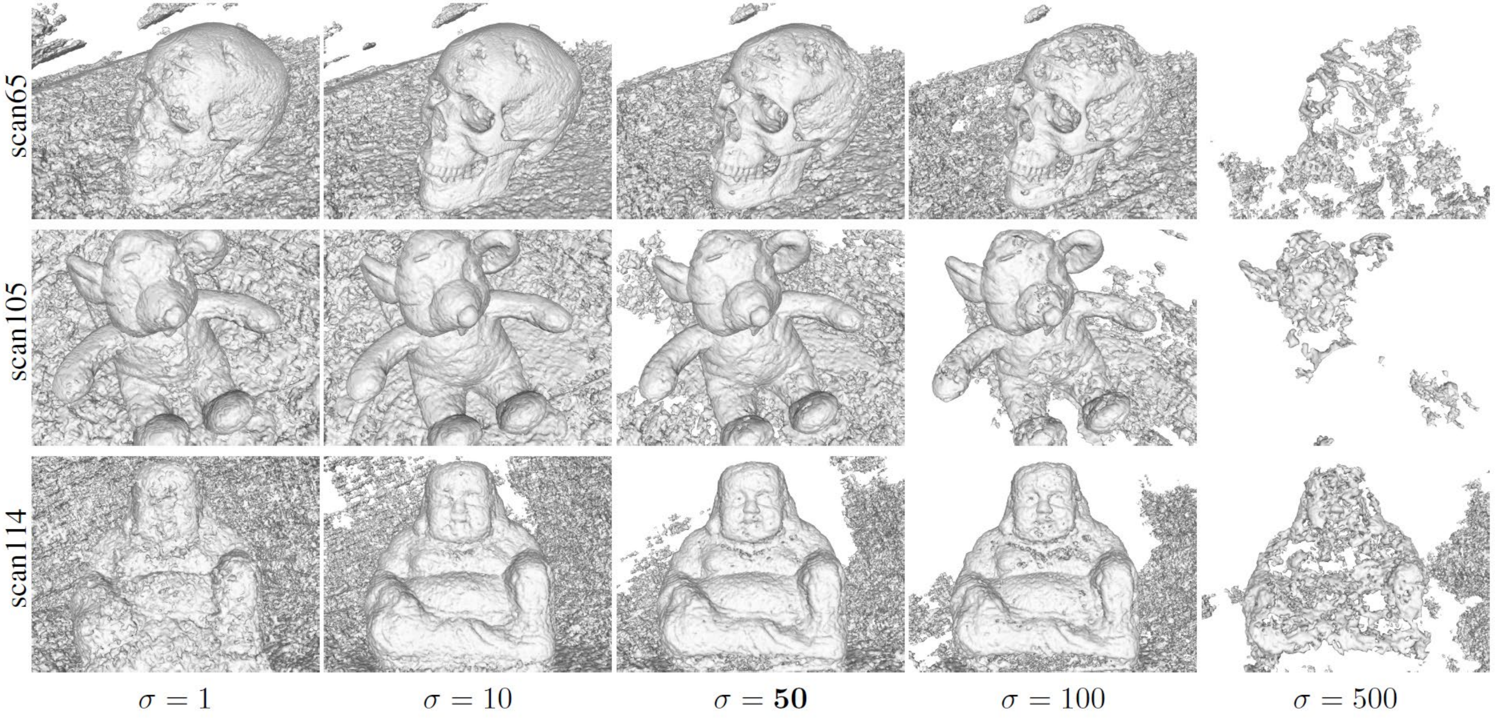
NeRF

Drawbacks / Future directions

- Trained per scene, not generalizable
- Limited by the dense cover of the scene
- Glossy/transparent surfaces are not modelled well
- The surface geometry is not characterized well by the density σ

Neural rendering

- Representing the surface itself, why?
- Volume rendering or estimation of volume density does not admit accurate surface reconstruction



Volume density thresholds of NeRF

UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction, Oechsle et al., 2021

Next time, by Lior Yariv

Neural Surface Rendering and Reconstruction