Lecture: Neural Fields Part 1

How does Computer Vision & Graphics work together?

Current Image

Explicit 3D Reconstruction with Neural Fields.

Vision

Edit Components

Graphics

New Image under different conditions

3D Intrinsic Components

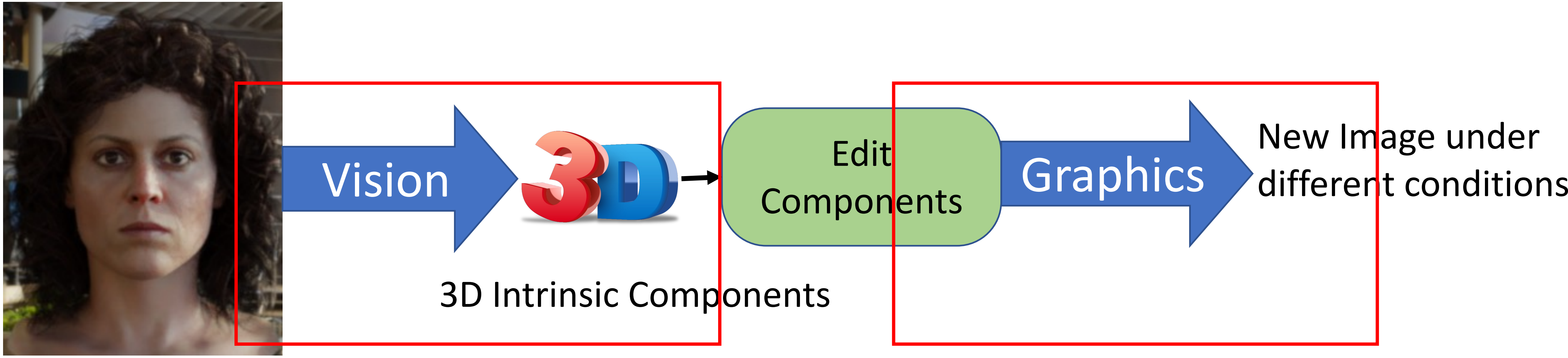
Explicit: Reconstruct 3D
(Introduction to Graphics Lectures)

Implicit: Neural Representation
(Generative Models Lectures

Change:
- Viewpoint
- Lighting
- Reflectance
- Background
- Attributes
- Many others…

Outline

- What is Neural Fields & why it got so much attention?
- The Prelude: Neural Implicit Surfaces
- Introduction to Volume Rendering

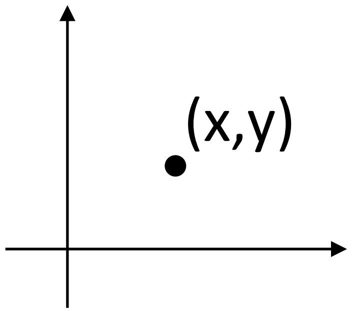


Current Image

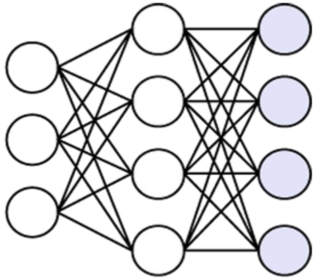
Outline

- What is Neural Fields & why it got so much attention?
- The Prelude: Neural Implicit Surfaces
- Introduction to Volume Rendering

What are neural fields?

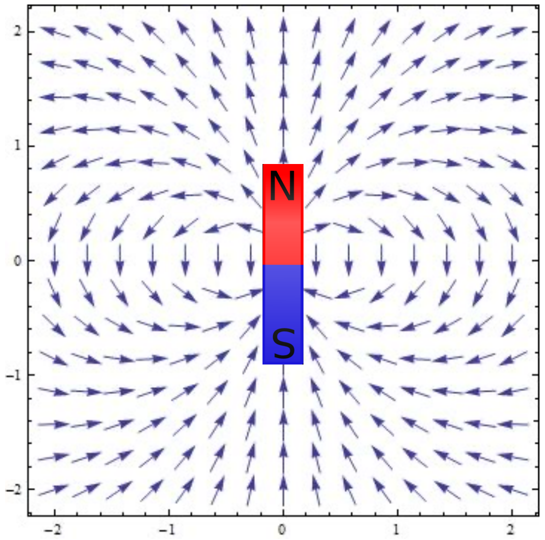


$\Phi: \mathbb{R}^2 \rightarrow \mathbb{R}^2$

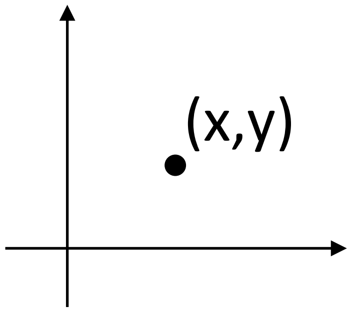


Neural Network

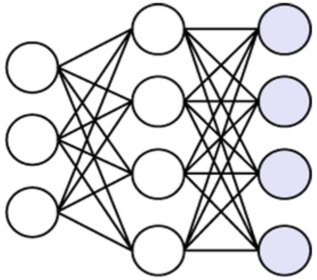
(Φ)



Magnetic Field

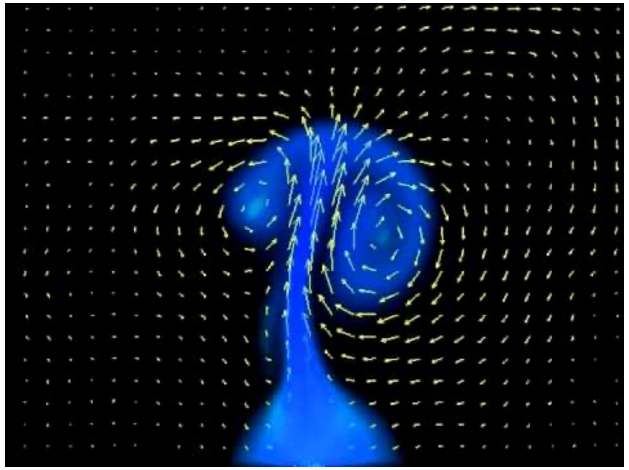


$\Phi: \mathbb{R}^2 \rightarrow \mathbb{R}^2$



Neural Network

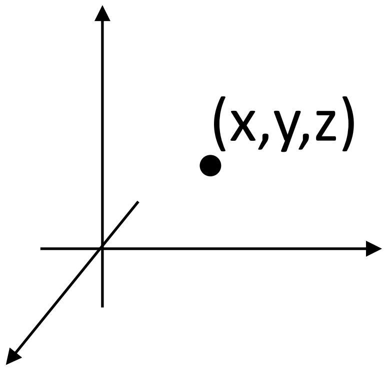
(Φ)



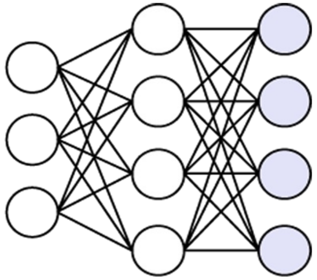
Eulerian Flow Field of a Fluid

[Koldora CC]

What are neural fields?

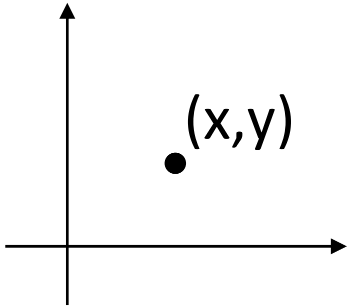


$\Phi: \mathbb{R}^n \rightarrow \mathbb{R}$

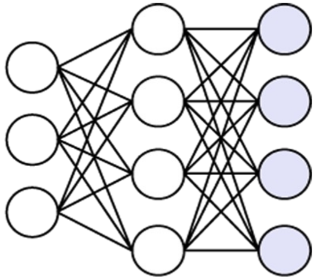


Neural Network
(Φ)

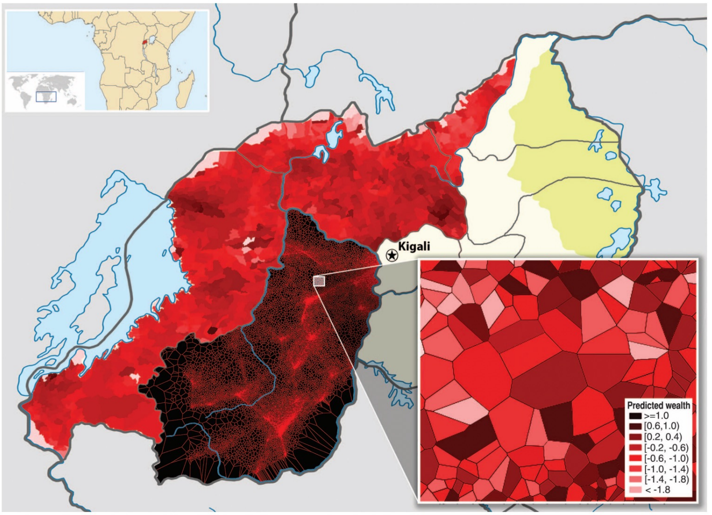
Signed Distance Function (SDF)



$\Phi: \mathbb{R}^2 \rightarrow \mathbb{R}^n$



Neural Network
(Φ)



Geospatial Data
[Blumenstock et al. 2015]

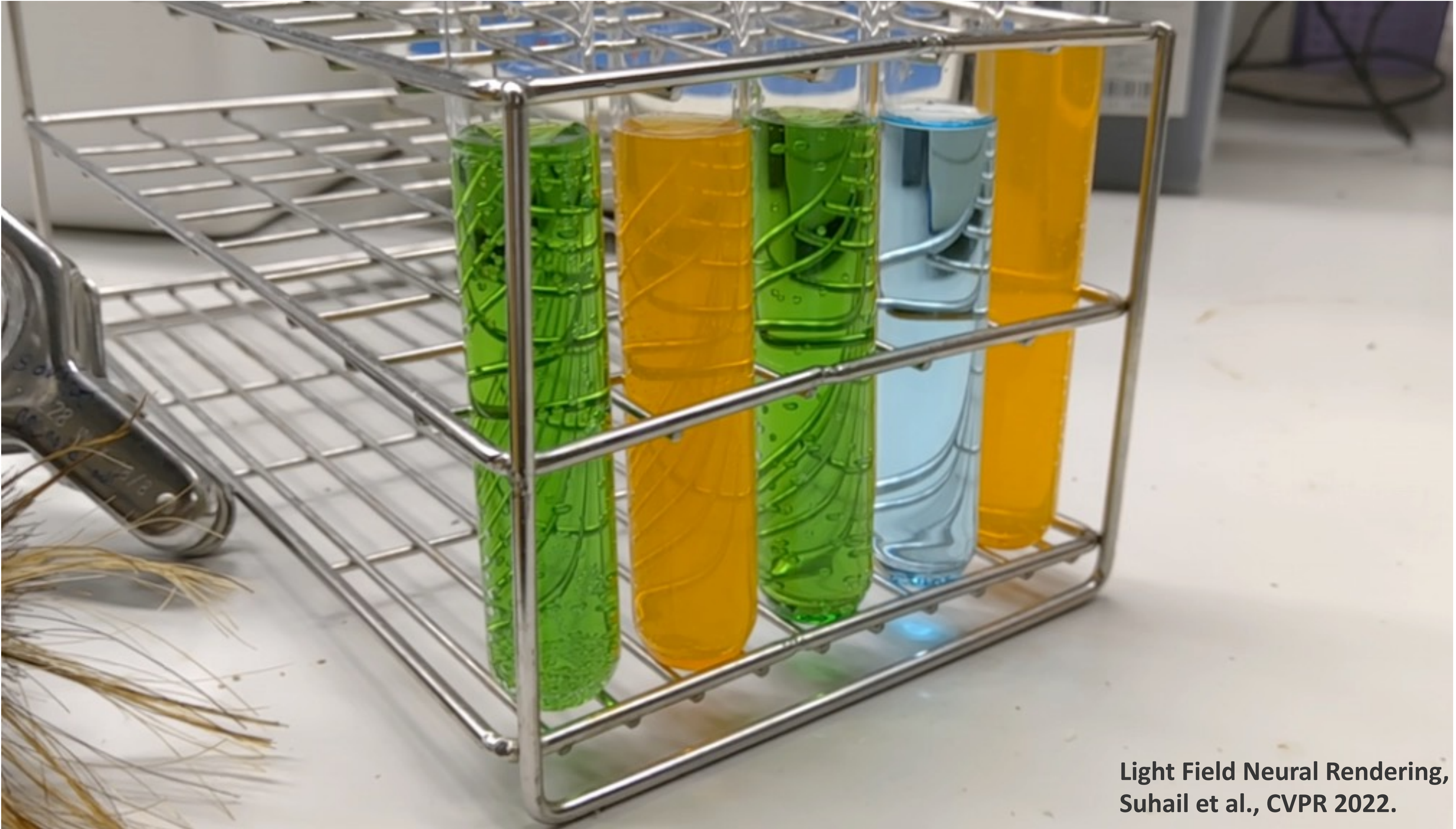
NeRF (Neural Radiance Field) has revolutionalized Computer Vision & Graphics in past 2 years!

Let’s look at some of the stunning results it produced!

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis,
Ben Mildenhall*, Pratul Srinivasan*, Matthew Tancik*, Jonathan Barron, Ravi Ramamoorthi, Ren Ng, ECCV 2020.

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis,
Ben Mildenhall*, Pratul Srinivasan*, Matthew Tancik*, Jonathan Barron, Ravi Ramamoorthi, Ren Ng, ECCV 2020.

Block-NeRF: Scalable Large Scene Neural View Synthesis, CVPR 2022.



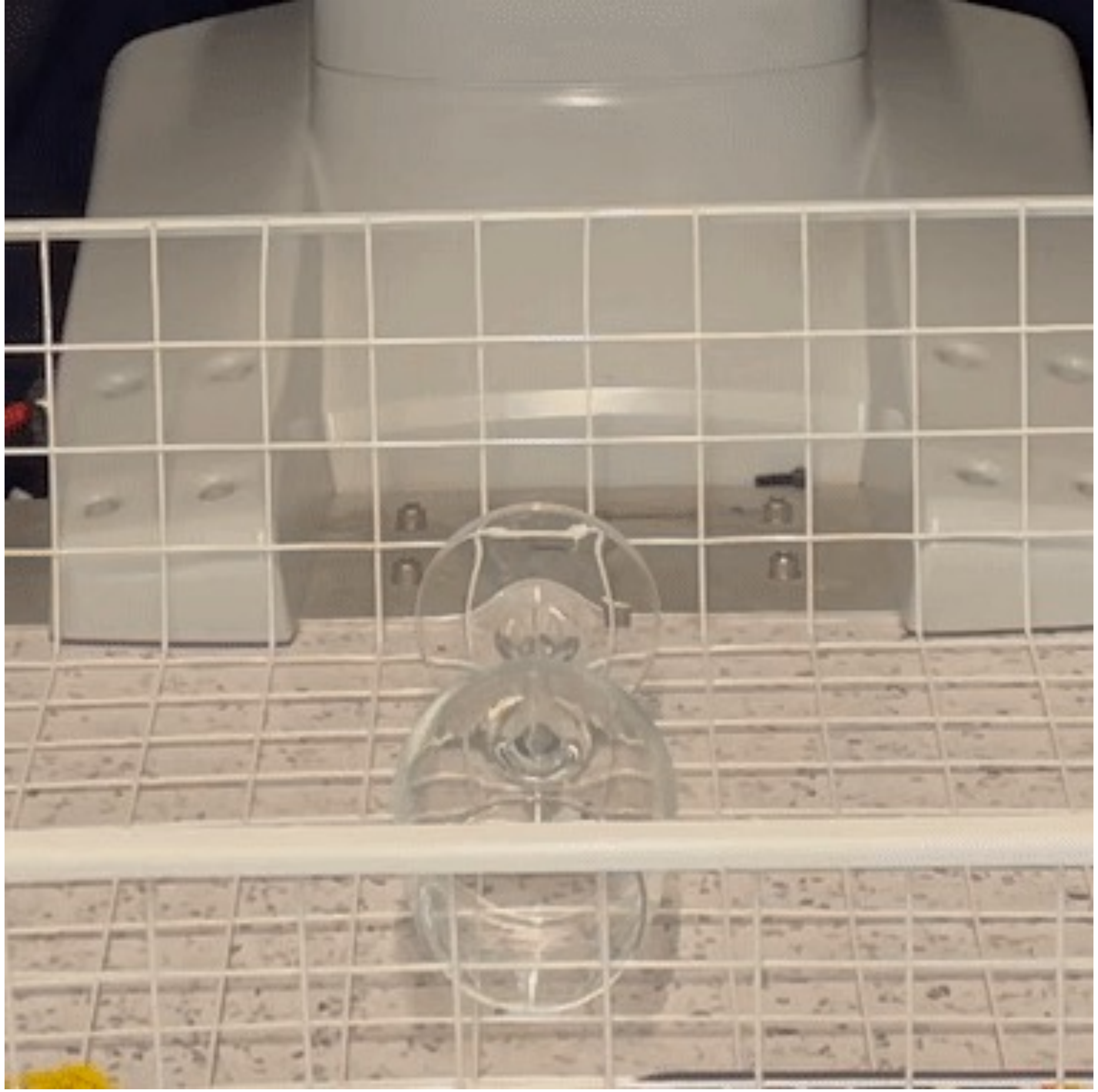
Light Field Neural Rendering,
Suhail et al., CVPR 2022.

(a) Capture Process (b) Input (c) Nerfie (d) Nerfie Depth

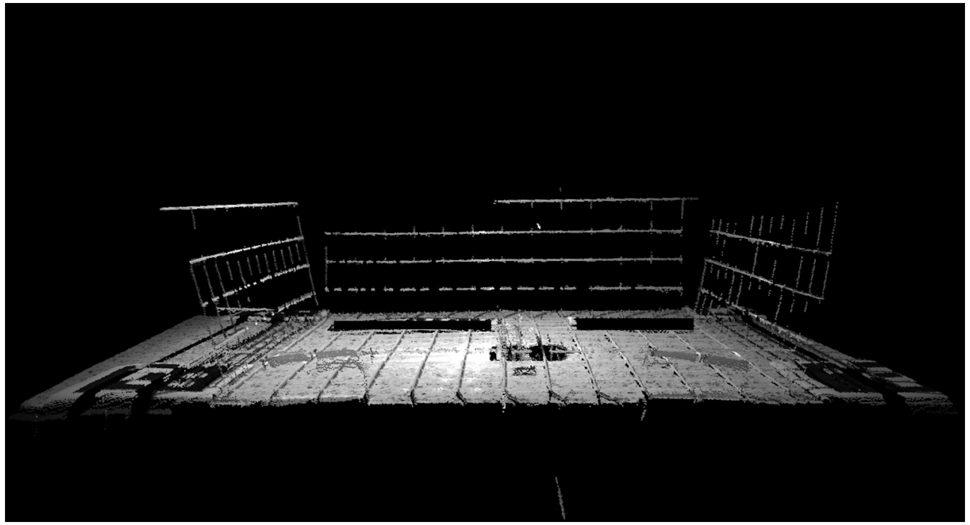
NeRFies: Deformable Neural Radiance Fields, Keunhong Park et al., ICCV 2021.

Neural 3D Video Synthesis
from Multi-view Video,
Li et al., CVPR 2022

“GRAM: Generative Radiance Manifolds for 3D-Aware Image Generation”, Deng at al., CVPR 2022.



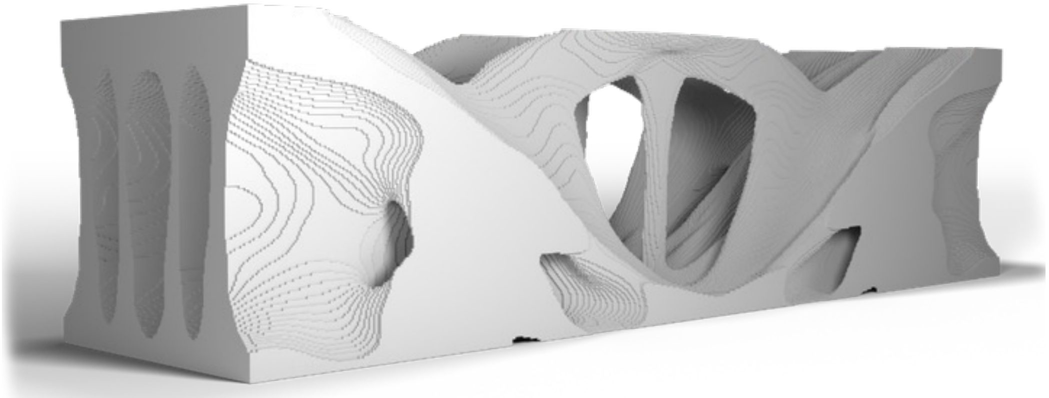
Grasp shown in red



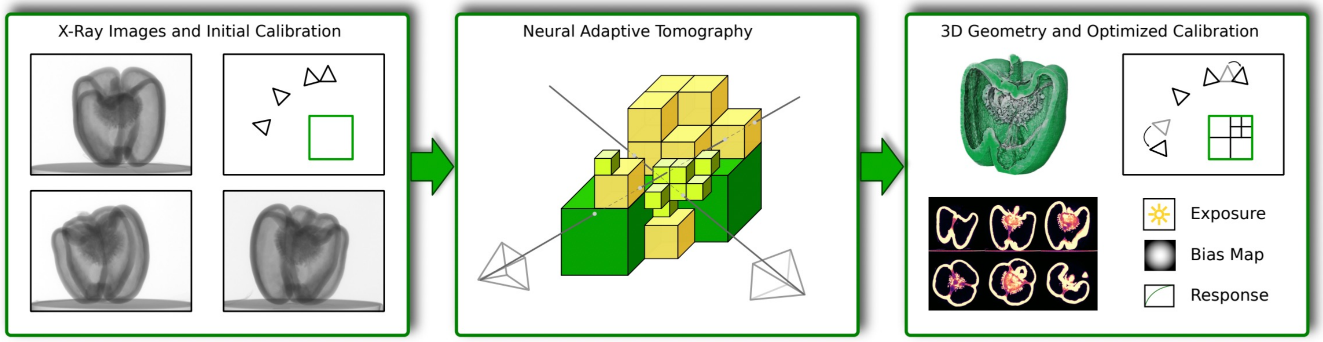
Point cloud from sensor

Dex-NeRF: Grasping Transparent Objects using NeRF, Ichnowski et al., CoRL 2021

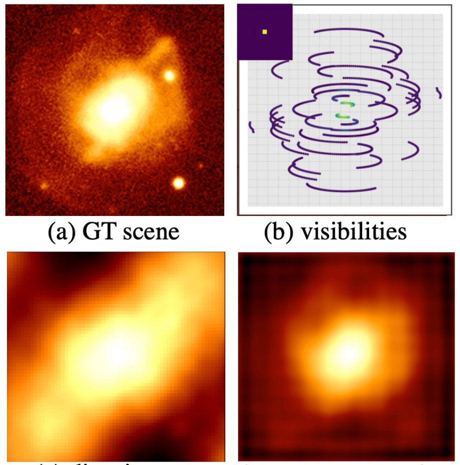
Neural Fields for Science and Engineering



Topology Optimization [Doosti et al. 2021]



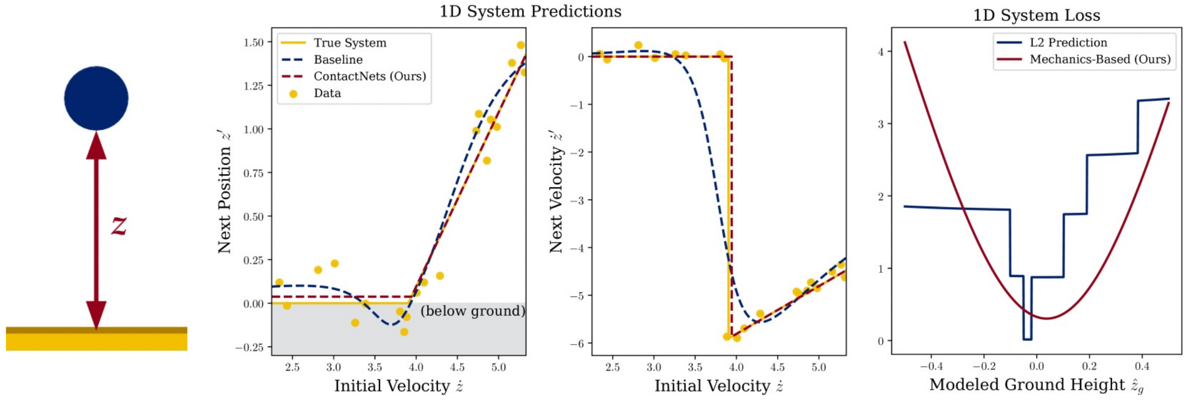
Tomographic Reconstruction [Ruckert et al. 2022]



(a) GT scene (b) visibilities

(c) dirty image (d) our reconstruction

Astronomical Interferometry [Wu et al. 2021]



(a) 1D System (b) Model Predictions (c) Loss Landscape

Contact Dynamics [Pfrommer, Halm et al. 2022]

The “Cambrian Explosion” of Neural Fields

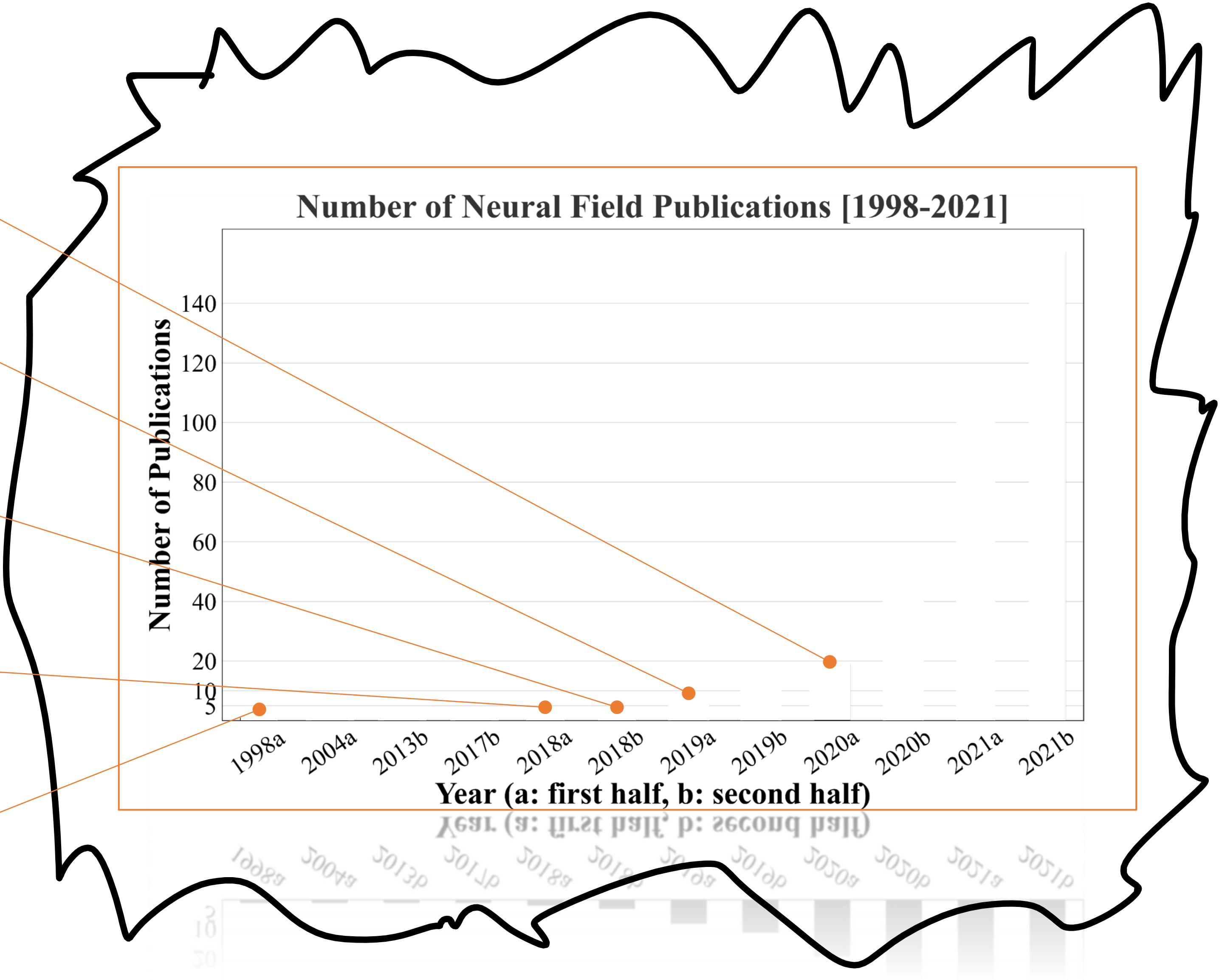
NeRF

DeepSDF, PIFu

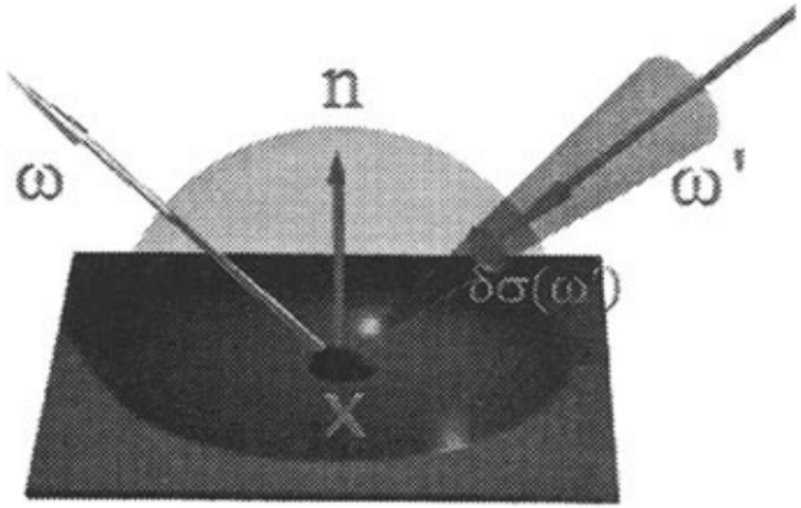
Occupancy Networks, IM-NET

AtlasNet

Approximating Reflectance Functions using Neural Networks

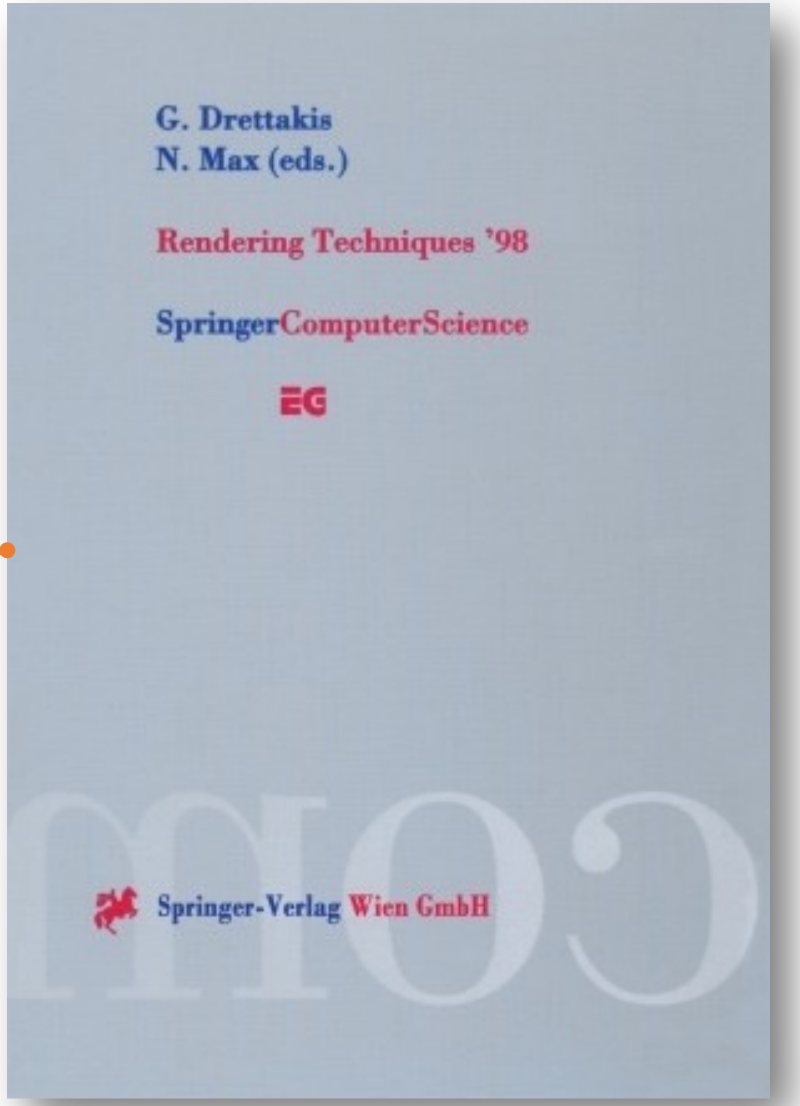


The “Cambrian Explosion” of Neural Fields

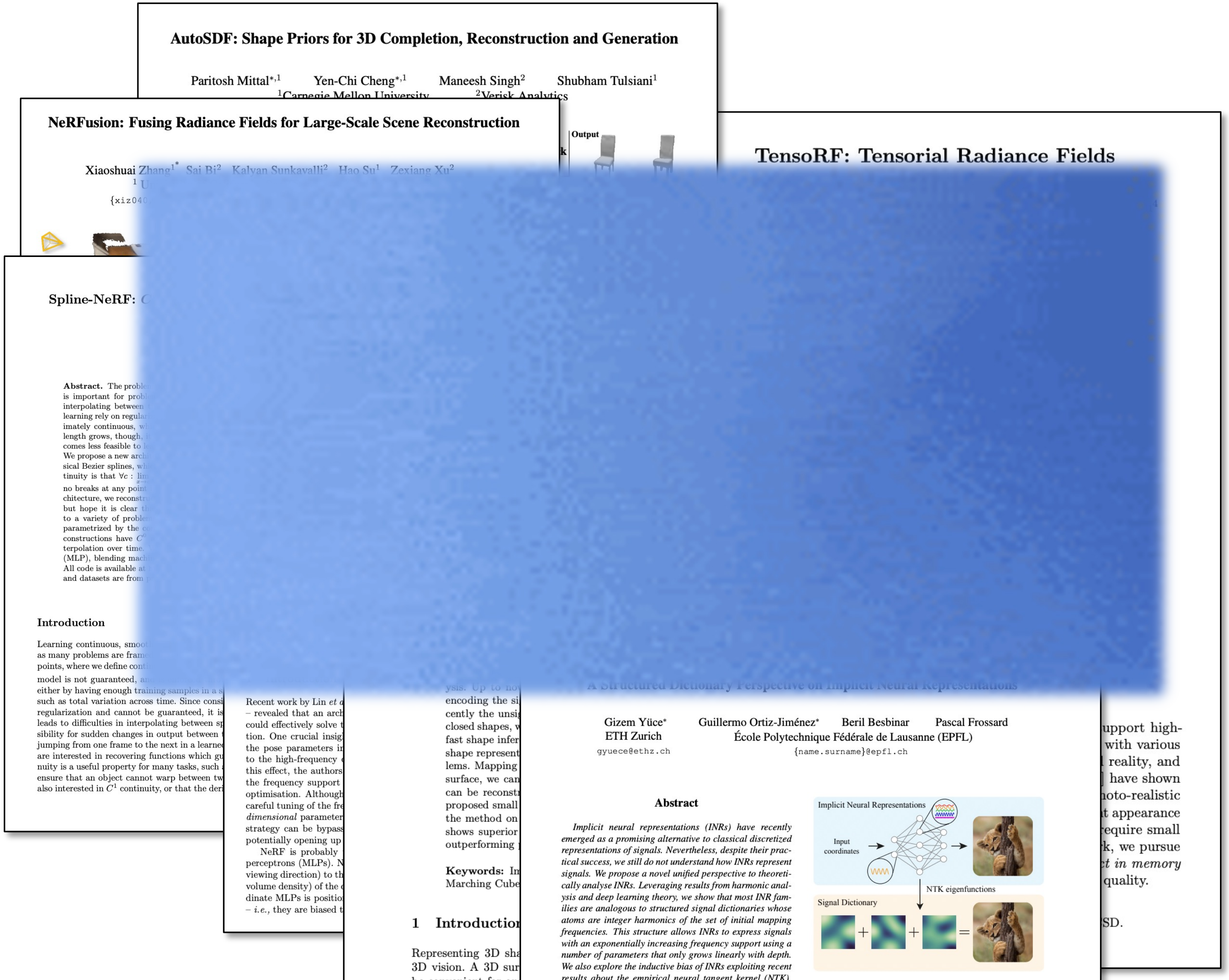


[Gargan and Neelamkavil 1998]

Approximating Reflectance Functions using Neural Networks



Why is the community so excited?



Outline

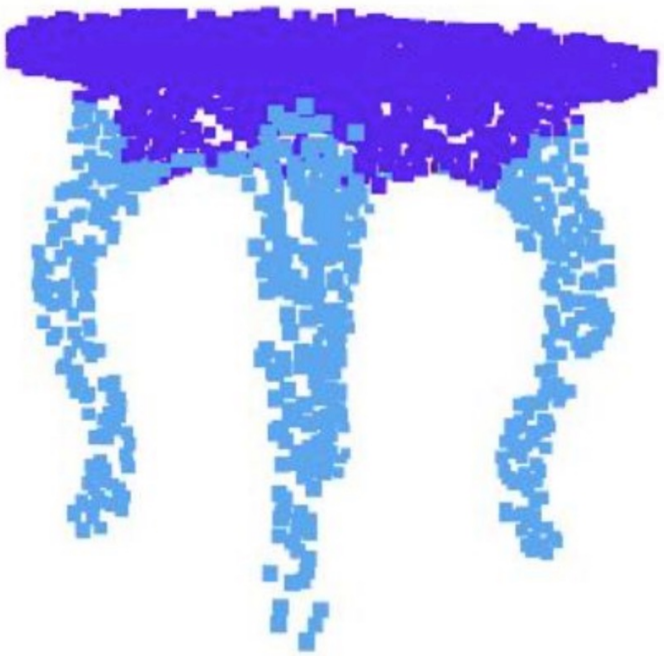
- What is Neural Fields & why it got so much attention?
- The Prelude: Neural Implicit Surfaces
- Introduction to Volume Rendering

Representations for 3D Deep Learning

Voxel

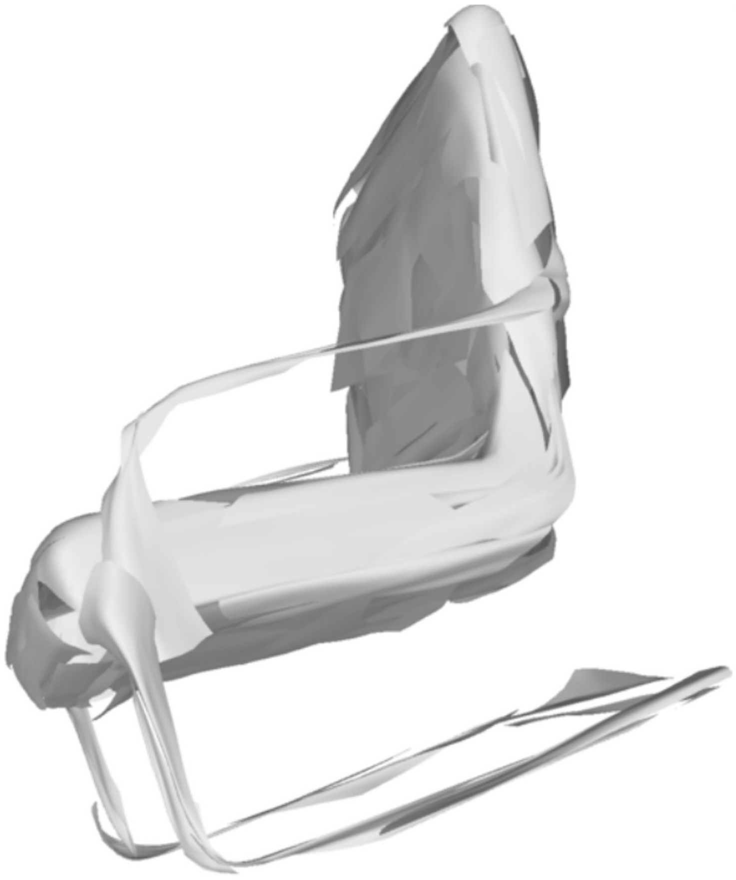
Wu et al. 2016

Points



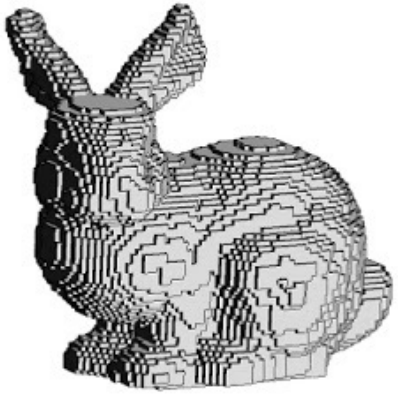
Qi et al. 2017

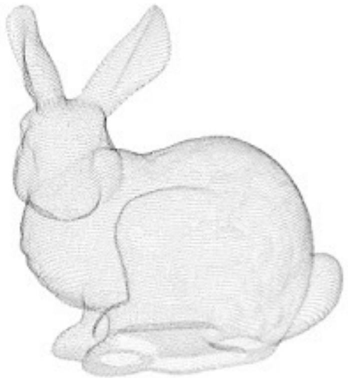
Meshes

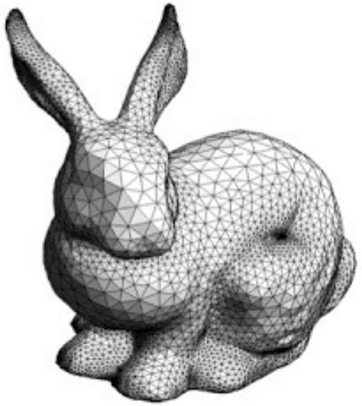


Groueix et al. 2018

3D Representations (Explicit)







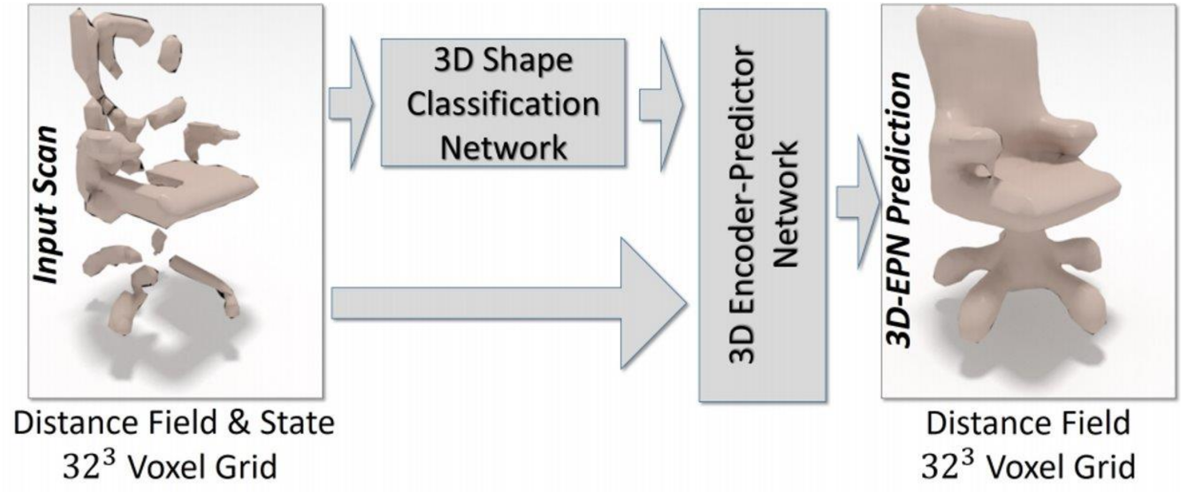
	Voxel	Point cloud	Polygon mesh
Memory efficiency	Poor	Not good	Good
Textures	Not good	No	Yes
For neural networks	Easy	Not easy	Not easy

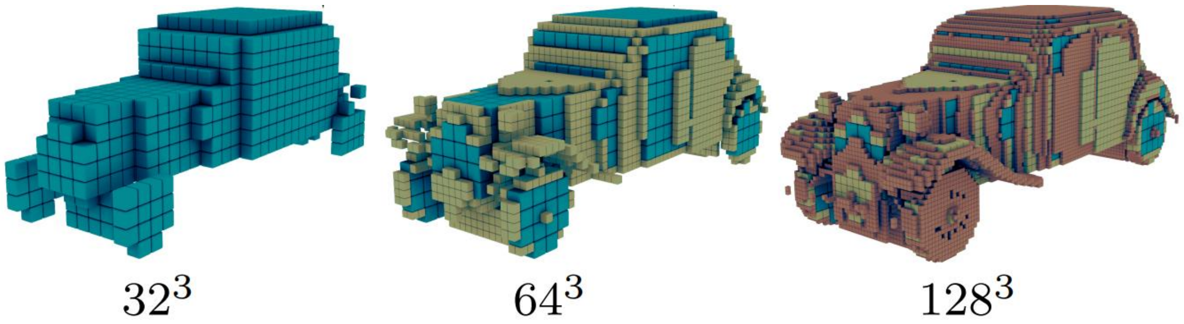
We adopt **polygon mesh** for its high potential

Images are from
http://cse.iitkgp.ac.in/~pb/research/3dpoly/3dpoly.html
http://waldyrious.net/learning-holography/pb-cgh-formulas.xhtm
http://www.cs.mun.ca/~omeruvia/philosophy/images/BunnyWire.gif

Voxel Representation

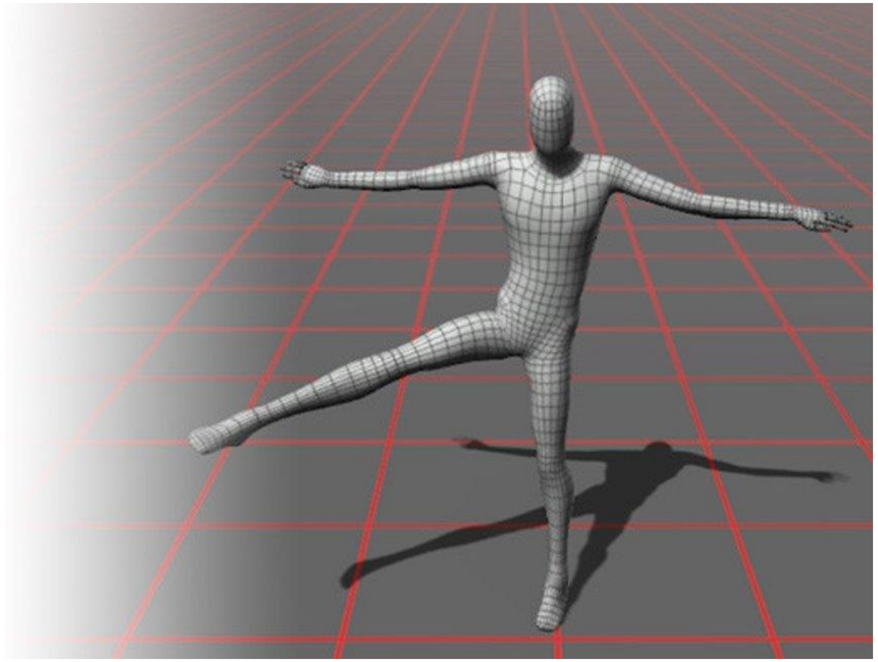
- Memory Intensive, Computationally Expensive (N^3)

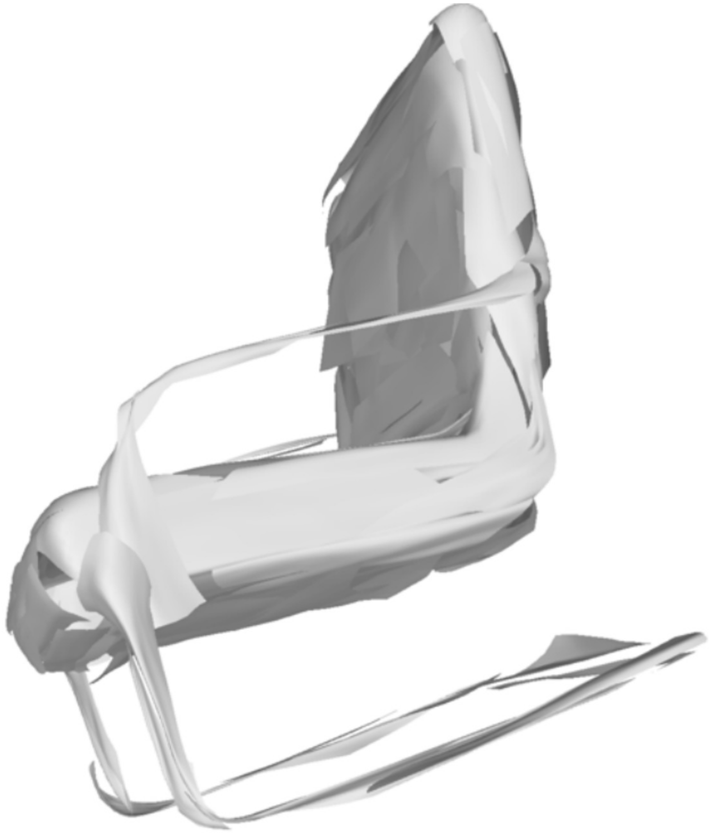




Mesh Representation

- Fixed Topologies (relies on Templates)
- Combinatorial Problem → Discrete Vertices and Connections





Groueix et al. 2018

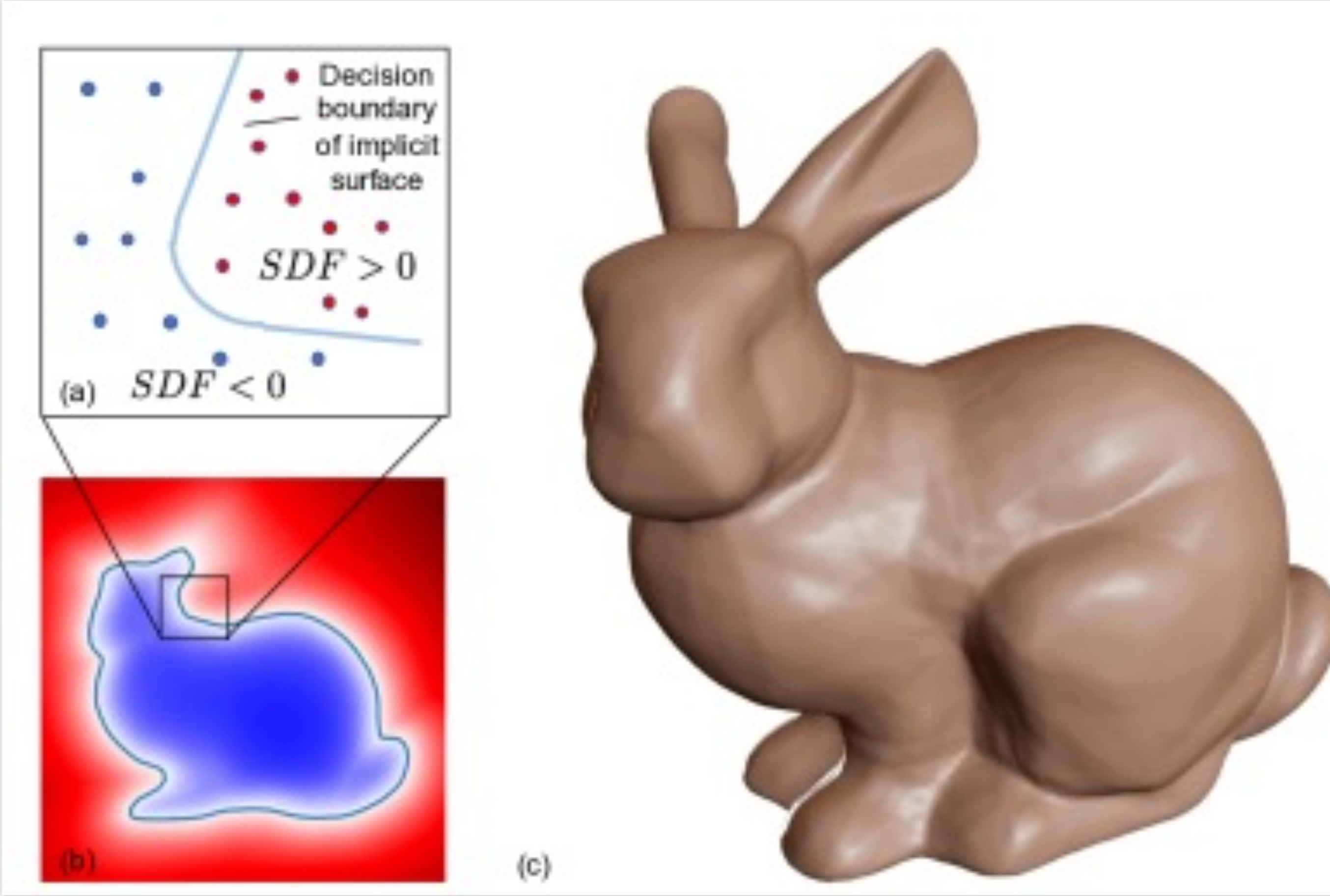
Point Cloud Representation

- Does not Define a Surface
- Not suitable for Visualization, Texturing, etc

Achlioptas et al.

Surface Representation: Signed Distance Function (SDF) - implicit representation via level set

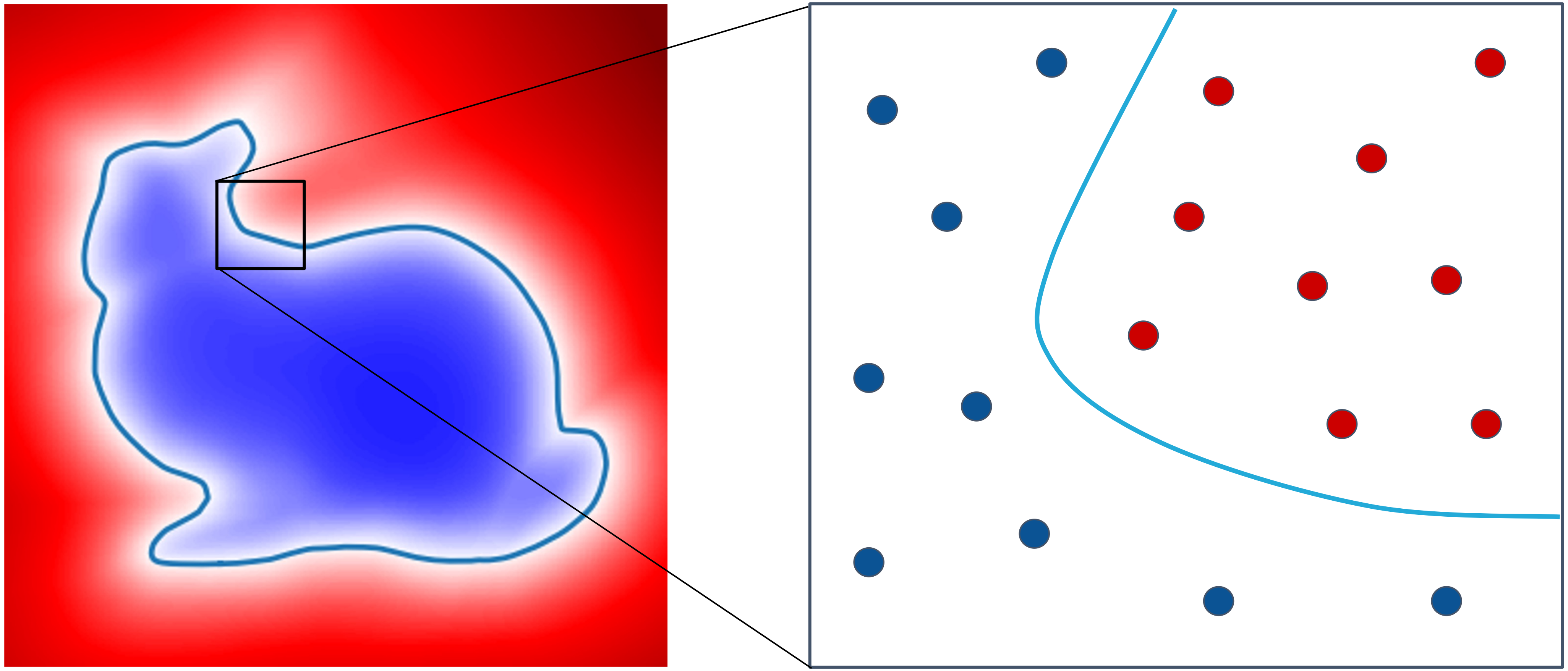
SDF(X) = 0, when X is on the surface.
SDF(X) > 0, when X is outside the surface
SDF(X) < 0, when X is inside the surface



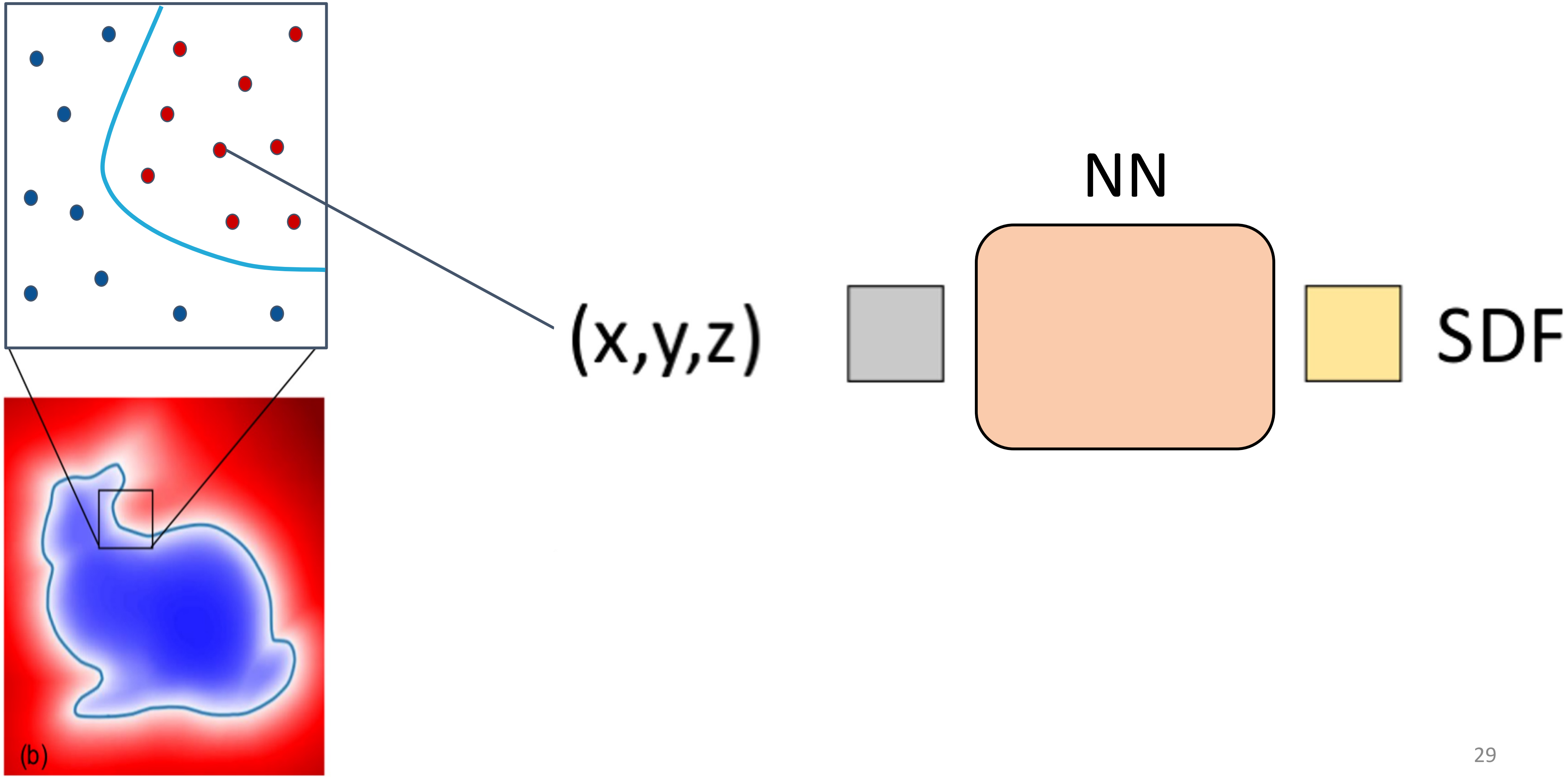
Note: SDF is an implicit representation! Suitable for neural networks but hard to import inside existing graphics software.

Deep SDF: Use a neural network (co-ordinate based MLP) to represent the SDF function.

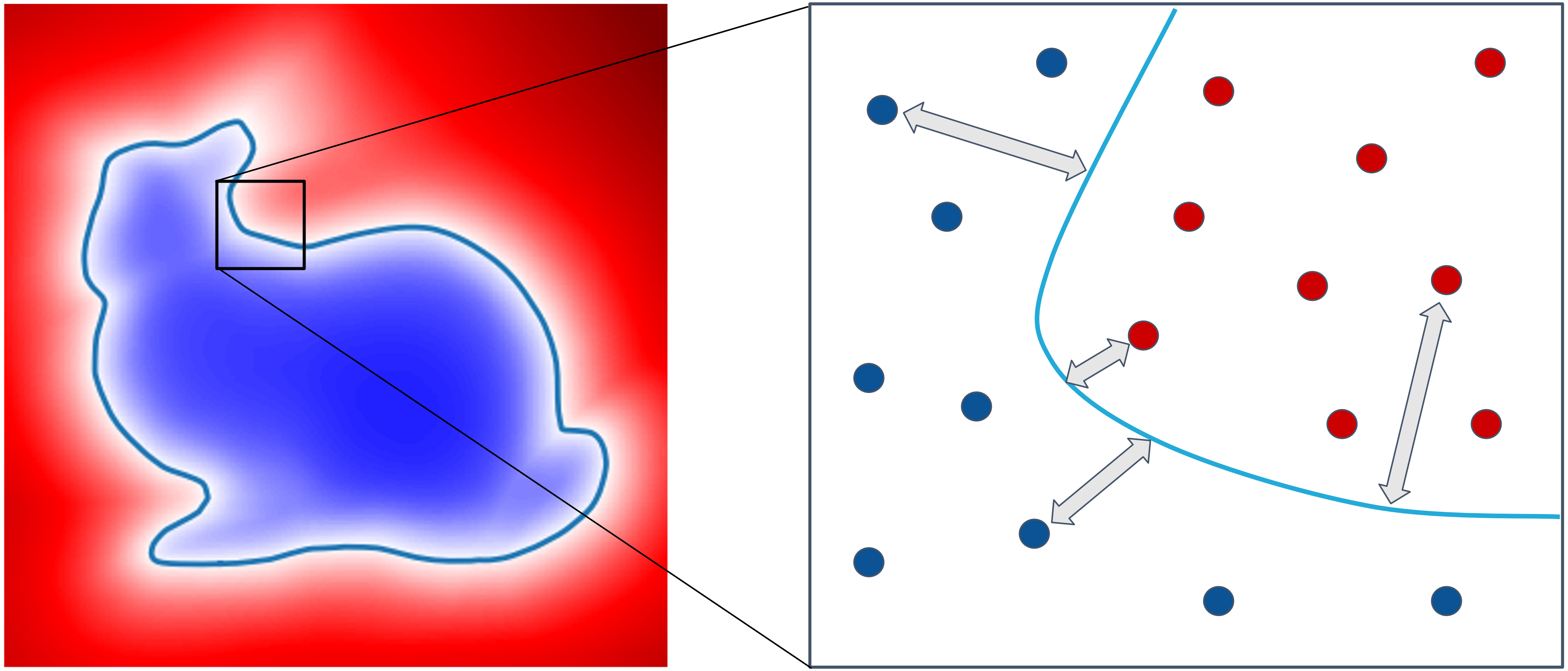
Surface as Decision Boundary



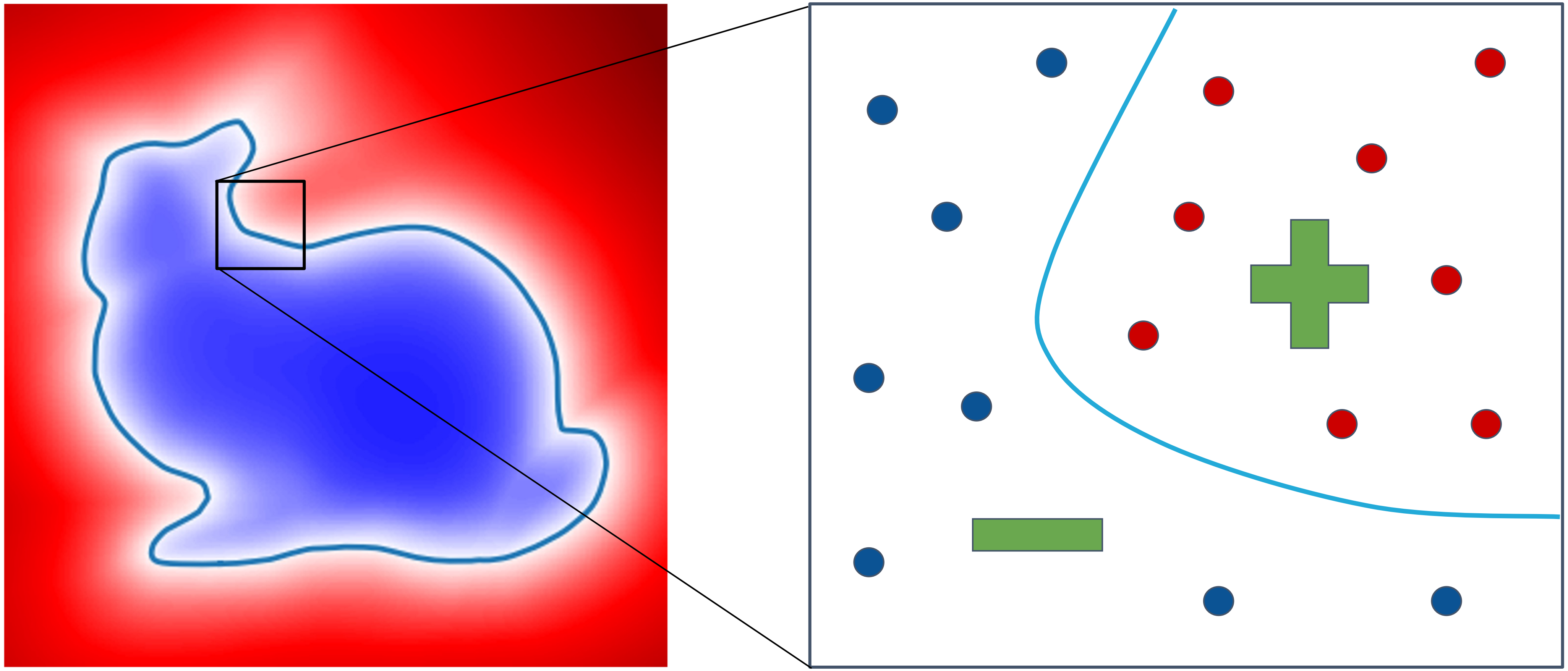
Regression of Continuous SDF



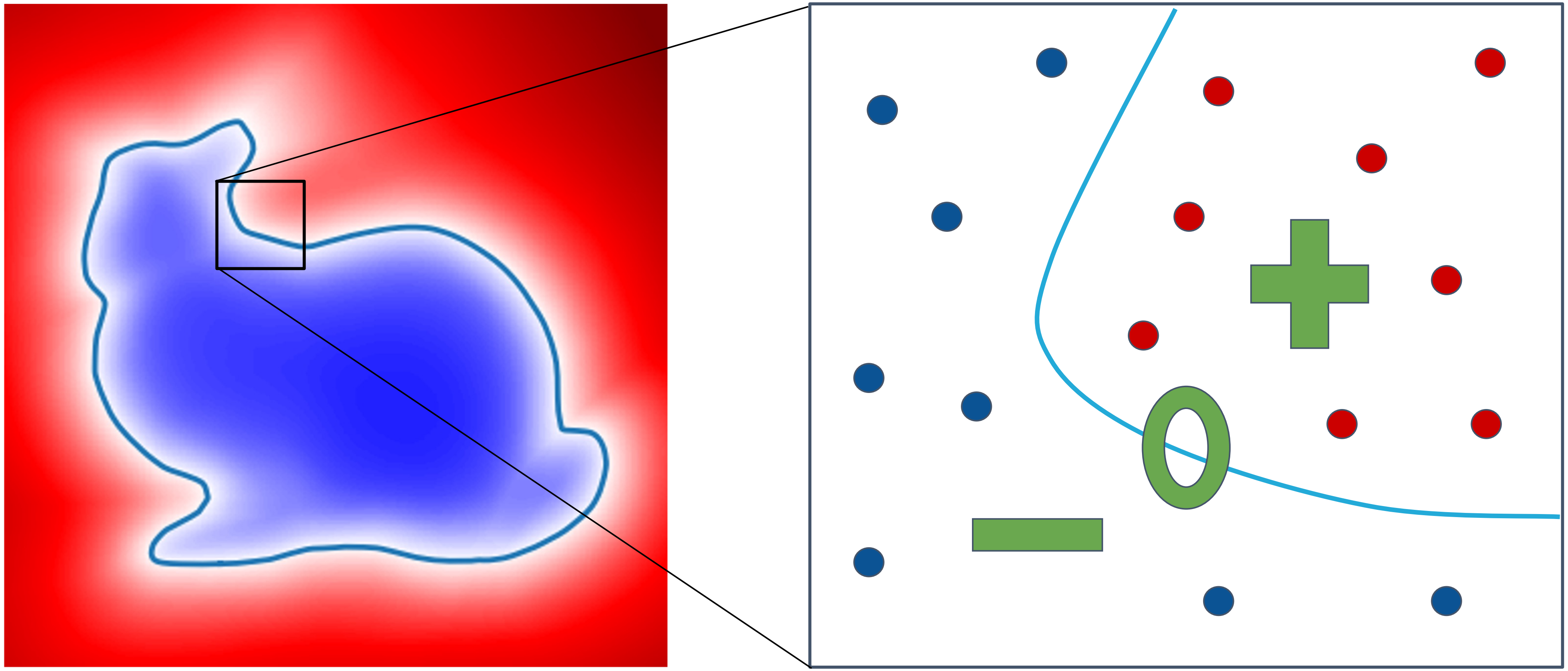
Signed Distance Function



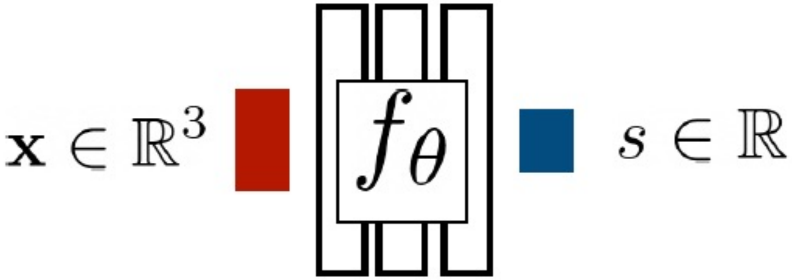
Signed Distance Function

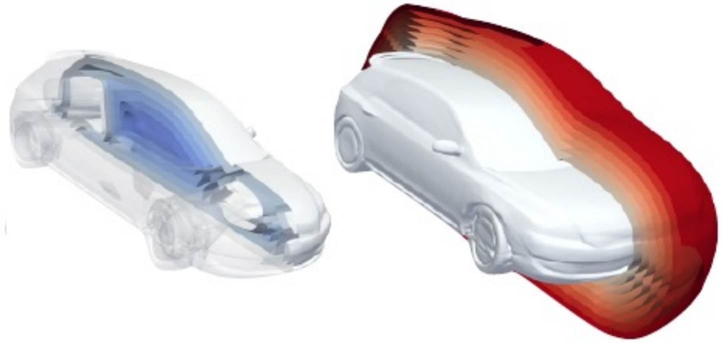


Signed Distance Function



Instance-specific SDFs





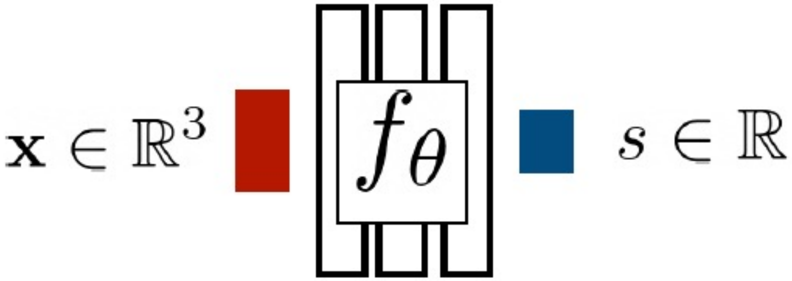
$$f_\theta : \mathbb{R}^3 \to \mathbb{R}$$

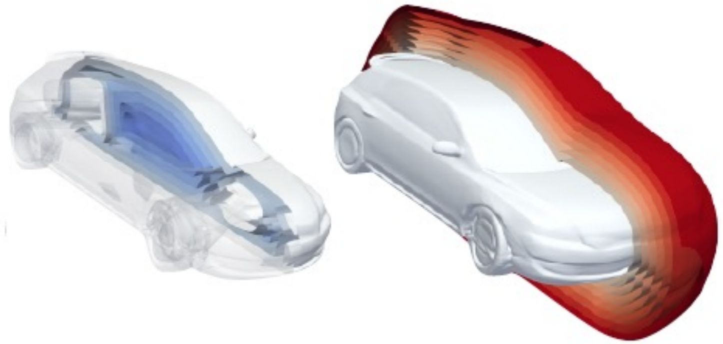
Signed Distance Field (for a single instance):
(position) → (distance)

if 6 layer network with 1000-dim feature space, about 6M parameters per instance!

Questions we want to answer

- How do we create a 3D mesh from a SDF function? (SDF rendering)
- How do we generalize this to any objects? (Training Deep SDF)
- How do we use this during inference? (Testing DeepSDF)
- How do DeepSDF concept extends to NeRF and other methods?



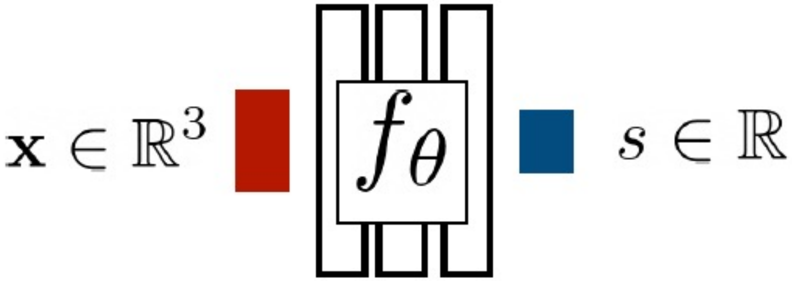


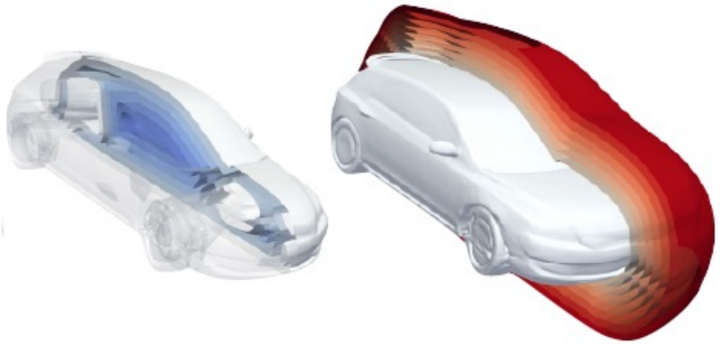
$$f_\theta : \mathbb{R}^3 \to \mathbb{R}$$

Signed Distance Field (for a single instance):
(position) → (distance)

Questions we want to answer

- How do we create a 3D mesh from a SDF function? (SDF rendering)
- How do we generalize this to any objects? (Training Deep SDF)
- How do we use this during inference? (Testing DeepSDF)
- How do DeepSDF concept extends to NeRF and other methods?





$$f_\theta : \mathbb{R}^3 \to \mathbb{R}$$

Signed Distance Field (for a single instance):
(position) → (distance)

Neural Implicit Surfaces

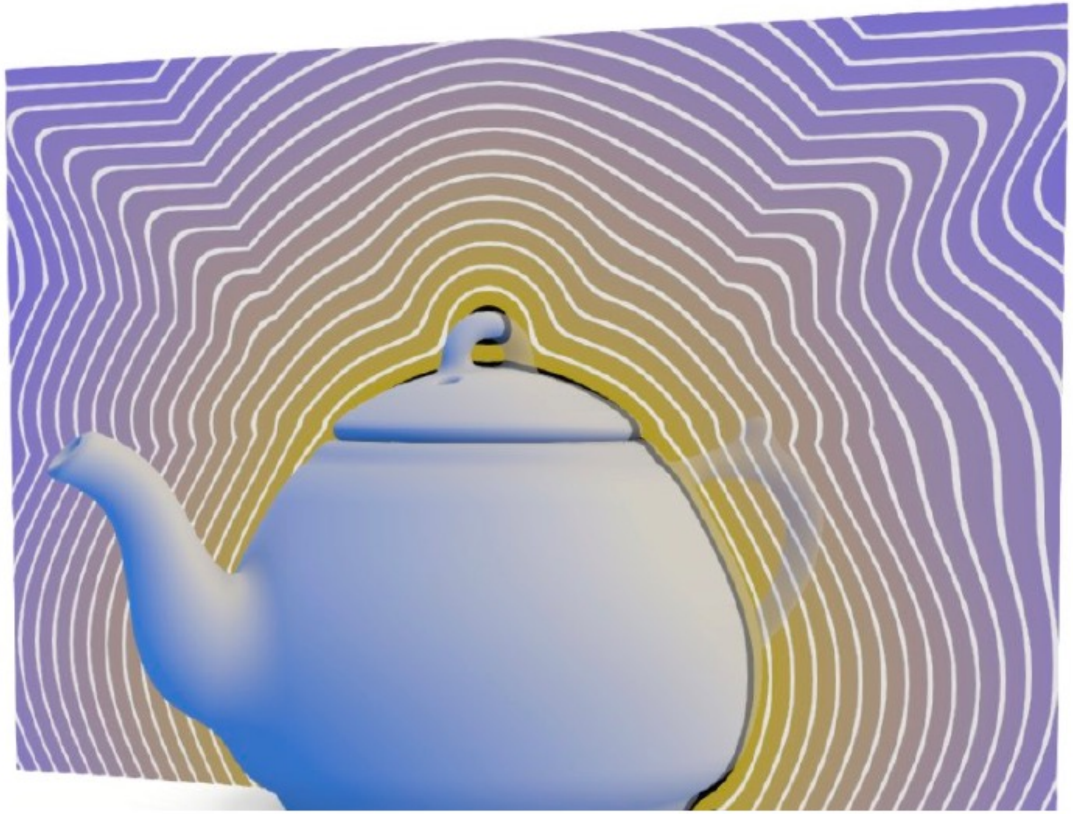
$$\{\mathbf{p} \mid f_\theta(\mathbf{p})\} = 0$$

Neural network with parameters θ

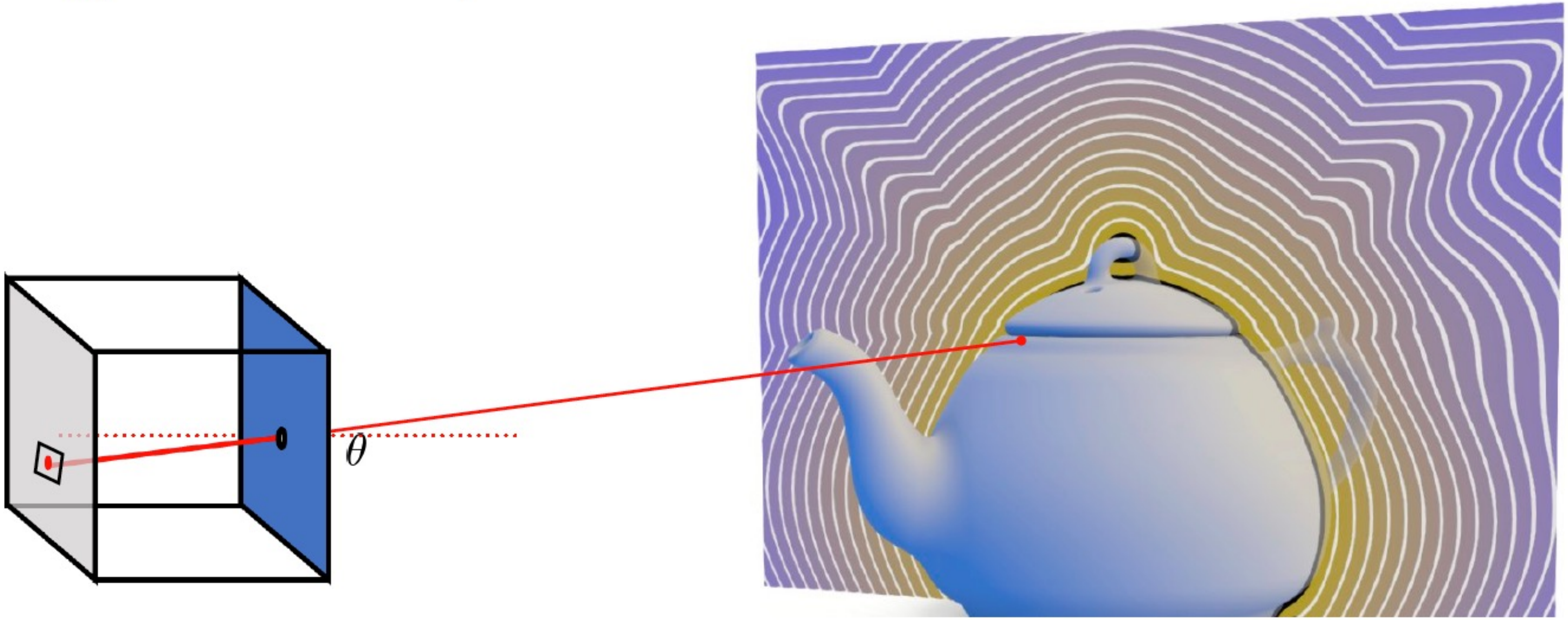
Allows representing complex geometry

Q: What additional constraint is required for a signed distance field?

$$||\nabla f|| = 1$$



Rendering Neural Implicit Surfaces

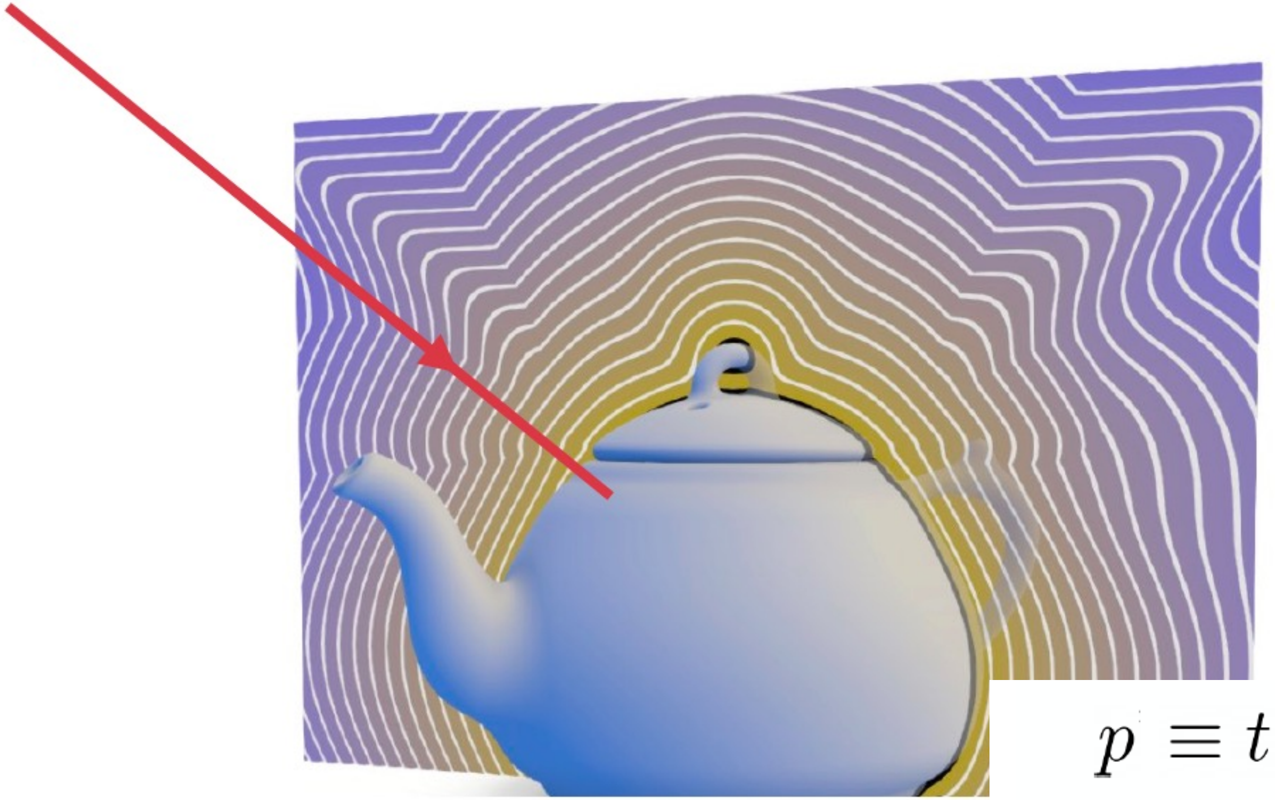


The appearance at each pixel is determined by a unique surface point

Where does a ray intersect an implicit surface?

How to model appearance?

Intersections of Rays with Implicit Surfaces



Input:

Implicit Representation $f_\theta(\cdot)$

Query Ray $r : x_0 + td$

Output:

Intersection point: p

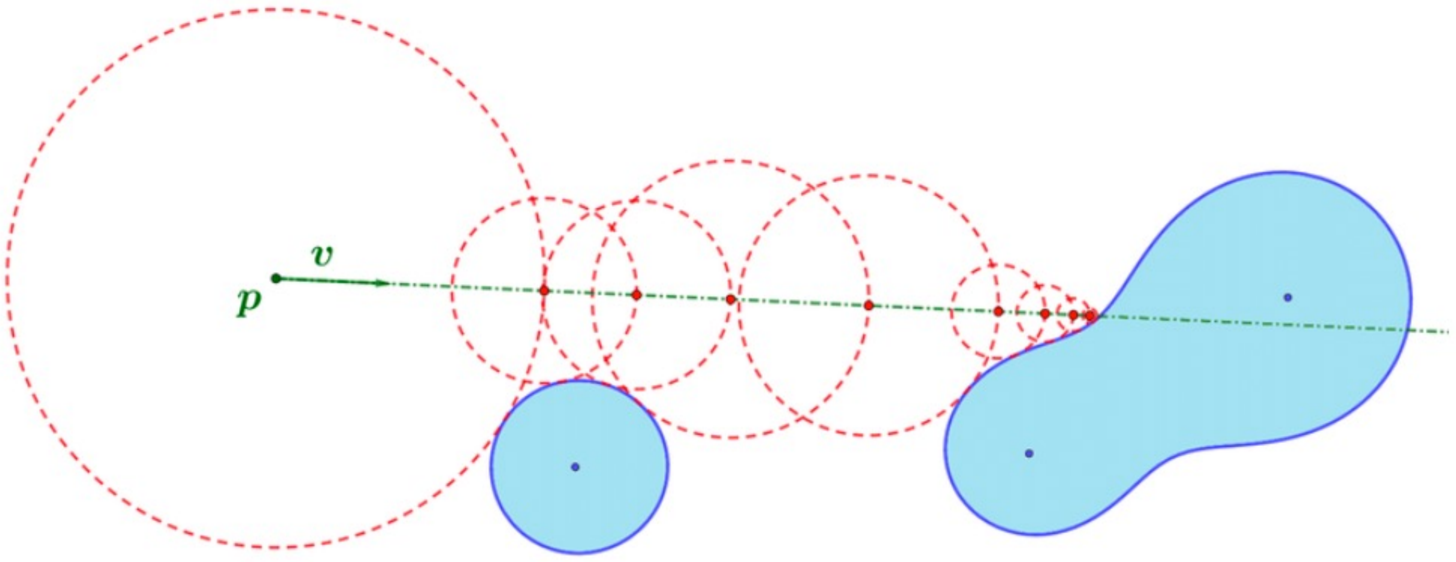
$$p \equiv t^* d + x_0$$
$$\text{s.t.} \quad f_\theta(p) = 0$$
$$\bar{f}_\theta(t) \equiv f_\theta(x_0 + td)$$

a 1-d search problem!

d = unit vector denoting the direction of the ray.
t = scalar distance between any point p on the ray and the origin x0.

How to solve this?
- Use root-finding methods (e.g. secant method)

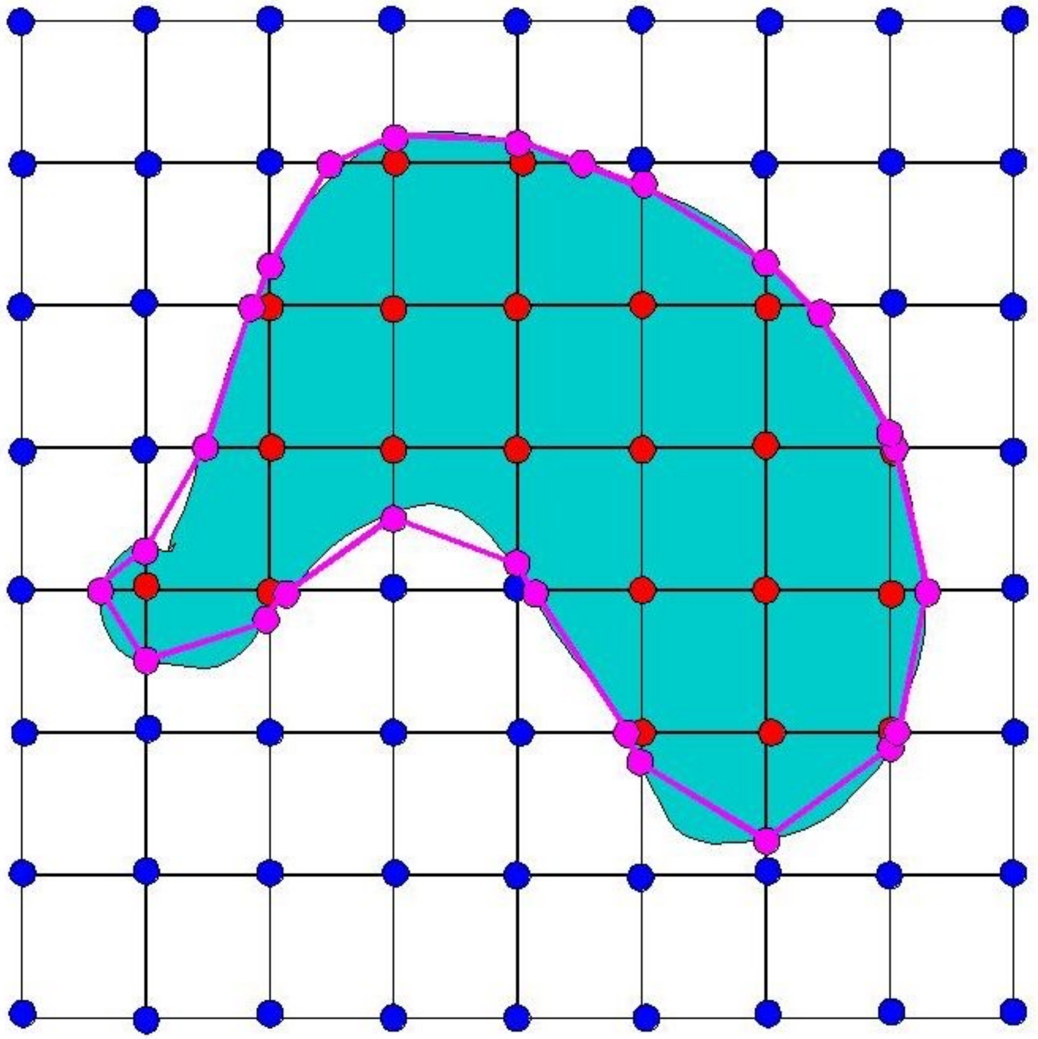
Intersections of Rays with SDFs: Sphere Tracing



$$\text{While } f(p) > \epsilon :$$
$$t = t + f(p)$$
$$p = x_0 + td$$

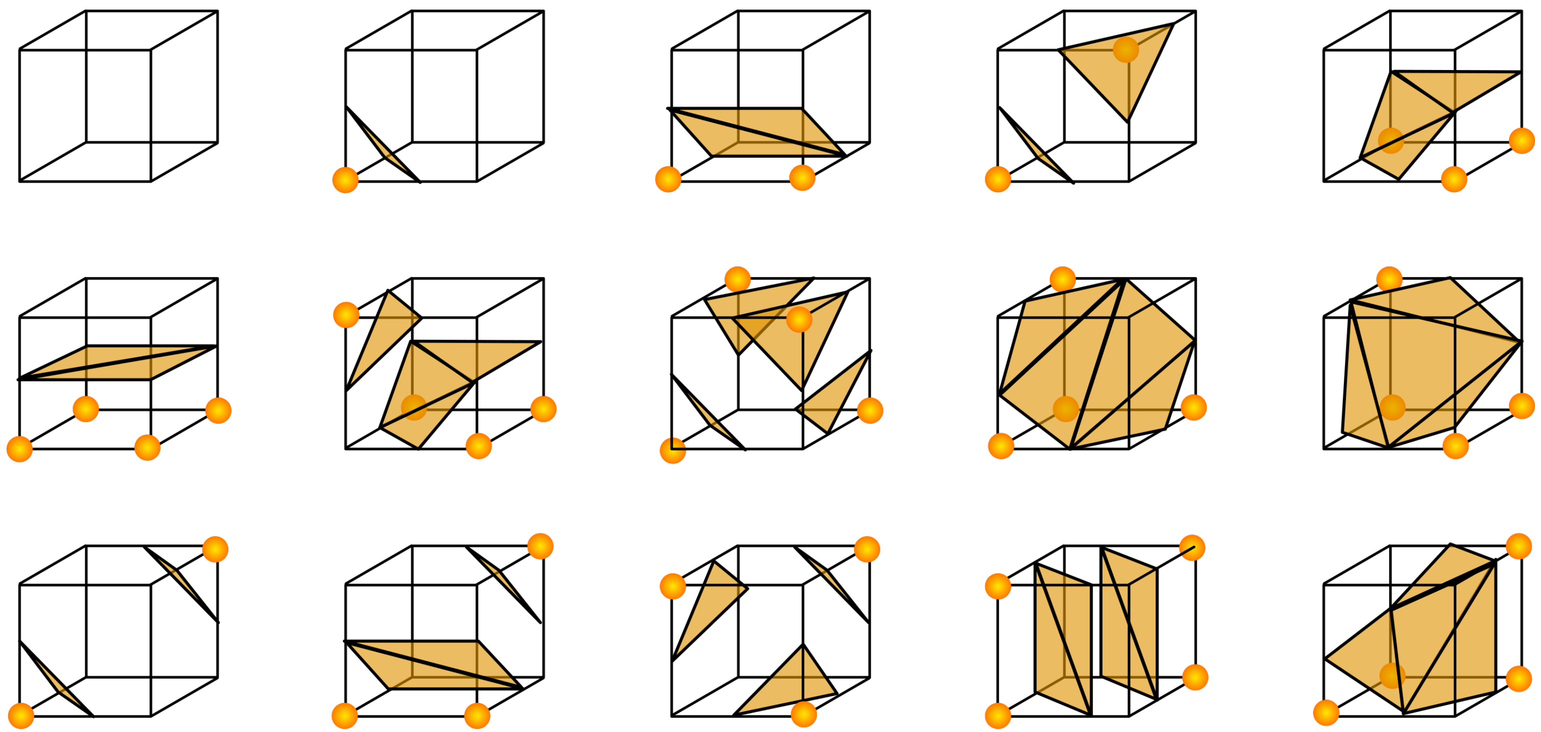
Basically, walk along the ray until close to a surface (or we exceed max-steps)

Marching Cubes in 2D



- Each vertex of a cell can be labelled +ve or –ve, so total 2^4 possibilities.
- After labeling each vertex draw boundary between +ve and –ve.
- Refine the boundary.

Marching Cubes in 3D



In 3D coloring vertex +ve/–ve has 2^8 possibilities. Brute force search is expensive.

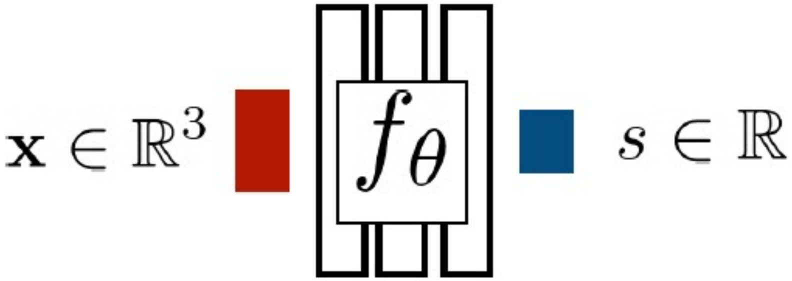
However, we can obtain only 15 unique possibilities from 2^8

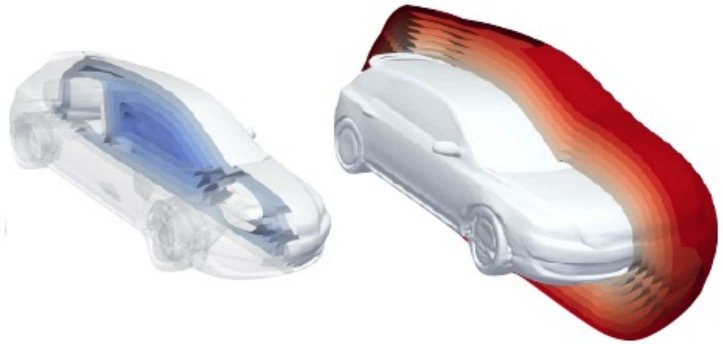
How do we render SDF into a mesh?

- Secant Method (Find roots of a 1-d search problem along each ray)
- Sphere Tracing
- Marching Cubes

Questions we want to answer

- How do we create a 3D mesh from a SDF function? (SDF rendering)
- How do we generalize this to any objects? (Training Deep SDF)
- How do we use this during inference? (Testing DeepSDF)
- How do DeepSDF concept extends to NeRF and other methods?

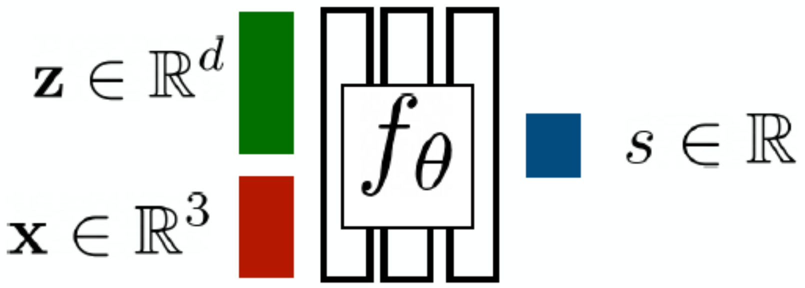




$$f_\theta : \mathbb{R}^3 \to \mathbb{R}$$

Signed Distance Field (for a single instance):
(position) → (distance)

Latent Conditioning based SDFs



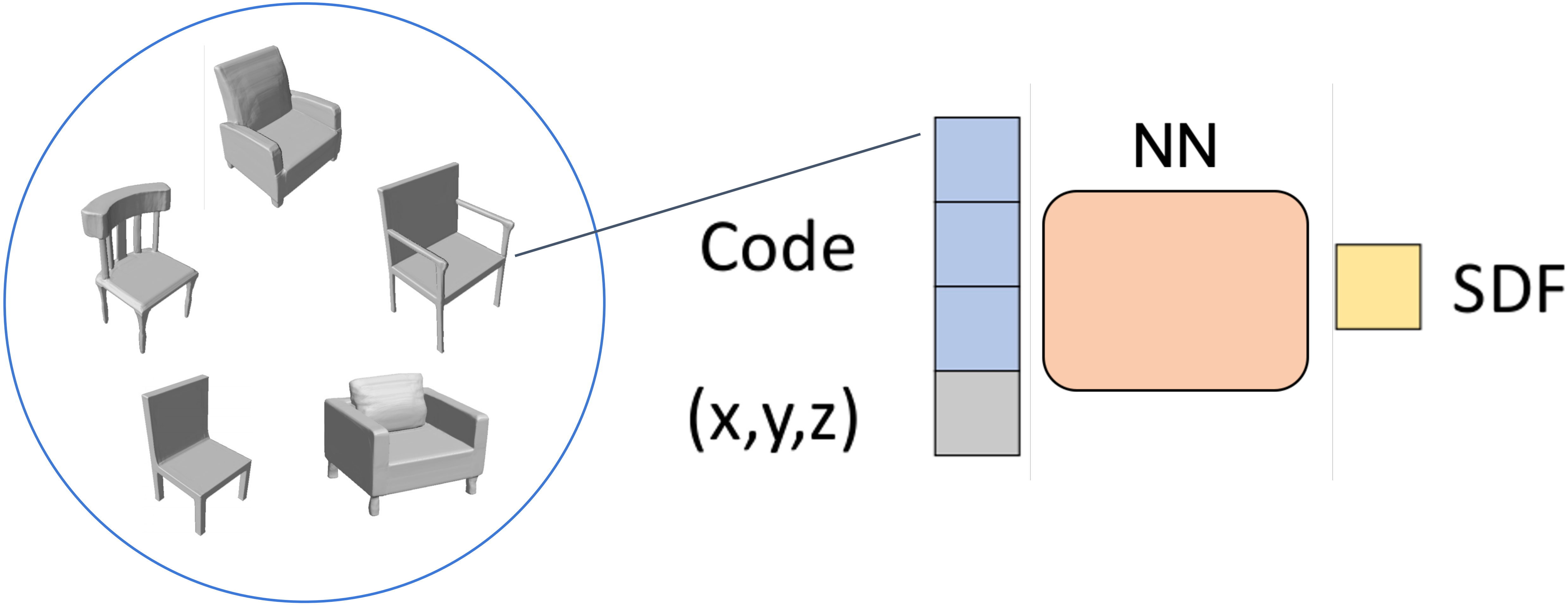
$$f_\theta : \mathbb{R}^3 \times \mathbb{R}^d \to \mathbb{R}$$

Generalizable Signed Distance Field:
(latent code, position) → (distance)

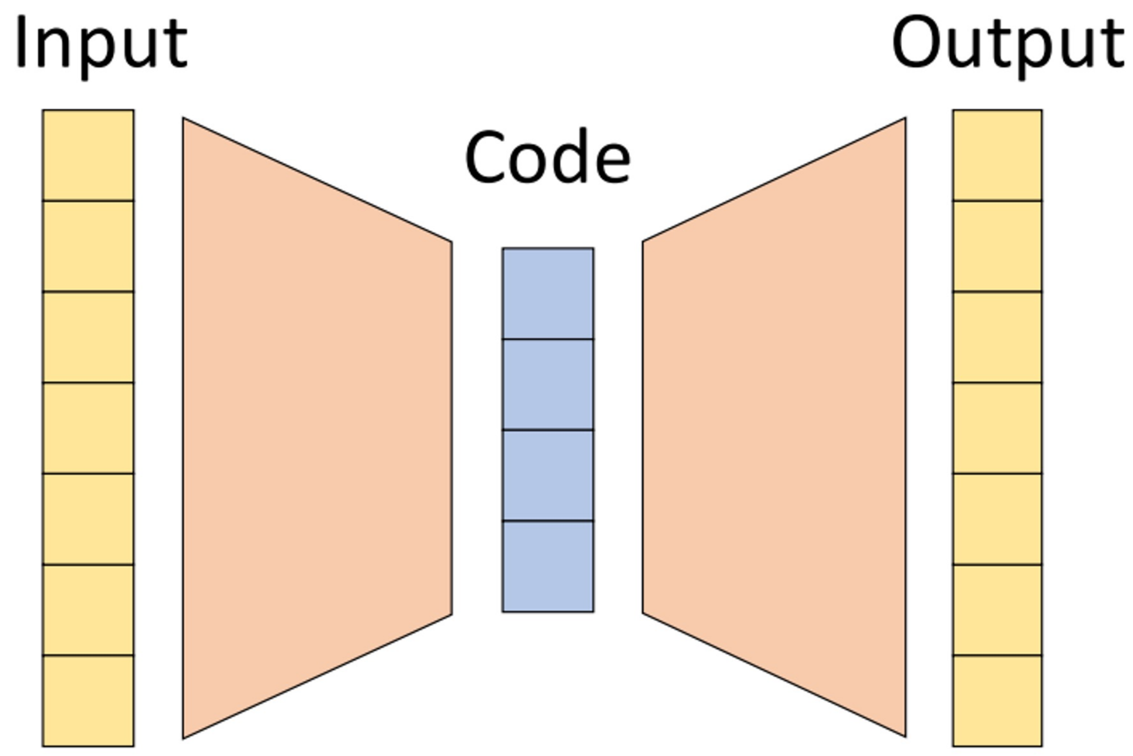
Each object is represented by a corresponding latent code (only *d* parameters per instance)

The same neural net parameters across **all** objects

Coding Multiple Shapes

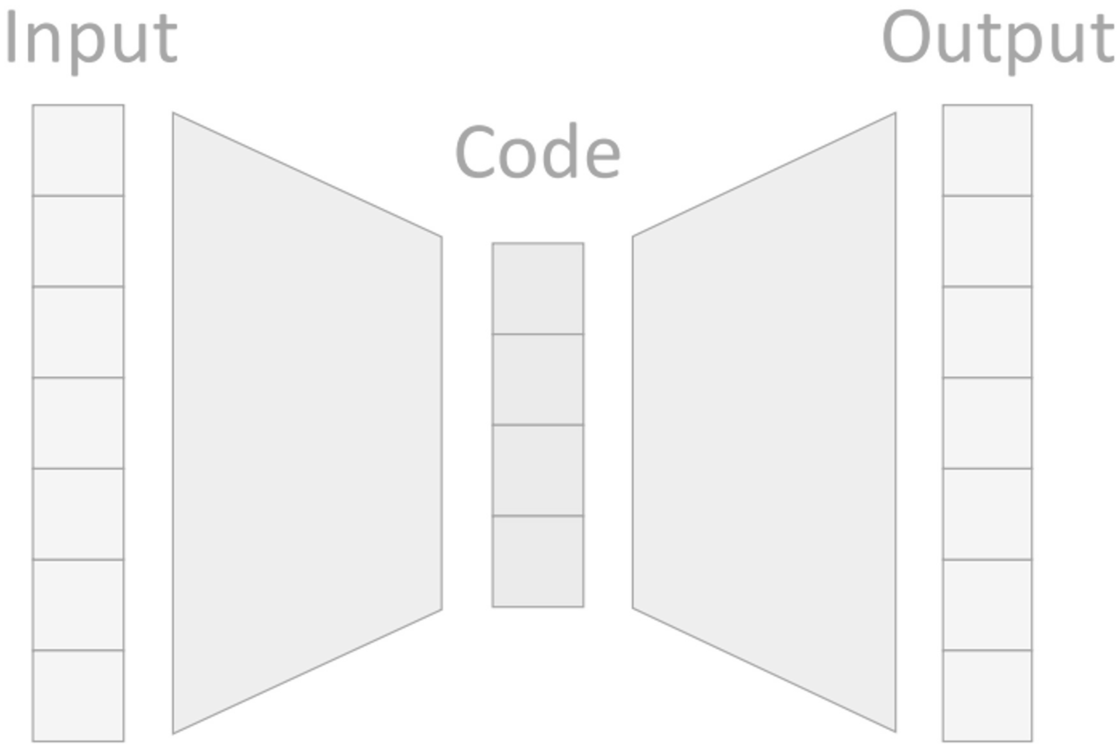


Auto-Encoder

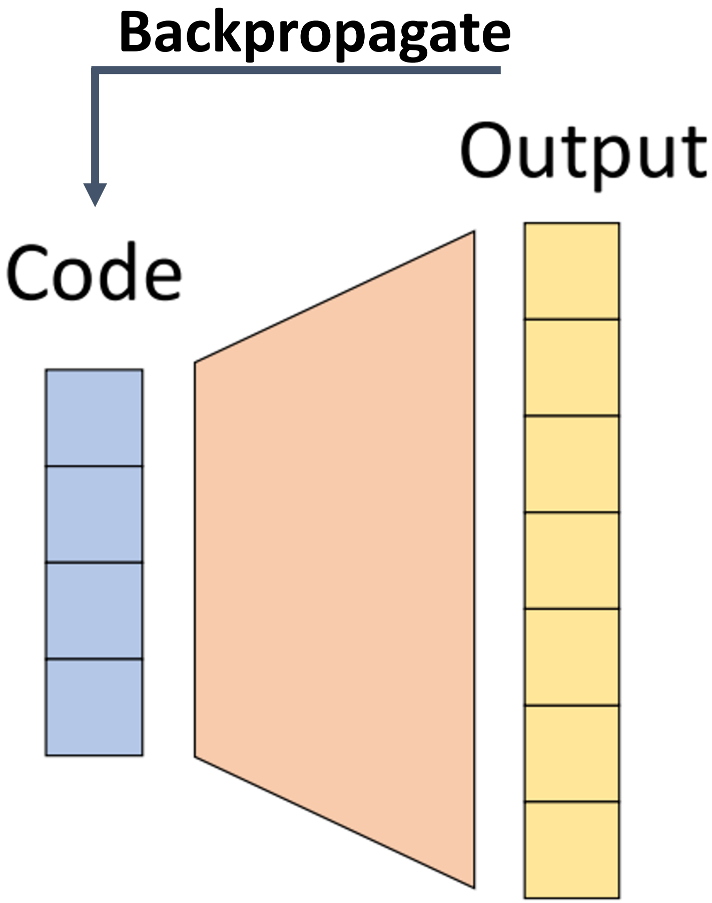


Auto-Encoder

Auto-Decoder

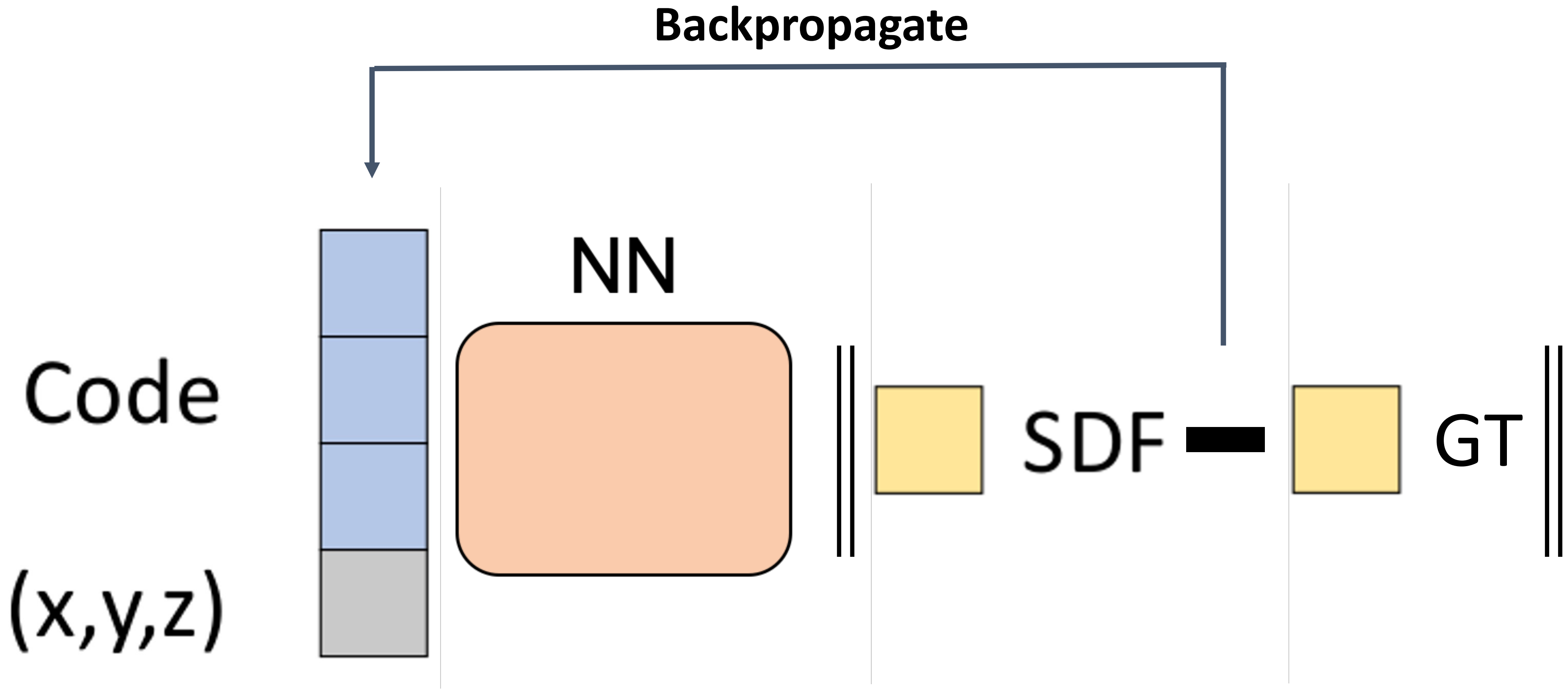


Auto-Encoder



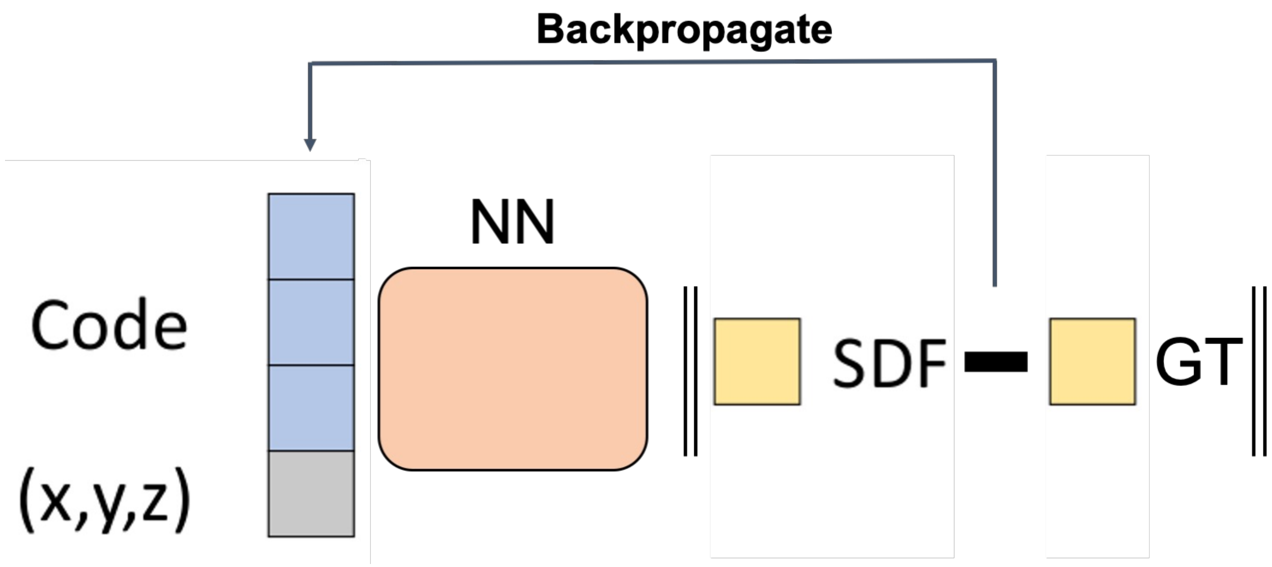
Auto-Decoder

Auto-Decoder



During Training: Optimize for both NN parameters and Code

Benefits of Auto-Decoder

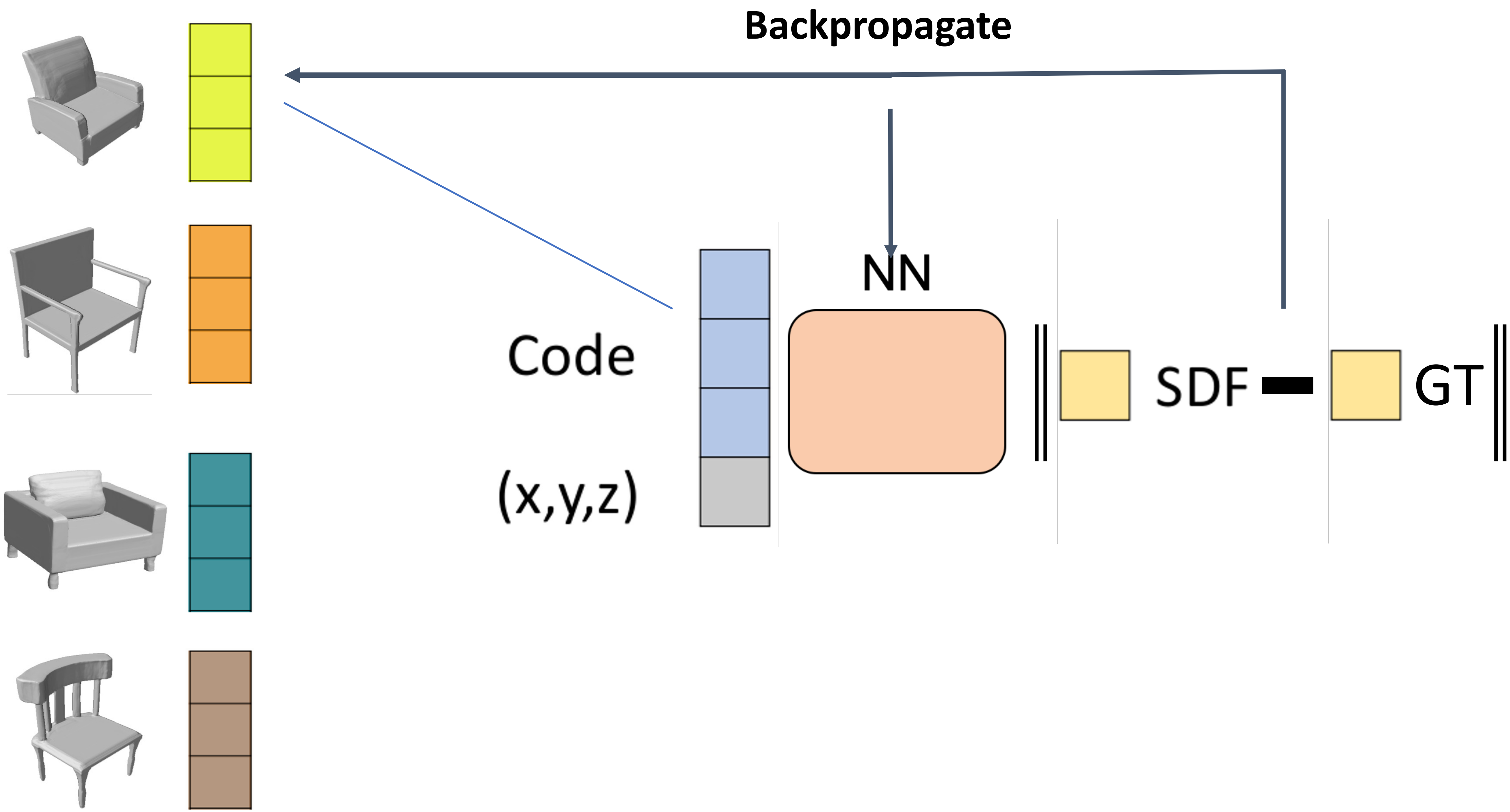


Benefits during Inference

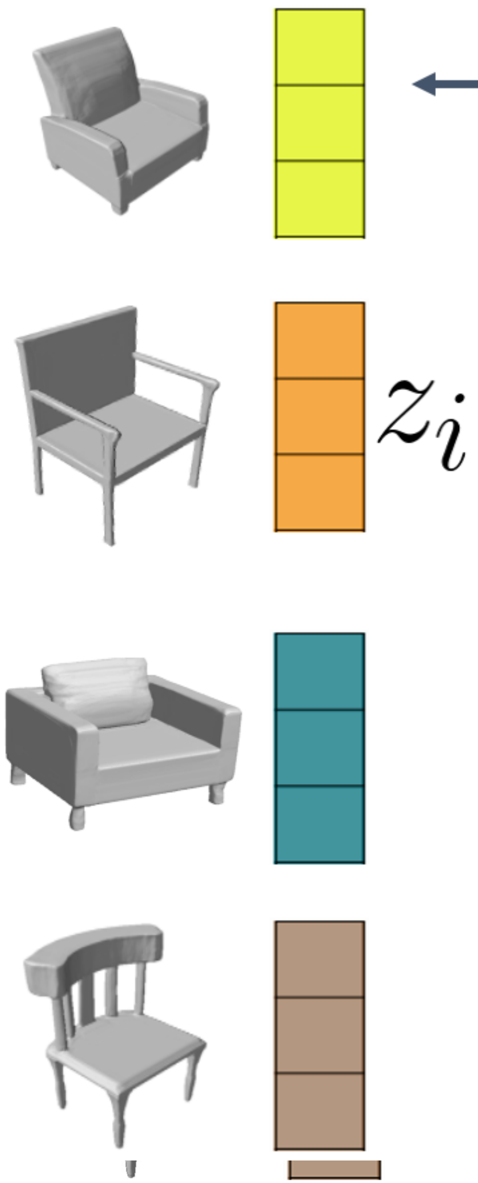
1. Any Number of Observations – Partial

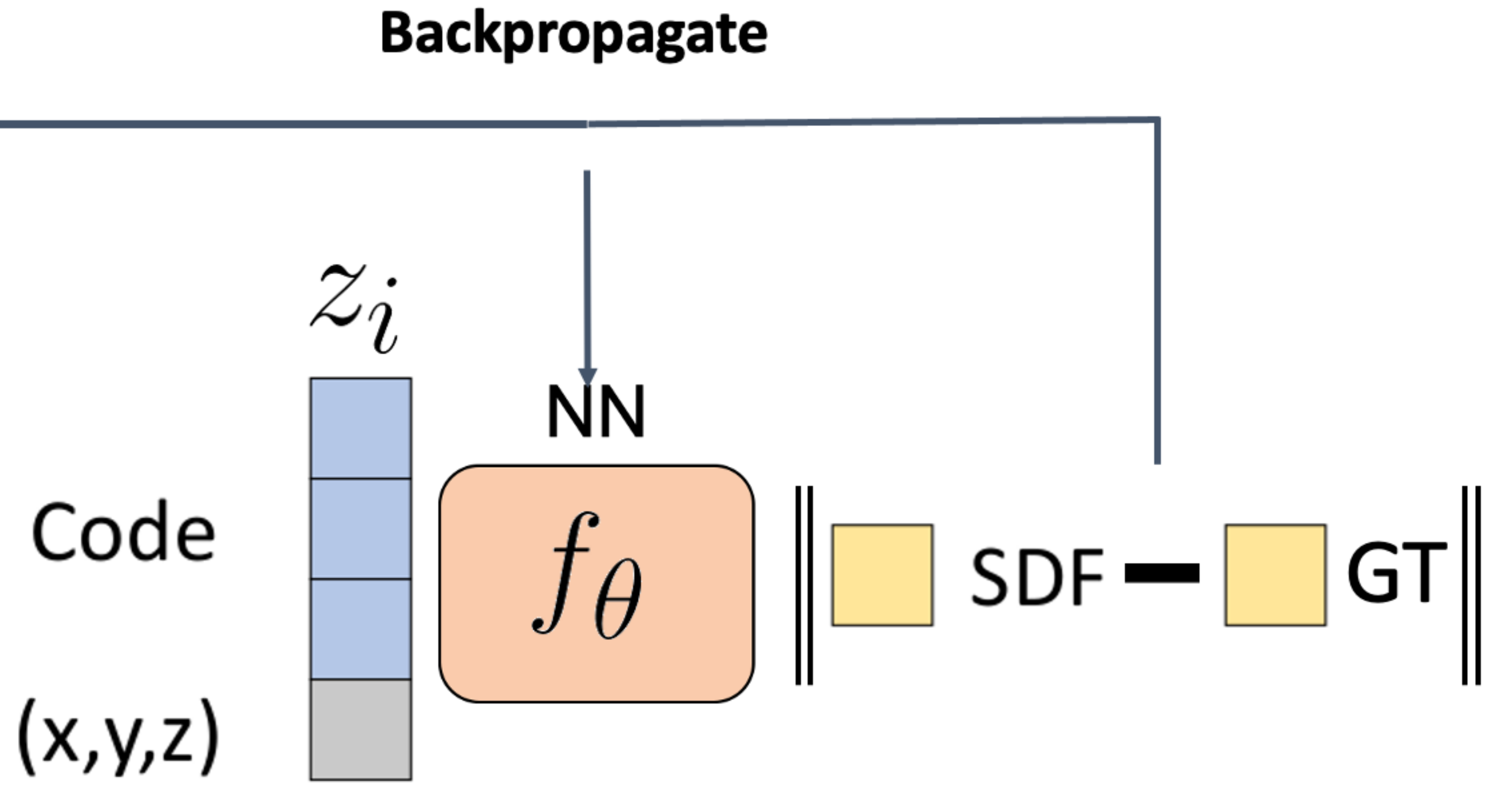
2. More Controlled Inference – e.g. Accuracy, Priors

Auto-Decoder Training



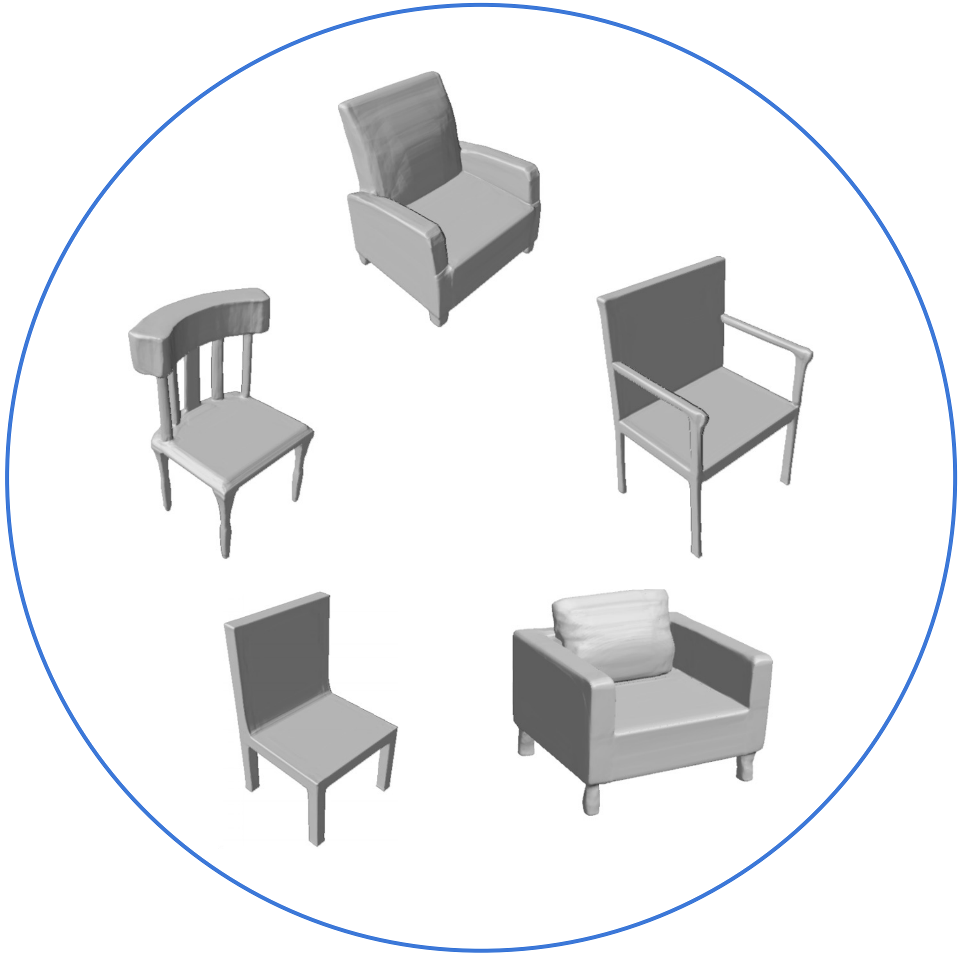
Auto-Decoder Training

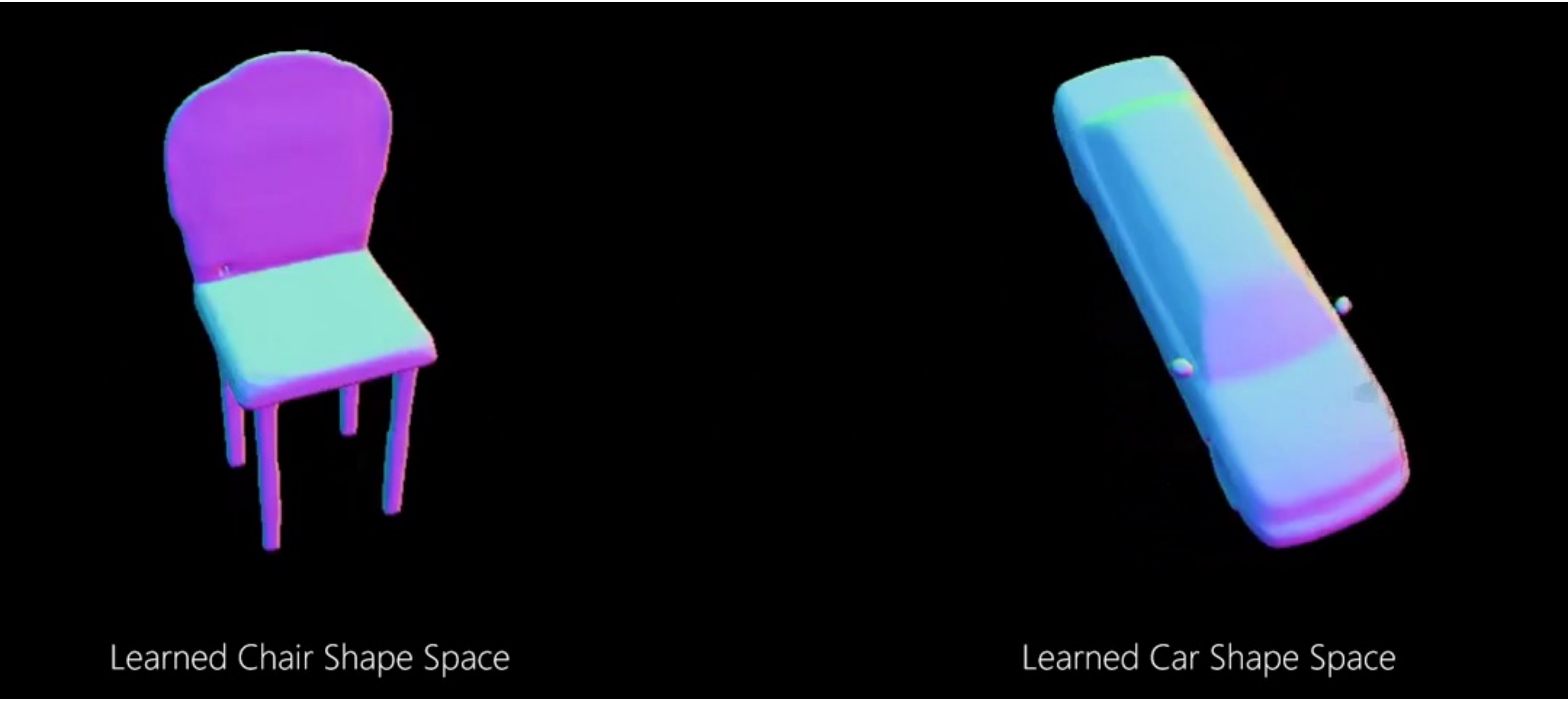




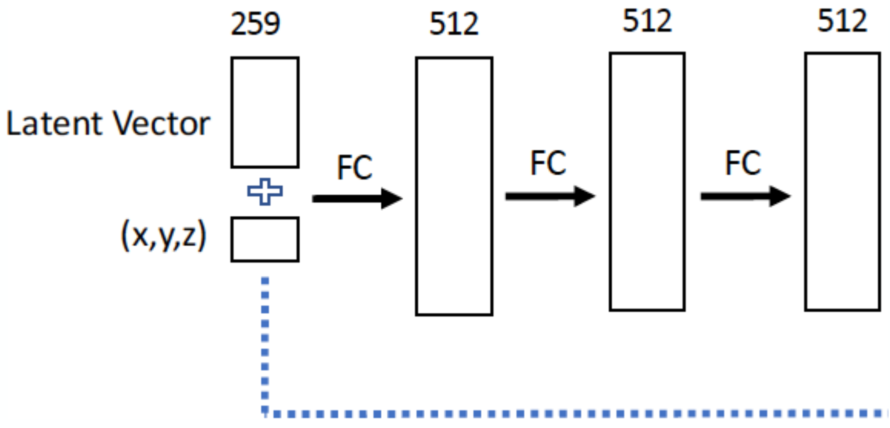
$$\arg\min_{\theta, \{\boldsymbol{z}_i\}_{i=1}^{N}} \sum_{i=1}^{N} \left(\sum_{j=1}^{K} \mathcal{L}(f_\theta(\boldsymbol{z}_i, \boldsymbol{x}_j), s_j) \right)$$

Latent Space of Shapes



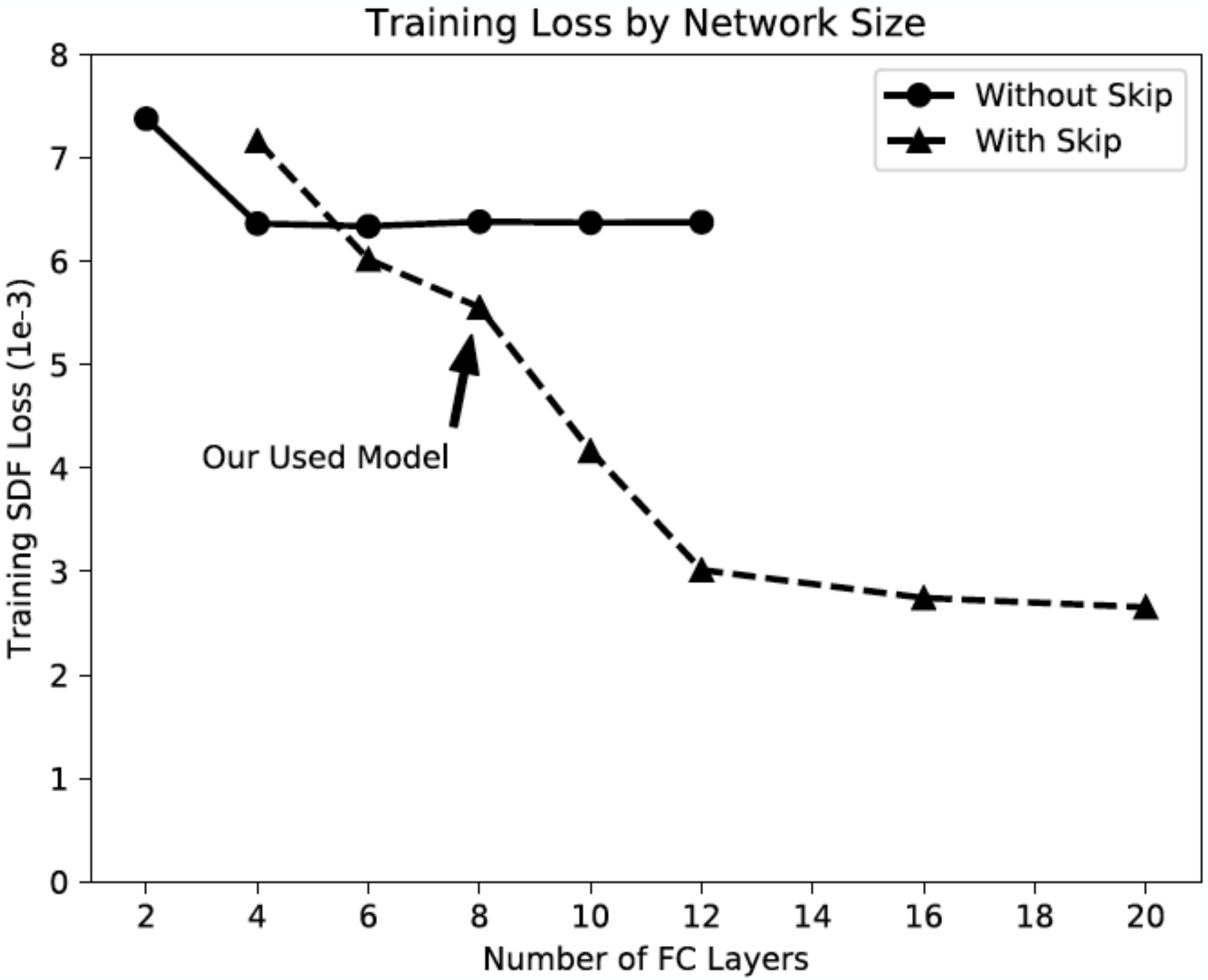

Learned Chair Shape Space
Learned Car Shape Space

Conditional Neural SDFs



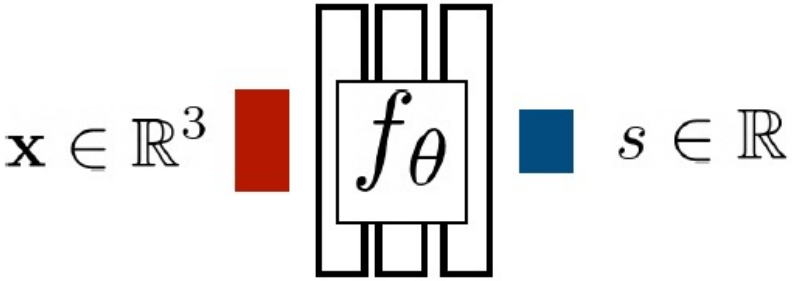
Simplest approach — just conca

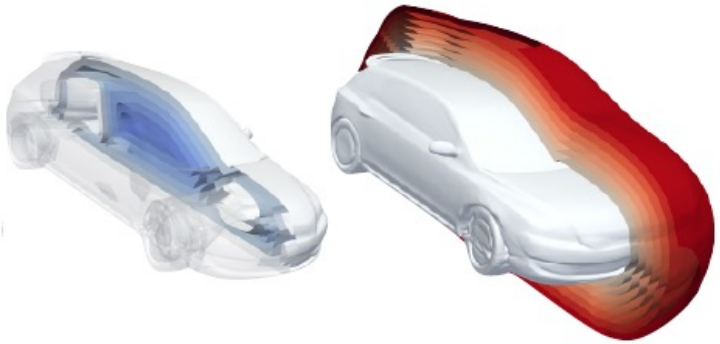
Skip connecti



Questions we want to answer

- How do we create a 3D mesh from a SDF function? (SDF rendering)
- How do we generalize this to any objects? (Training Deep SDF)
- How do we use this during inference? (Testing DeepSDF)
- How do DeepSDF concept extends to NeRF and other methods?

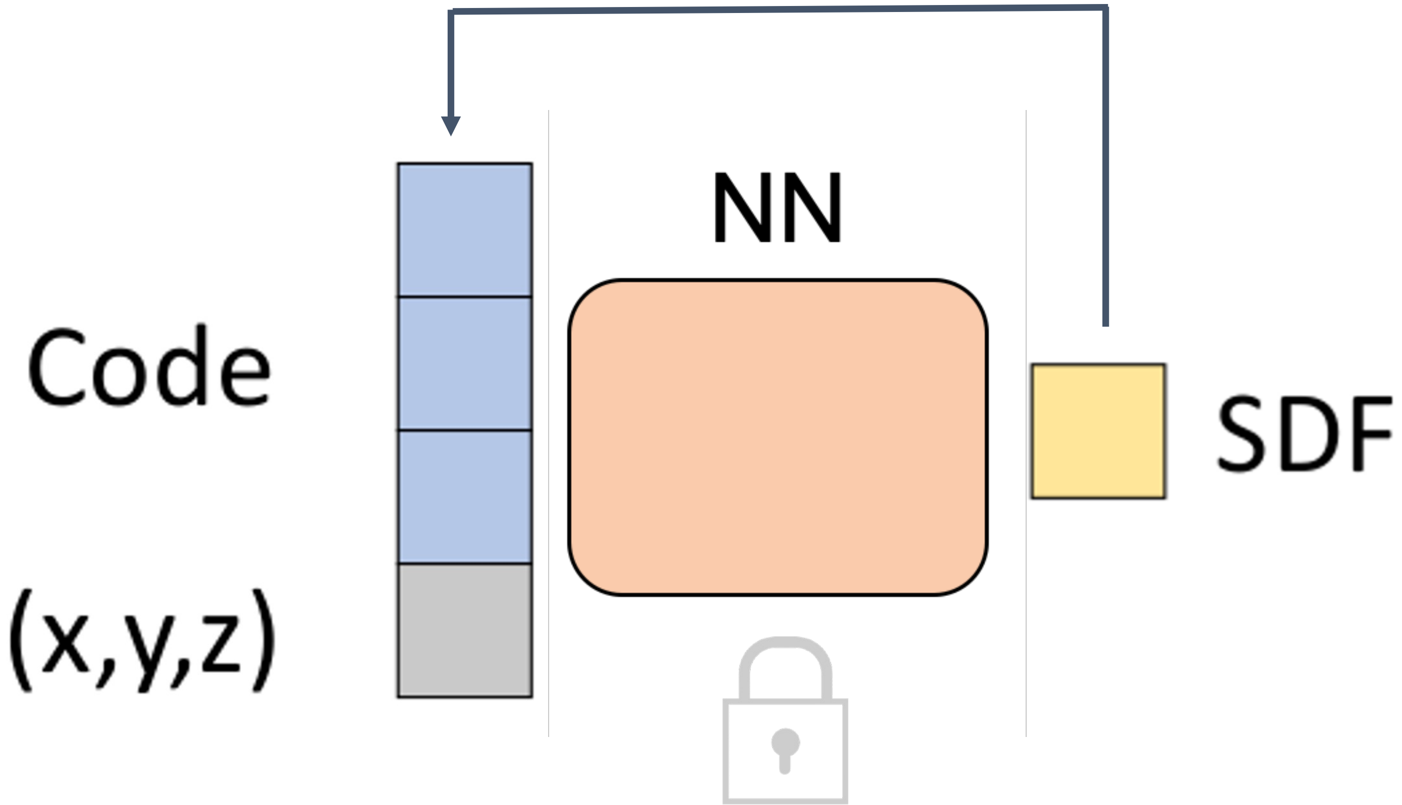




$$f_\theta : \mathbb{R}^3 \to \mathbb{R}$$

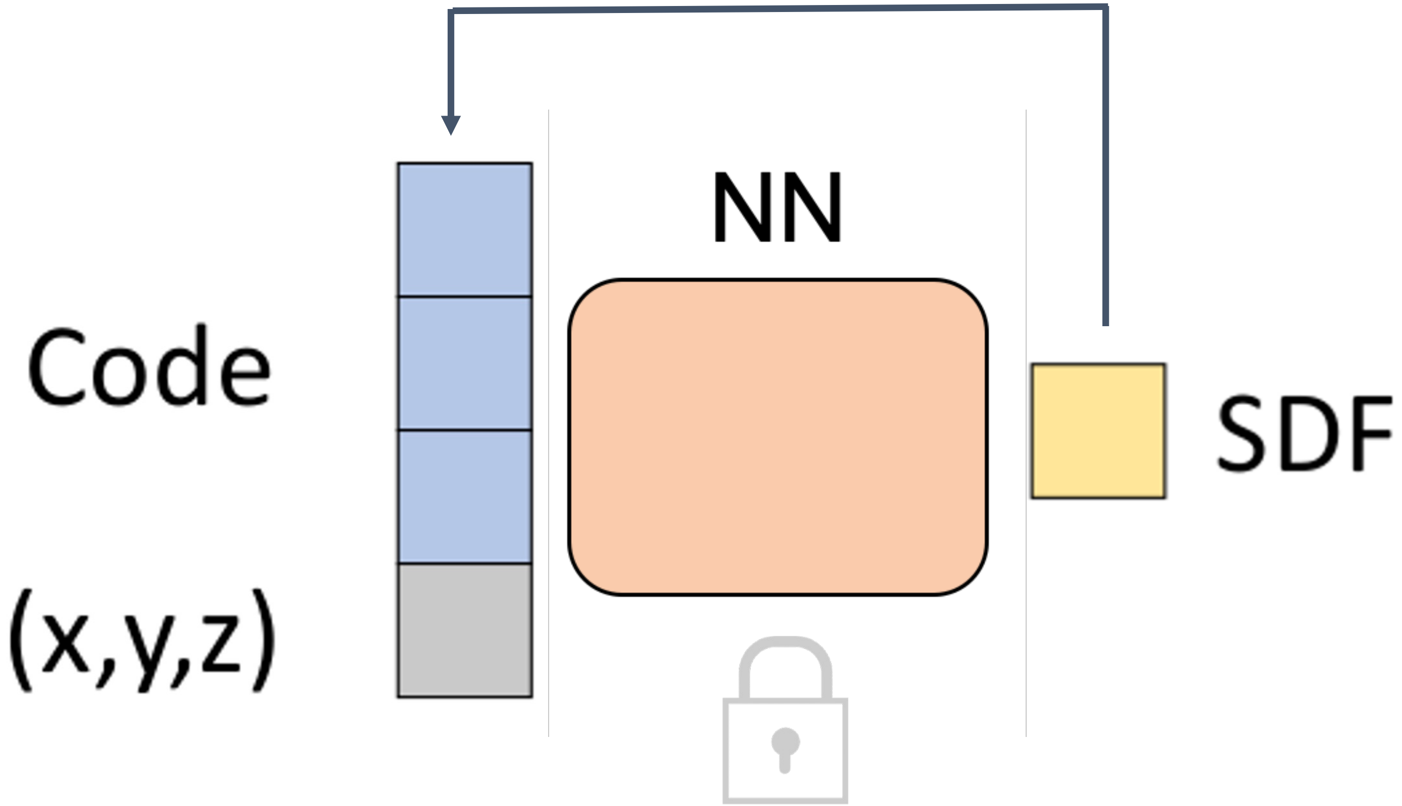
Signed Distance Field (for a single instance):
(position) → (distance)

Auto-Decoder Inference



Test Shape

Auto-Decoder Inference

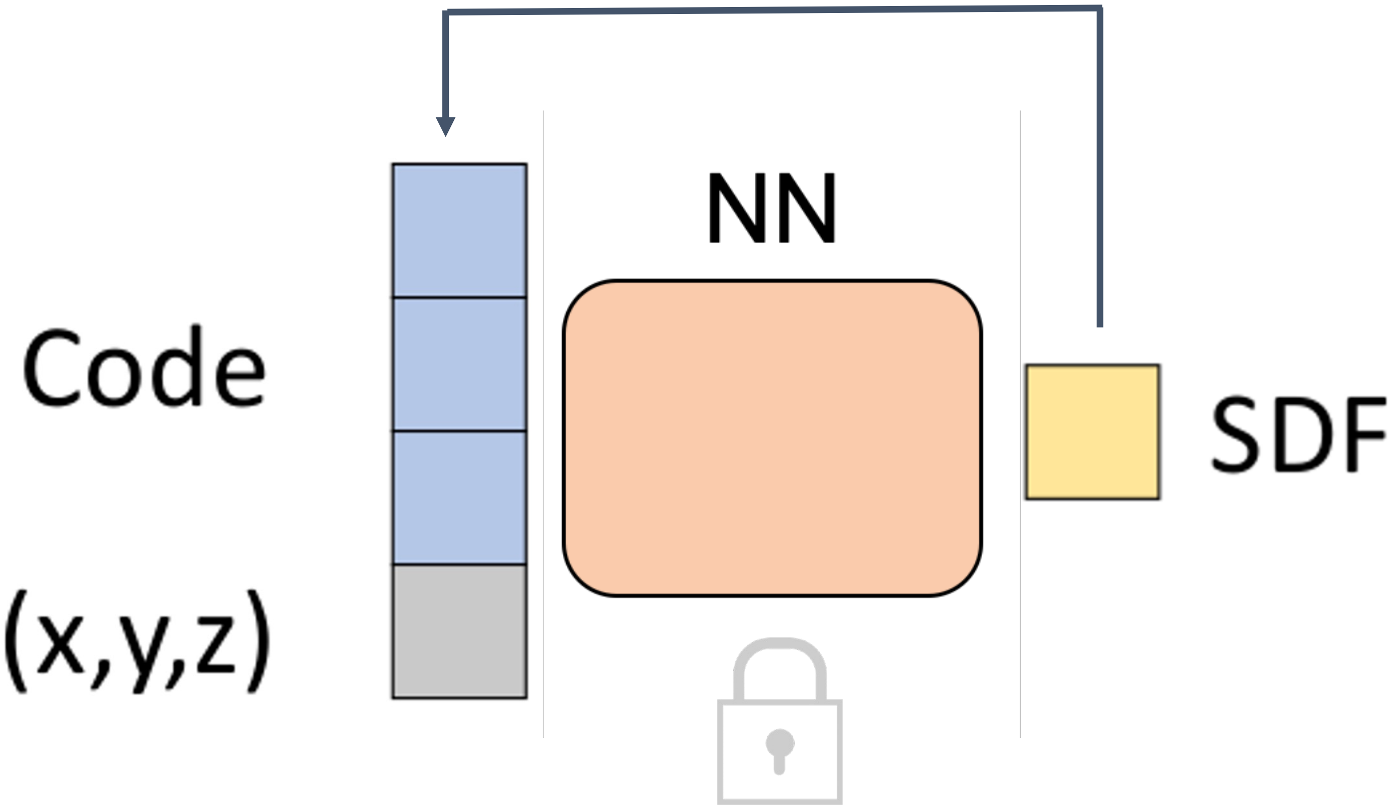


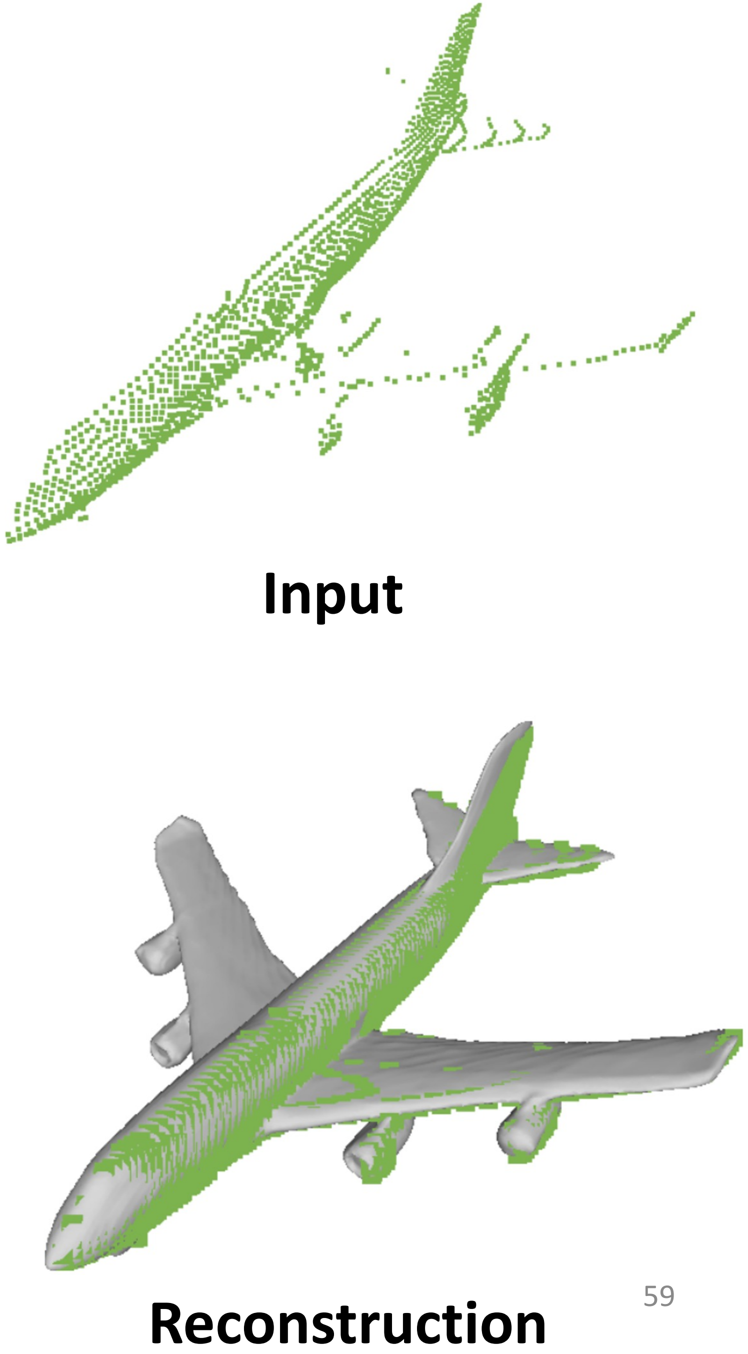
Optimize the Code until the SDF matches the Test Shape

Test Shape

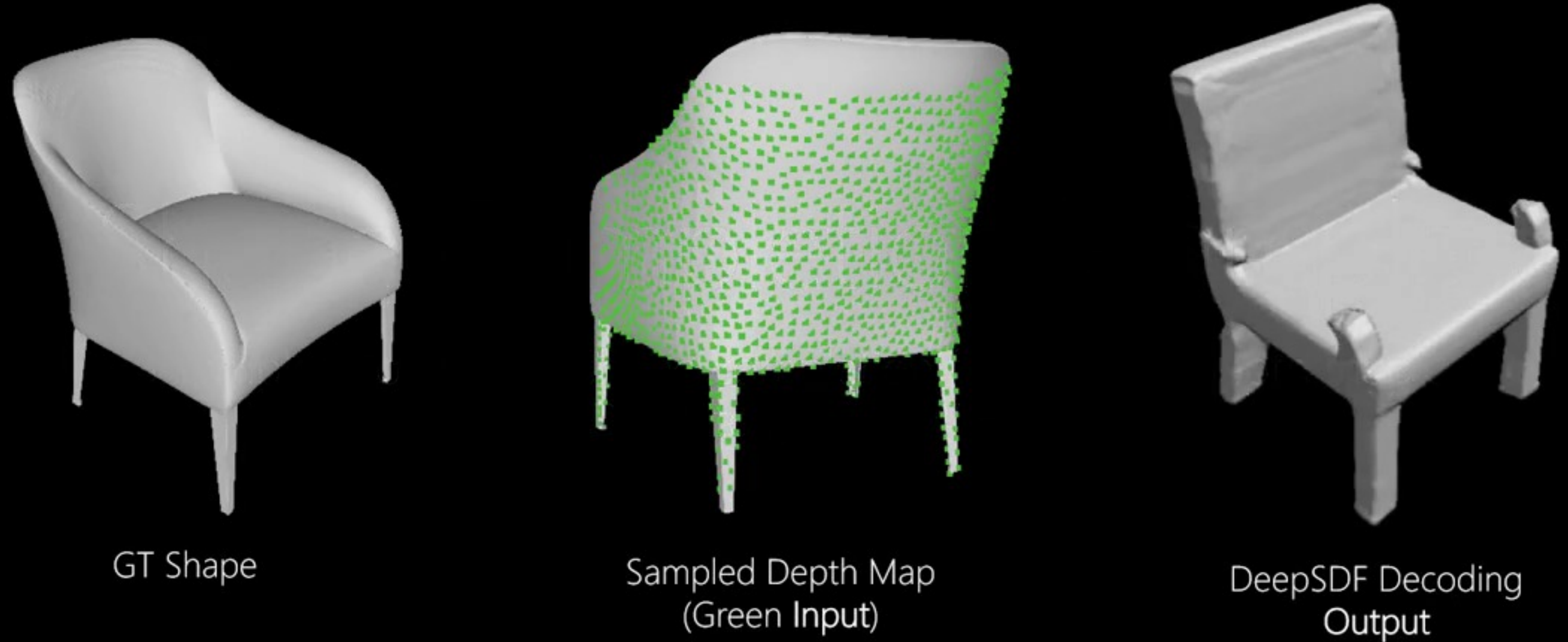
Reconstruction

Auto-Decoder Inference
Input = Partial Observation





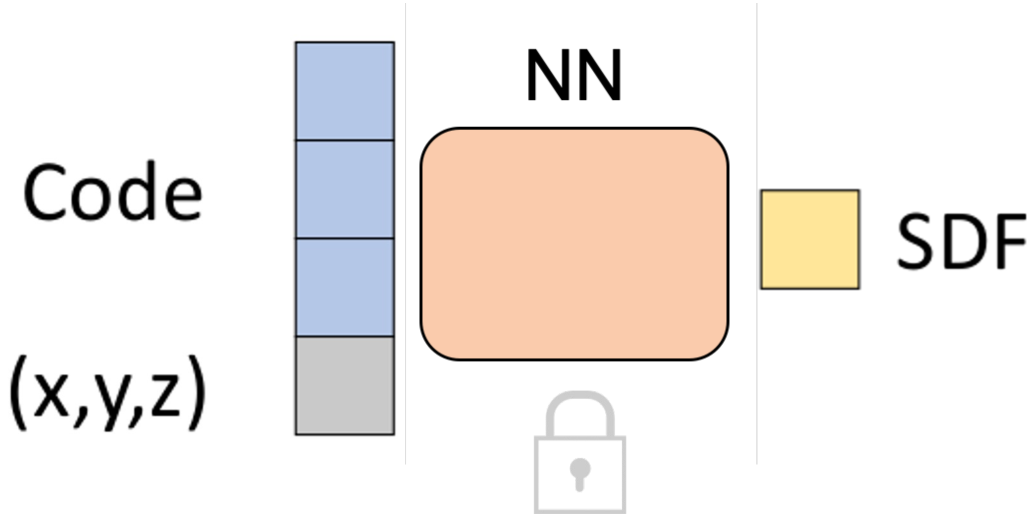
DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation



Example evolution of shape completion for a partial depth map (ADAM iteration: 1-400)

Adding Priors to Inference

$$\hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}} \sum_{(\boldsymbol{x}_j, \boldsymbol{s}_j) \in X} \mathcal{L}(f_\theta(\boldsymbol{z}, \boldsymbol{x}_j), s_j)$$



Adding Priors to Inference

$$\hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}} \sum_{(\boldsymbol{x}_j, \boldsymbol{s}_j) \in X} \mathcal{L}(f_\theta(\boldsymbol{z}, \boldsymbol{x}_j), s_j)$$

Distribution Prior: $\frac{1}{\sigma^2} ||\boldsymbol{z}||_2^2$

SDF Regularization: $\left(\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x};\theta)\| - 1 \right)^2$ (Matan et al. 2020)

Normal Regularization: $\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_i;\theta) - \boldsymbol{n}_i\|$

Results

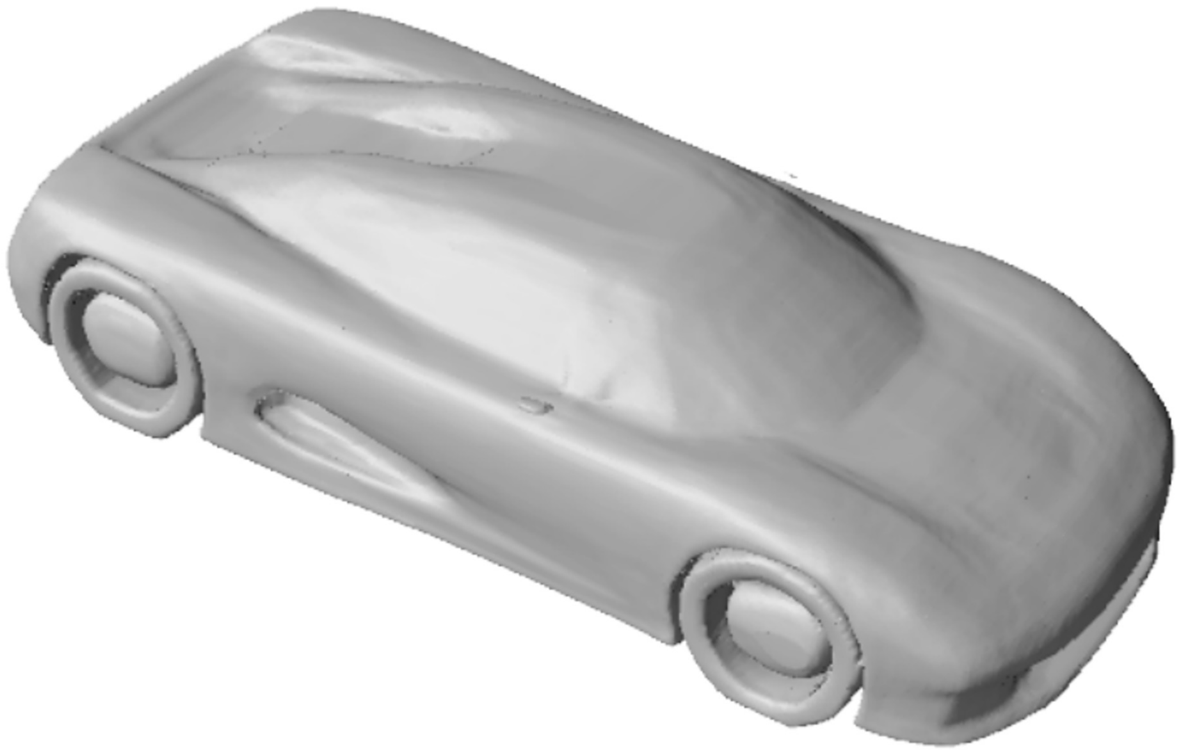
Auto-encoding unknown shapes

CD, median					
AtlasNet-Sph.	0.511	0.079	0.389	2.180	0.330
AtlasNet-25	0.276	0.065	0.195	0.993	0.311
DeepSDF	**0.072**	**0.036**	**0.068**	**0.219**	**0.088**

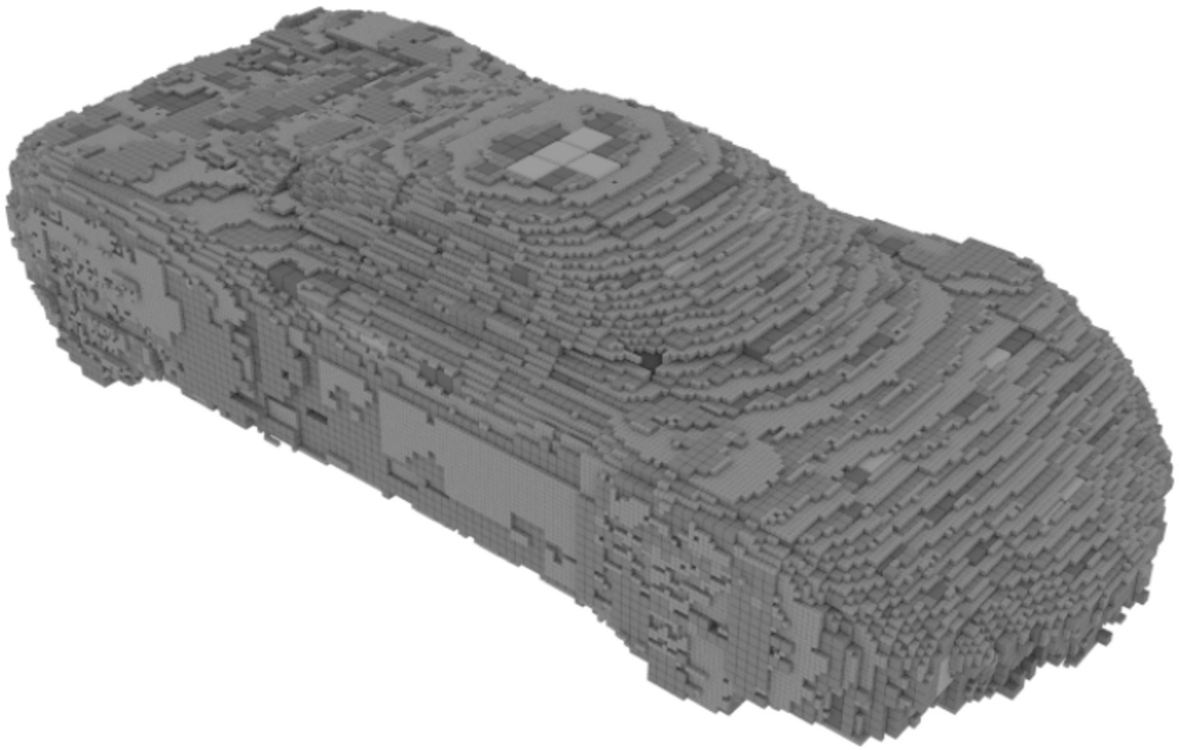
Shape completion

	lower is better				*higher is better*	
Method \Metric	CD, med.	CD, mean	EMD	Mesh acc.	Mesh comp.	Cos sim.
chair						
3D-EPN	2.25	2.83	0.084	0.059	0.209	0.752
DeepSDF	**1.28**	**2.11**	**0.071**	**0.049**	**0.500**	**0.766**
plane						
3D-EPN	1.63	2.19	0.063	0.040	0.165	0.710
DeepSDF	**0.37**	**1.16**	**0.049**	**0.032**	**0.722**	**0.823**

Results: Comparison with Octree-Based



Our Reconstruction

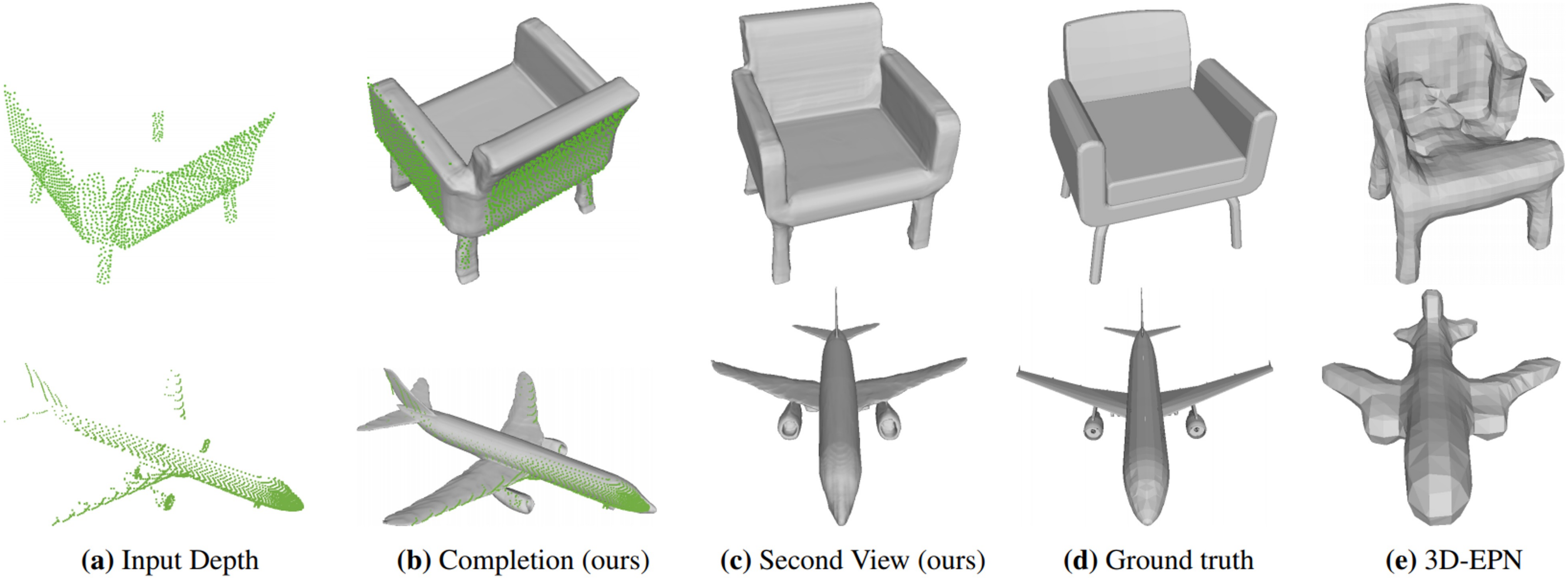


Octree Based

Results: Comparisons with Mesh-Based

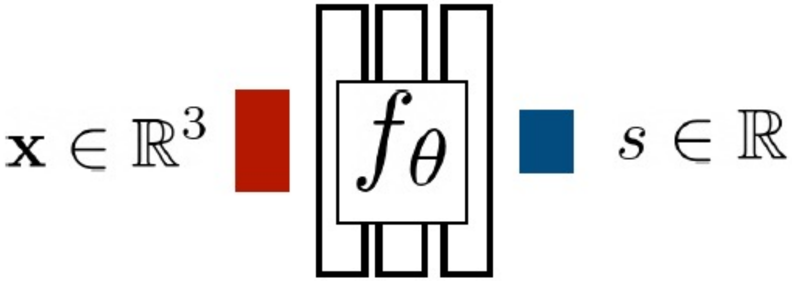
Ground Truth | Our Reconstruction | Atlasnet (25 Patches) | Atlasnet (1 Patch)

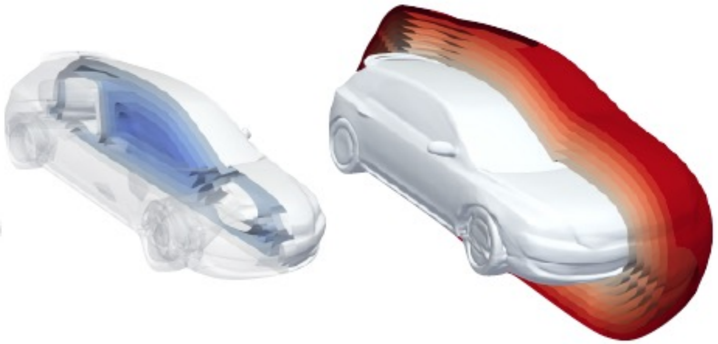
Shape Completion



Questions we want to answer

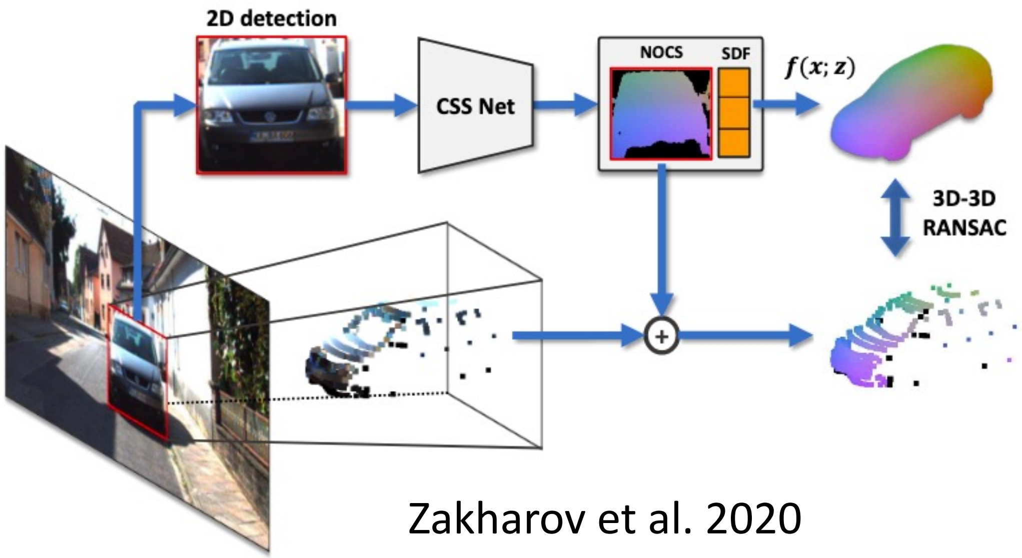
- How do we create a 3D mesh from a SDF function? (SDF rendering)
- How do we generalize this to any objects? (Training Deep SDF)
- How do we use this during inference? (Testing DeepSDF)
- How do DeepSDF concept extends to NeRF and other methods?



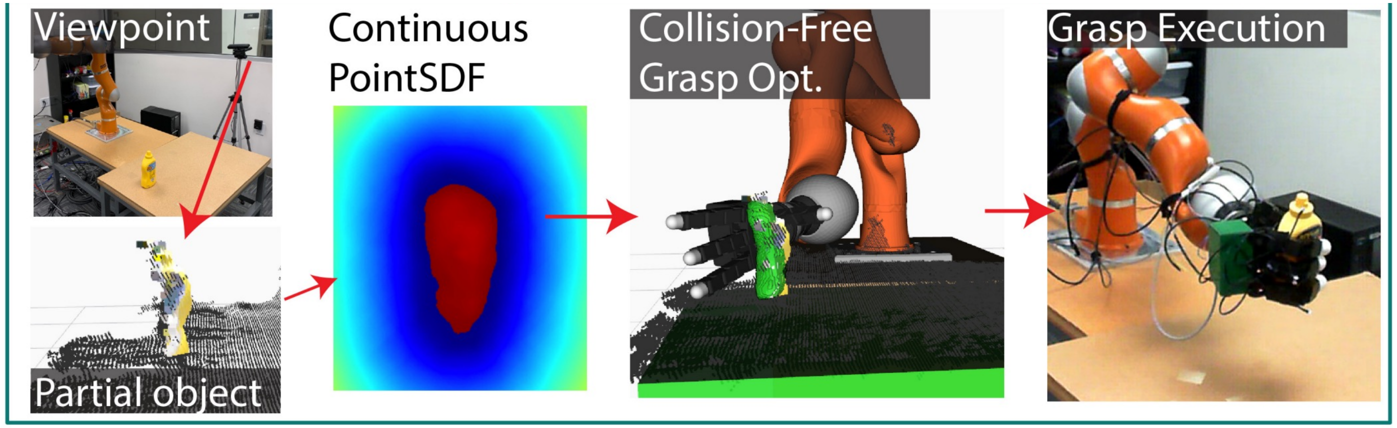


$$f_\theta : \mathbb{R}^3 \rightarrow \mathbb{R}$$

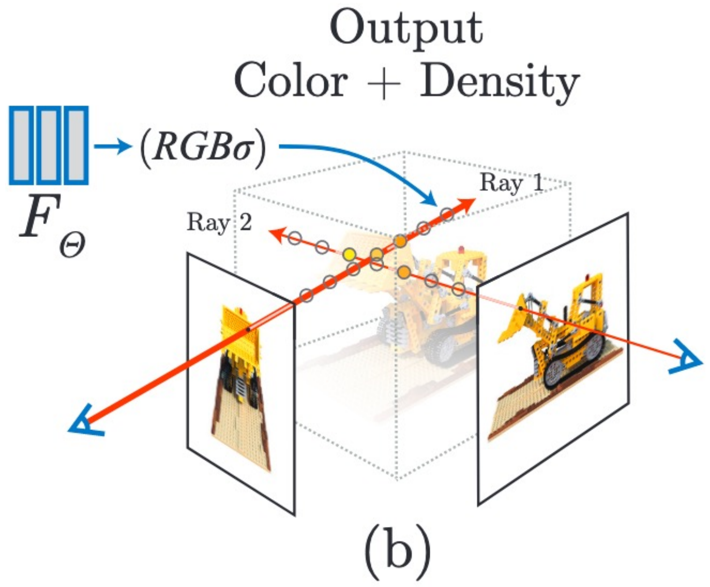
Signed Distance Field (for a single instance):
(position) → (distance)



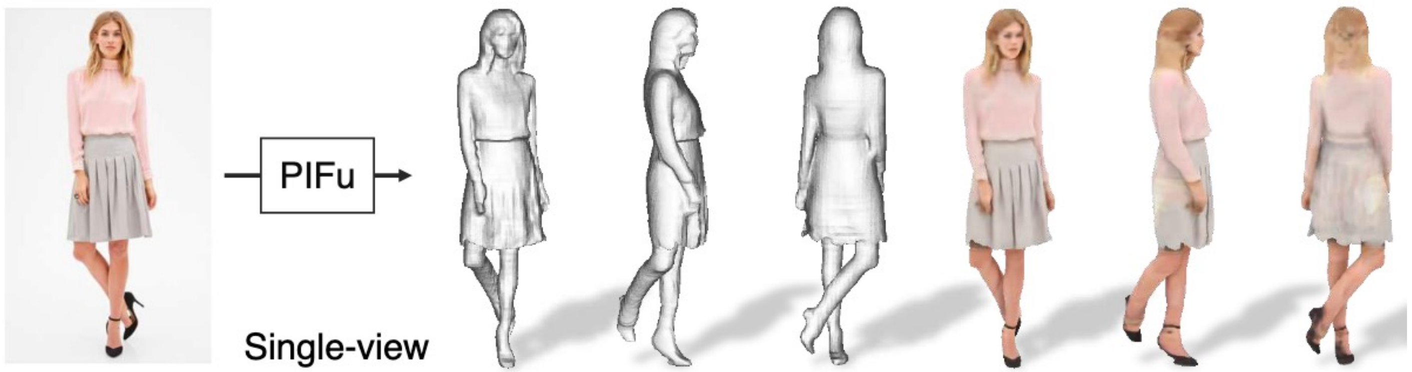
Zakharov et al. 2020



Merwe et al. 2020

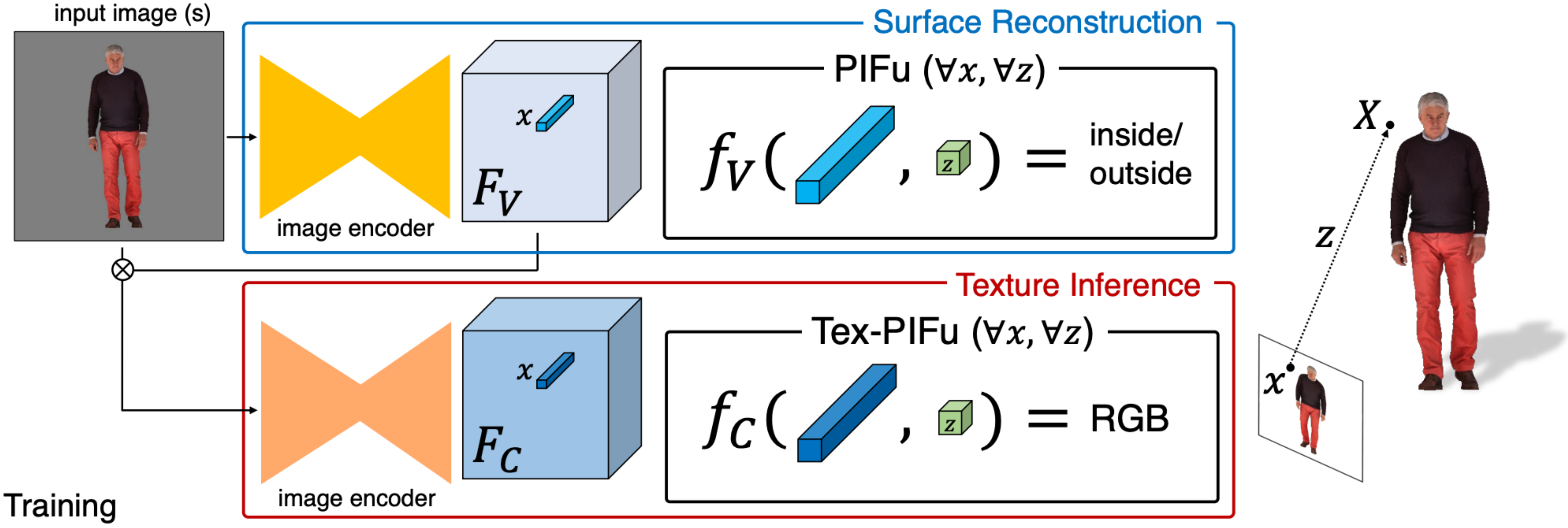


Mildenhall et al. 2020



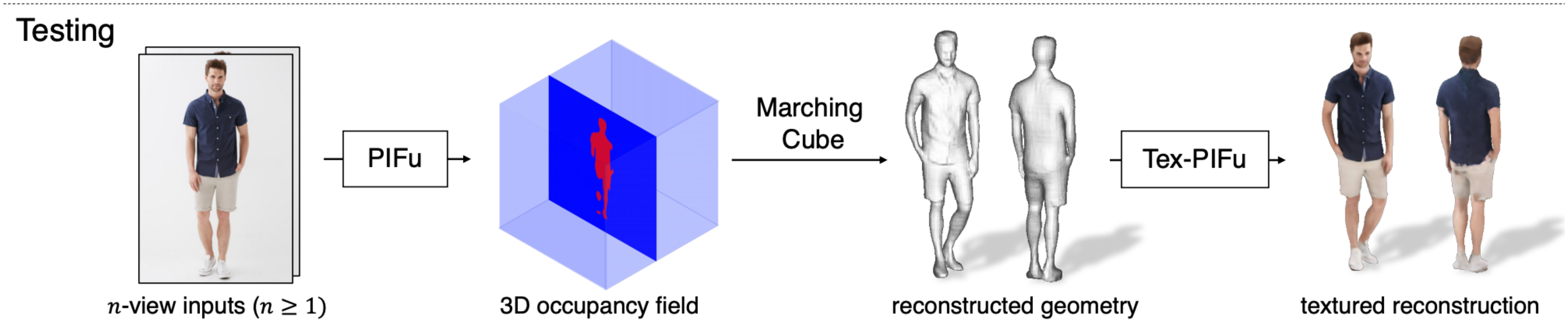
Saito et al. 2019

DeepSDF Extensions: PiFU



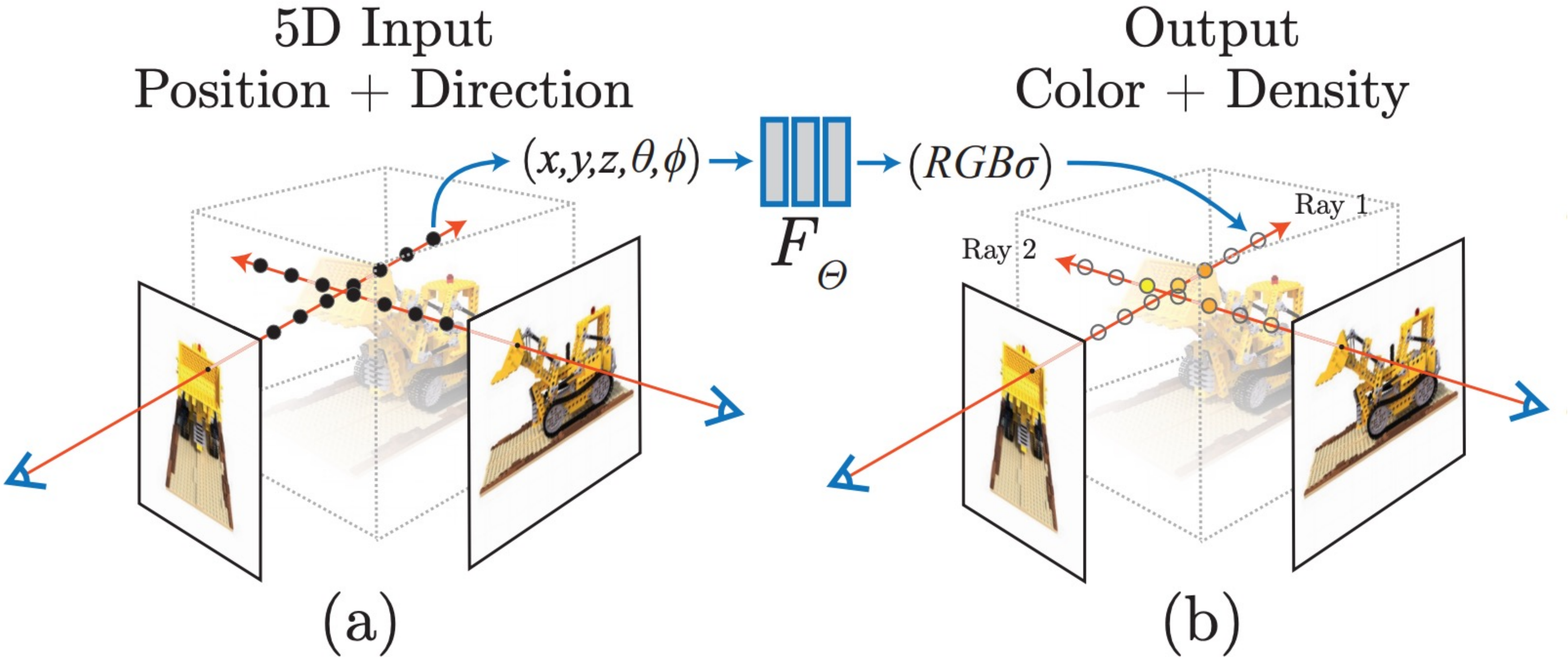
Saito et al. 2019

DeepSDF Extensions: PiFU



Saito et al. 2019

DeepSDF Extensions: NeRF



More in next class

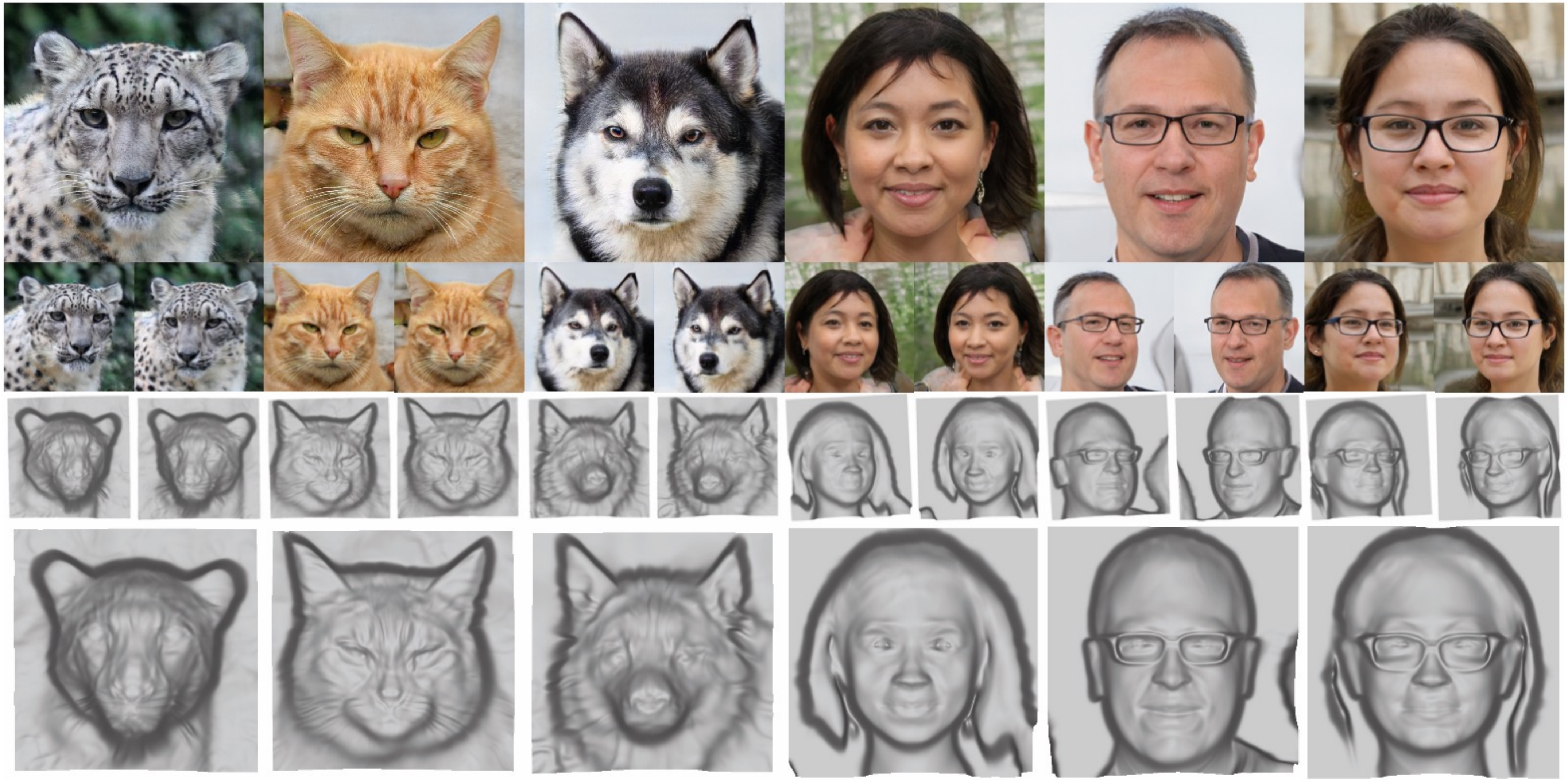
Mildenhall et al. 2020

DeepSDF Extensions: NeRF

Mildenhall et al. 2020

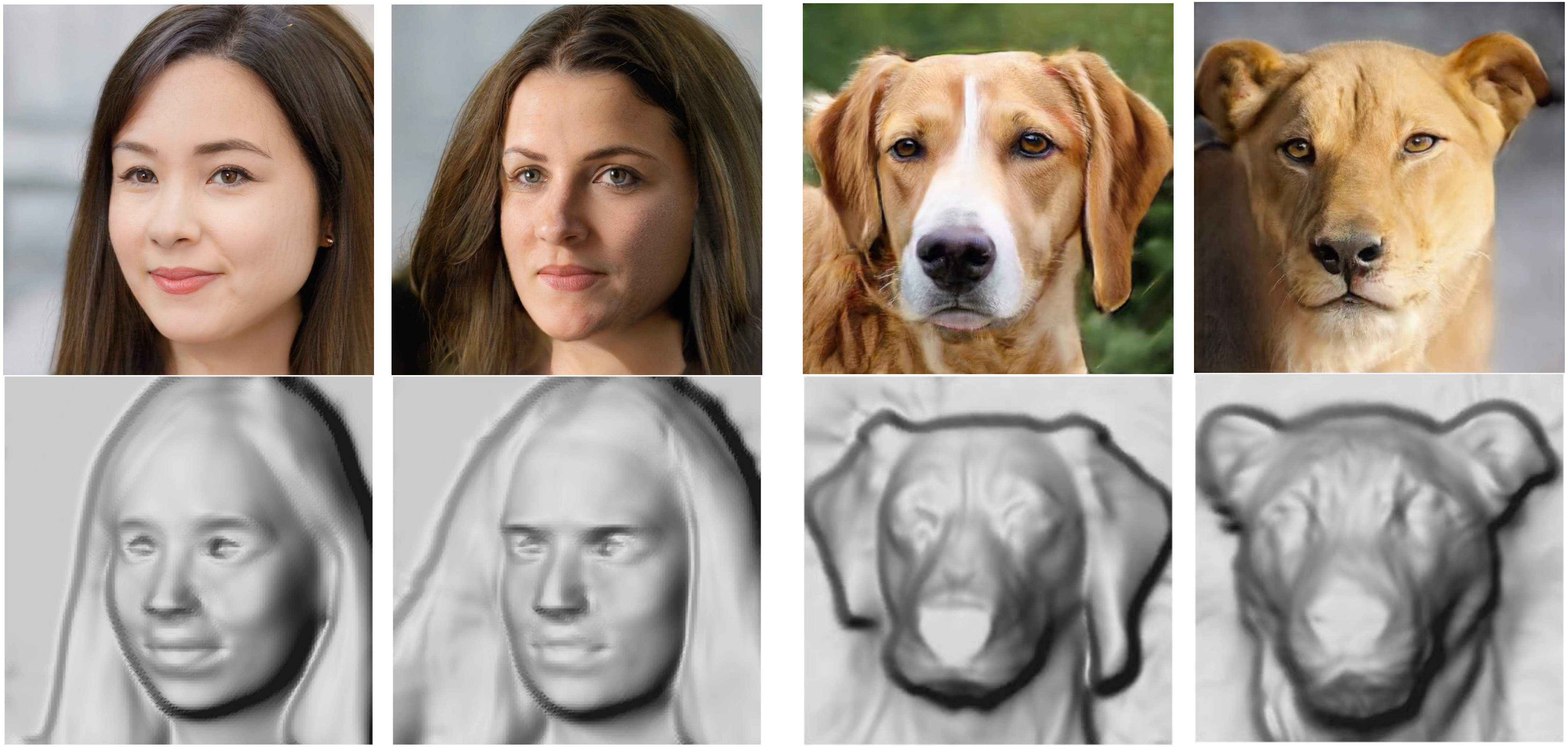
DeepSDF Extension: StyleSDF

- A 3D GAN using DeepSDF + NeRF modeling

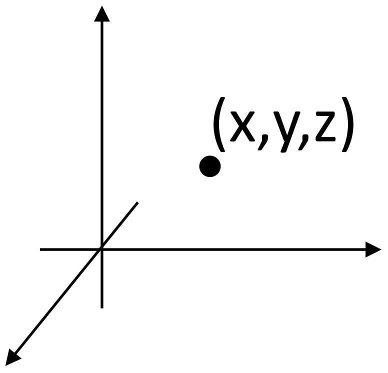


Or-El et al. 2021

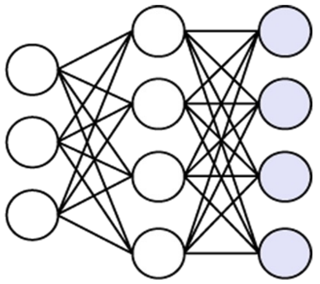
DeepSDF Extension: StyleSDF



What are neural fields?



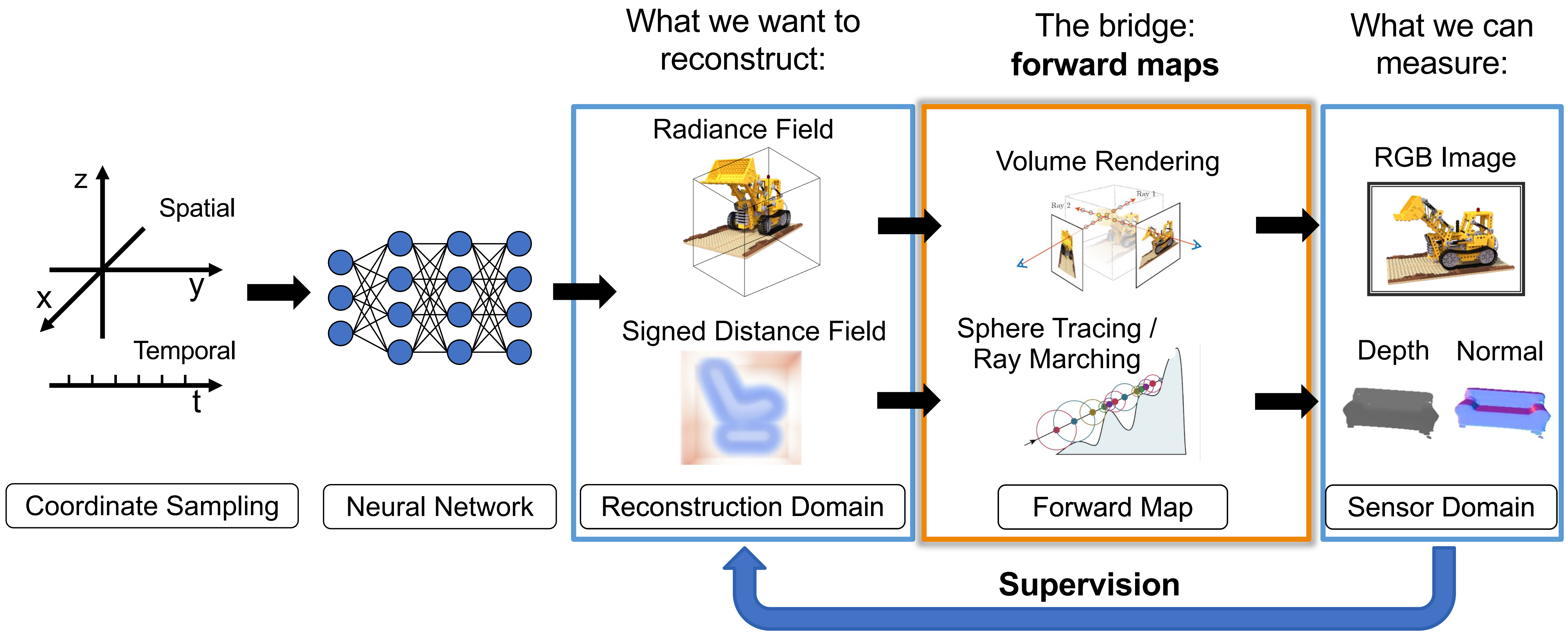
$\Phi: \mathbb{R}^n \rightarrow \mathbb{R}$



Neural Network
(Φ)

Signed Distance Function (SDF)

Neural Field General Framework

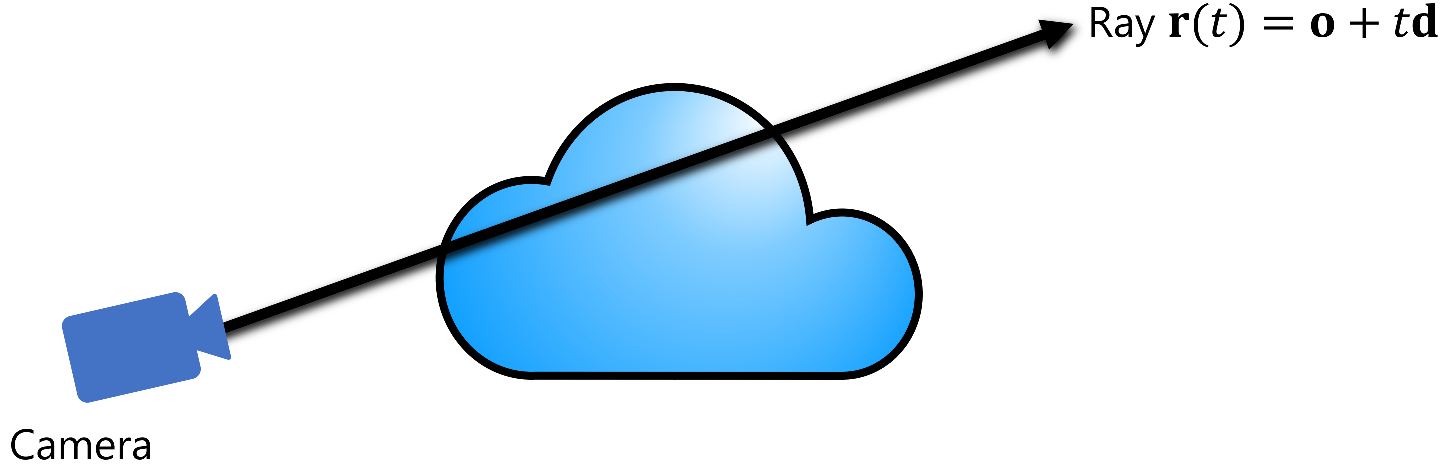


Outline

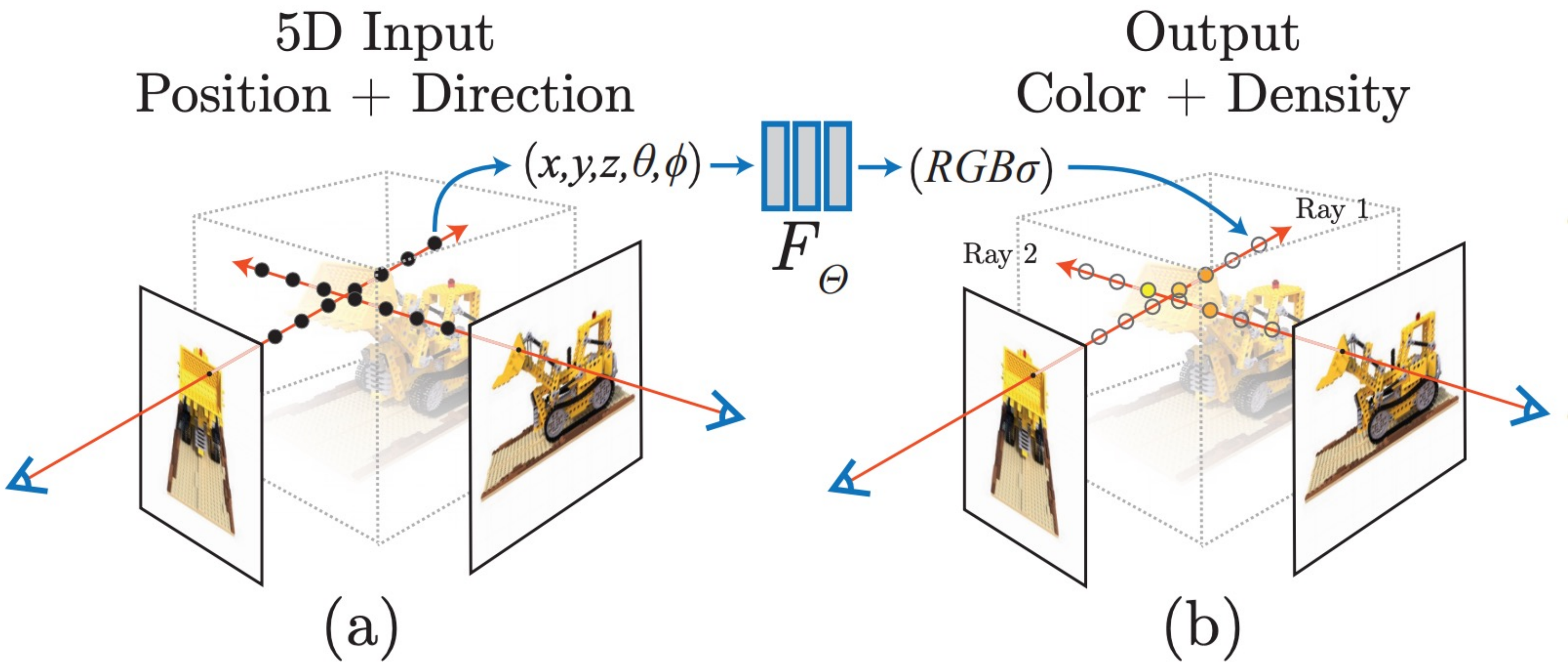
- What is Neural Fields & why it got so much attention?
- The Prelude: Neural Implicit Surfaces
- Introduction to Volume Rendering

What is Volume Rendering?

- Assume a cloud of tiny colored particles in 3D. Each particle has a RGB color and a density.
- Take a pixel on image plane, and shoot a ray from the camera center, through the pixel and into the 'cloud of tiny colored particles'
- What should be the color for that pixel?



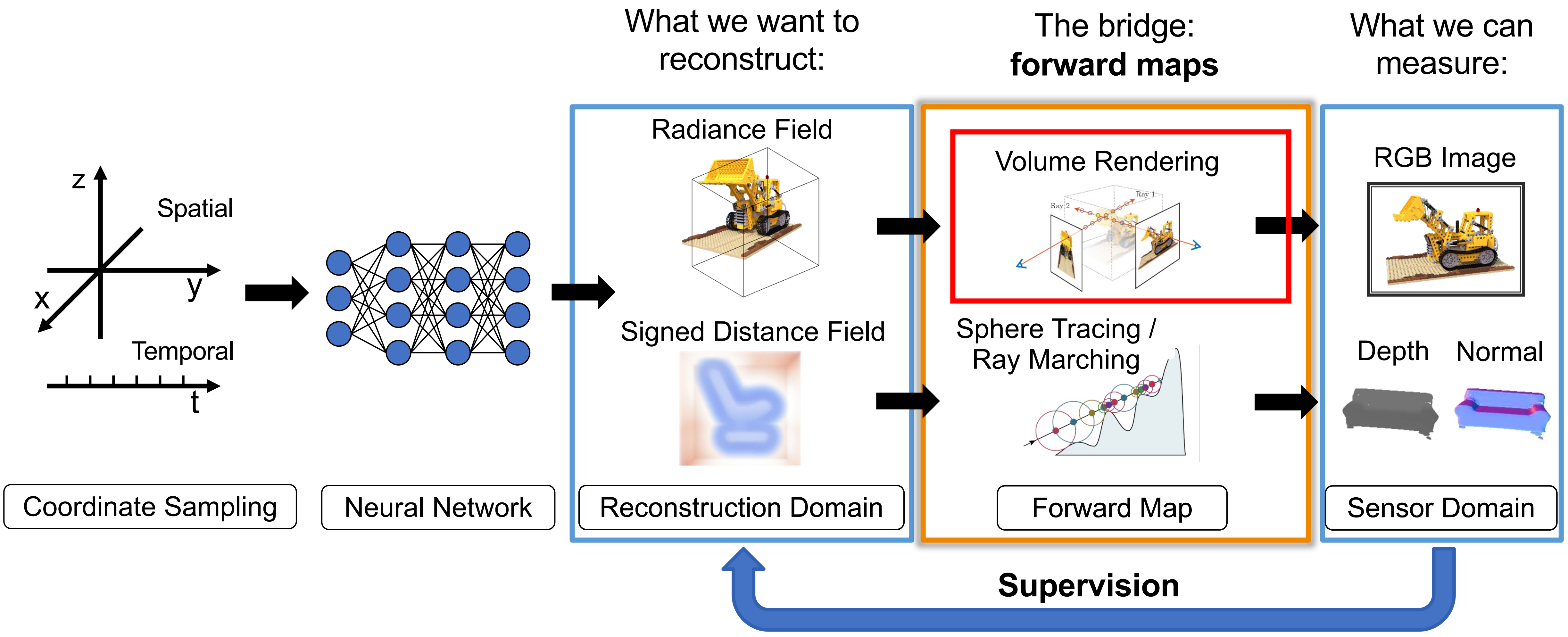
Volume Rendering in NeRF



In NeRF, we model a 3D scene with a ‘cloud of tiny colored particles’.

Neural Field General Framework

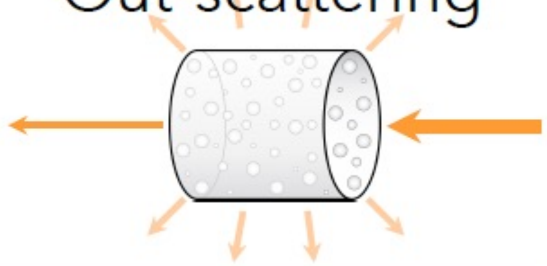
Volume Rendering tells us how to take 'cloud of tiny colored particles' and create an image.



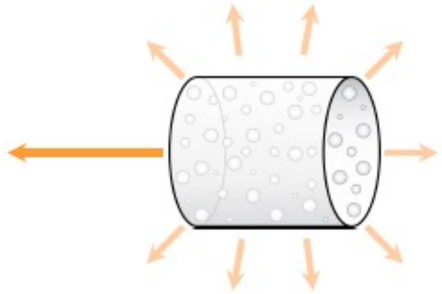
Radiative Transfer Equation

Absorption

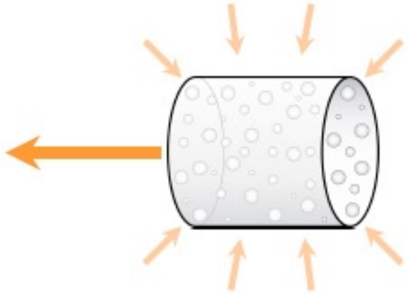
Out-scattering



$$
\begin{aligned}
dL(\mathbf{x},\vec{\omega}) = & \boxed{-\,\sigma_a(\mathbf{x})L(\mathbf{x},\vec{\omega})dz \quad -\,\sigma_s(\mathbf{x})L(\mathbf{x},\vec{\omega})dz} \quad \text{Losses} \\
& \boxed{+\,\sigma_a(\mathbf{x})L_e(\mathbf{x},\vec{\omega})dz \; +\,\sigma_s(\mathbf{x})L_s(\mathbf{x},\vec{\omega})dz} \quad \text{Gains}
\end{aligned}
$$

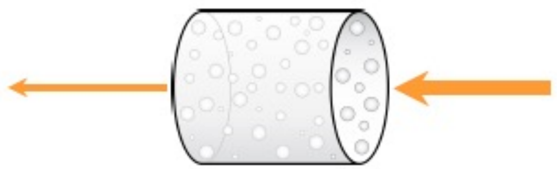


Emission

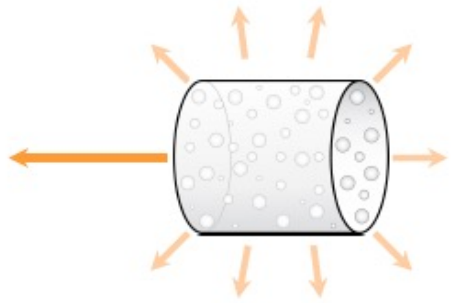


In-scattering

Emission-Absorption Model (Ignoring Scattering)



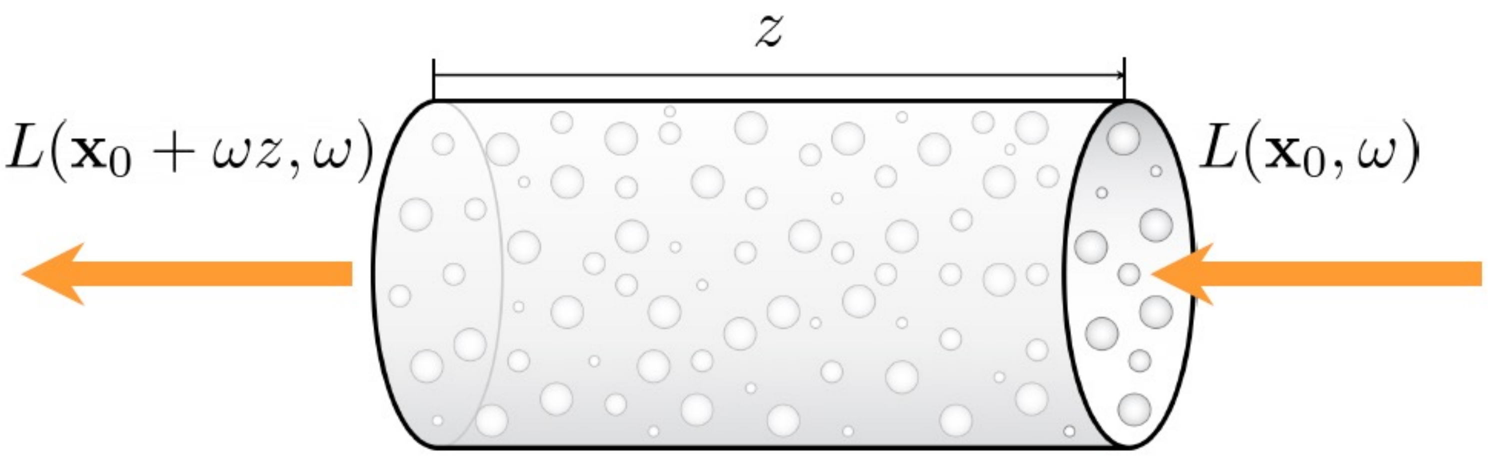
$$dL(\mathbf{x},\vec{\omega}) = -\,\sigma_a(\mathbf{x})L(\mathbf{x},\vec{\omega})dz \quad - \sigma_s(\mathbf{x})L(\mathbf{x},\vec{\omega})dz$$
$$+\,\sigma_a(\mathbf{x})L_e(\mathbf{x},\vec{\omega})dz + \sigma_s(\mathbf{x})L_s(\mathbf{x},\vec{\omega})dz$$



Will drop the subscript moving forward

$$\sigma_a \equiv \sigma$$

Absorption-only Volume Rendering



Q: What if we have a homogenous medium? (uniform coefficient)

$$dL(\mathbf{x}, \omega) = -\sigma L(\mathbf{x}, \omega) dz$$

Can you prove this?

$$L(\mathbf{x}_0 + \omega z, \omega) = e^{-\sigma z} L(\mathbf{x}_0, \omega)$$

What if we have a non-homogenous medium? In non-homogenous medium, coefficient of absorption σ varies with location

$$T(\mathbf{x}, \mathbf{y}) = \underbrace{e^{-\int_{t=0}^{z} \sigma(\mathbf{x}+\omega \mathbf{t}) dt}}_{\text{Transmittance}} L(\mathbf{x}_0, \omega)$$

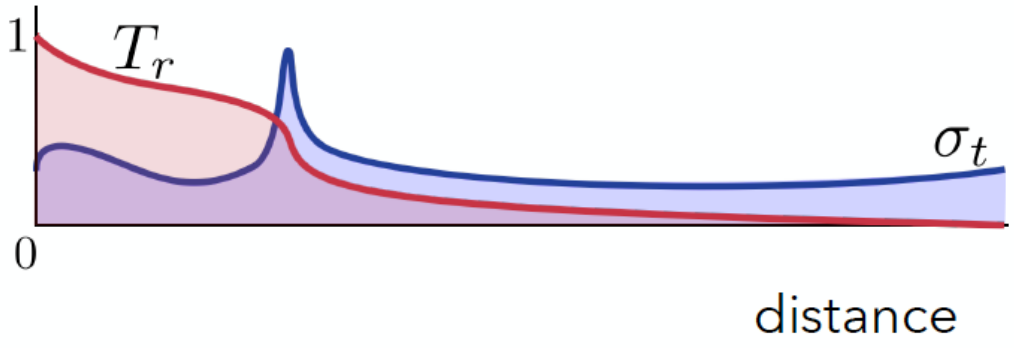
Transmittance

$$T(\mathbf{x},\mathbf{y}) = \underbrace{e^{-\int_{t=0}^{z}\sigma(\mathbf{x}+\omega\mathbf{t})dt}}_{\text{Transmittance}} L(\mathbf{x}_0,\omega)$$

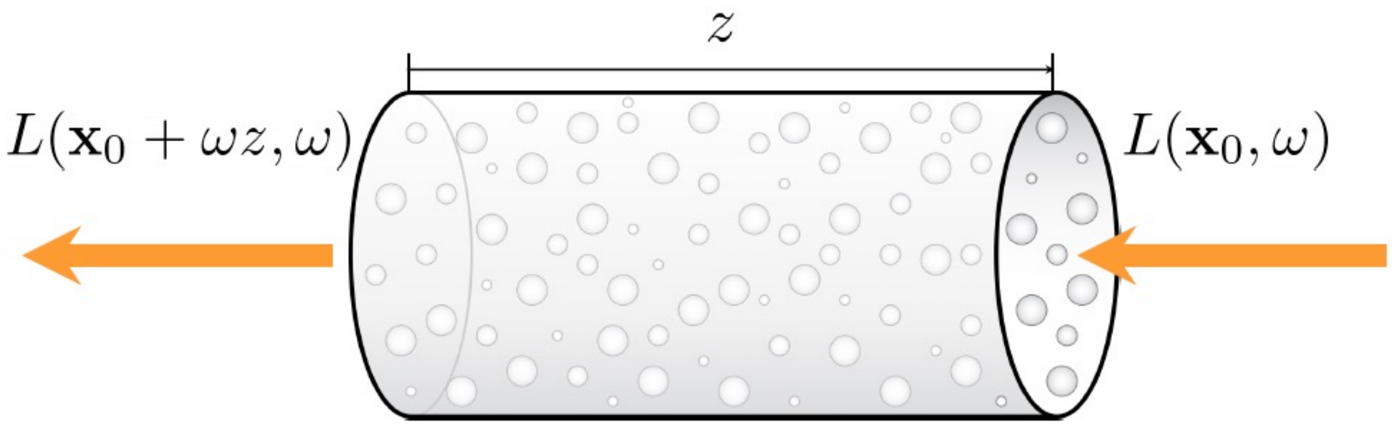
$T(\mathbf{x},\mathbf{y})$ What fraction of radiance at **x** in direction of **y**, reaches **y** ?
(along a straight line under absorption-only model)

Homogenous Medium: $e^{-\sigma\|\mathbf{x}-\mathbf{y}\|}$

Non-Homogenous Medium: $e^{-\int_{t=0}^{\|\mathbf{x}-\mathbf{y}\|}\sigma(\mathbf{x}+\omega\mathbf{t})}$

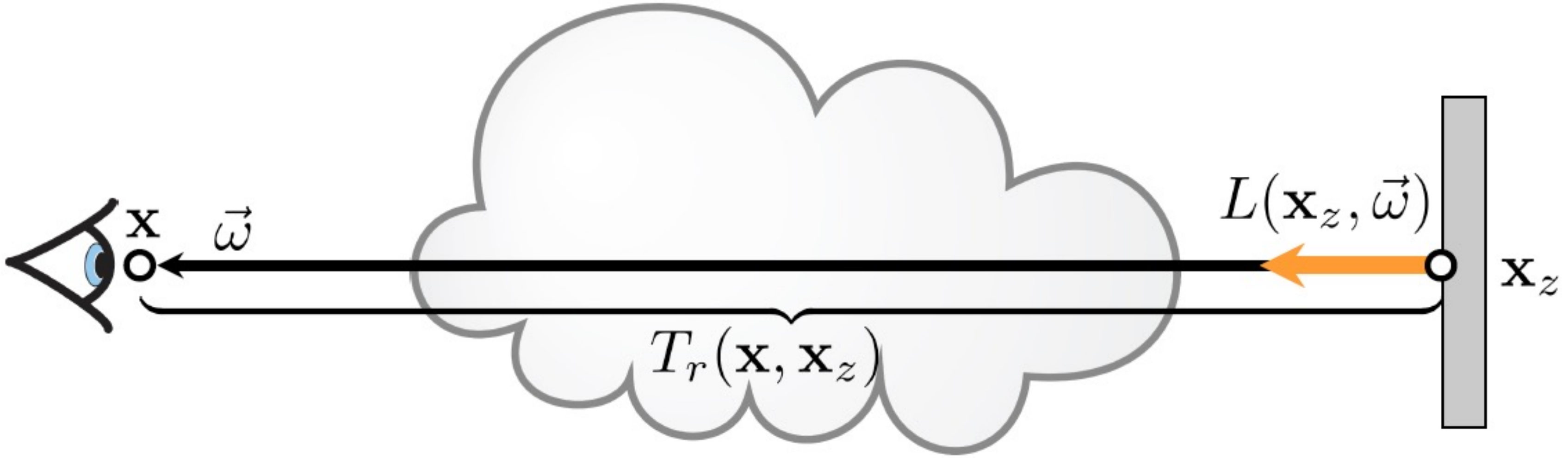


Absorption-only Volume Rendering



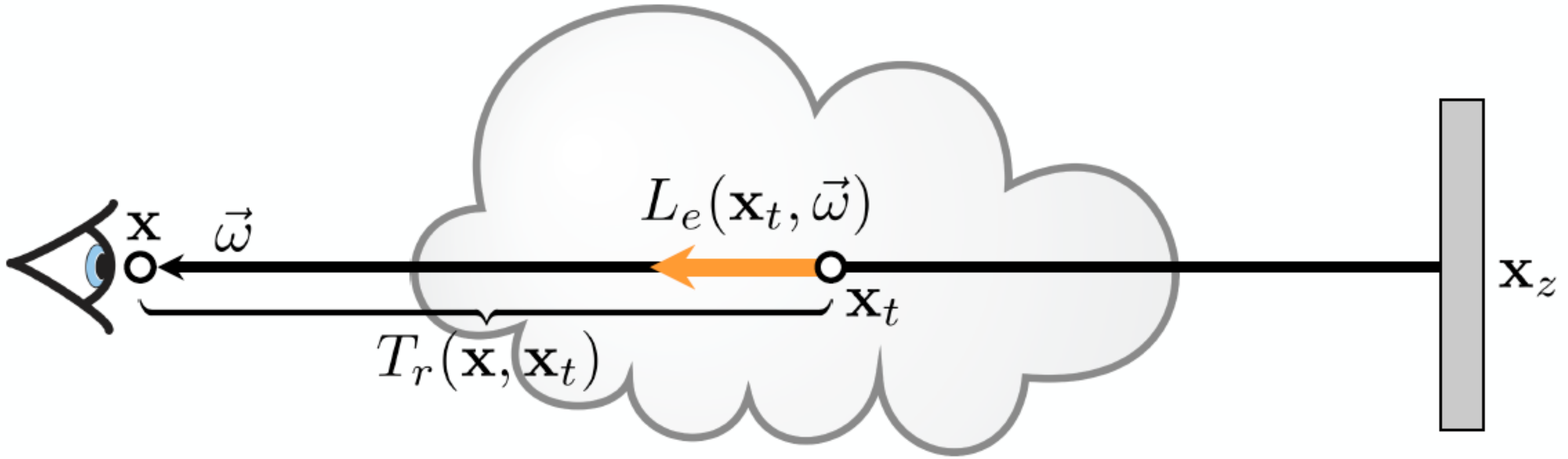
$$L(\mathbf{x}_0 + \omega z, \omega) = T(\mathbf{x}_0, \mathbf{x}_0 + \omega z) L(\mathbf{x}_0, \omega)$$

Absorption-only Volume Rendering



$$L(\mathbf{x}, \omega) = T(\mathbf{x}, \mathbf{x}_z) \underbrace{L(\mathbf{x}_z, \omega)}_{\text{Radiance from 'outside' the medium}}$$

Emission-Absorption Volume Rendering



$$L(\mathbf{x}, \omega) = T(\mathbf{x}, \mathbf{x}_z) L(\mathbf{x}_z, \omega) + \int_0^z \underbrace{T(\mathbf{x}, \mathbf{x}_t)\sigma(\mathbf{x}_t) L_e(\mathbf{x}_t, \omega) dt}_{\text{Accumulated Emitted Radiance from inside}}$$

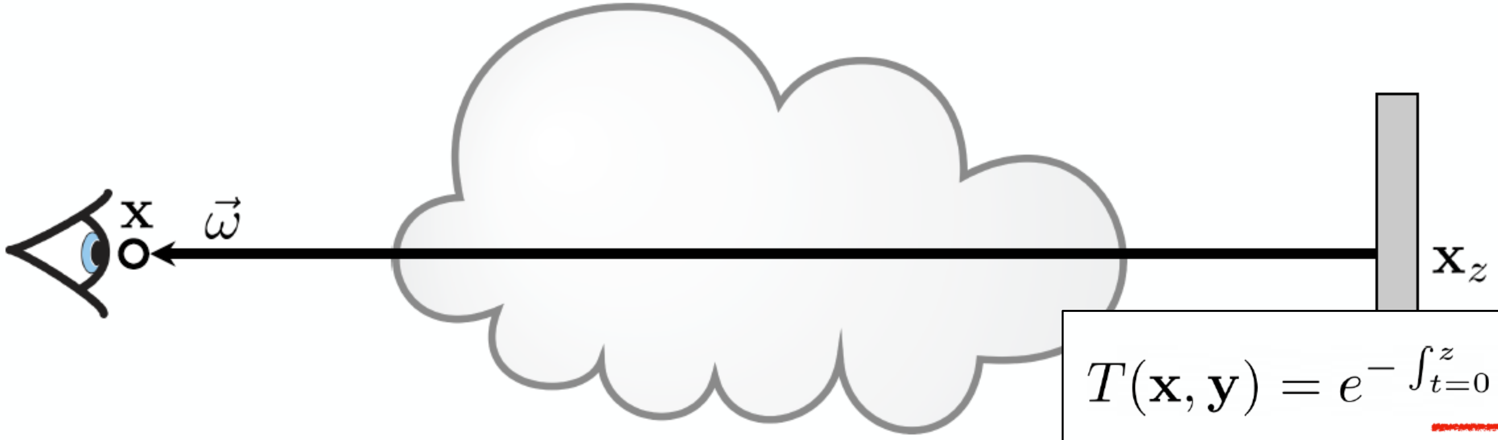
Emission-Absorption Volume Rendering
Special Case: Homogenous non-emitting medium

$$T(\mathbf{x}, \mathbf{y}) = e^{-\int_{t=0}^{z} \sigma(\mathbf{x}+\omega \mathbf{t})dt} L(\mathbf{x}_0, \omega)$$

$$L(\mathbf{x}, \omega) = T(\mathbf{x}, \mathbf{x}_z) L(\mathbf{x}_z, \omega)$$

$$T(\mathbf{x}, \mathbf{x}_z) = e^{-\sigma \Delta}$$

Emission-Absorption Volume Rendering
Special Case: Homogenous emitting medium



$$T(\mathbf{x},\mathbf{y}) = \underbrace{e^{-\int_{t=0}^{z}\sigma(\mathbf{x}+\omega\mathbf{t})dt}}_{\text{Transmittance}} L(\mathbf{x}_0,\omega)$$

$$L(\mathbf{x},\omega) = e^{-\sigma\Delta}L(\mathbf{x}_z,\omega) + \int_0^z T(\mathbf{x},\mathbf{x}_t)\sigma(\mathbf{x}_t)L_e(\mathbf{x}_t,\omega)dt$$

$$\boxed{L_e(\mathbf{x},\omega) \equiv C}$$

$$\Rightarrow$$

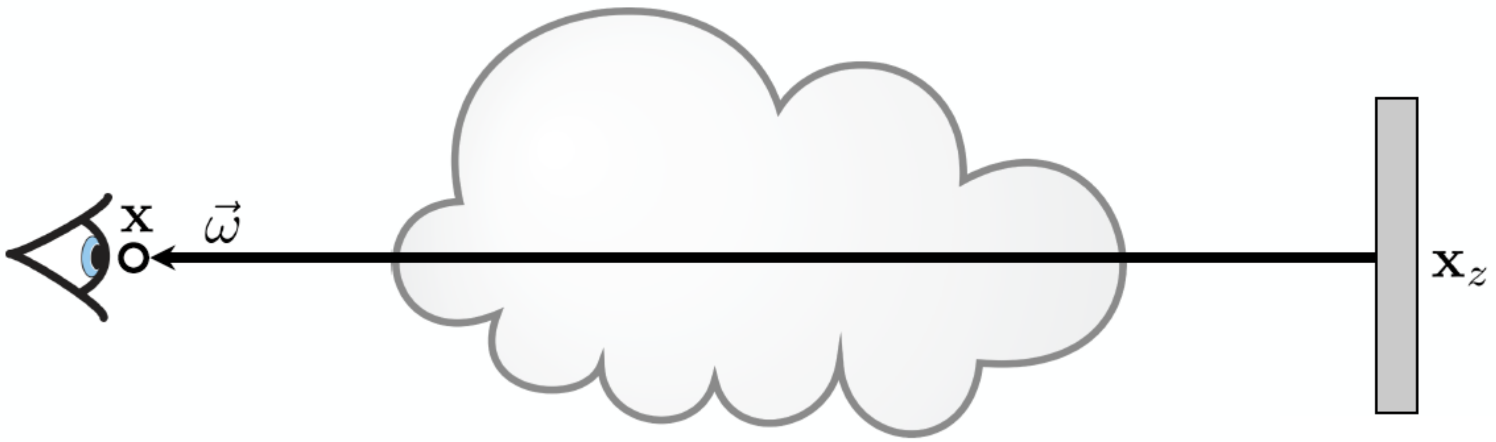
$$L(\mathbf{x},\omega) = e^{-\sigma\Delta}L(\mathbf{x}_z,\omega) + \int_{\delta=0}^{\Delta} e^{-\sigma\delta}\sigma C\, d\delta$$

$$\Downarrow$$

$$L(\mathbf{x},\omega) = e^{-\sigma\Delta}L(\mathbf{x}_z,\omega) + (1-e^{-\sigma\Delta})C$$

$$\boxed{\int \lambda e^{-\lambda x}dx = -e^{\lambda x} + c}$$

Emission-Absorption Volume Rendering
Special Case: Homogenous emitting (only) medium



$$L(\mathbf{x}, \omega) = \int_0^z T(\mathbf{x}, \mathbf{x}_t)\sigma(\mathbf{x}_t)L_e(\mathbf{x}_t, \omega)dt$$

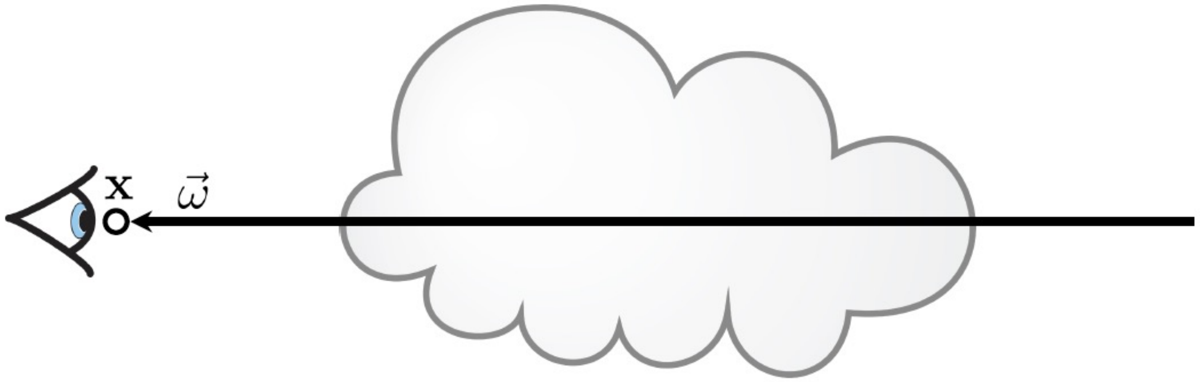
Let's ignore light from outside medium (for now)

$$L(\mathbf{x}, \omega) = e^{-\sigma\Delta}L(\mathbf{x}_z, \omega)$$

$$+ \int_0^z T(\mathbf{x}, \mathbf{x}_t)\sigma(\mathbf{x}_t)L_e(\mathbf{x}_t, \omega)dt$$

$$\Downarrow$$

$$L(\mathbf{x}, \omega) = e^{-\sigma\Delta}L(\mathbf{x}_z, \omega)$$
$$+(1 - e^{-\sigma\Delta})C$$



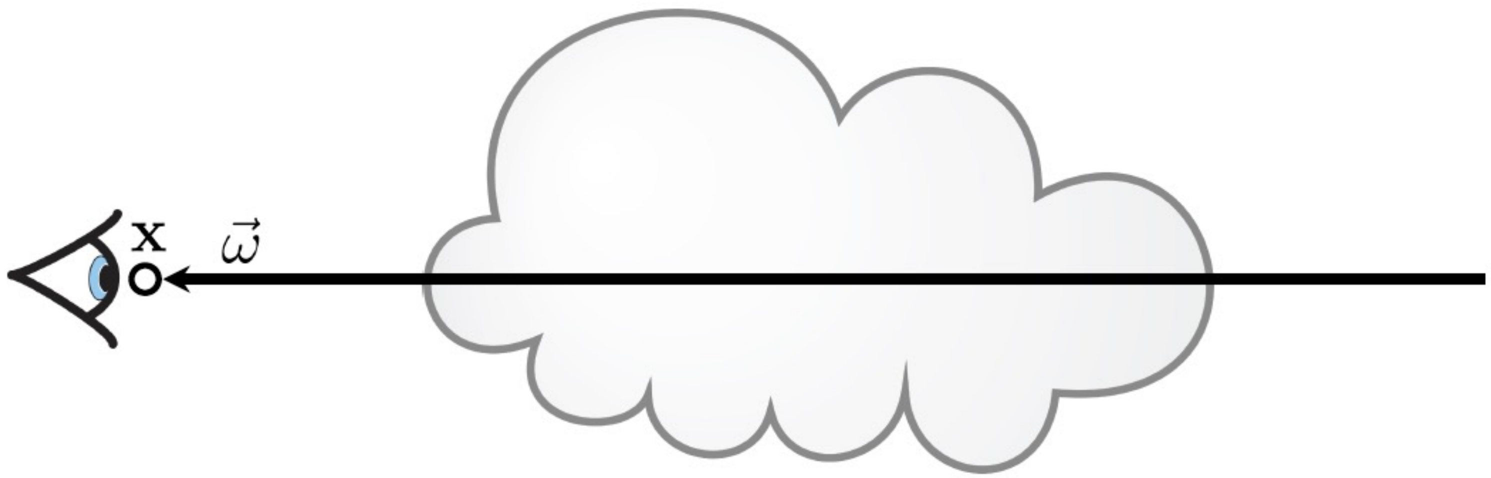
$$L(\mathbf{x}, \omega) = (1 - e^{-\sigma\Delta})C$$

Emission from a homogenous segment of length Δ

Emission-Absorption Volume Rendering
Special Case: Homogenous emitting (only) medium

Volume Rendering Model to be used in NeRF

Assumption: Ignore light from outside medium.

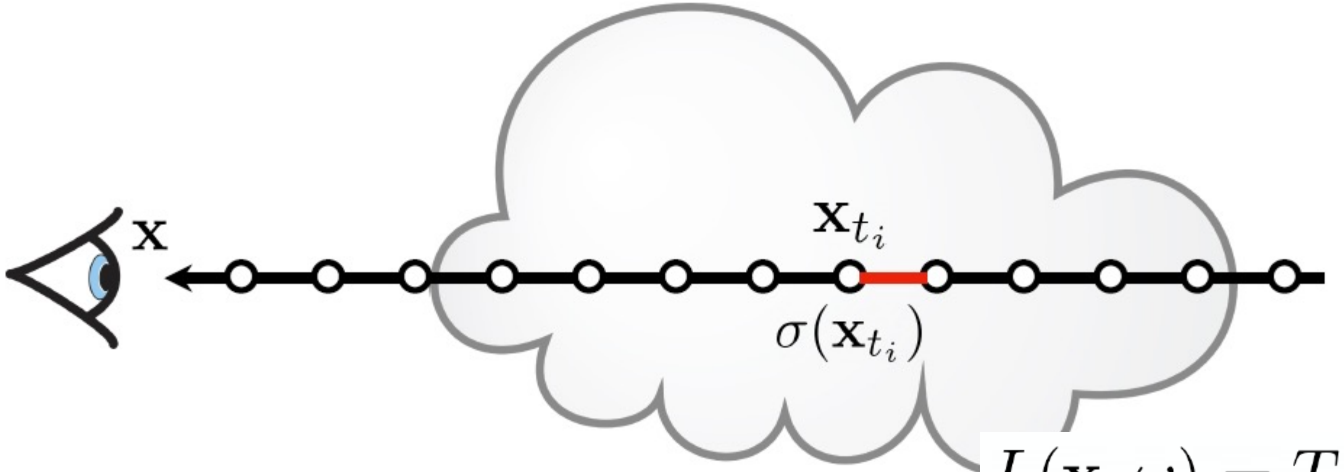


For NeRF this means: the object you are trying to render only emits light, There is no lighting being emitted by the background.

$$L(\mathbf{x}, \omega) = (1 - e^{-\sigma\Delta})C$$

Emission from a homogenous segment of length Δ

Computational Volume Rendering: Ray Marching



$$L(\mathbf{x}, \omega) = T(\mathbf{x}, \mathbf{x}_z) L(\mathbf{x}_z, \omega)$$

$$L(\mathbf{x}, \omega) = \sum_{i=1}^{N} \text{(contribution from i}^{\text{th}}\text{ segment)} \Rightarrow L(\mathbf{x}, \omega) = \sum_{i=1}^{N} T(\mathbf{x}, \mathbf{x}_{t_i}) \times \text{(emission from i}^{\text{th}}\text{ segment)}$$

Approximate with a discrete sum

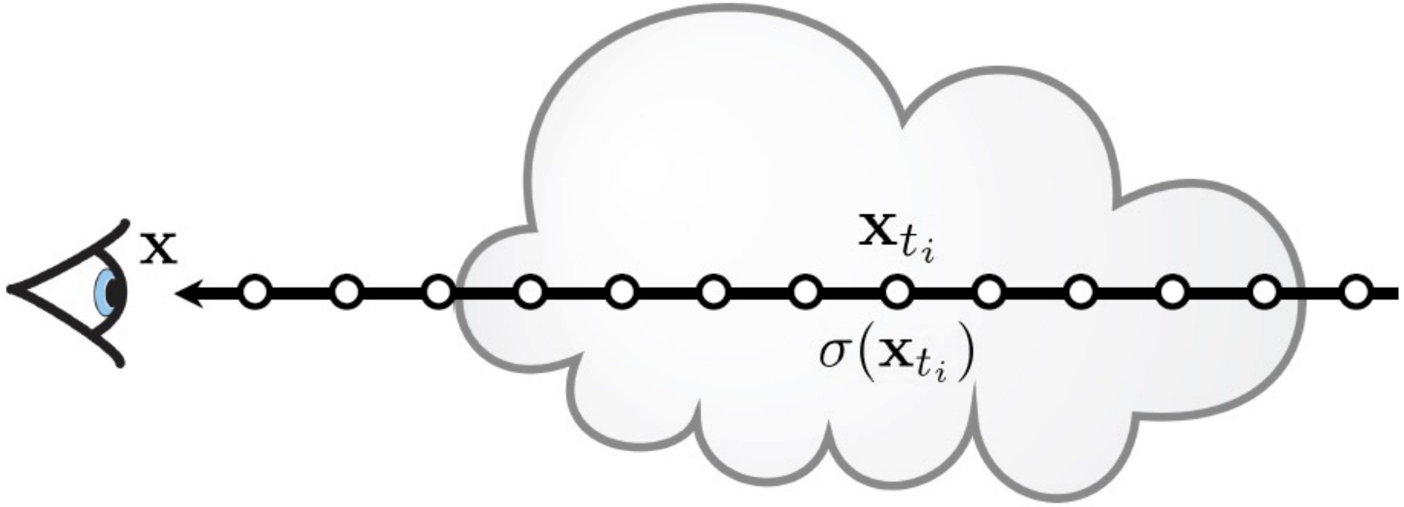
$\mathbf{x}_{t_i}$: i-th sample along ray at depth t_i

Δt : distance between successive samples

$$L(\mathbf{x}, \omega) = (1 - e^{-\sigma\Delta})C$$

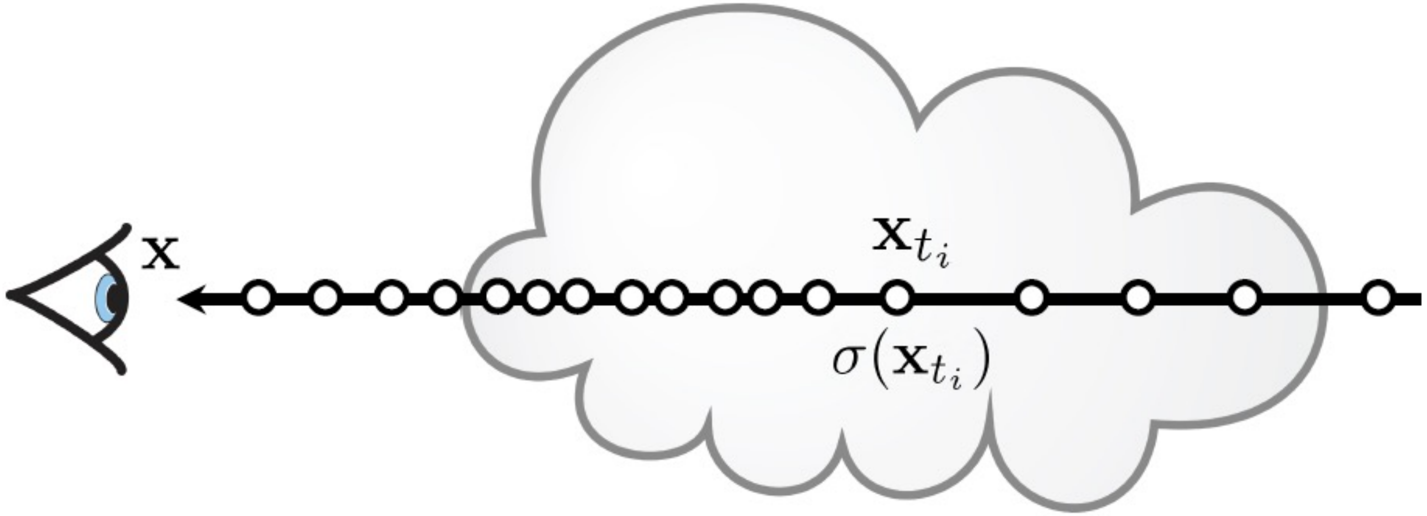
$$L(\mathbf{x}, \omega) = \sum_{i=1}^{N} T(\mathbf{x}, \mathbf{x}_{t_i}) (1 - e^{\sigma_{t_i}\Delta t}) L_e(\mathbf{x}_{t_i}, \omega)$$

Computational Volume Rendering: Ray Marching



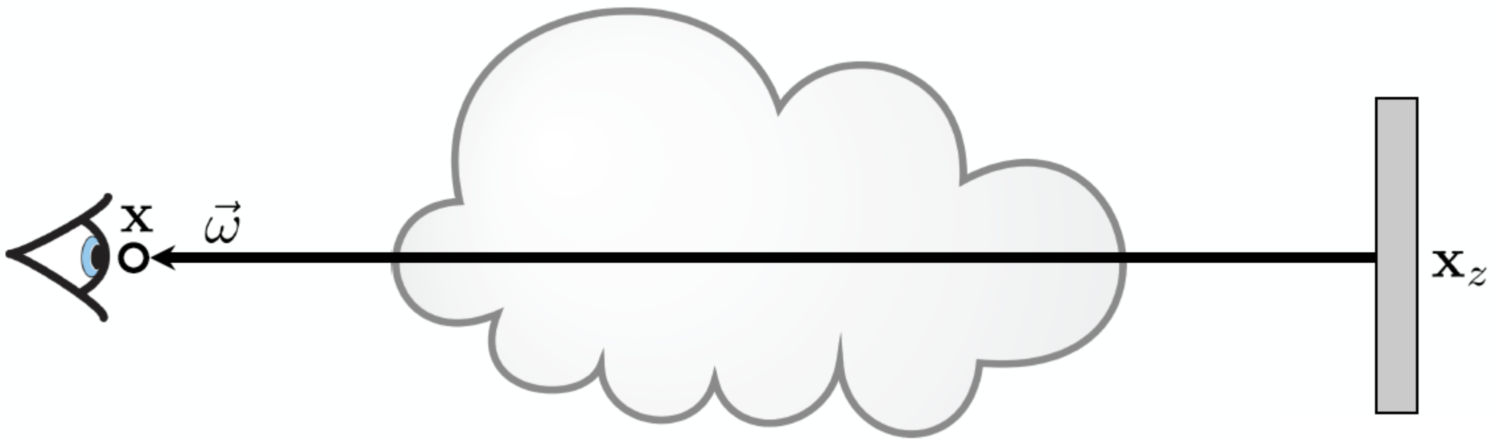
1. Draw uniform samples along a ray (N segments, or N+1 points)
2. Compute transmittance between camera and each sample
3. Aggregate contributions across segments to get overall radiance (color)

Computational Volume Rendering: Ray Marching



1. Draw **non-uniform** samples along a ray
2. Compute transmittance between camera and each sample
3. Aggregate contributions across segments to get overall radiance (color)

Emission-Absorption Volume Rendering
Special Case: Homogenous emitting (only) medium

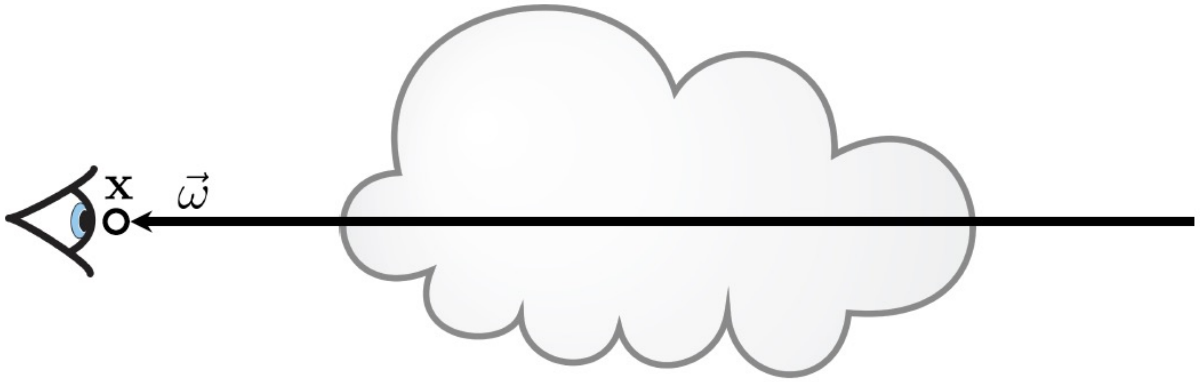


$$L(\mathbf{x},\omega) = \int_0^z T(\mathbf{x},\mathbf{x}_t)\sigma(\mathbf{x}_t)L_e(\mathbf{x}_t,\omega)dt$$

Let's ignore light from outside medium (for now)

$$L(\mathbf{x},\omega) = e^{-\sigma\Delta}L(\mathbf{x}_z,\omega) + \int_0^z T(\mathbf{x},\mathbf{x}_t)\sigma(\mathbf{x}_t)L_e(\mathbf{x}_t,\omega)dt$$

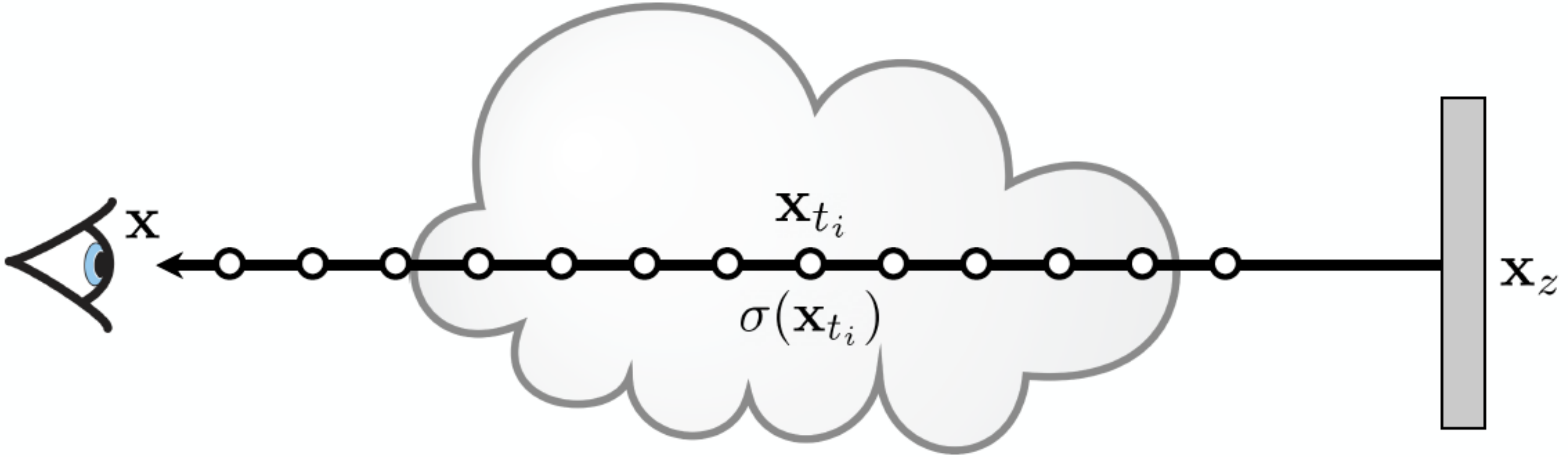
$$L(\mathbf{x},\omega) = e^{-\sigma\Delta}L(\mathbf{x}_z,\omega) + (1-e^{-\sigma\Delta})C$$



$$L(\mathbf{x},\omega) = (1-e^{-\sigma\Delta})C$$

Emission from a homogenous segment of length Δ

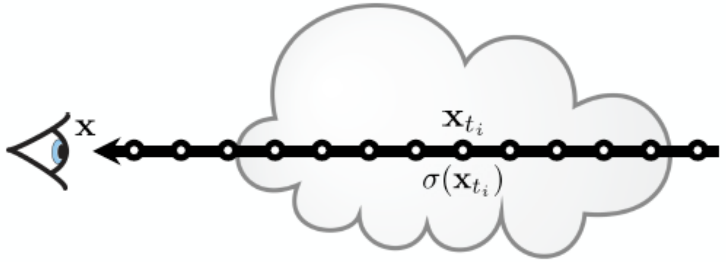
Enabling Background Radiance



$$L(\mathbf{x}, \omega) = \sum_{i=1}^{N} T(\mathbf{x}, \mathbf{x}_{t_i})(1 - e^{-\sigma_{t_i} \Delta t}) L_e(\mathbf{x}_{t_i}, \omega)$$

$$T(\mathbf{x}, \mathbf{x}_{t_i}) = T(\mathbf{x}, \mathbf{x}_{t_{i-1}}) e^{-\sigma_{t_{i-1}} \Delta t}$$

Computational Volume Rendering: A summary



$$\sigma_{t_i} \equiv \sigma(\mathbf{x}_{t_i})$$

$$L_e(\mathbf{x}_{t_i}, \omega) \longrightarrow L(\mathbf{x}, \omega)$$

$$L(\mathbf{x}, \omega) = \sum_{i=1}^{N} T(\mathbf{x}, \mathbf{x}_{t_i}) \underline{(1 - e^{\sigma_{t_i} \Delta t}) L_e(\mathbf{x}_{t_i}, \omega)}$$

If we can compute:

a) (per-point) density

b) (per-point, direction) emitted light,

we can render **any** ray through the medium

Equivalently, we can render an image from any camera viewpoint (using H*W rays)

Note: **Differentiable** process w.r.t. the **density**, **emitted light**

and also camera parameters if density, emission are differentiable functions of position, direction

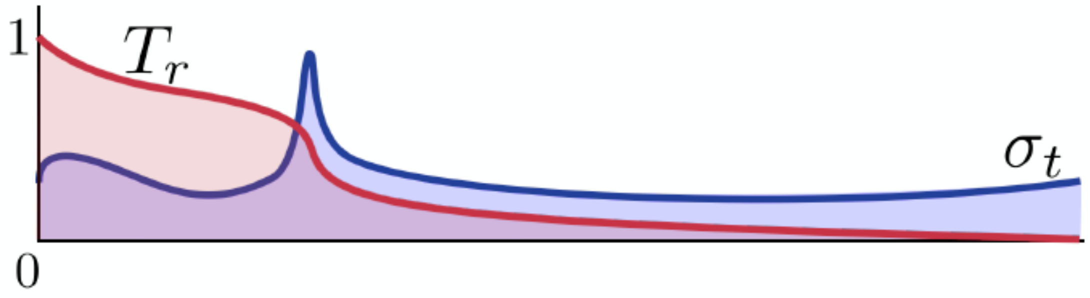
Volume Rendering in NeRF

$$L(\mathbf{x}, \omega) = \sum_{i=1}^{N} T(\mathbf{x}, \mathbf{x}_{t_i}) \underline{(1 - e^{\sigma_{t_i} \Delta t}) L_e(\mathbf{x}_{t_i}, \omega)}$$

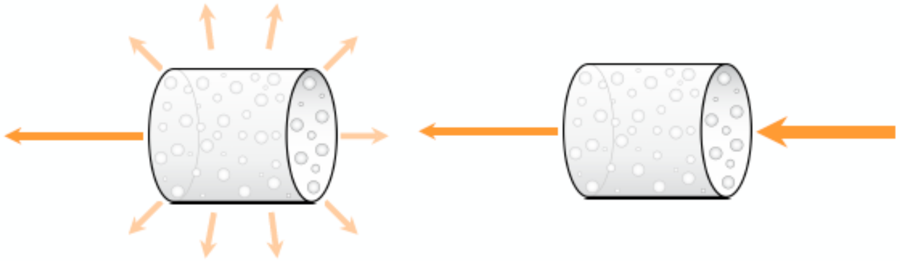
$L_e(.)$ = RGB color of cloud of tiny particles.

σ = density of tiny colored particles

Summary

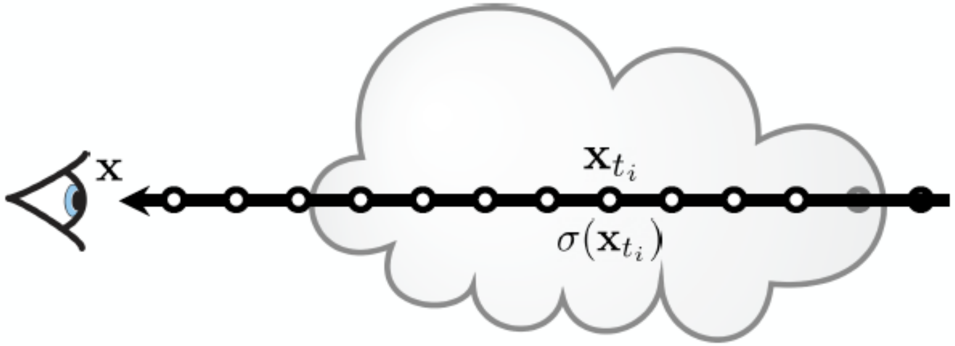


Transmittance



$$dL(\mathbf{x},\vec{\omega}) = -\,\sigma_a(\mathbf{x})L(\mathbf{x},\vec{\omega})dz \\ +\,\sigma_a(\mathbf{x})L_e(\mathbf{x},\vec{\omega})dz$$

Emission-Absorption Model



$$L(\mathbf{x},\omega) = \sum_{i=1}^{N} T(\mathbf{x},\mathbf{x}_{t_i})(1 - e^{\sigma_{t_i}\Delta t})L_e(\mathbf{x}_{t_i},\omega)$$

$$T(\mathbf{x},\mathbf{x}_{t_i}) = T(\mathbf{x},\mathbf{x}_{t_{i-1}})e^{-(\sigma_{t_i}\Delta t)}$$

A computational algorithm

Slide Credits

- "Neural Fields in Computer Vision", CVRP 2022 Tutorial.
- Shubham Tulsiani, "Learning for 3D Vision", Spring 2022, CMU
- Leo Guibas, JJ Park, "Neural Models for 3D geometry", Spring 2022, Stanford.