Neural Radiance Fields 2

CS194-26/294-26: Intro to Computer Vision and Computational Photography
Angjoo Kanazawa
UC Berkeley Fall 2022

Lots of content from ECCV 2022 Tutorial on Neural Volumetric Rendering for Computer Vision

Project 3 Winner!!

Original keyframe | My face morphed into keyframe shape

Joshua Chen

https://inst.eecs.berkeley.edu/~cs194-26/fa22/upload/files/proj3/cs194-26-adm/

Project 4 Highlight

Shinwoo Choi

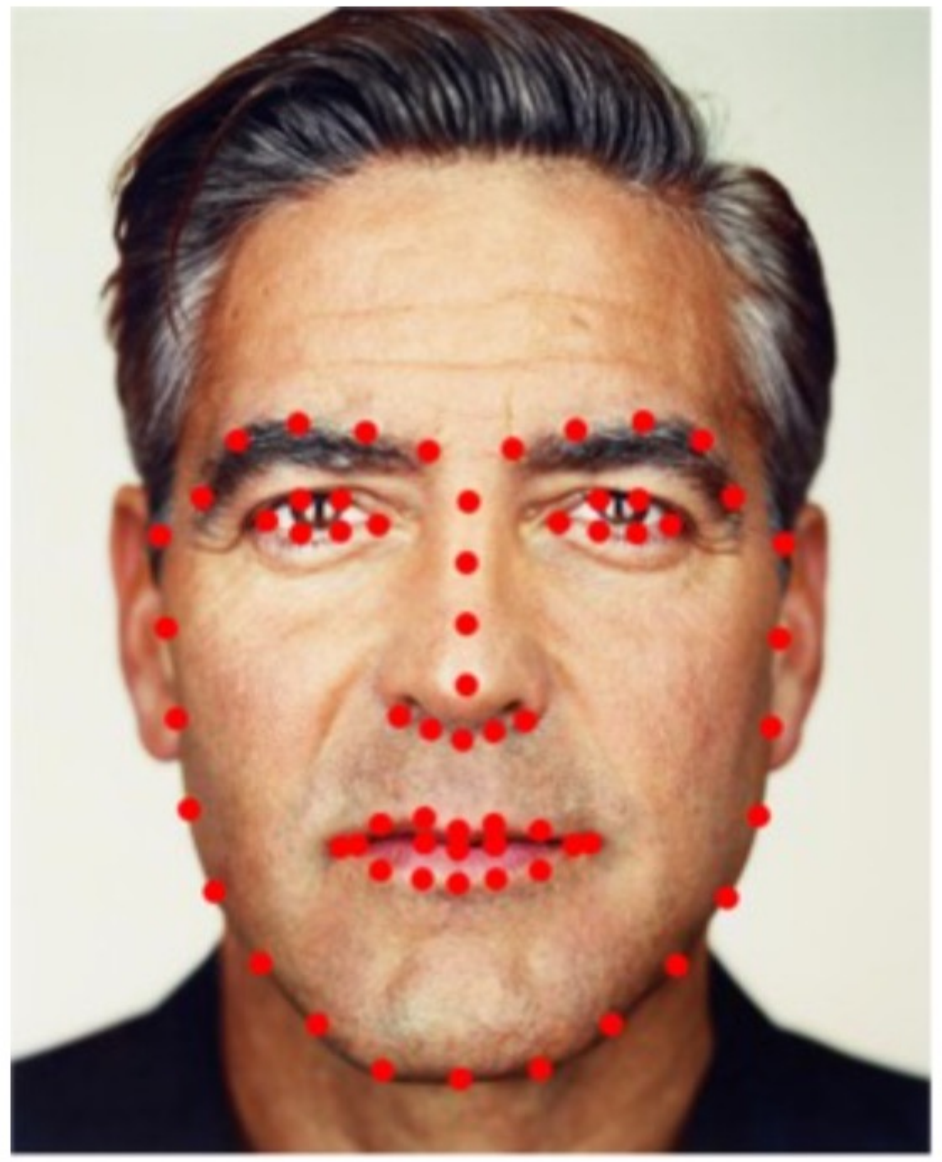
https://inst.eecs.berkeley.edu/~cs194-26/fa22/upload/files/proj4B/cs194-26-afm/

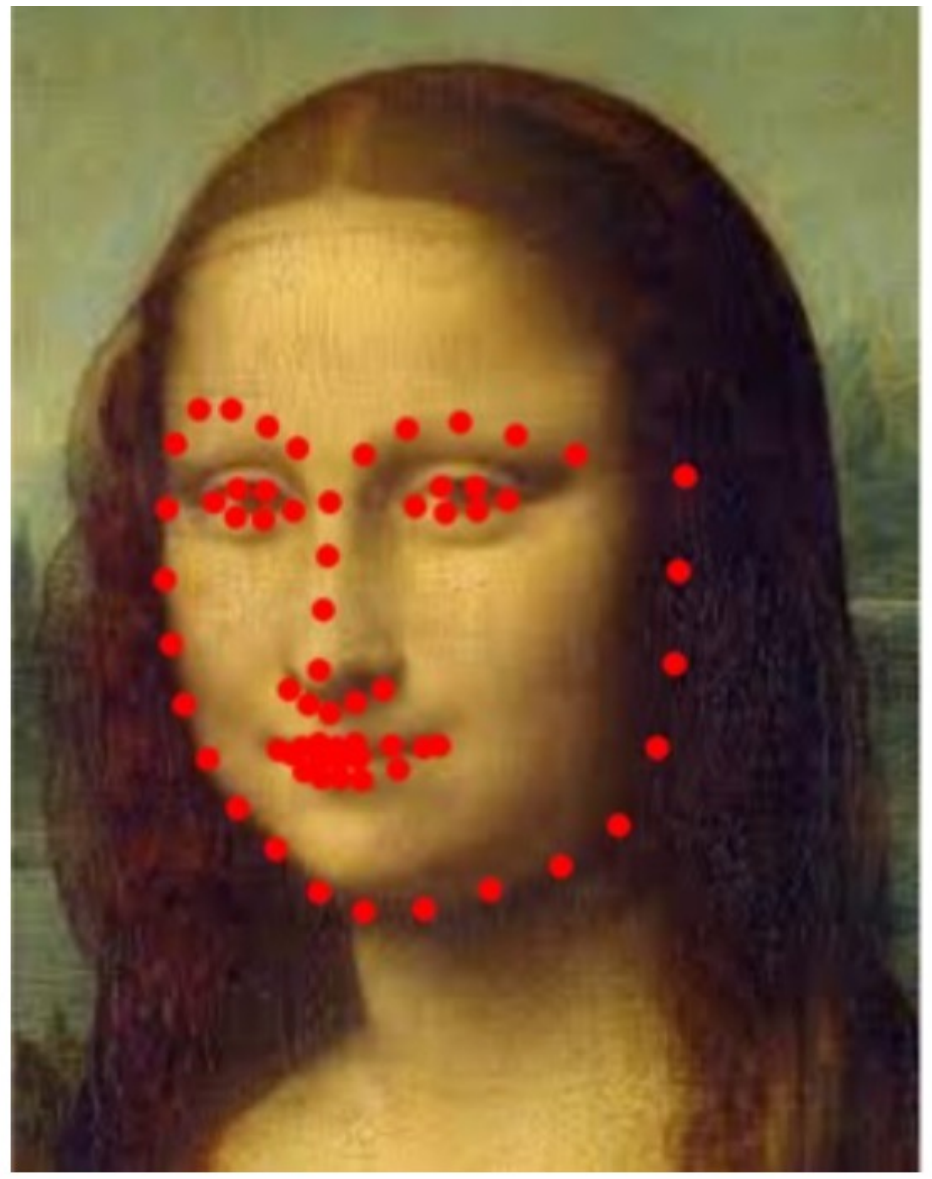
Class vote is happening now, do vote!

Project 5 Winner!!

Jules Dedieu

#	△	Team	Members	Score	Entries	Last	Code
1	—	Jules		5.06767	7	11d	
2	—	tna		5.47330	6	15d	
3	—	Val R		5.59510	7	14d	





Where we are

1. Birds Eye View & Background
2. **Volumetric Rendering Function**
3. Encoding and Representing 3D Volumes
4. Signal Processing Considerations
5. Challenges & Pointers

Simplify

Absorption

http://commons.wikimedia.org

Scattering

Emission

http://wikipedia.org

Slide credit: Novak et al 2018, *Monte Carlo methods for physically based volume rendering*

Summary: volume rendering integral estimate

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

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

weights

colors

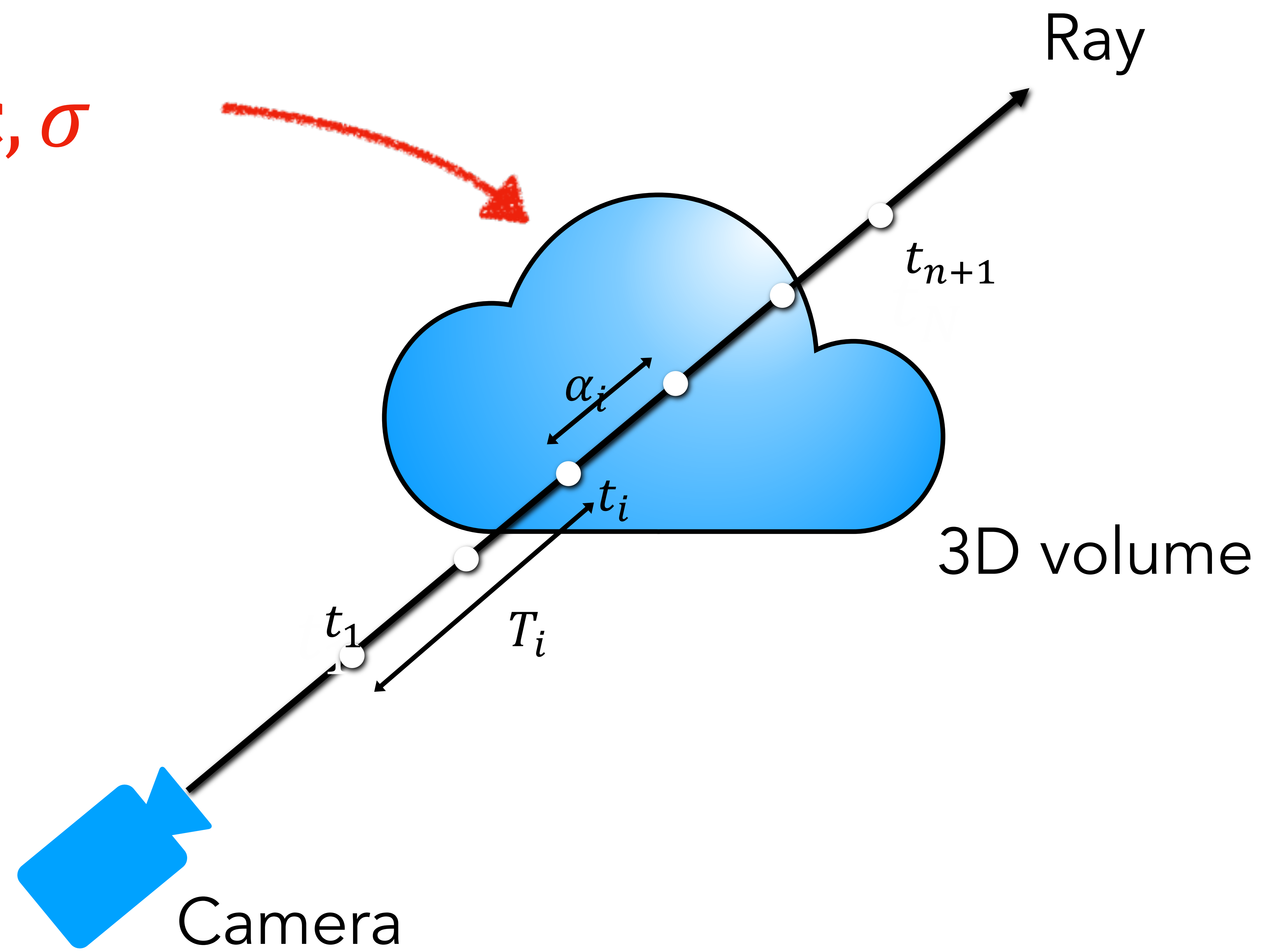
differentiable w.r.t. c, σ

How much light is blocked earlier along ray:

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

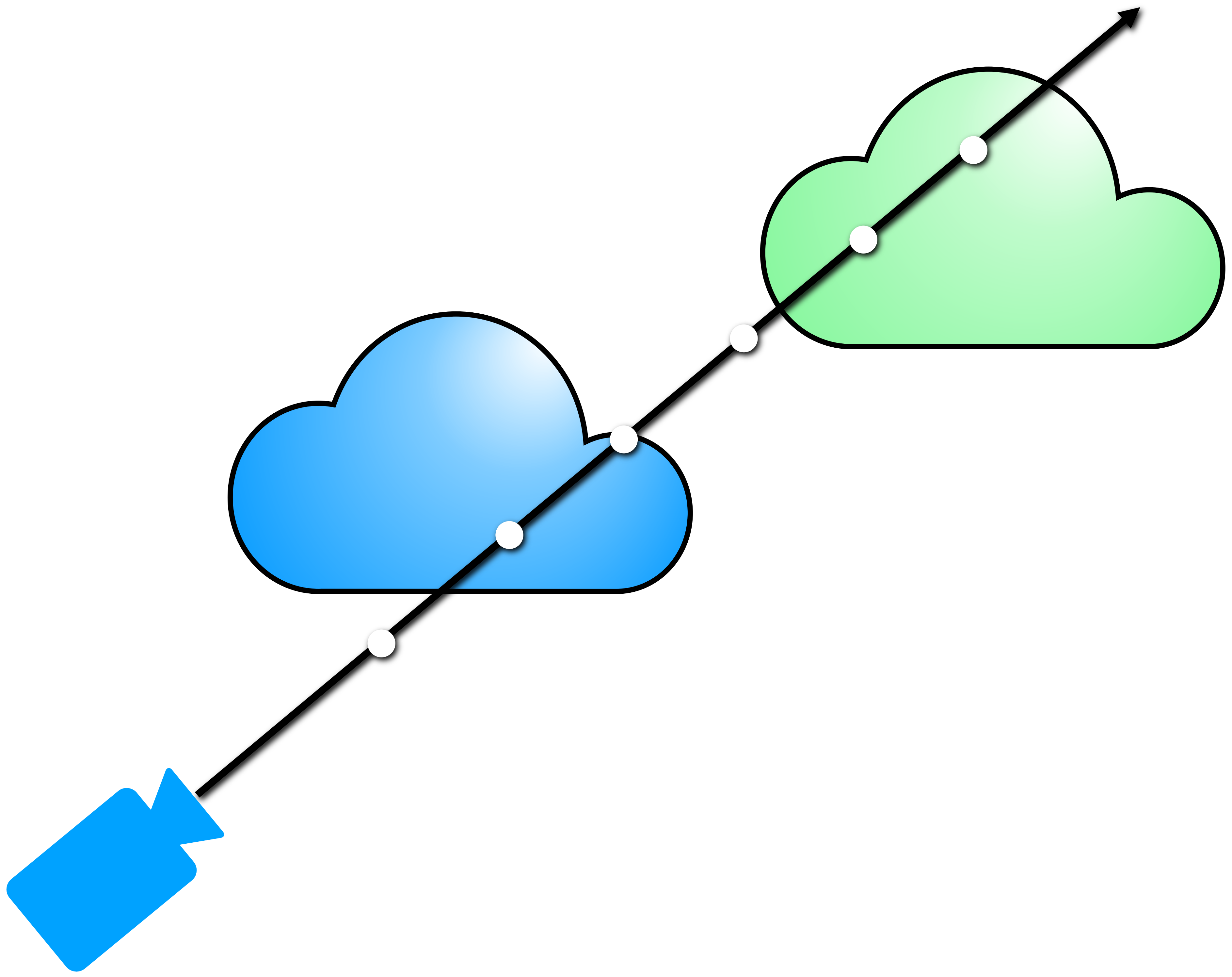
How much light is contributed by ray segment *i*:

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

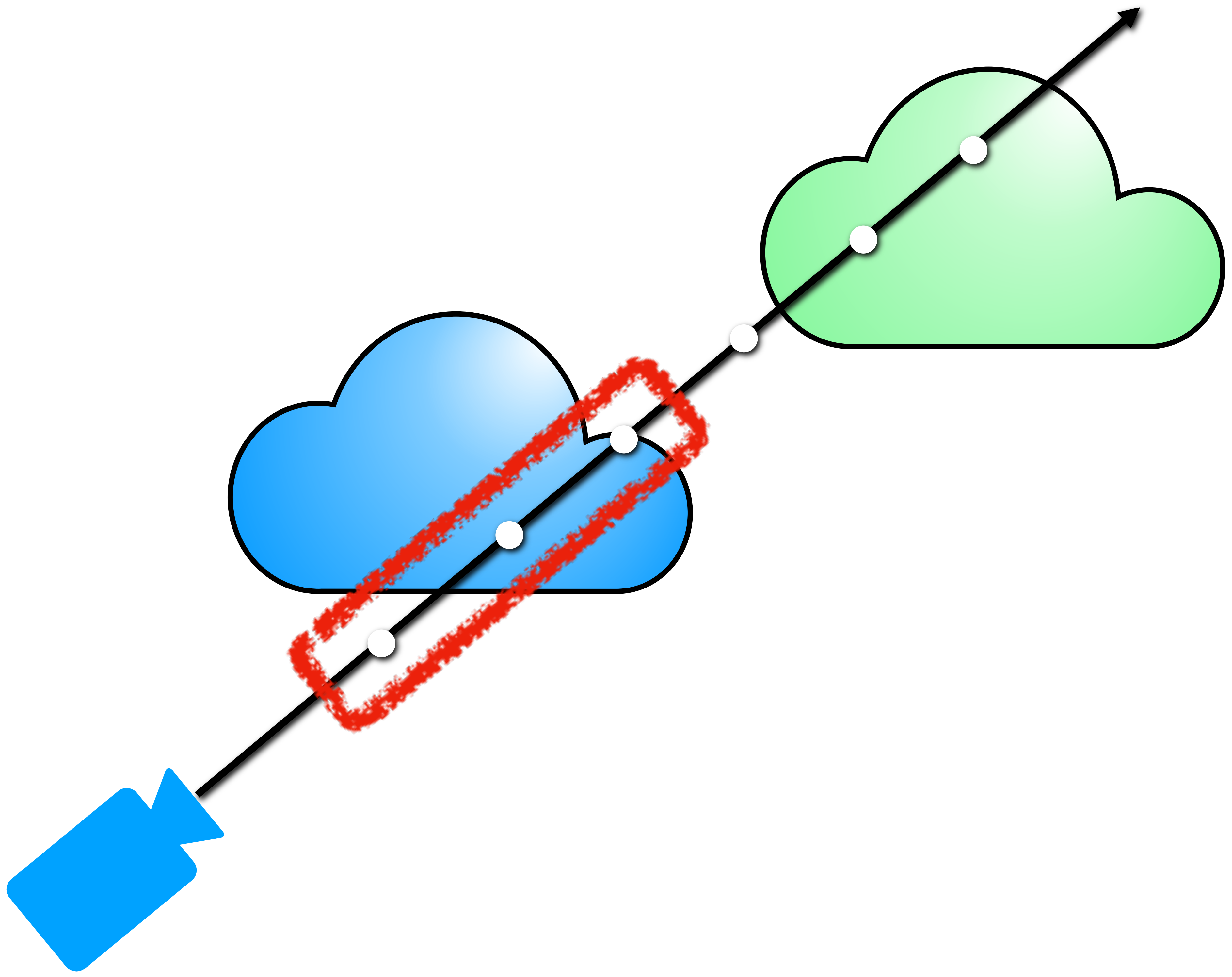


Further points on volume rendering

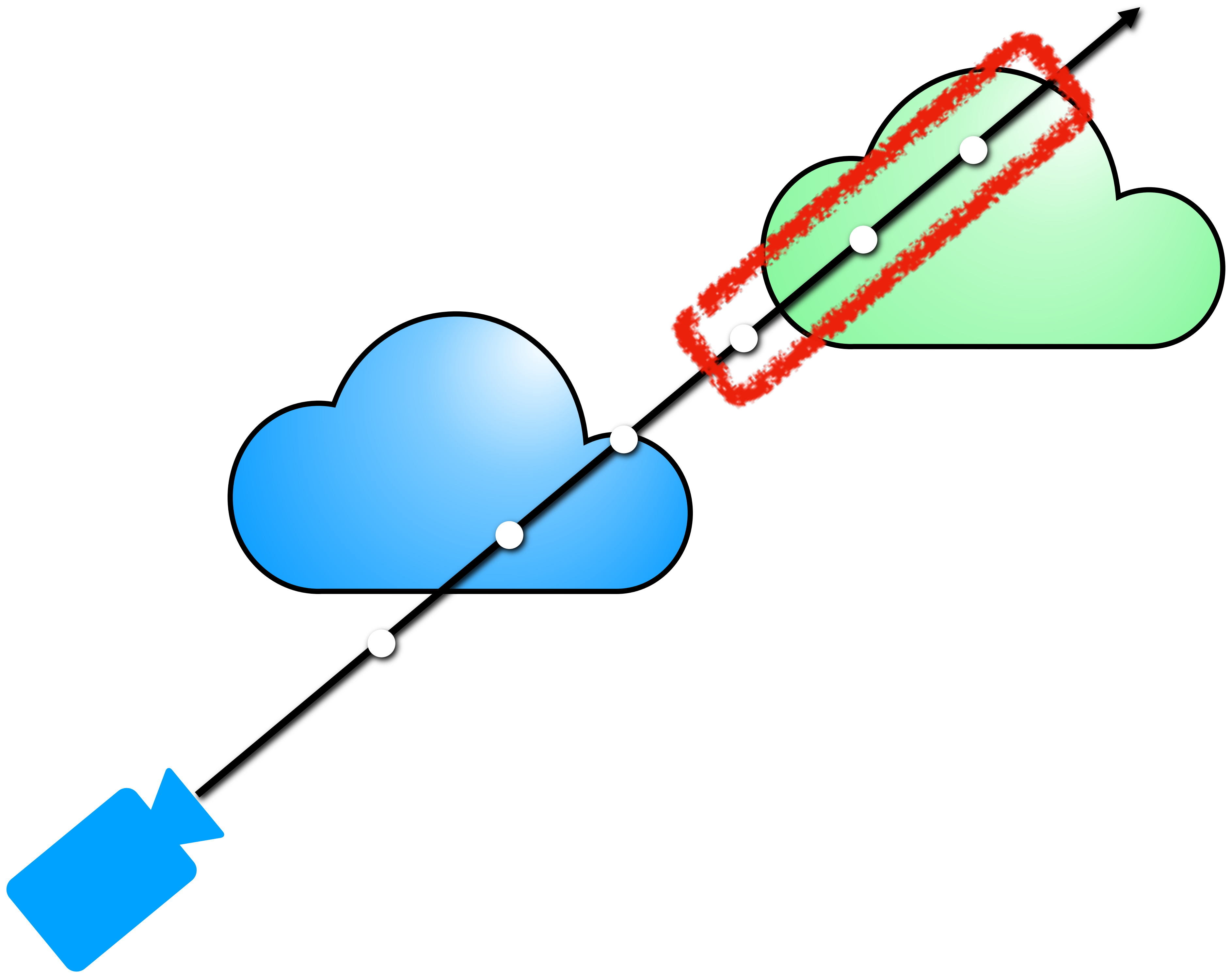
Alpha mattes and compositing



Alpha mattes and compositing



Alpha mattes and compositing



Alpha mattes and compositing

Mildenhall*, Srinivasan*, Tancik* et al 2020, *NeRF*
Poole et al 2022, *DreamFusion*
Tang et al 2022, *Compressible-composable NeRF via Rank-residual Decomposition*

Rendering weight PDF is important

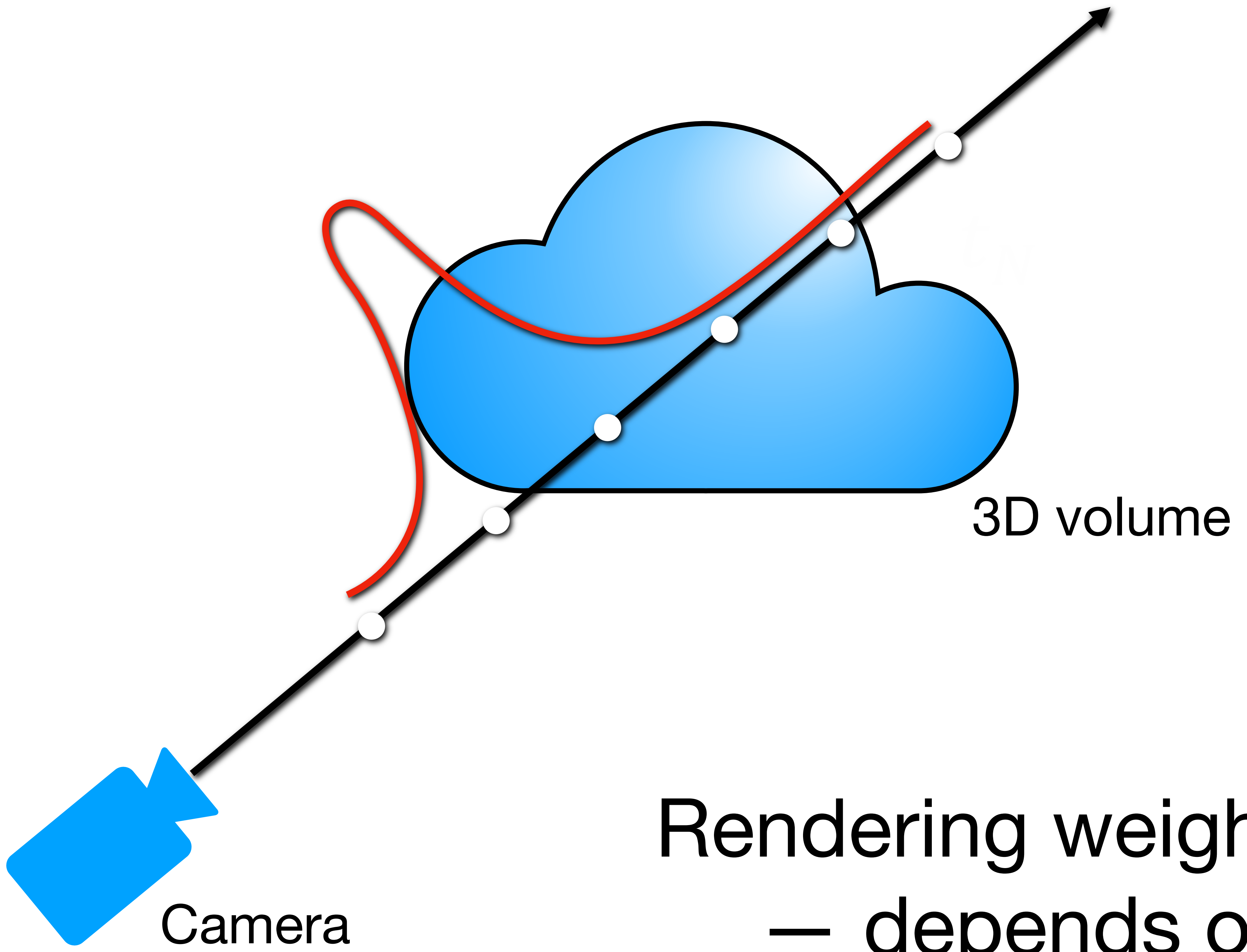
Remember, expected color is equal to

$$\int T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_i \underbrace{T_i\alpha_i}_{w_i}\mathbf{c}_i$$

$T(t)\sigma(t)$ and $T_i\alpha_i$ are "rendering weights" — probability distribution along the ray (continuous and discrete, respectively)

Visual intuition — rendering weights depend on the ray!

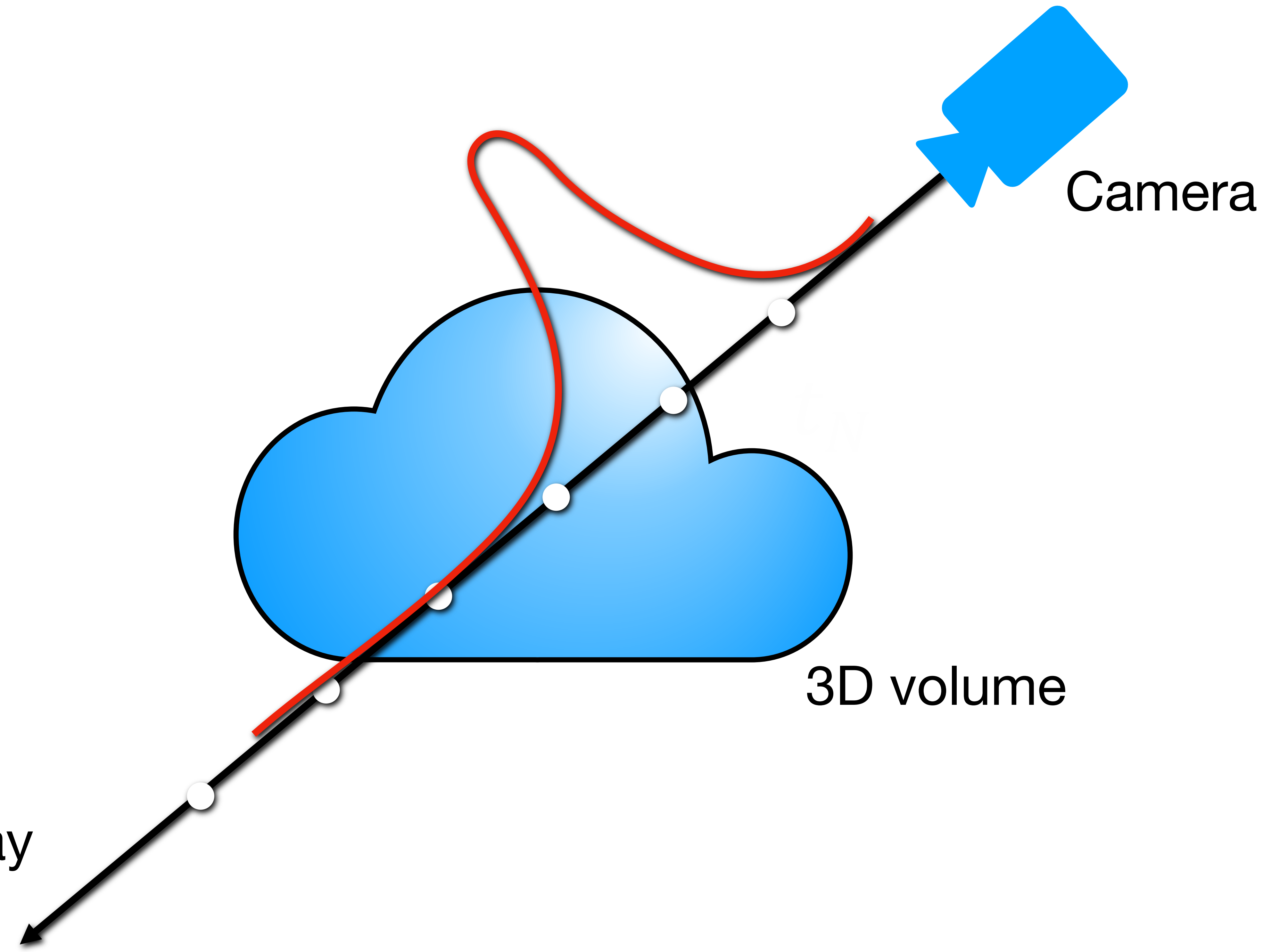
$$C \approx \sum_{i=1}^{N} \boxed{T_i \alpha_i} c_i$$



Rendering weights are not a 3D function — depends on the ray, because of transmittance!

Visual intuition — rendering weights depend on the ray!

$$C \approx \sum_{i=1}^{N} \boxed{T_i \alpha_i} c_i$$



Rendering weights are not a 3D function — depends on the ray, because of transmittance!

Use the rendering weight (PDF) to render depth

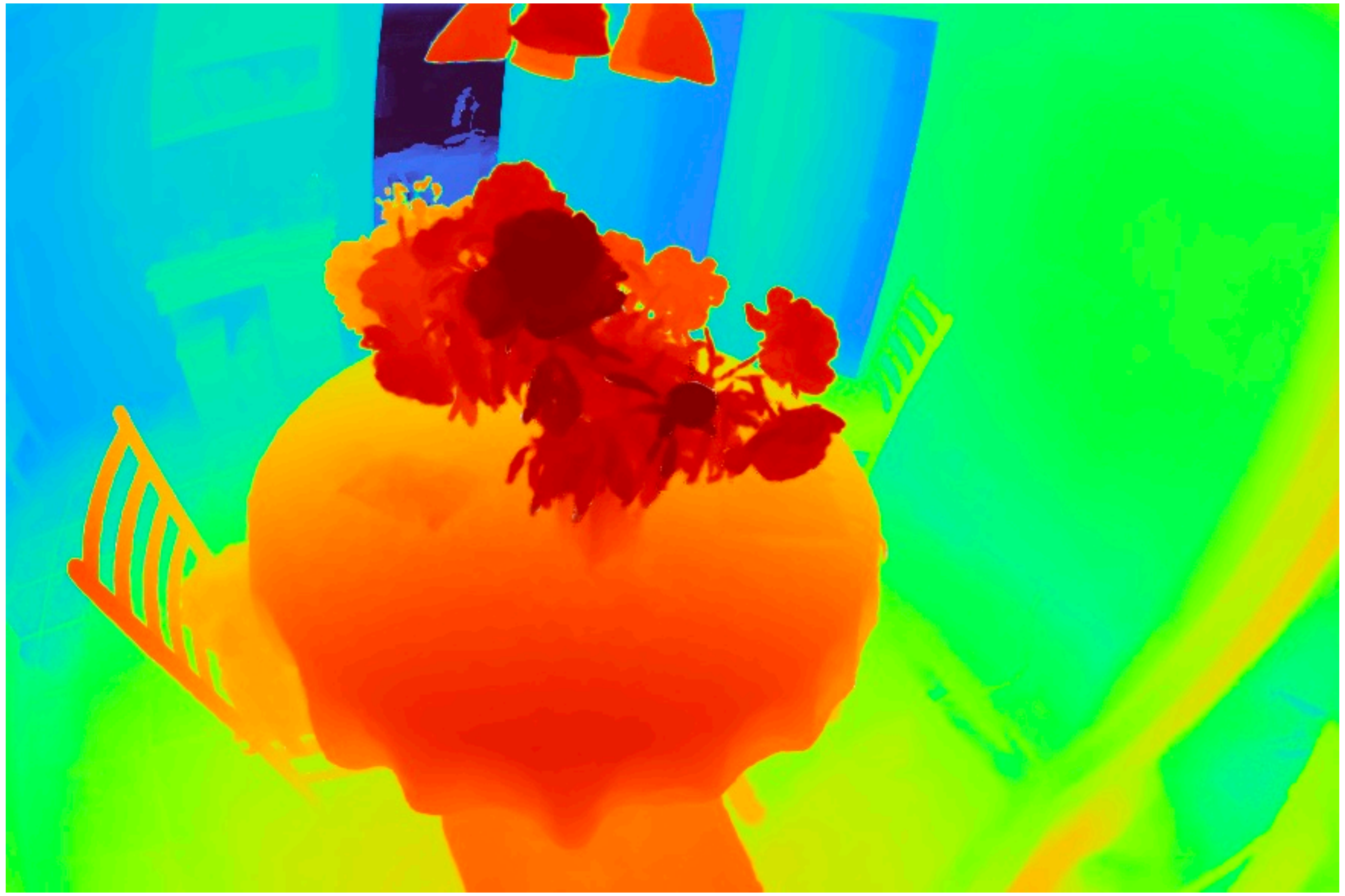
We can use this distribution to compute expectations for other quantities, e.g. “expected depth”:

$$\bar{t} = \sum_i T_i \alpha_i t_i$$

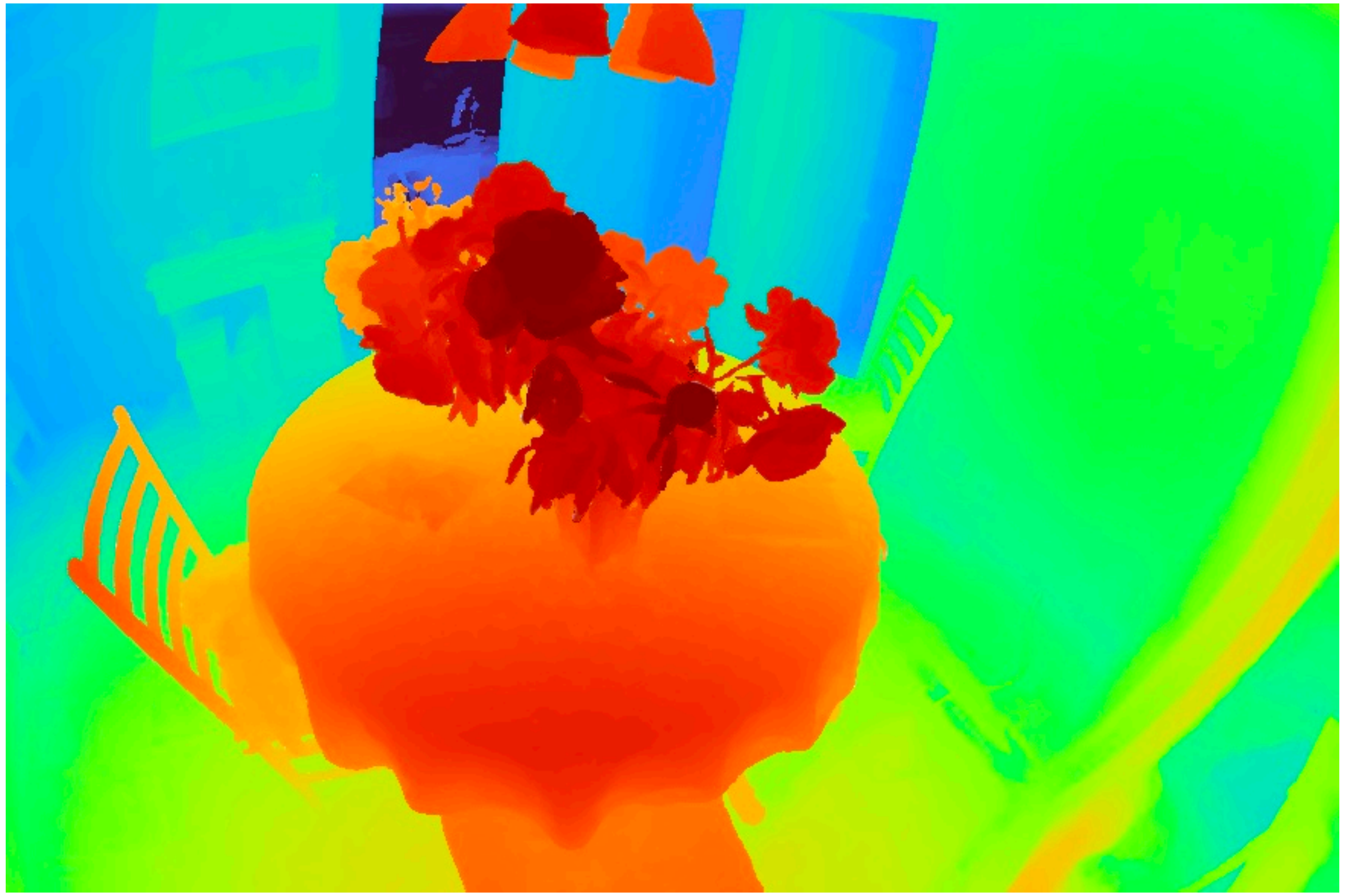
This is often how people visualise NeRF depth maps.

Alternatively, other statistics like mode or median can be used.

Rendering depth value with the PDF

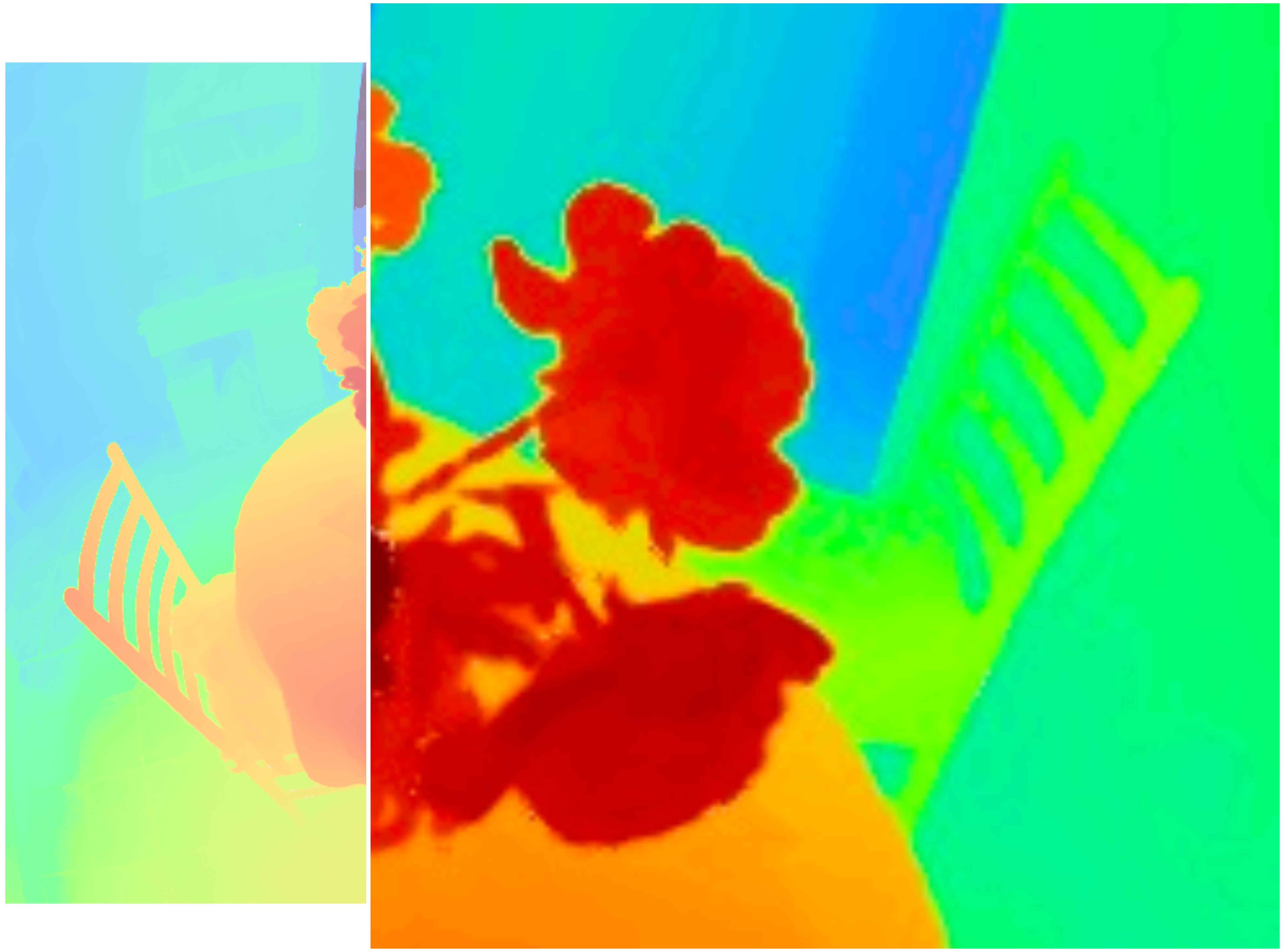


Mean depth

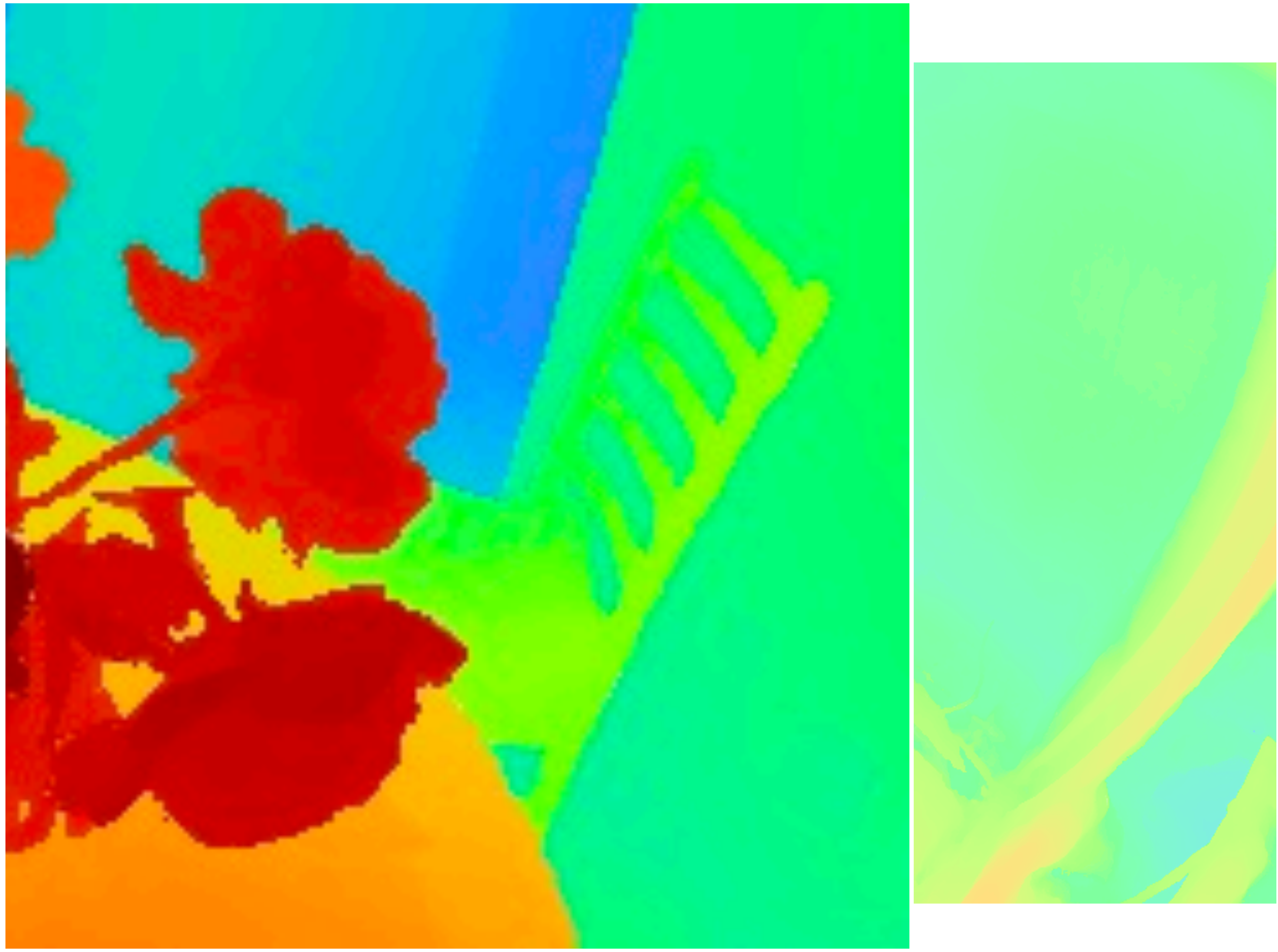


Median depth

Rendering depth value with the PDF



Mean depth



Median depth

Volume rendering other quantities

This idea can be used for any quantity we want to "volume render" into a 2D image. If $\mathbf{v}$ lives in 3D space (semantic features, normal vectors, etc.)

$$\sum_i T_i \alpha_i \mathbf{v}_i$$

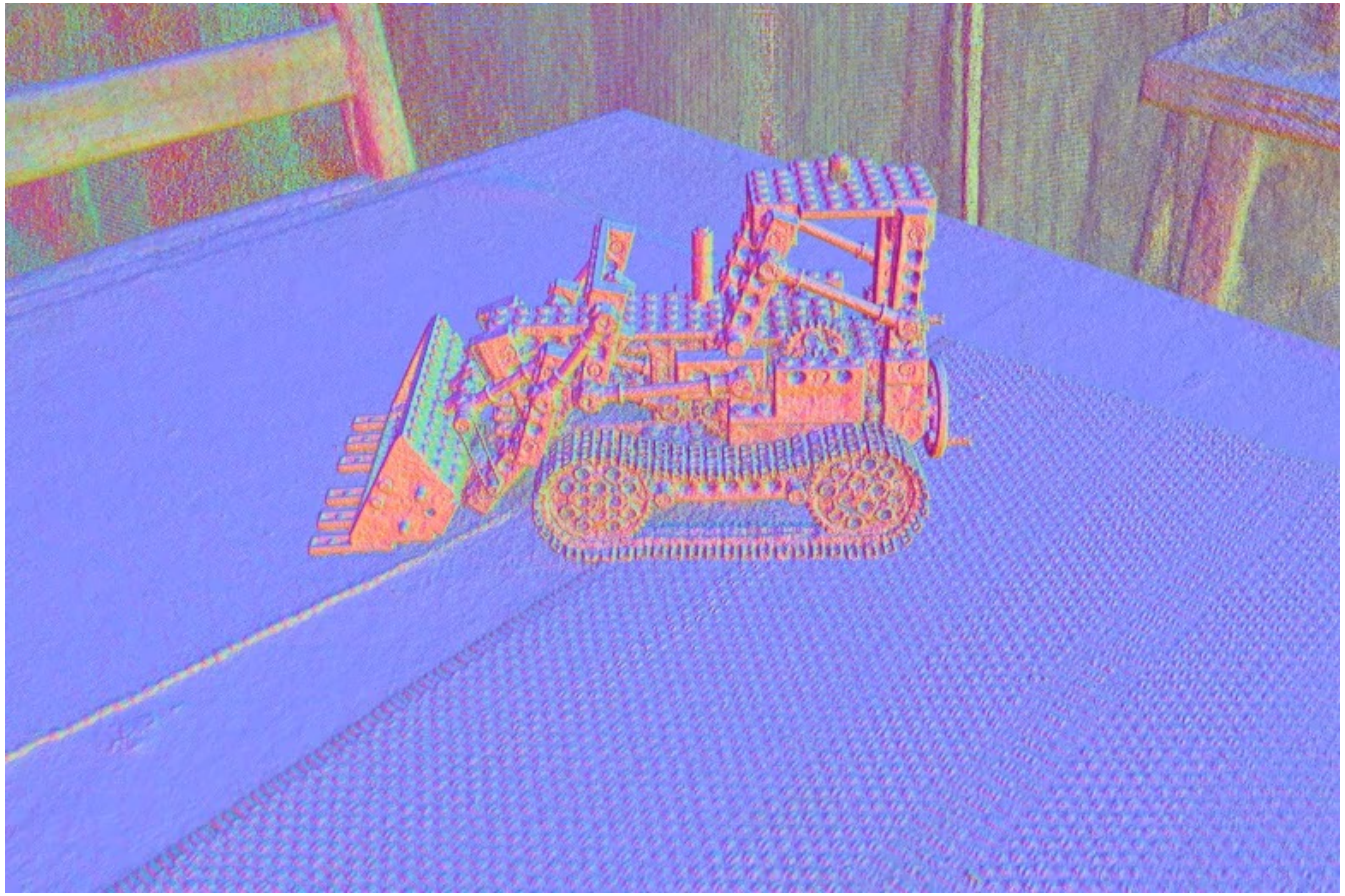
can be taken per-ray to produce 2D output images.

Volume rendering other quantities

Various recent works have used this idea to render higher-level semantic feature maps (e.g., *Feature Field Distillation* and *Neural Feature Fusion Fields*).

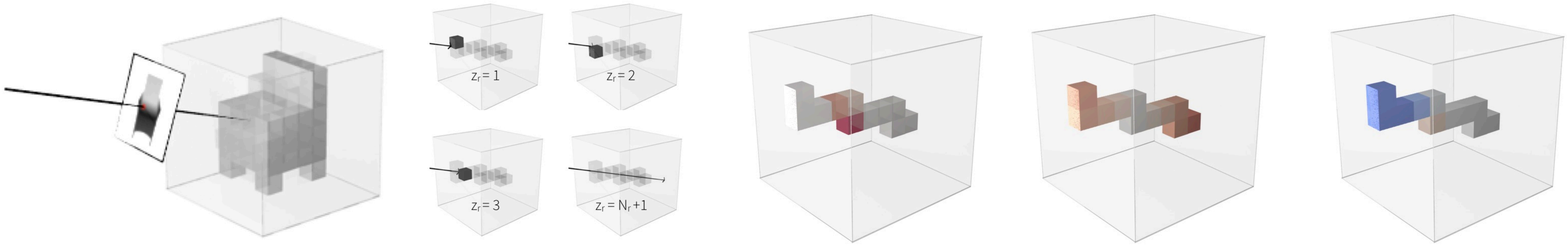
Kobayashi et al 2022, *Decomposing NeRF for Editing via Feature Field Distillation*, , Tschernezki et al. 2022 , Neural Feature Fusion Fields: 3D Distillation of Self-Supervised 2D Image Representation

Density as geometry



Normal vectors (from analytic gradient of density)

Alpha compositing model in ML/computer vision



Differentiable ray consistency work used a forward model with "probabilistic occupancy" to supervise 3D-from-single-image prediction. Same rendering model as alpha compositing!

$$p(z_r = i) = \begin{cases} (1 - x_i^r) \prod_{j=1}^{i-1} x_j^r, & \text{if } i \leq N_r \\ \prod_{j=1}^{N_r} x_j^r, & \text{if } i = N_r + 1 \end{cases}$$

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

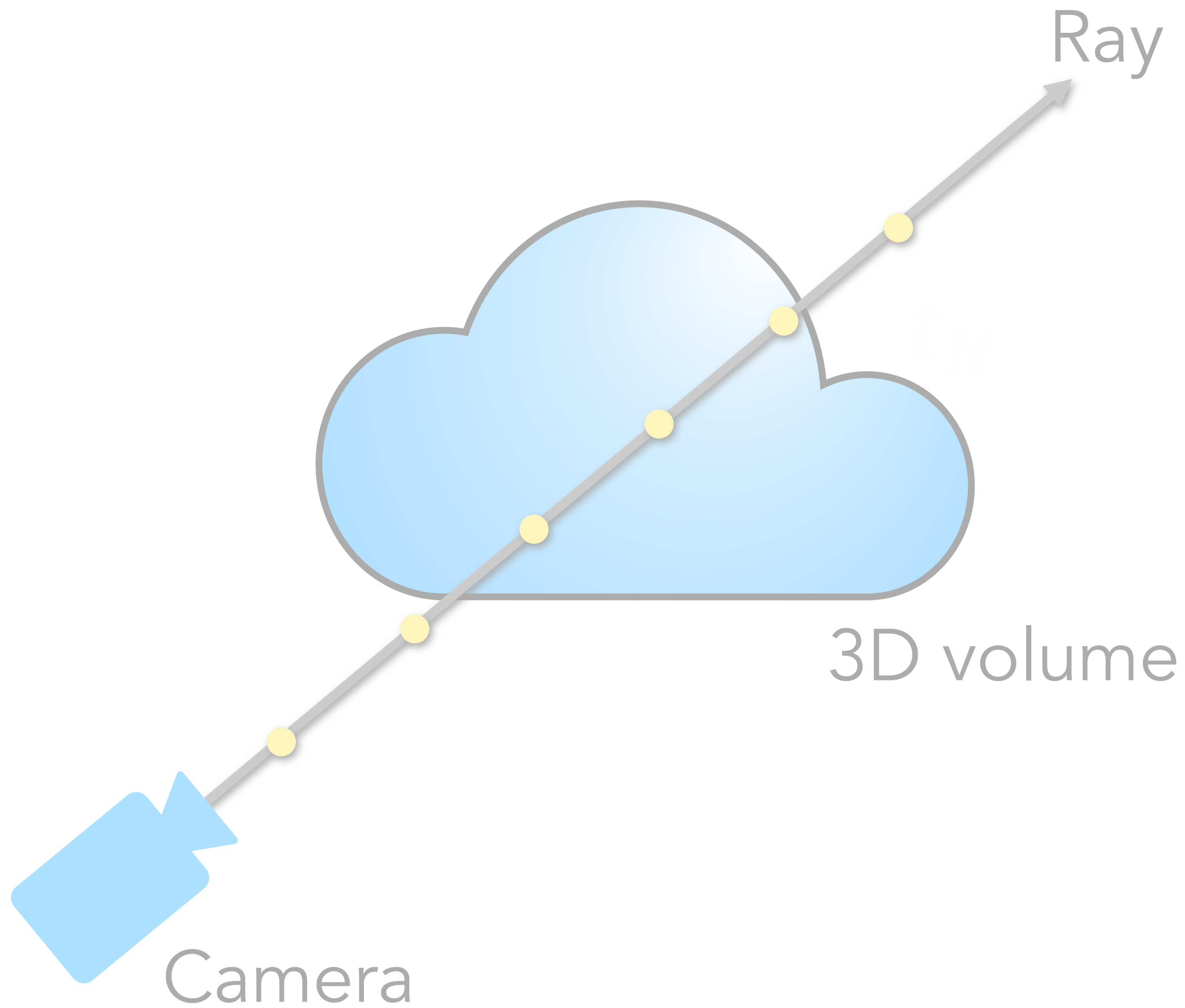
Where we are

1. Birds Eye View & Background
2. Volumetric Rendering Function
3. **Encoding and Representing 3D Volumes**
4. Signal Processing Considerations
5. Challenges & Pointers

Three Key Components

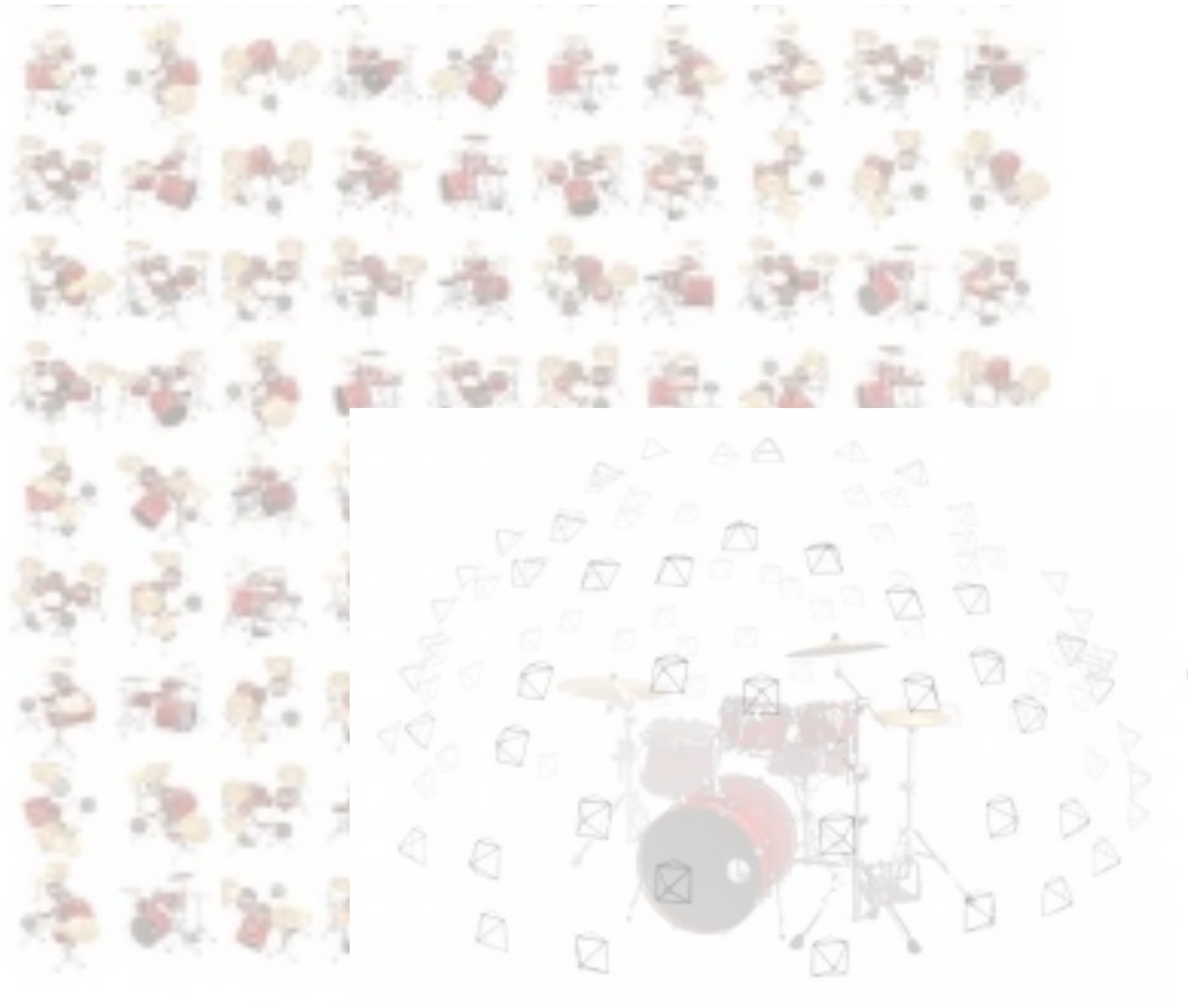
$$(x, y, z, \theta, \phi) \rightarrow F_\Omega \rightarrow (r, g, b, \sigma)$$

Neural Volumetric 3D Scene Representation



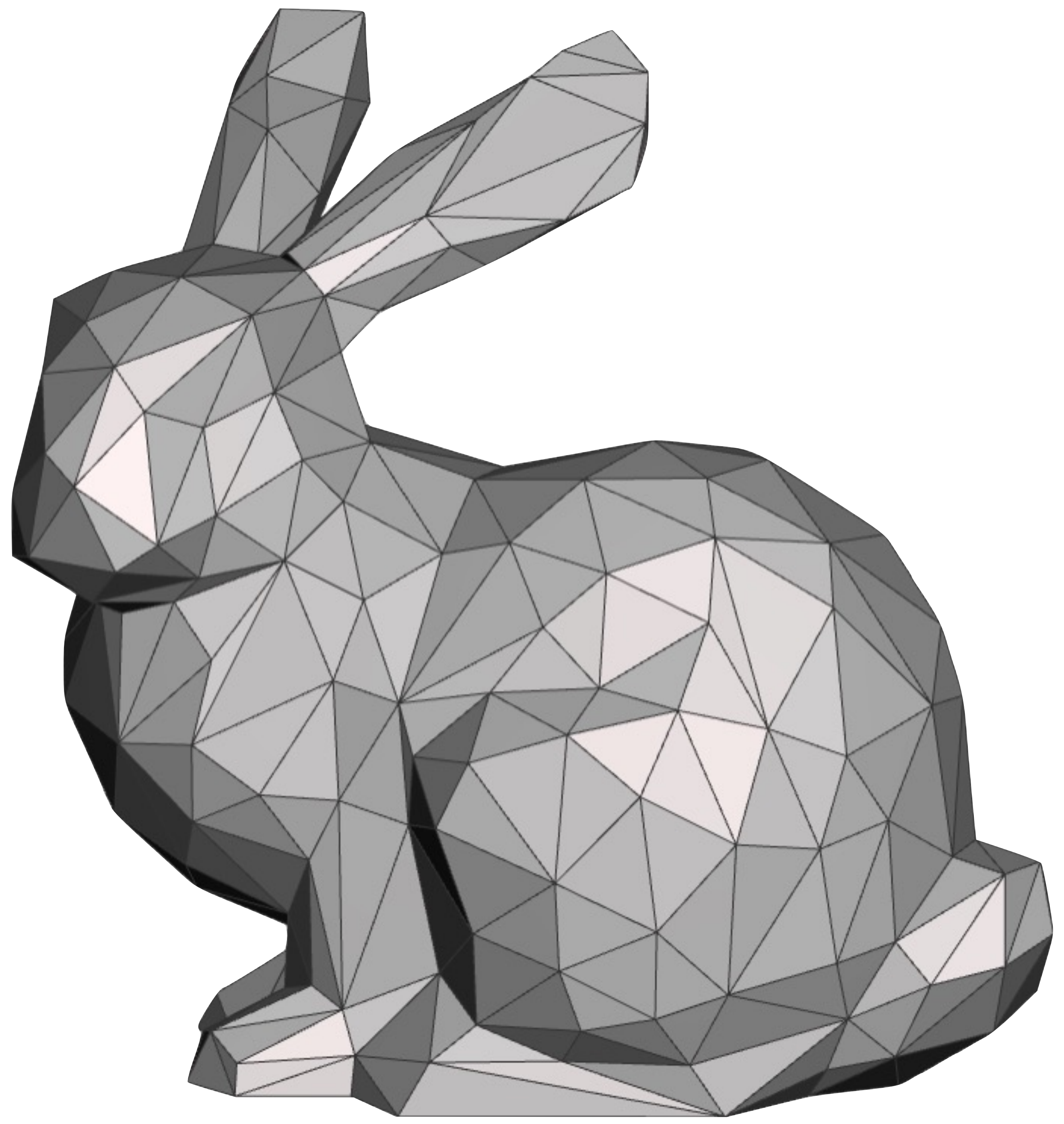
Differentiable Volumetric Rendering Function

Objective: Synthesize all training views

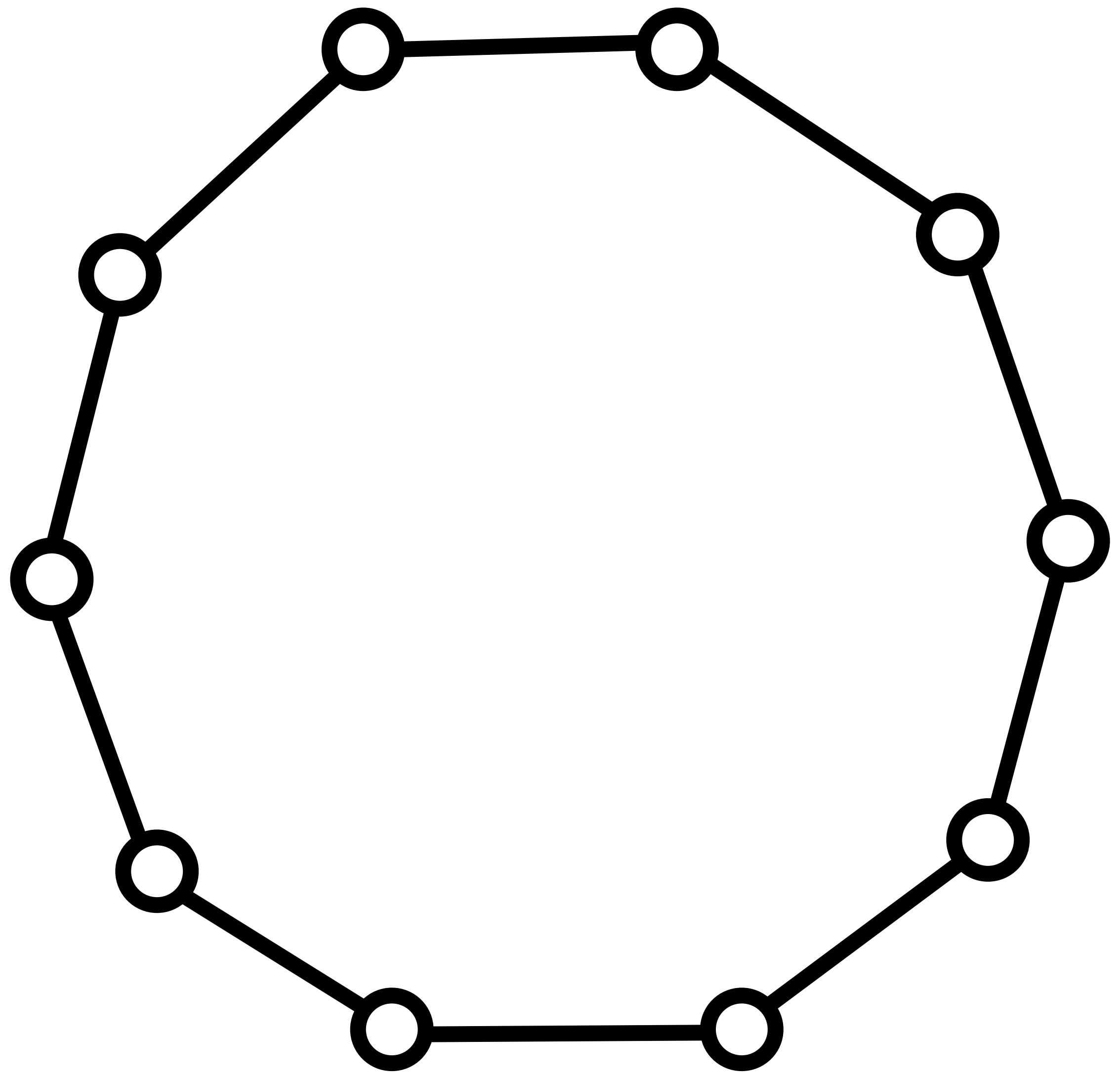


Optimization via Analysis-by-Synthesis

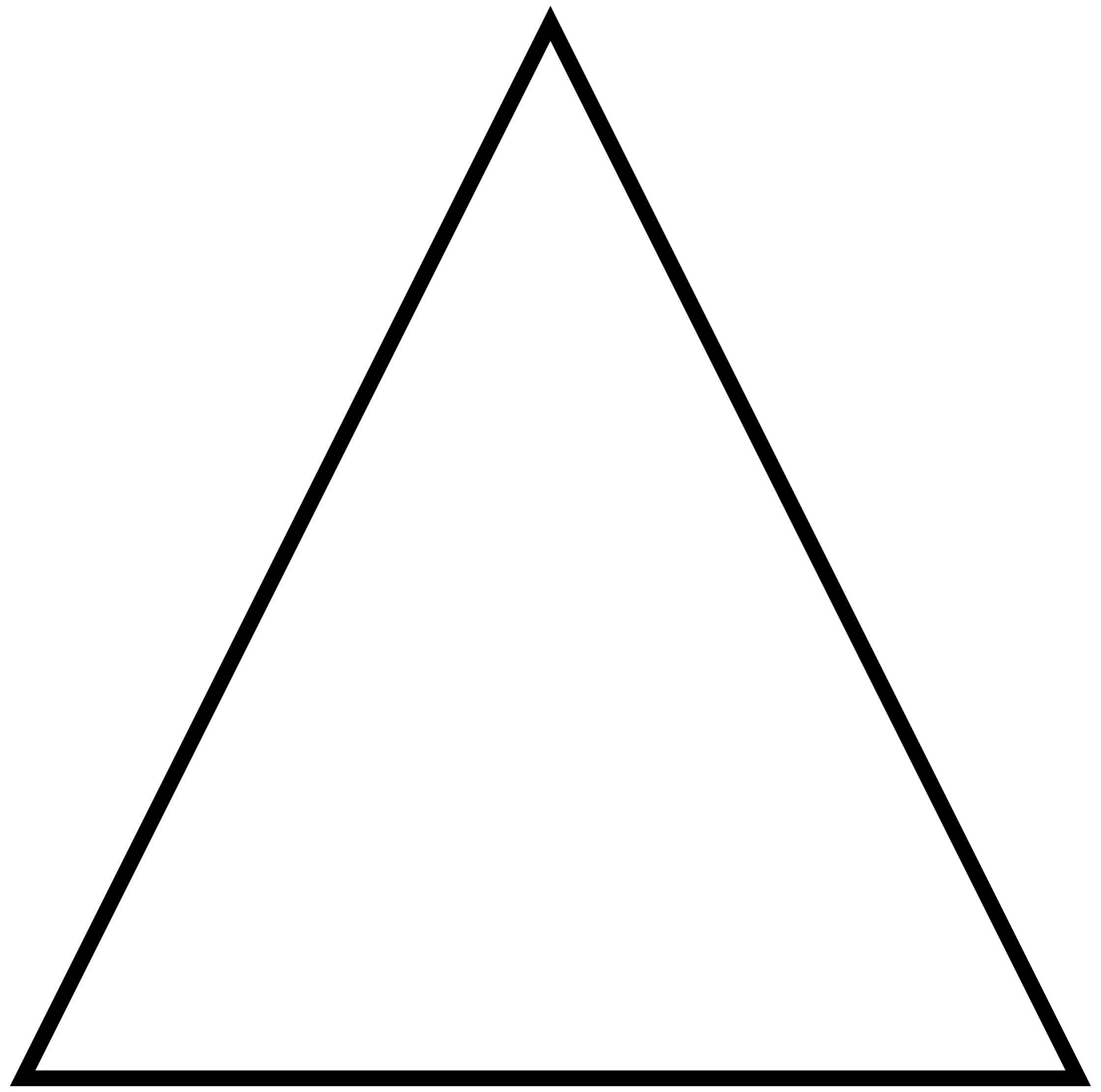
Mesh Representation



Gradient Based Optimization

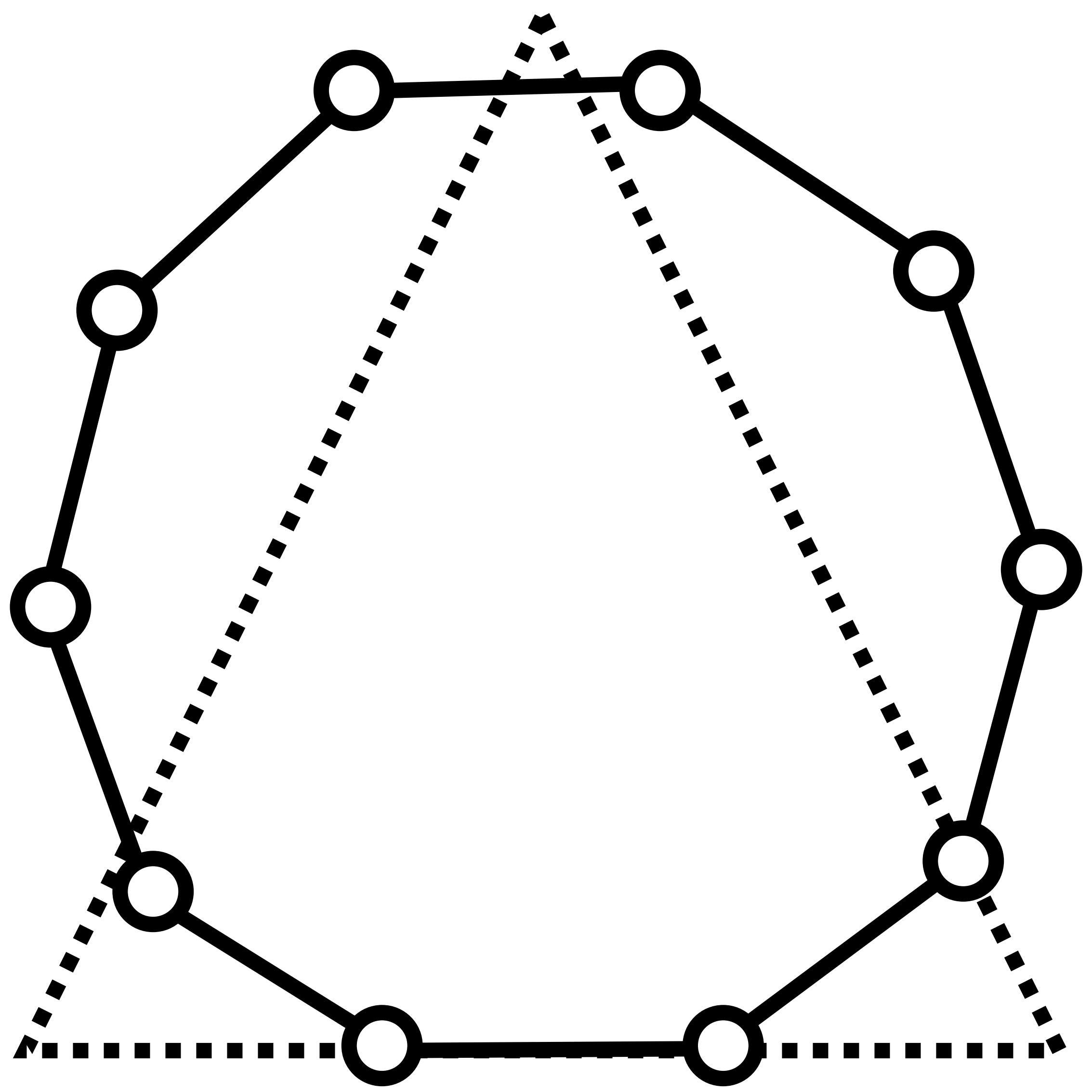


Initial Geometry

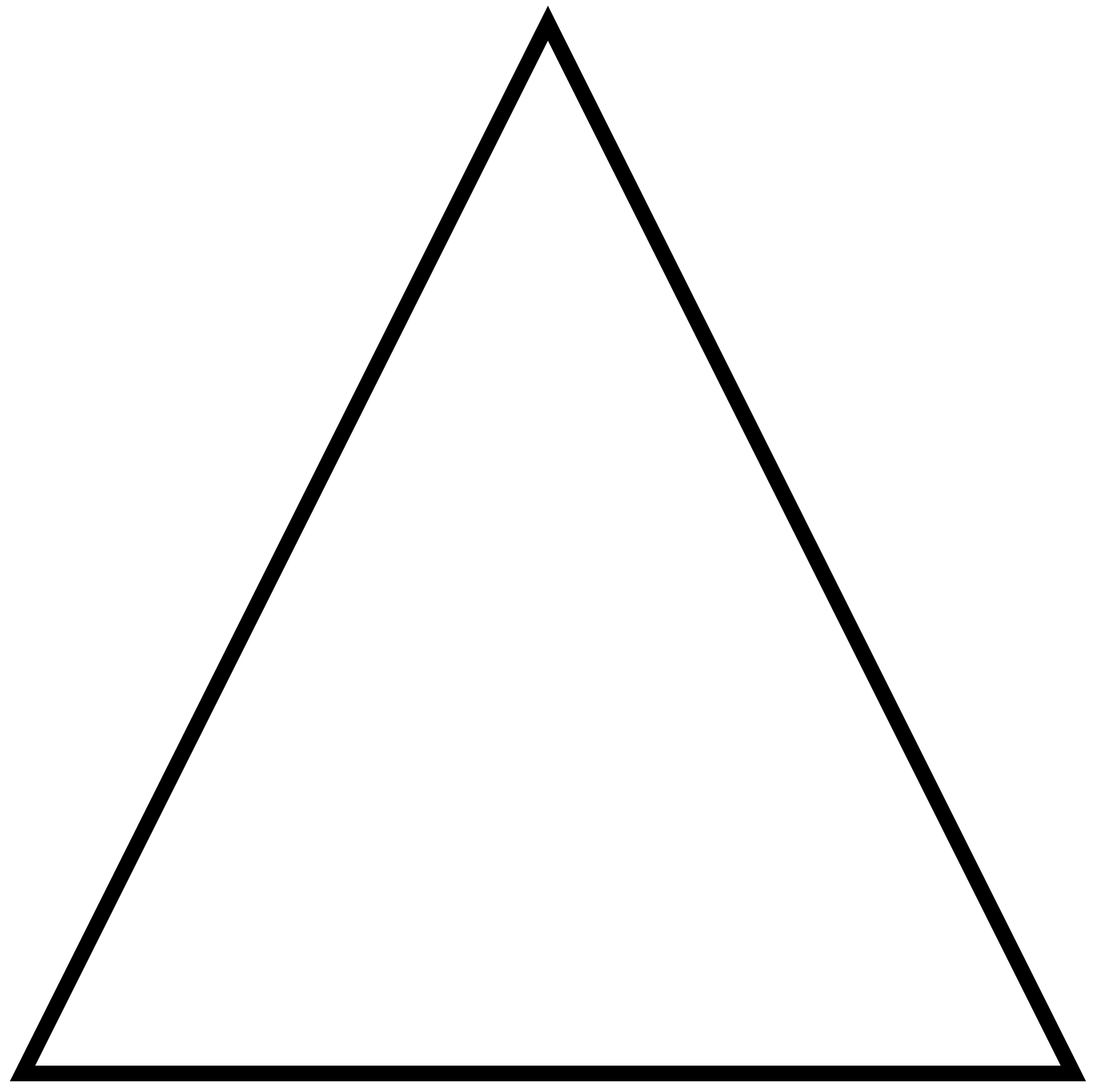


Target Geometry

Gradient Based Optimization

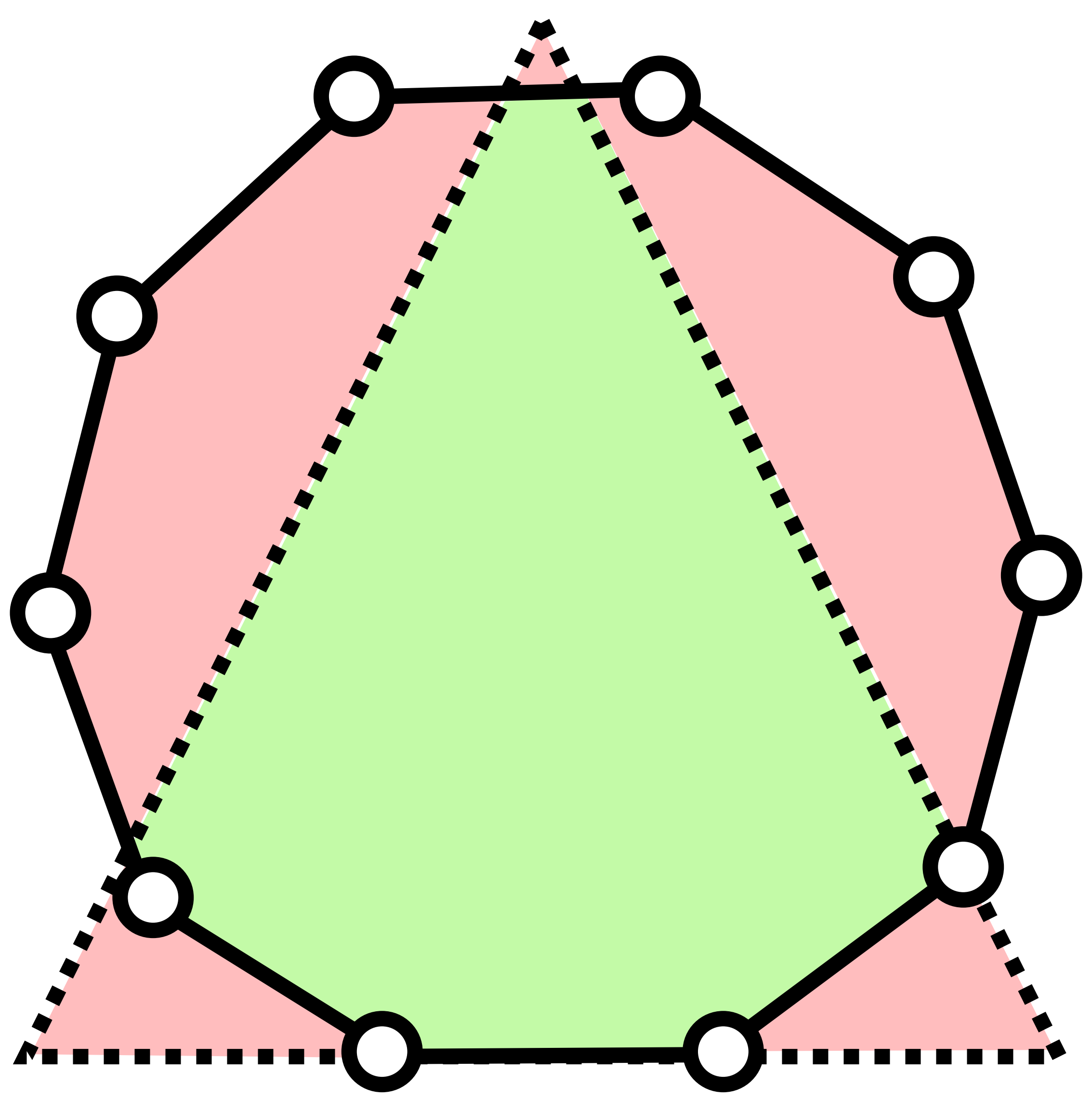


Initial Geometry

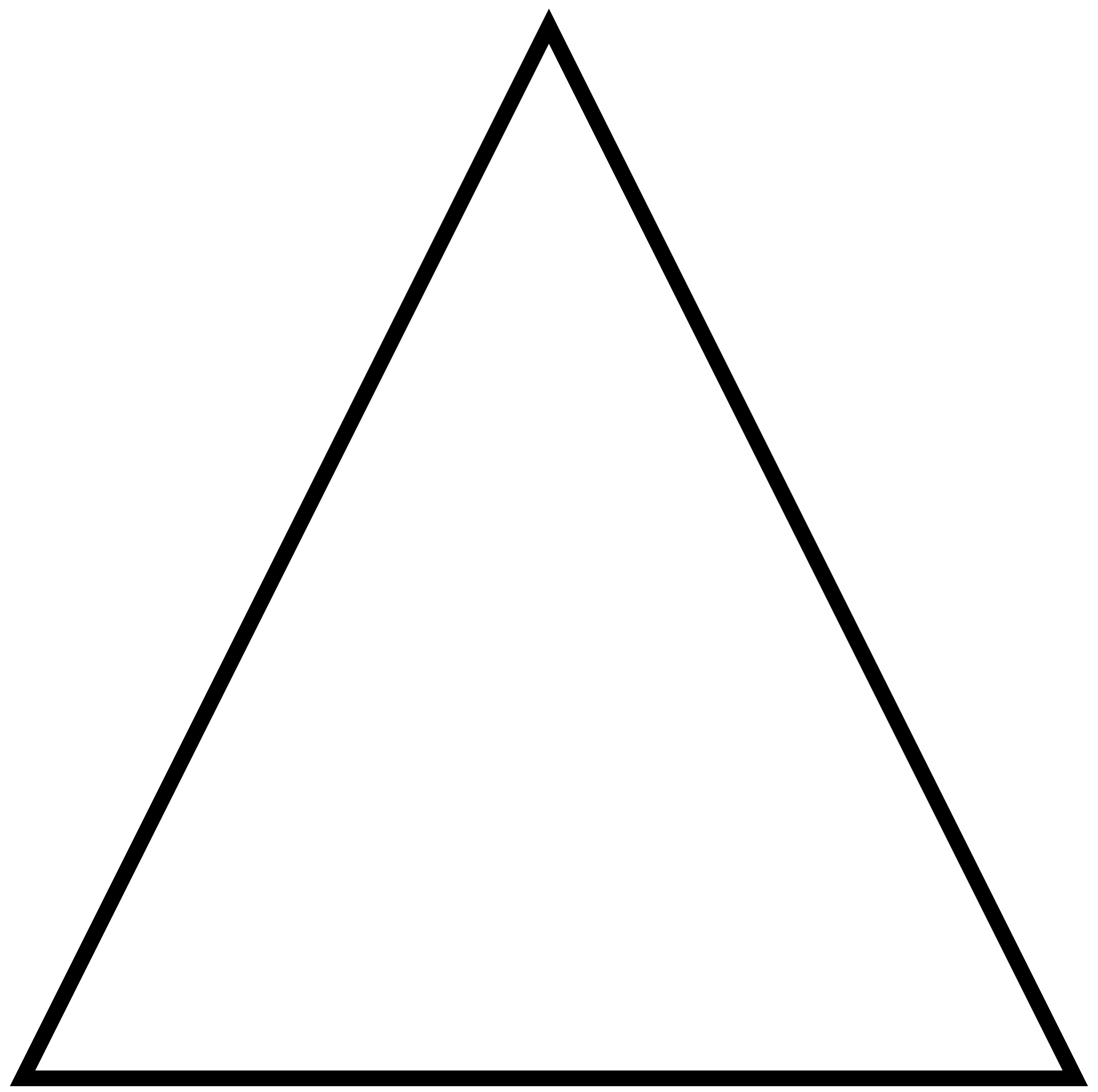


Target Geometry

Gradient Based Optimization

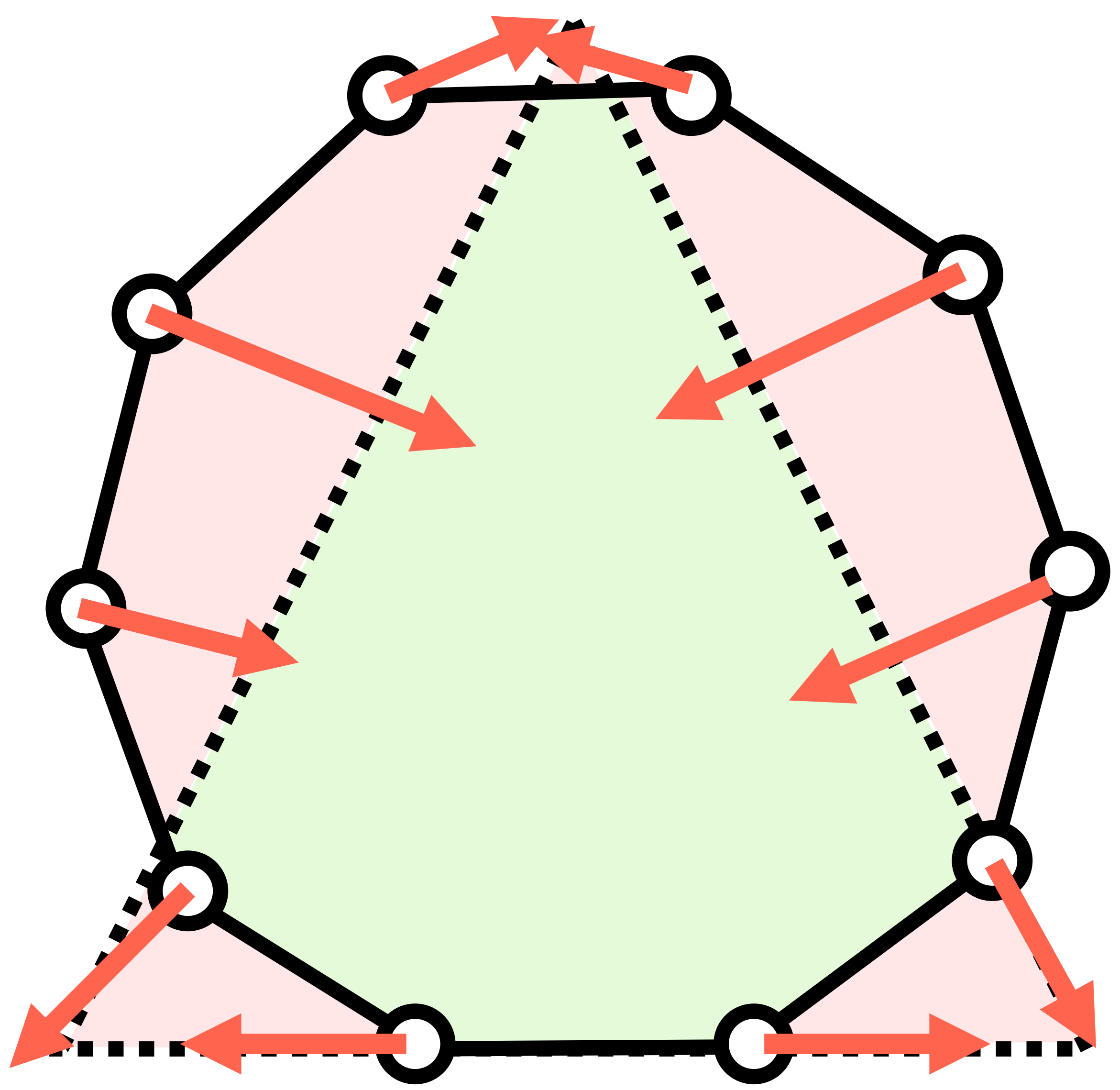


Compute Gradients

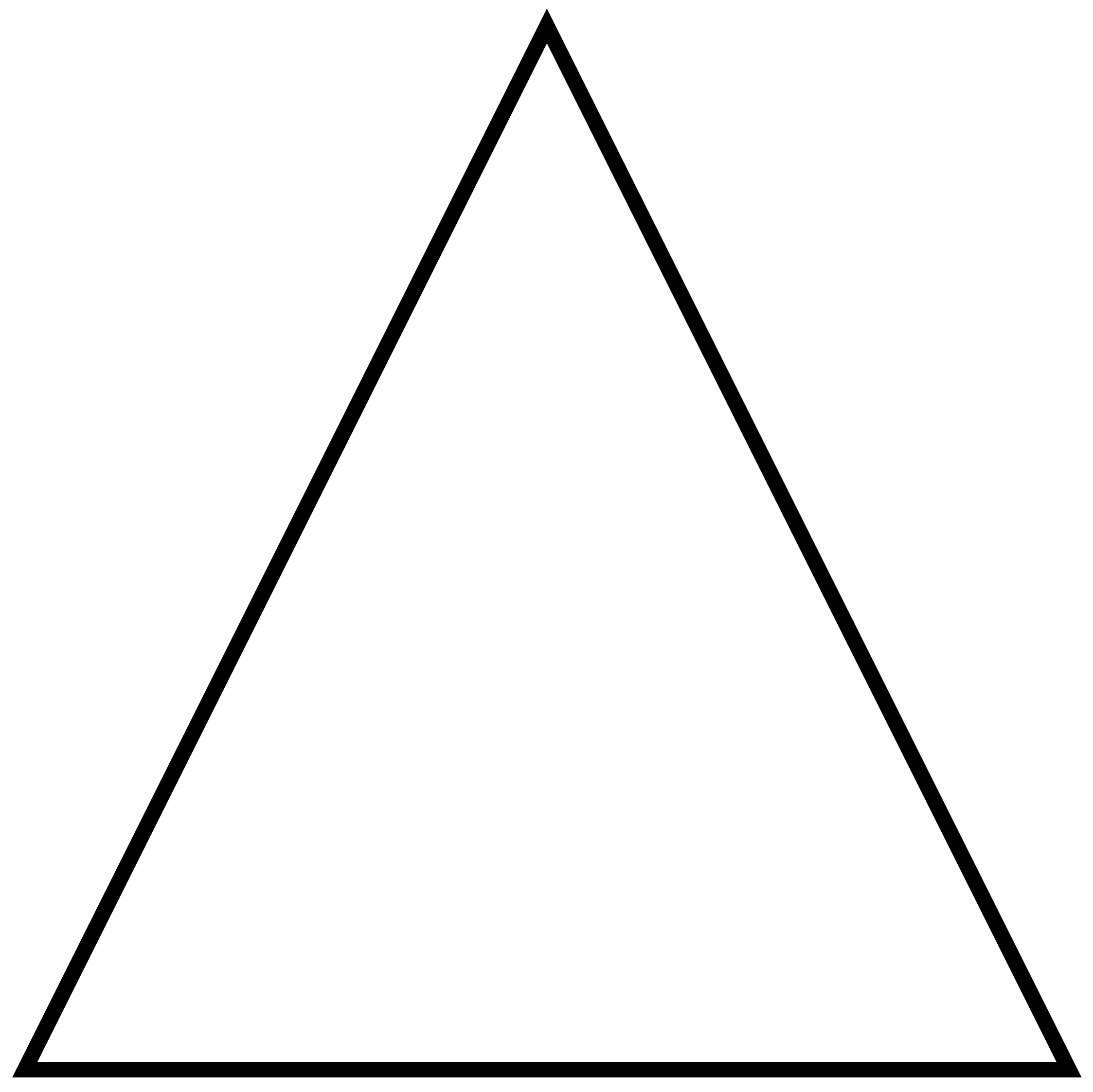


Target Geometry

Gradient Based Optimization

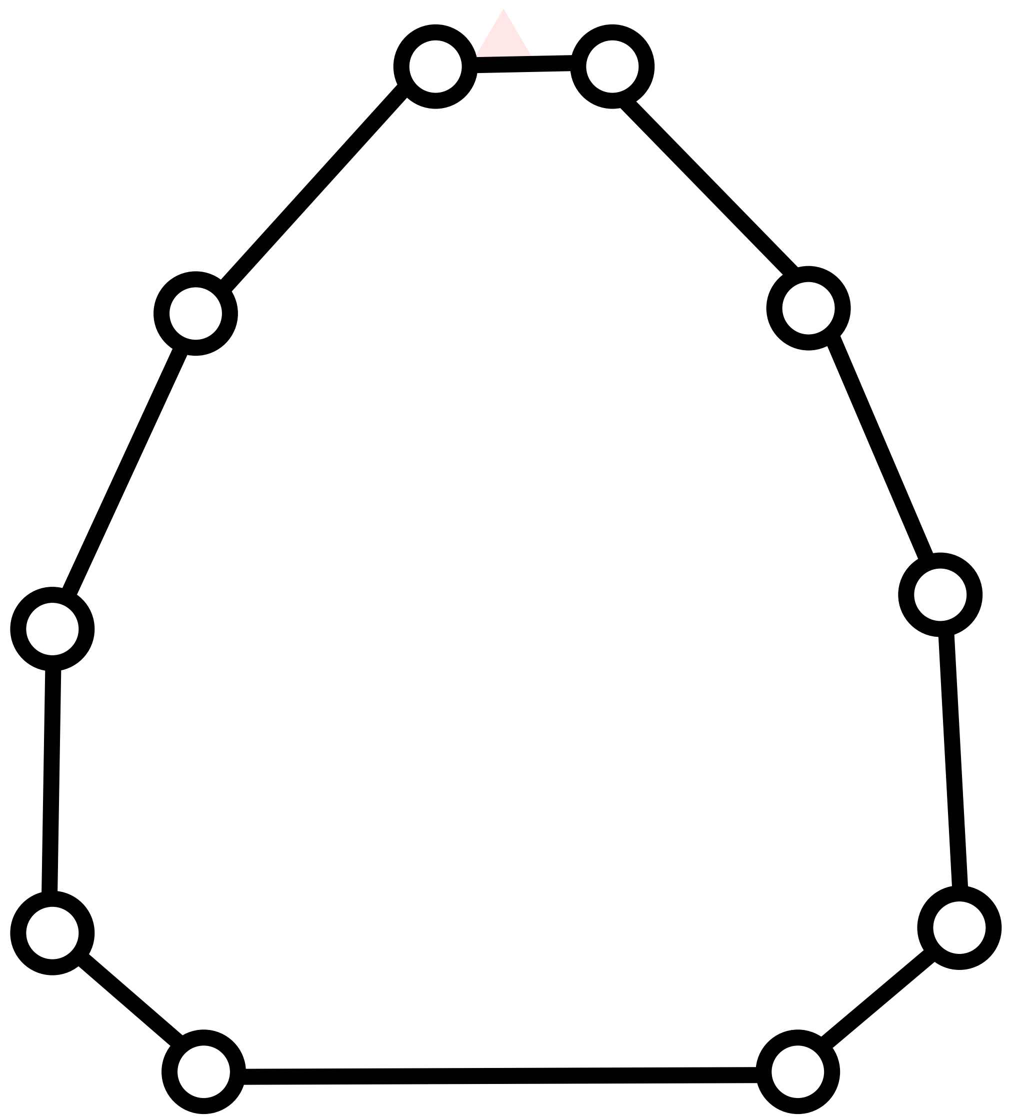


Compute Gradients

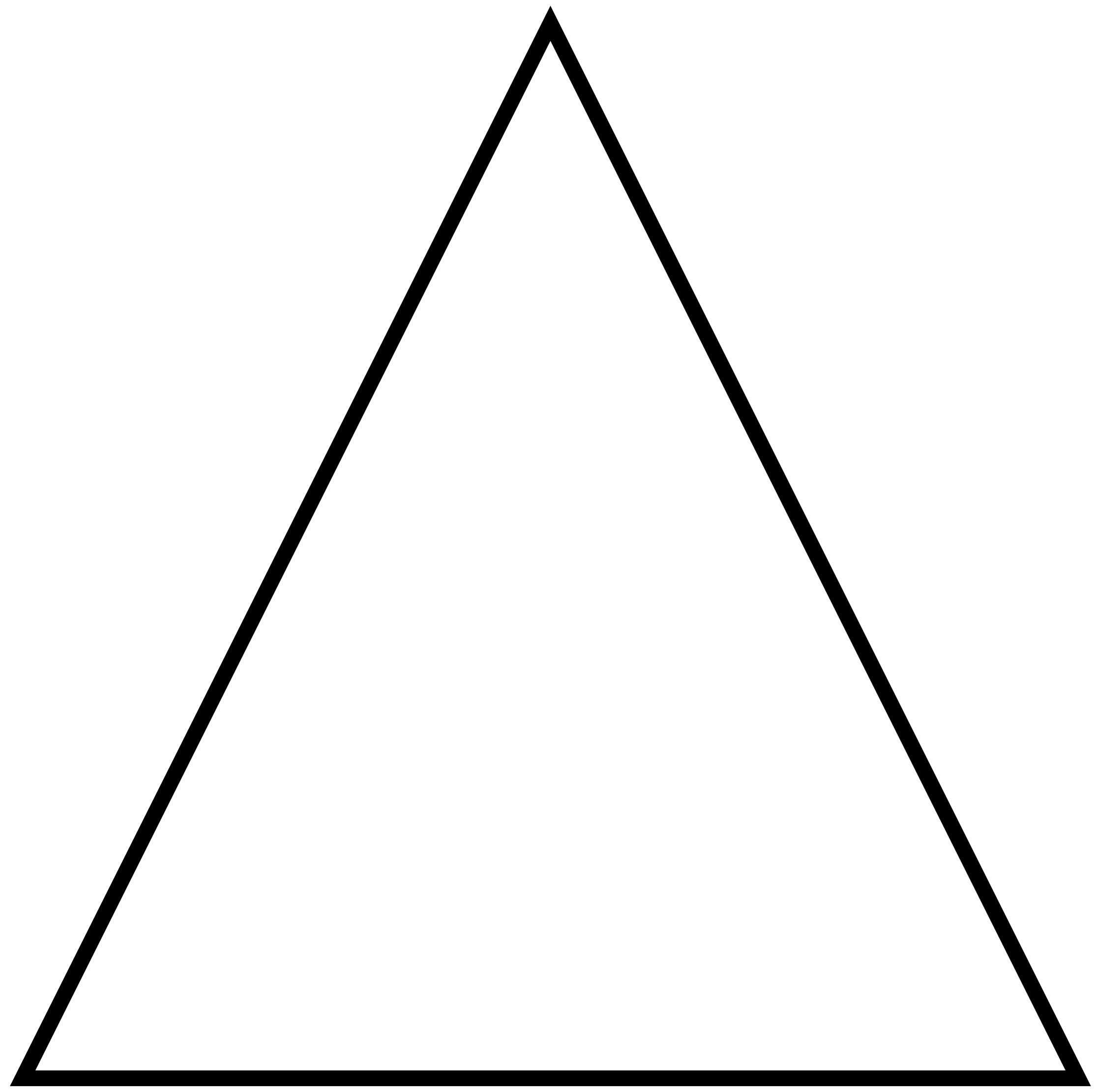


Target Geometry

Gradient Based Optimization

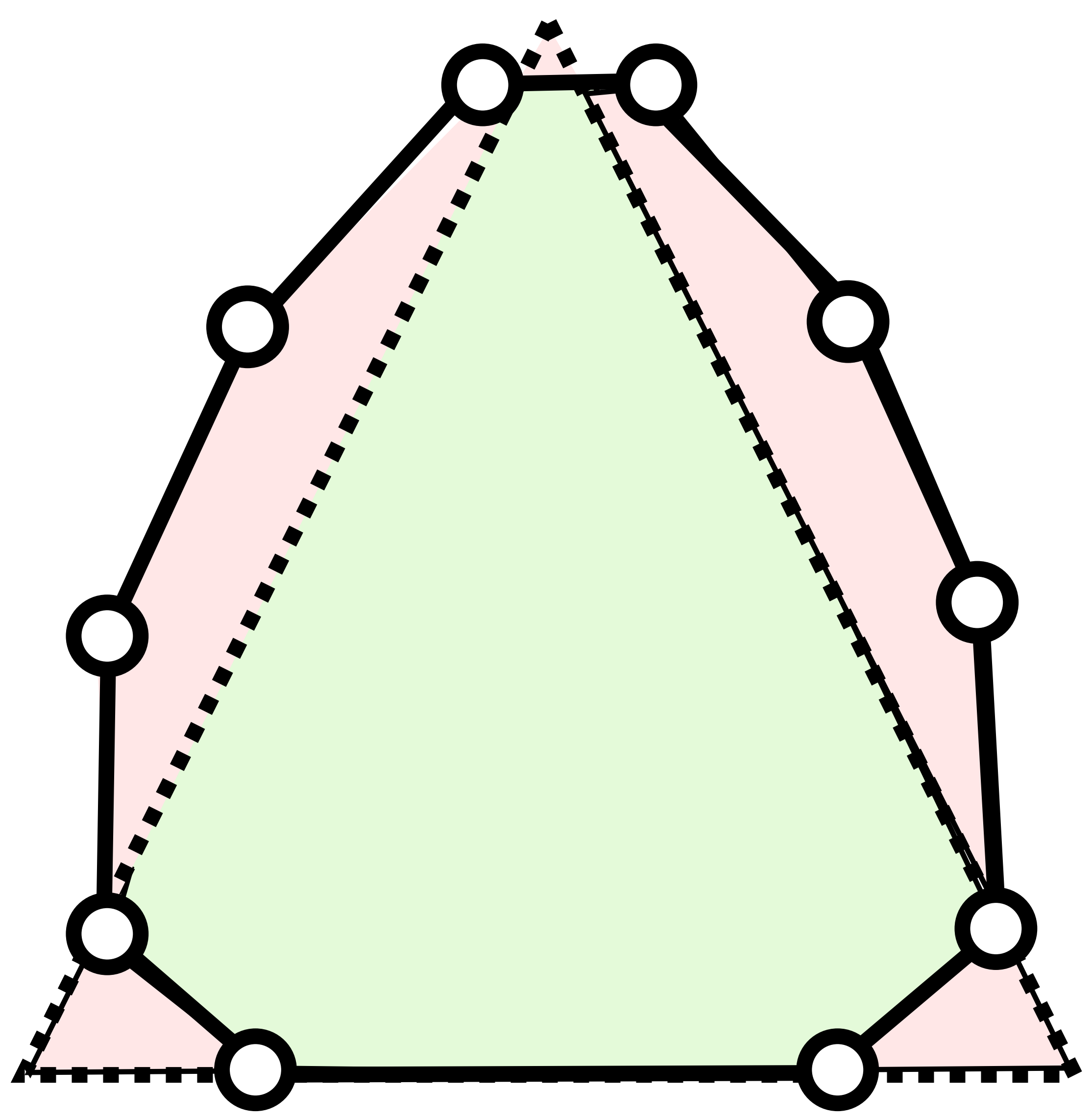


Update positions

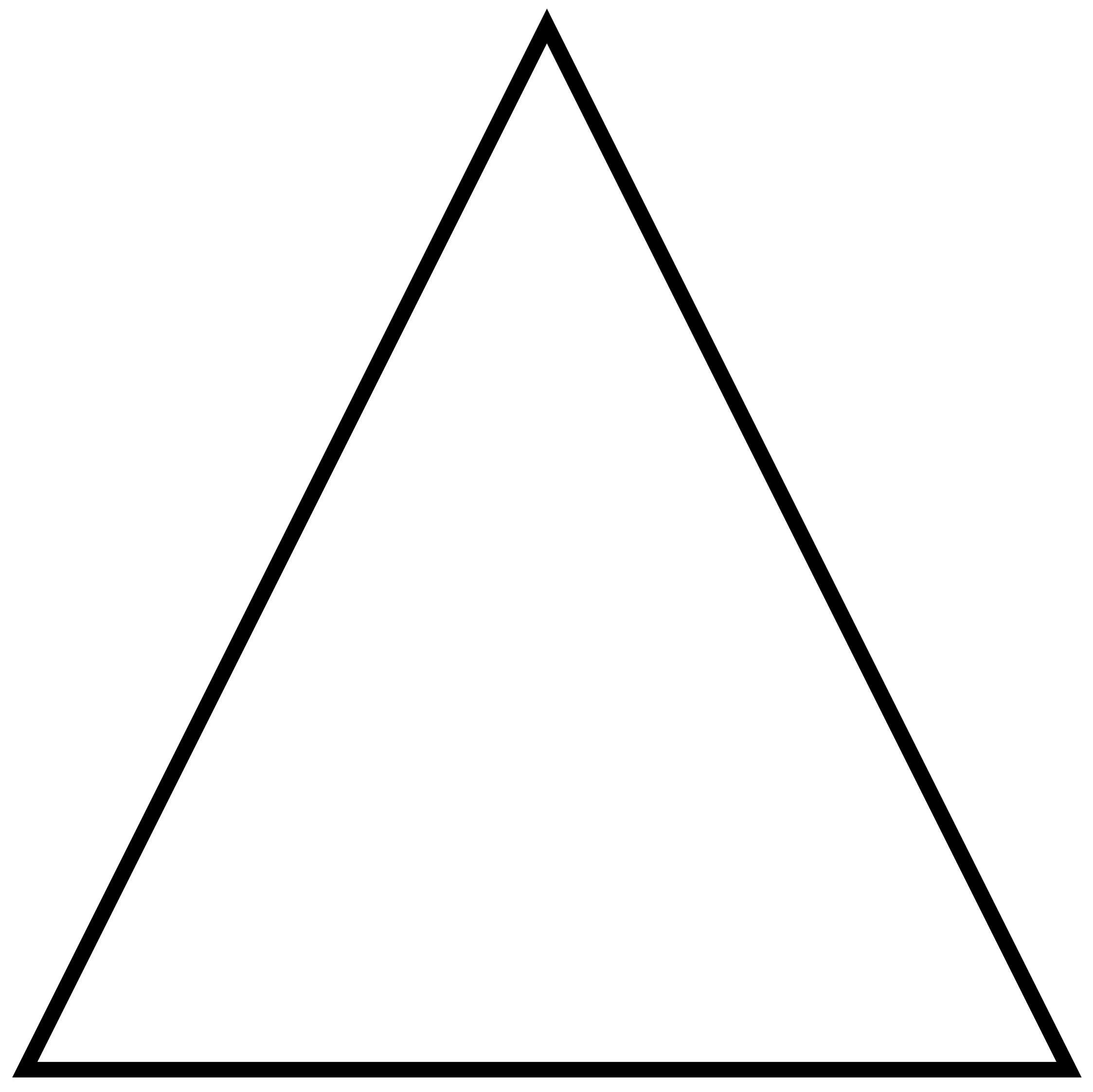


Target Geometry

Gradient Based Optimization

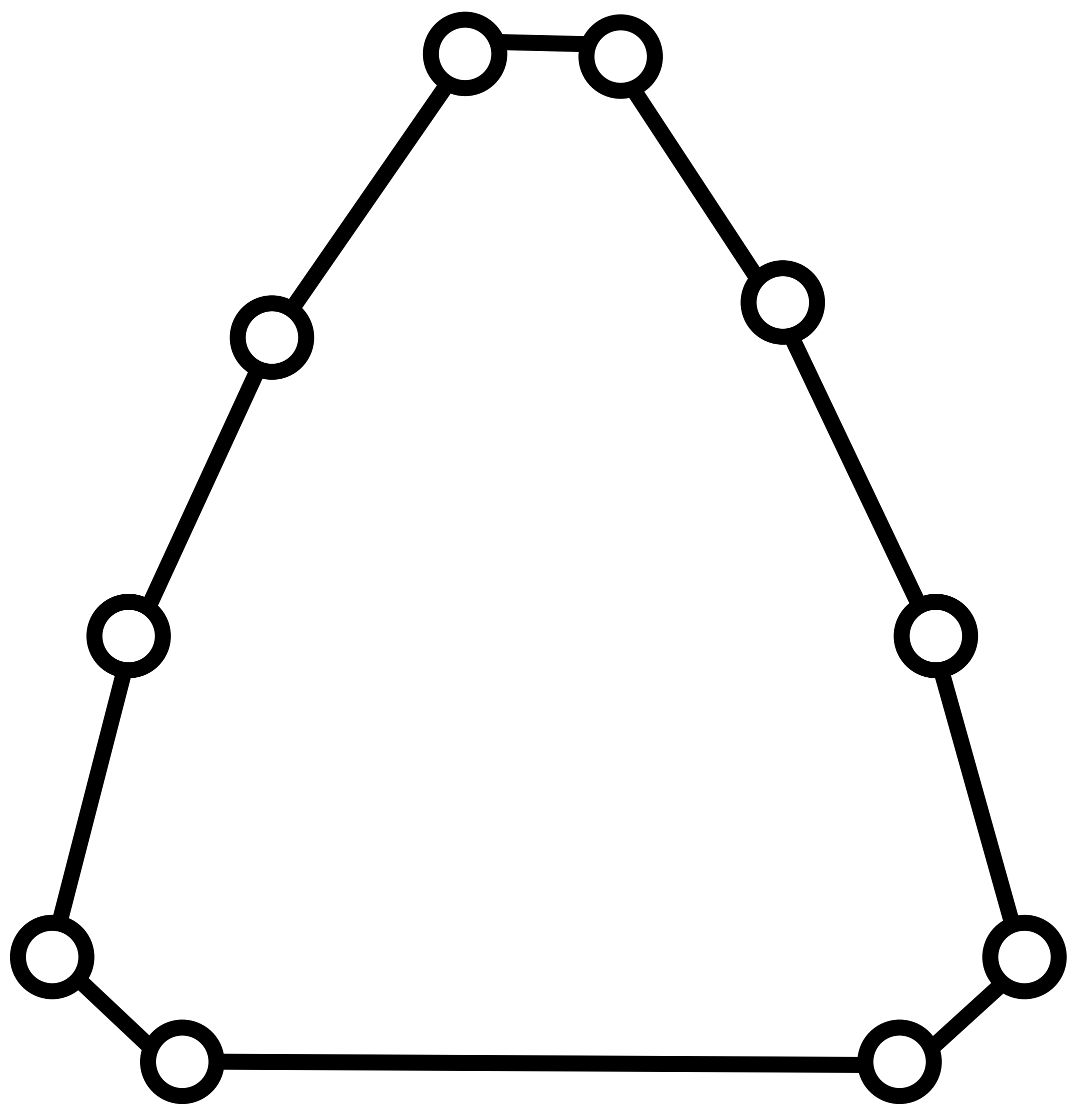


Compute New Error

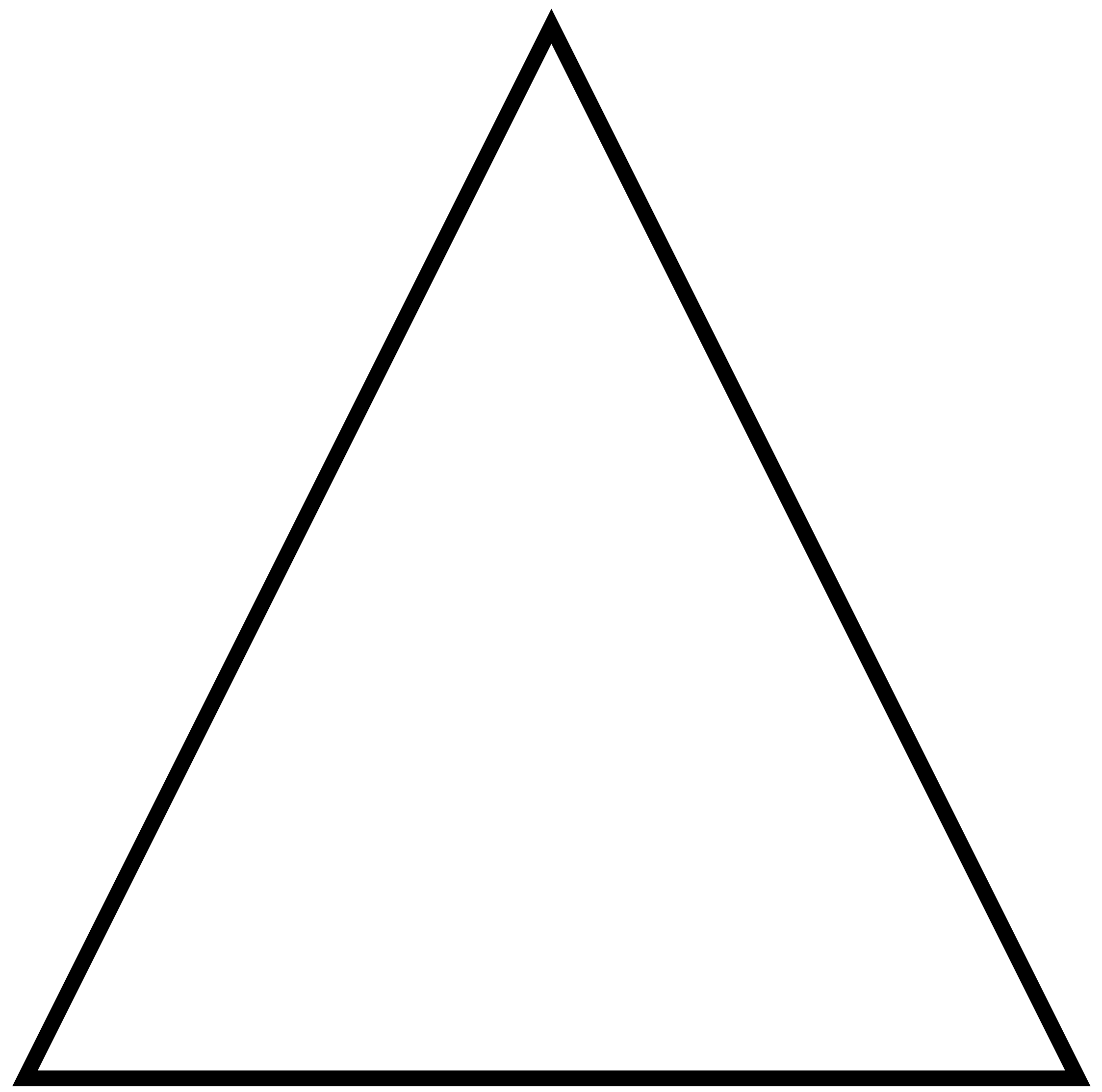


Target Geometry

Gradient Based Optimization

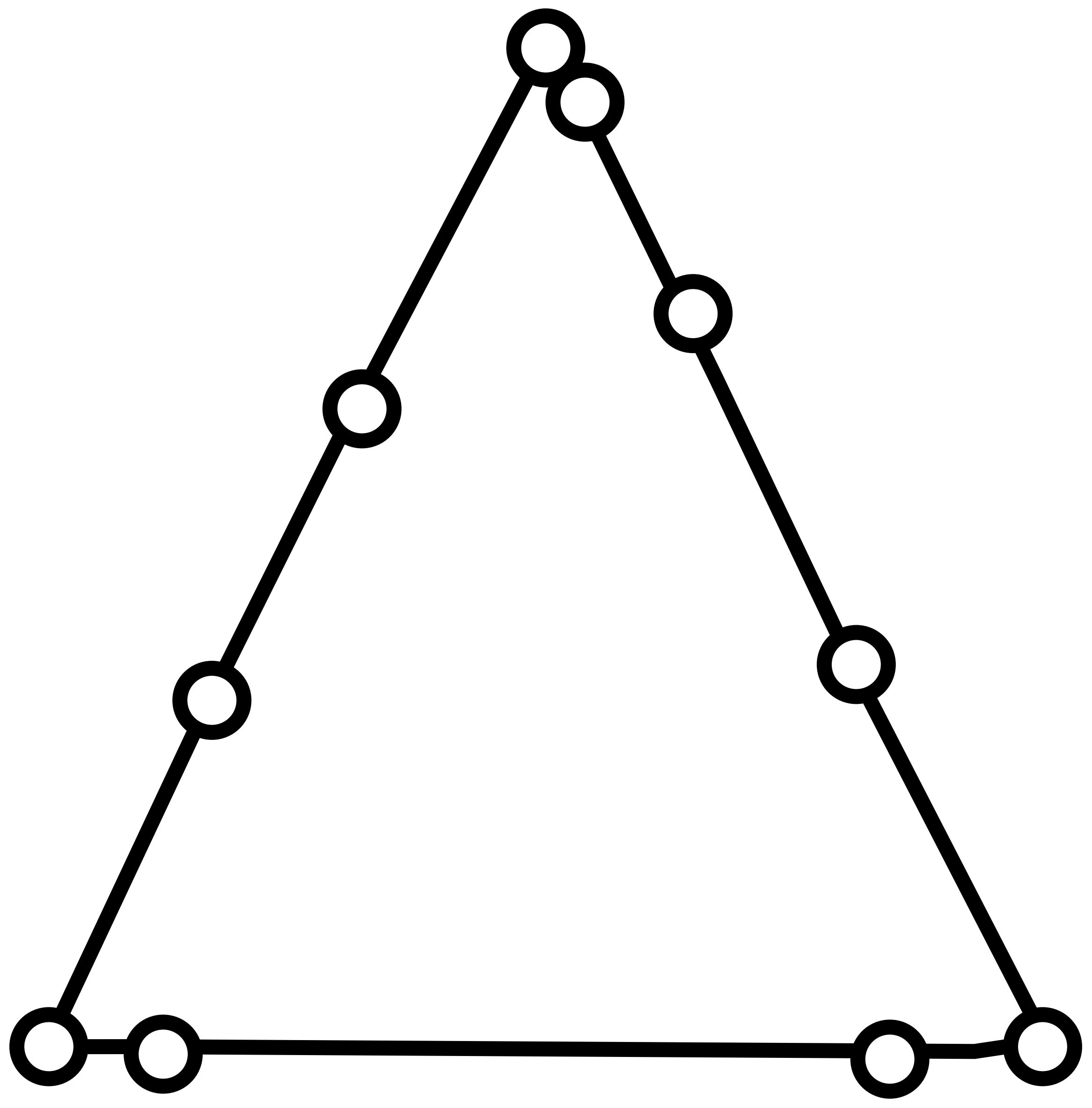


Repeat

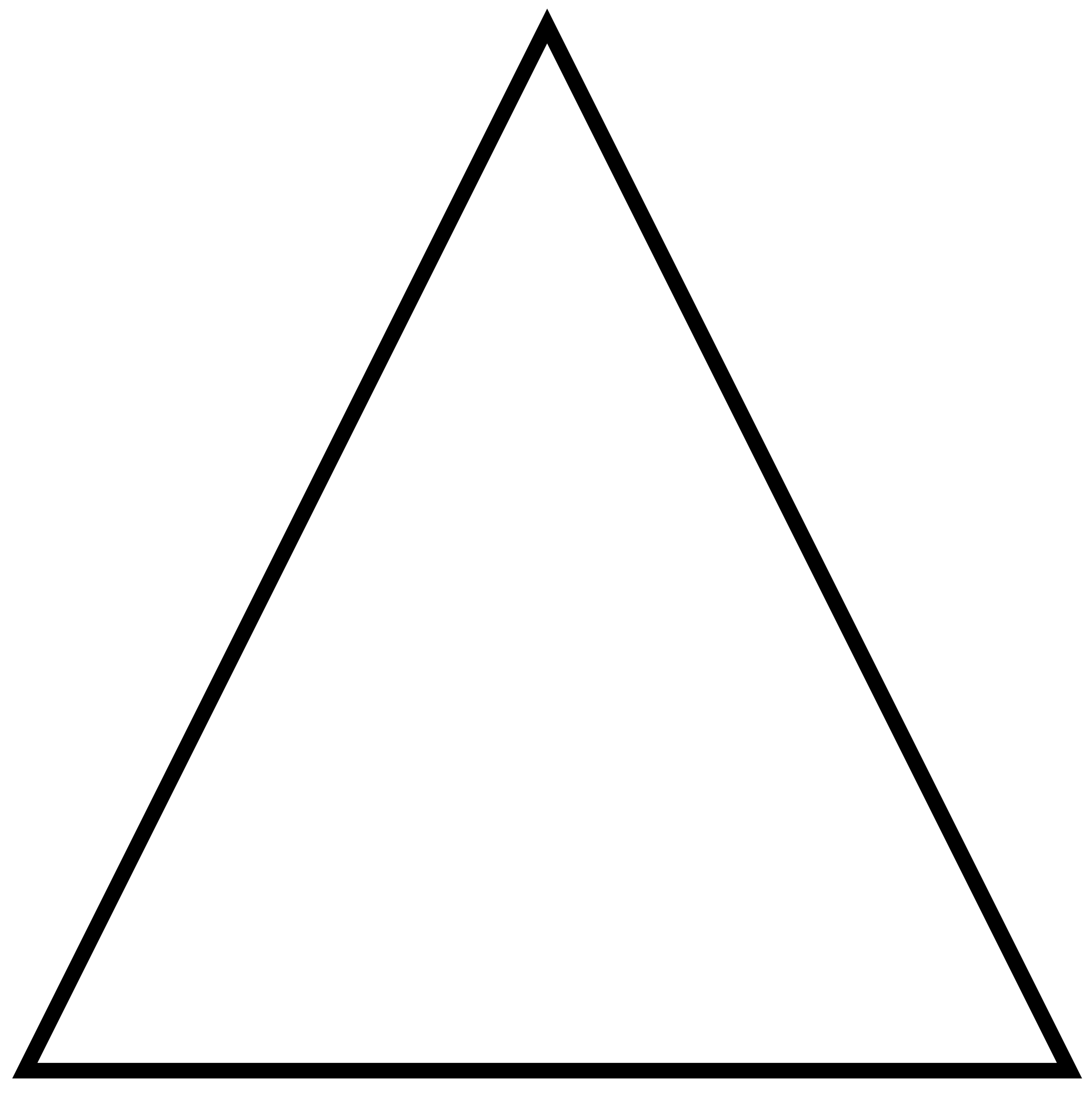


Target Geometry

Gradient Based Optimization

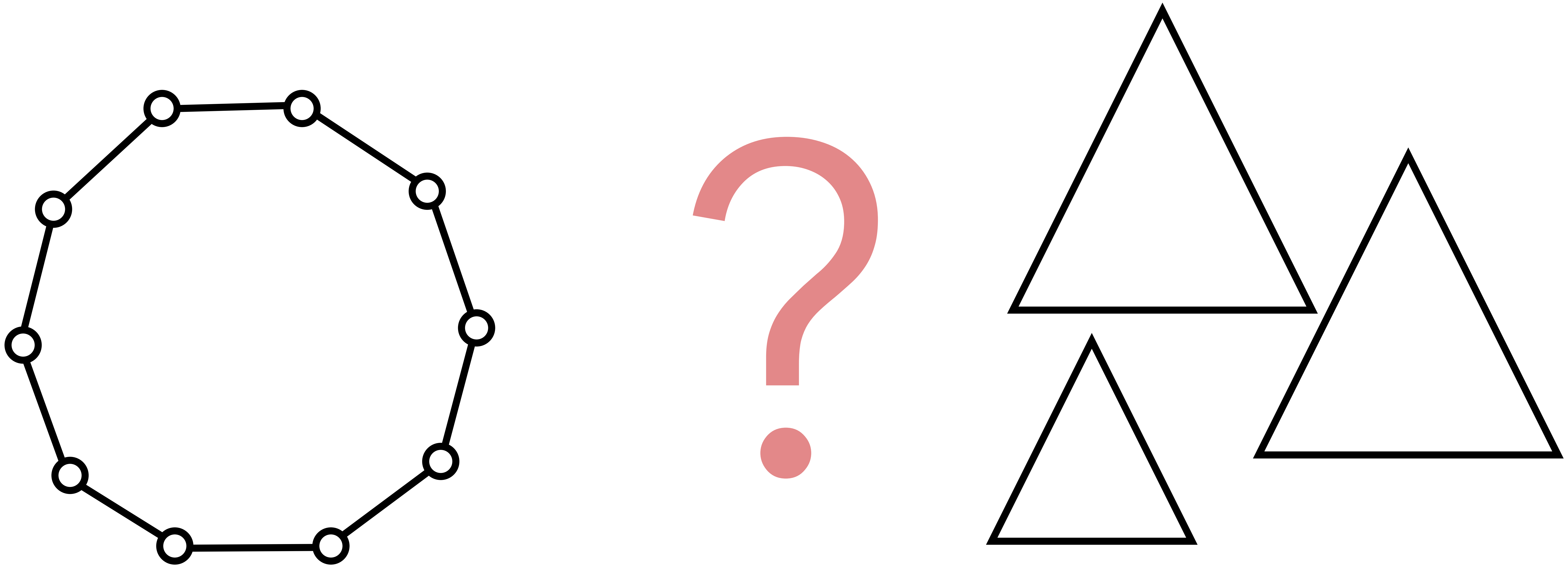


Repeat



Target Geometry

Gradient Based Optimization



Initial Geometry

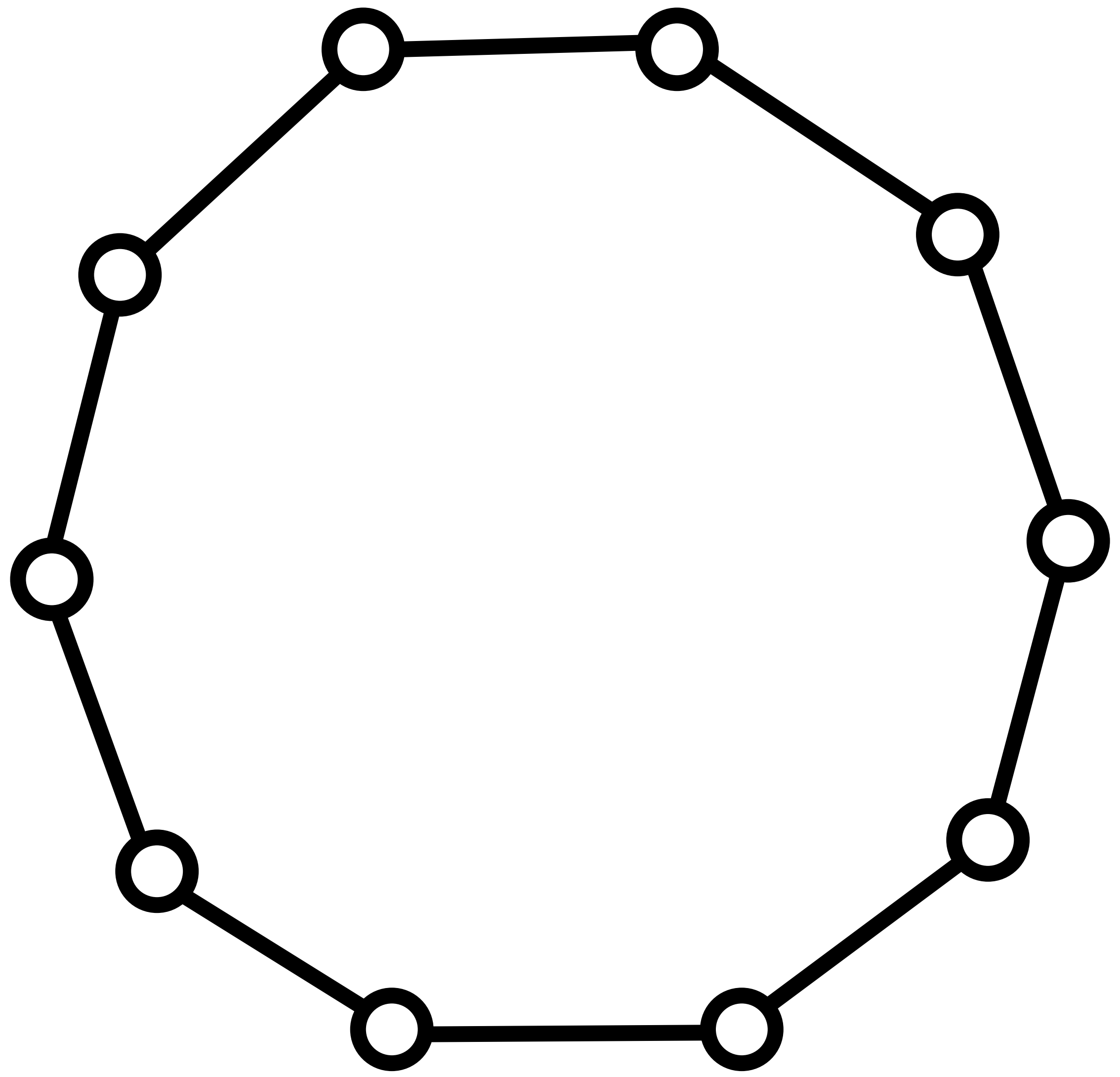
Target Geometry

Gradient Based Optimization

Initial Geometry

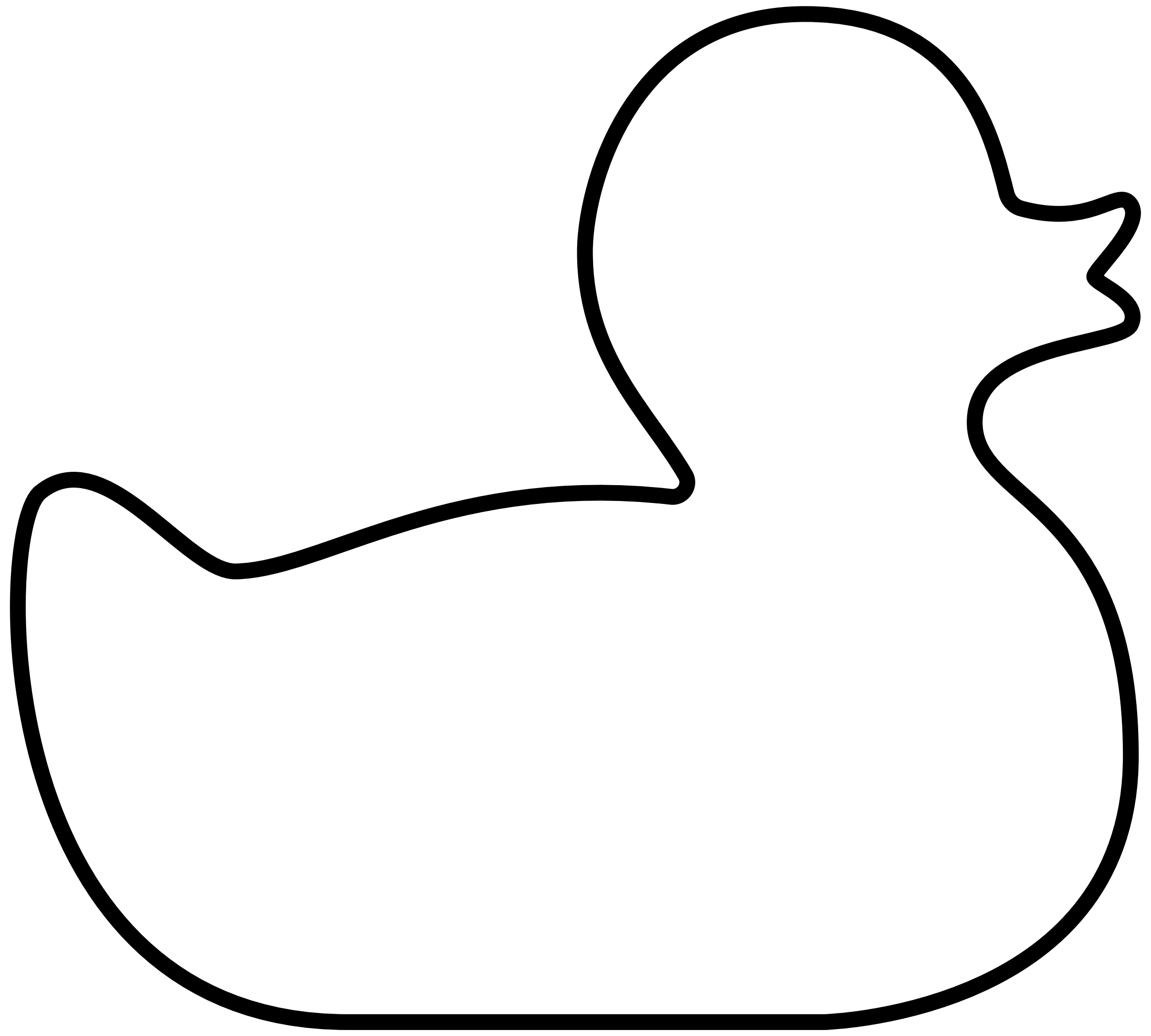
Target Geometry

Gradient Based Optimization



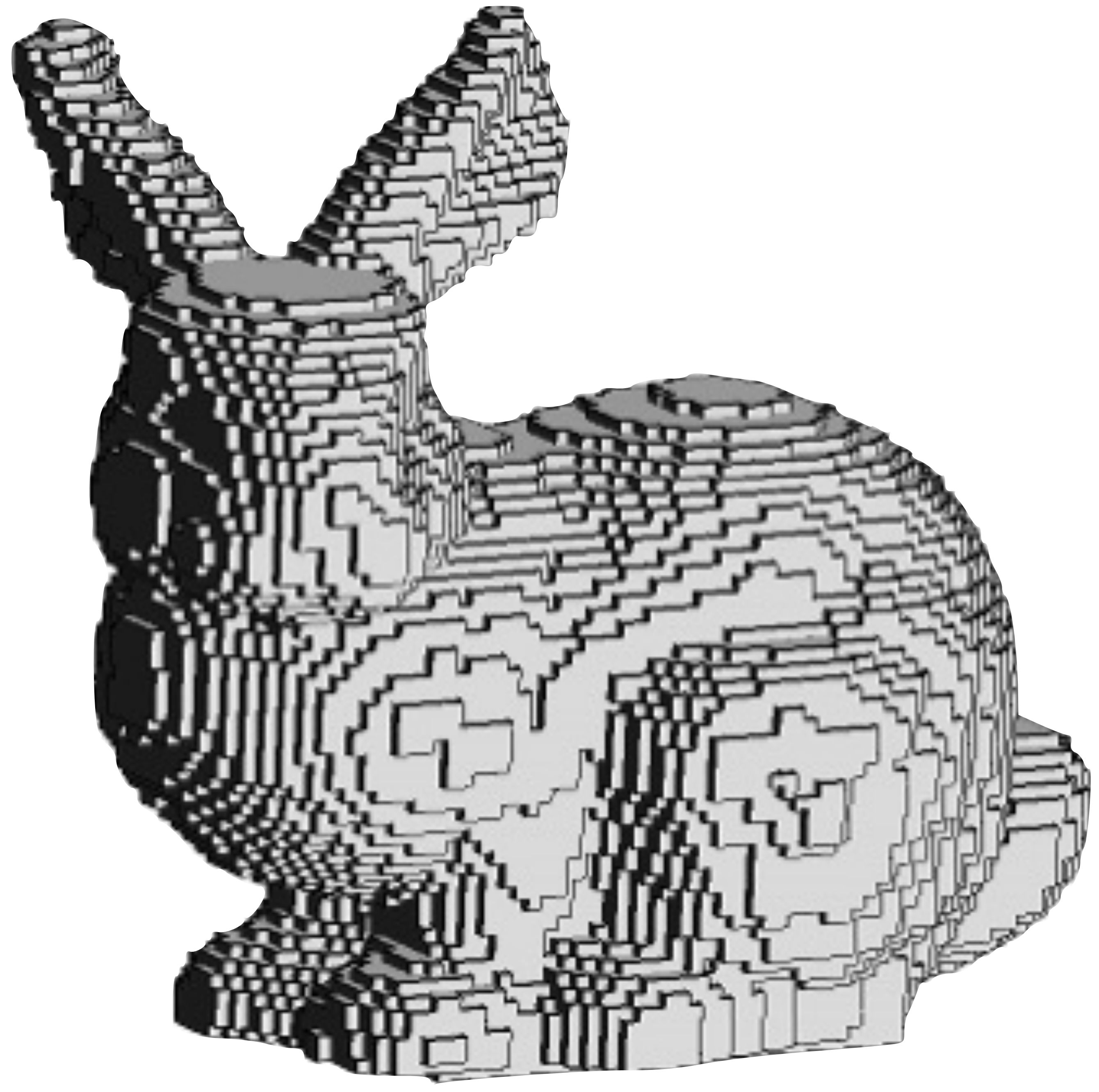
Initial Geometry

?

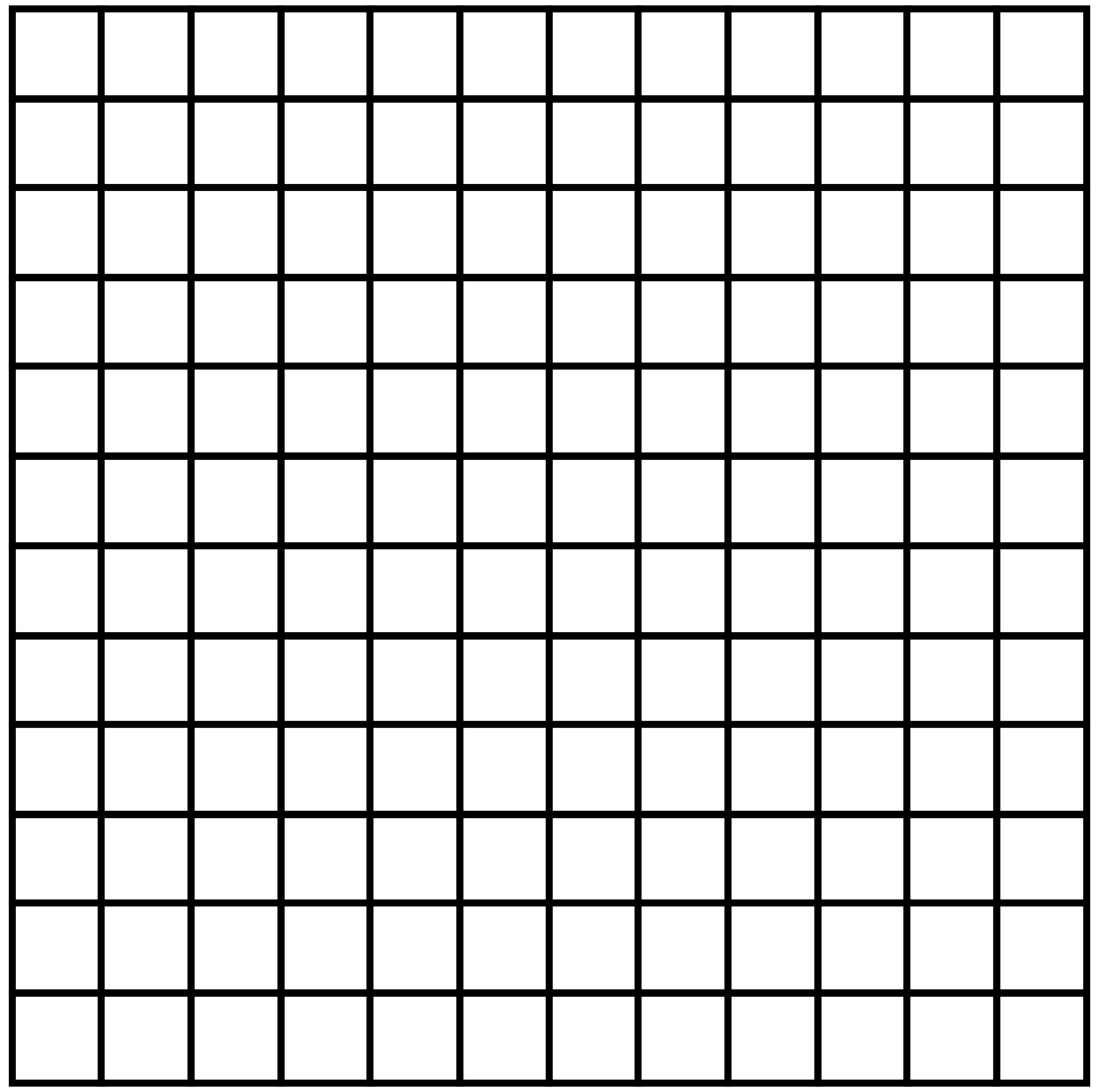


Target Geometry

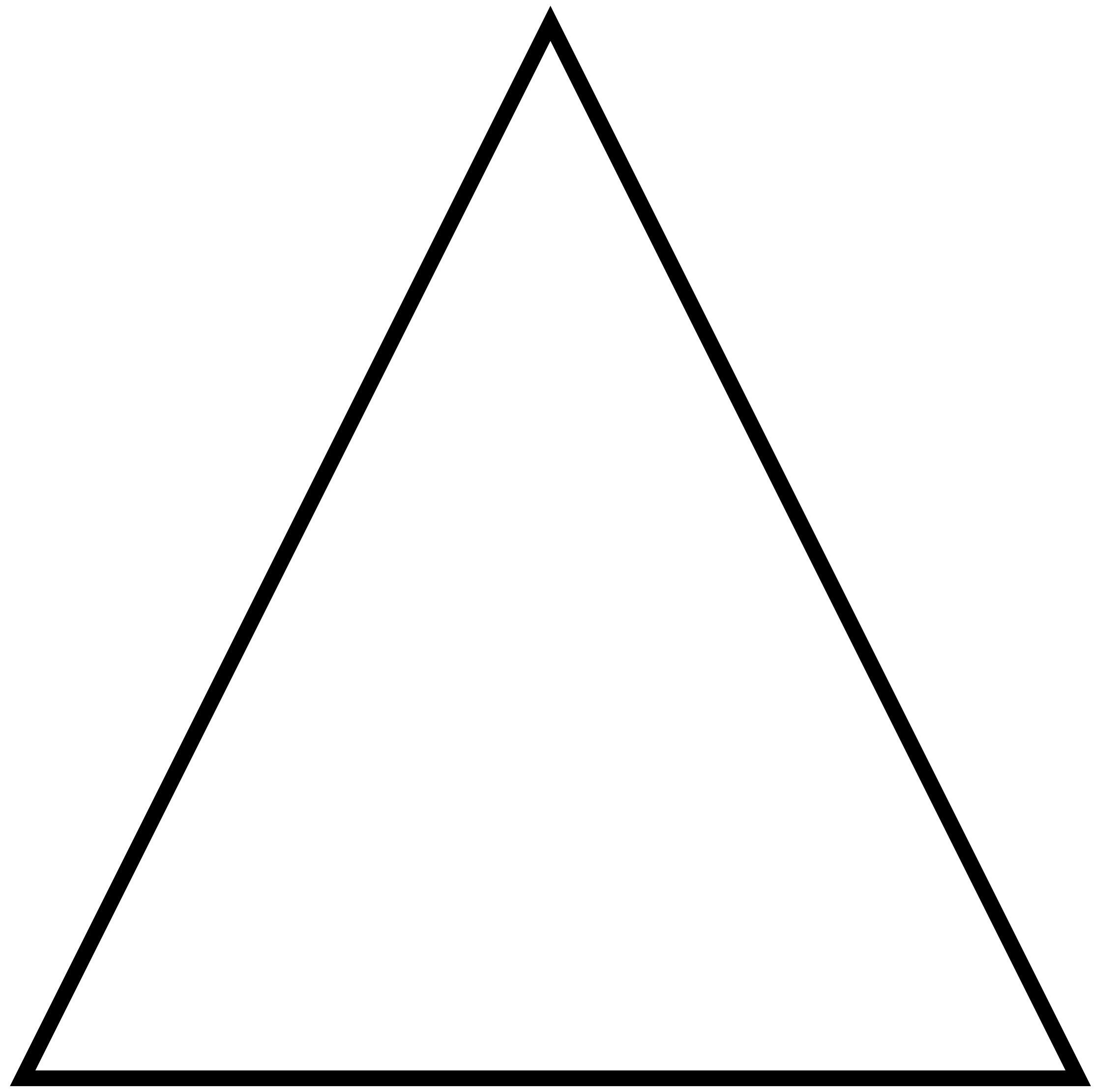
Voxel Representation



Gradient Based Optimization

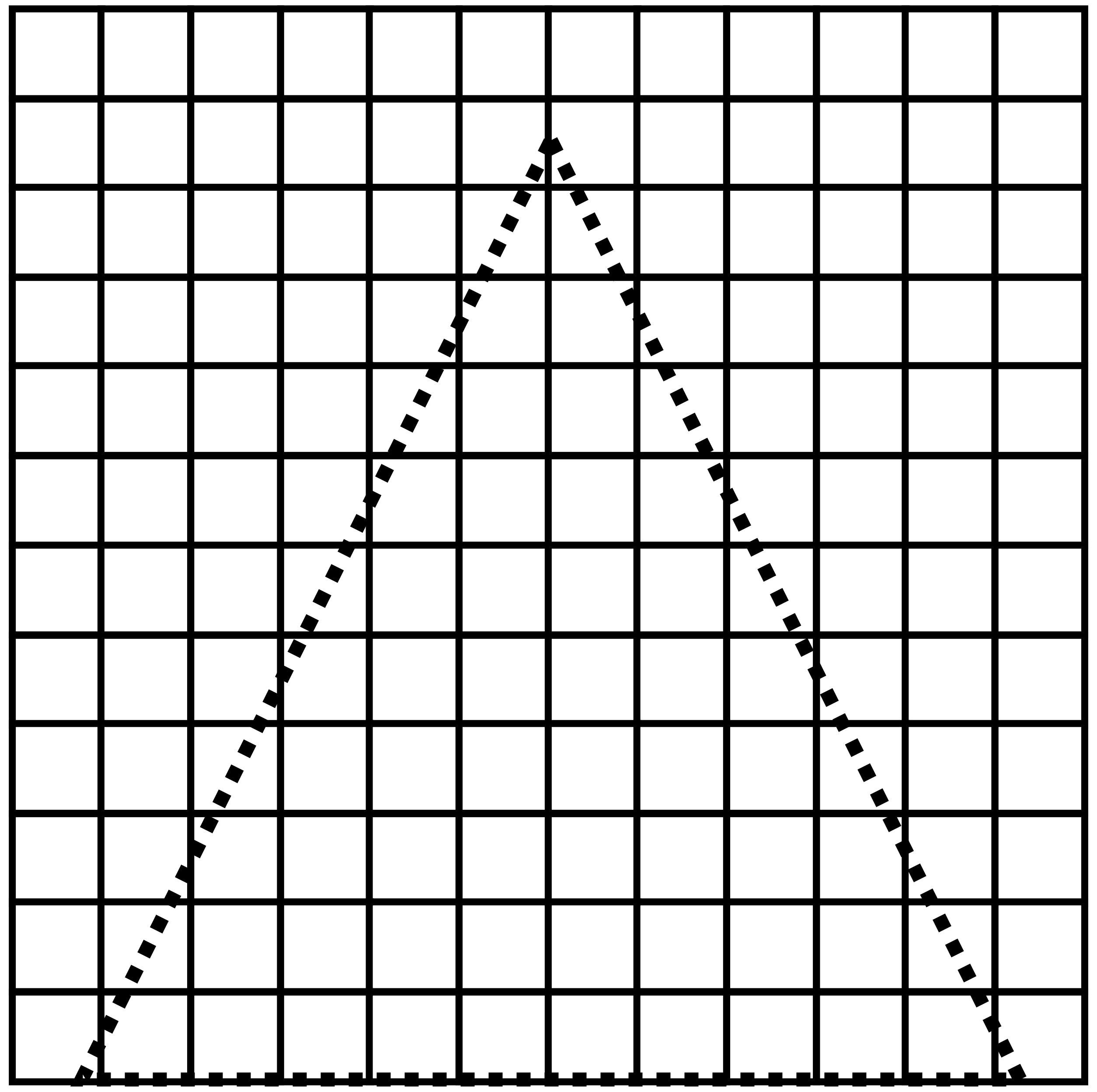


Initialized Grid

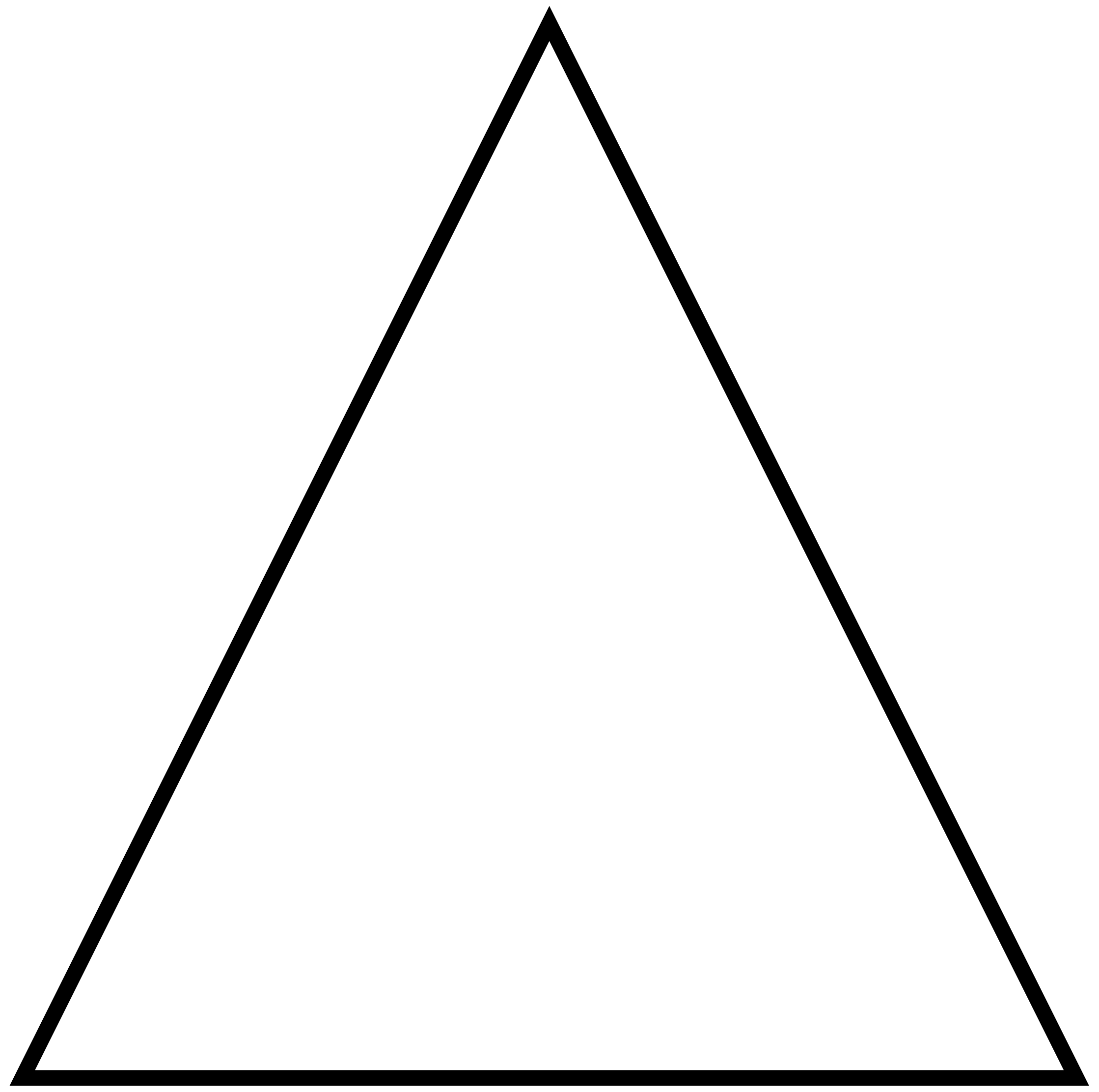


Target Geometry

Gradient Based Optimization

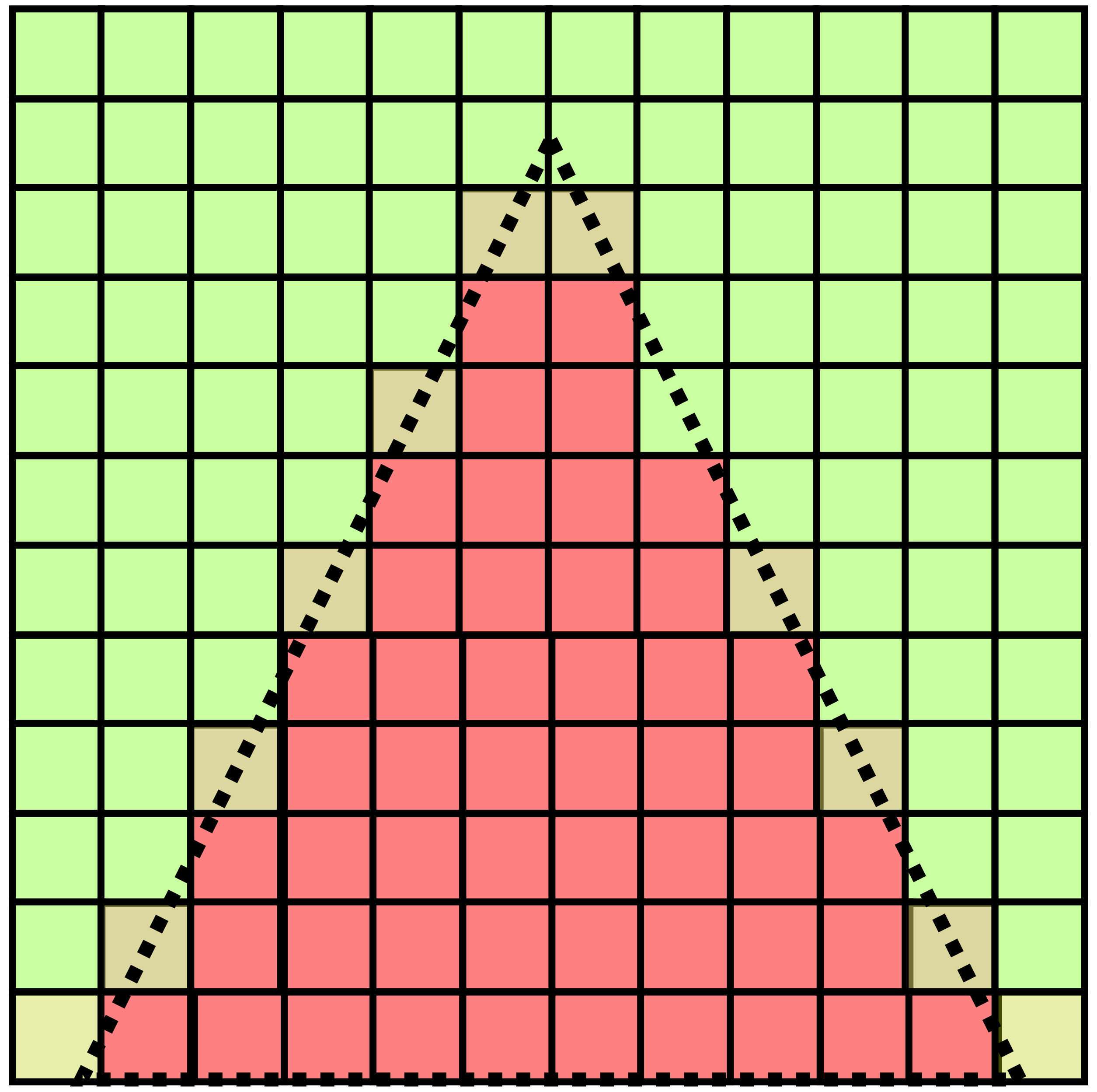


Initialized Grid

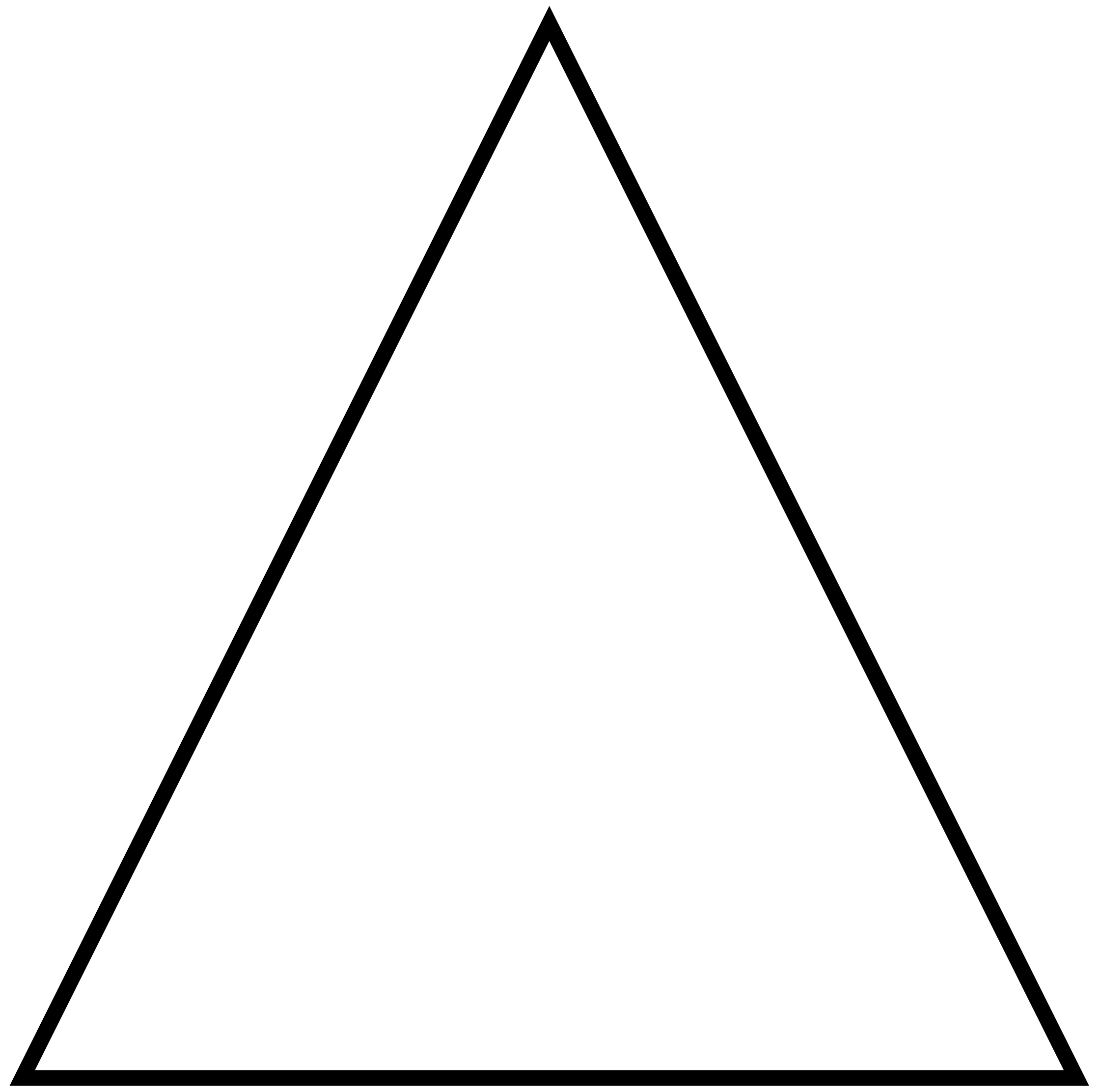


Target Geometry

Gradient Based Optimization

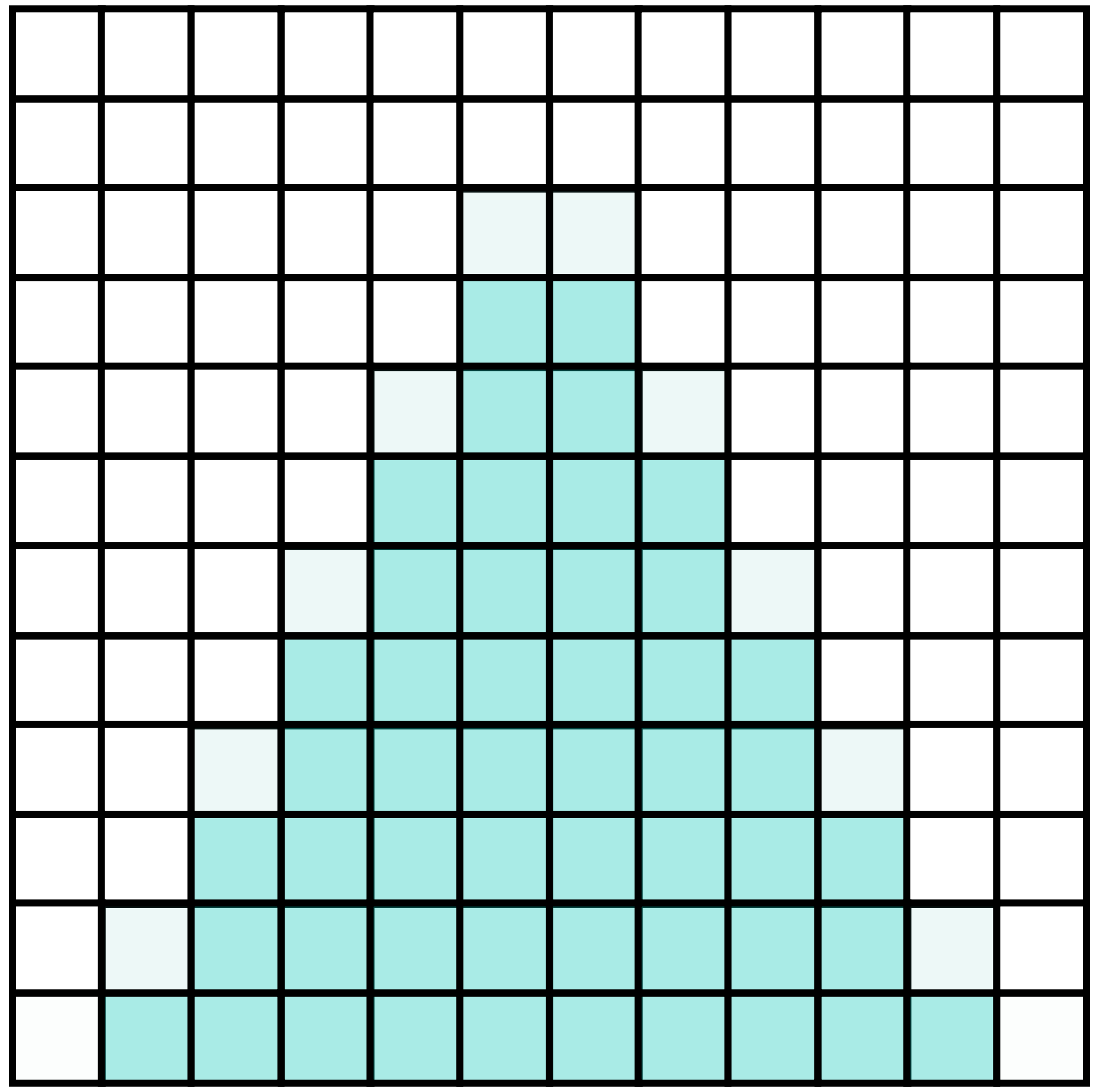


Loss

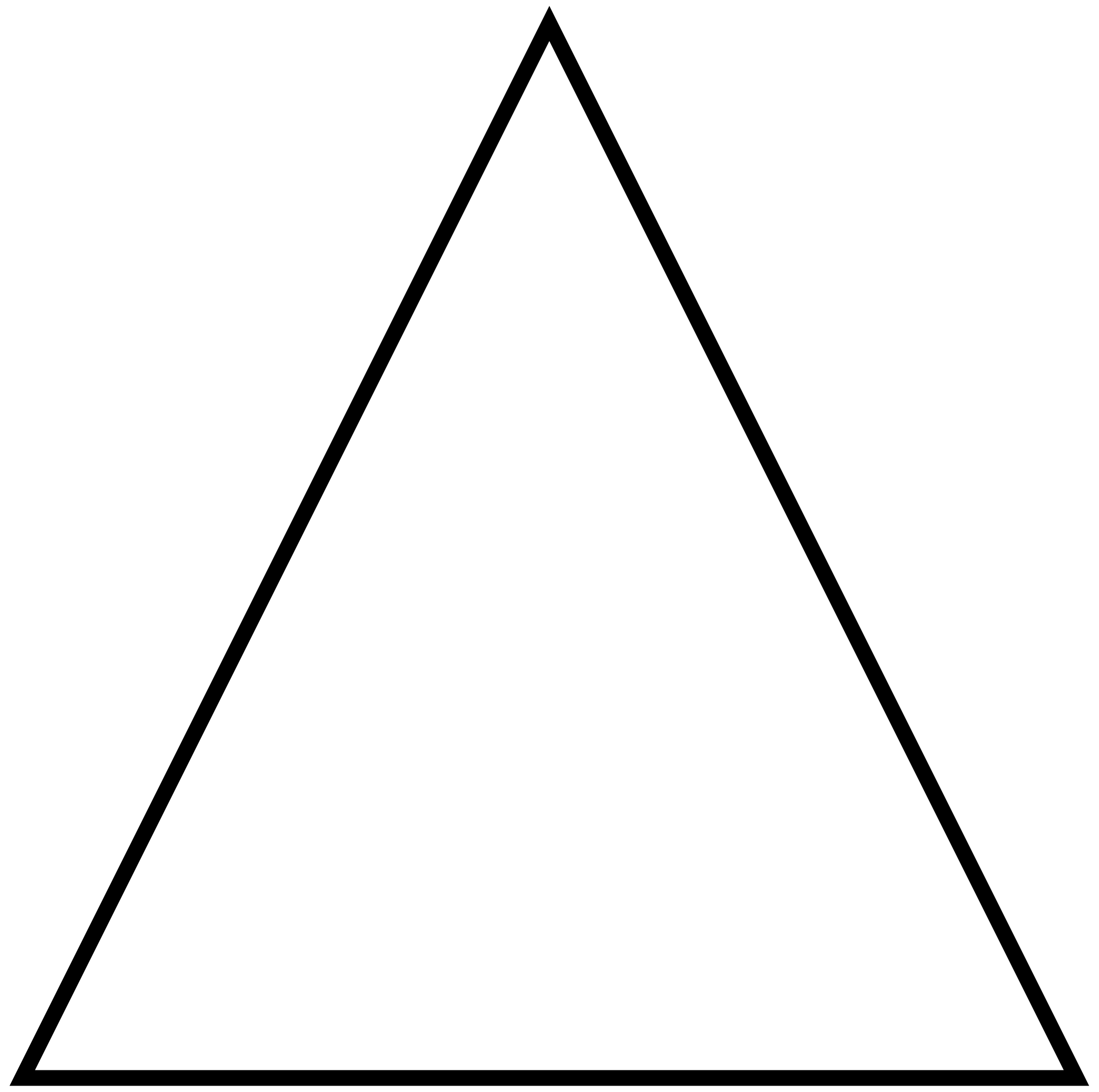


Target Geometry

Gradient Based Optimization

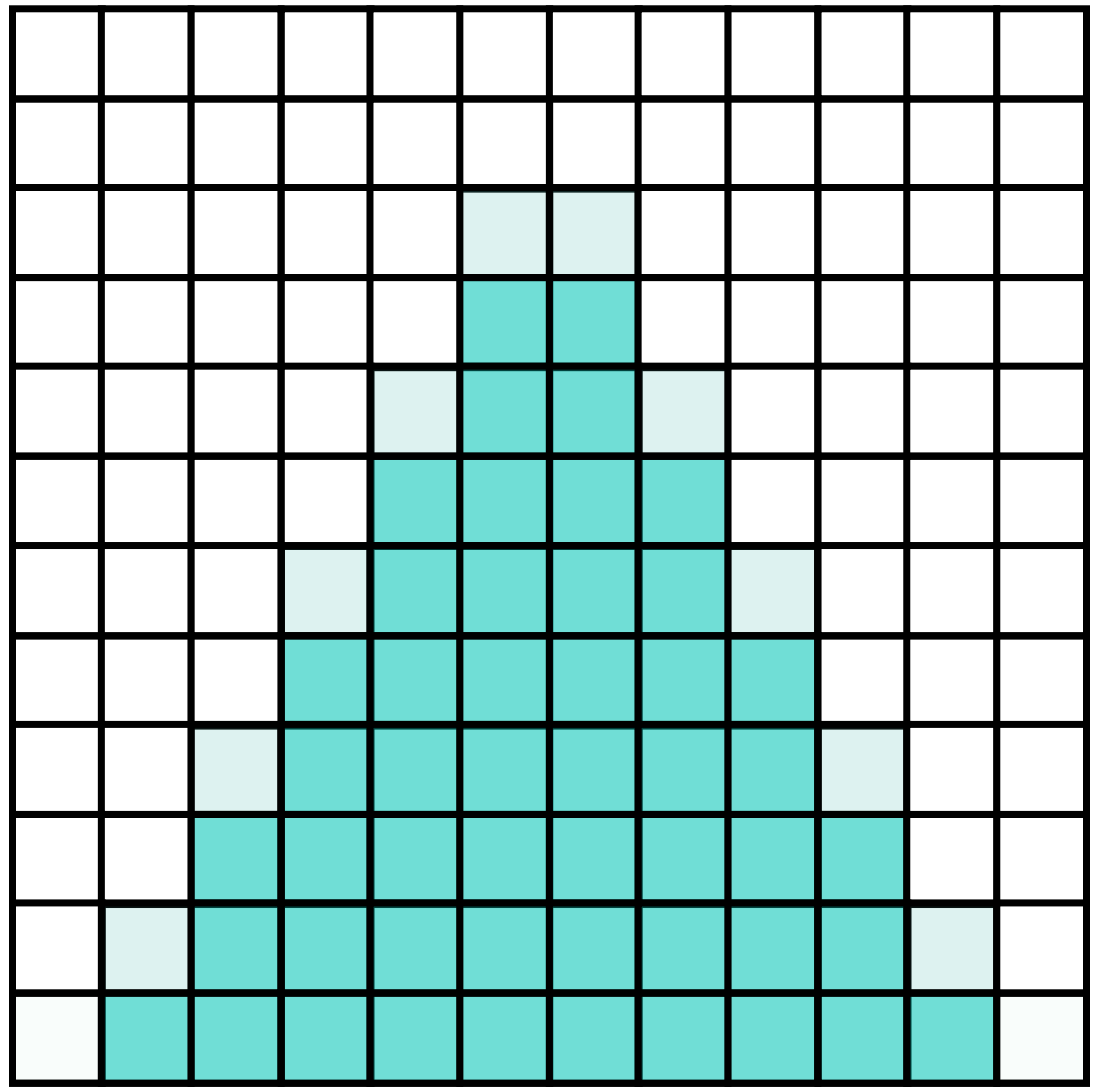


Gradient Step

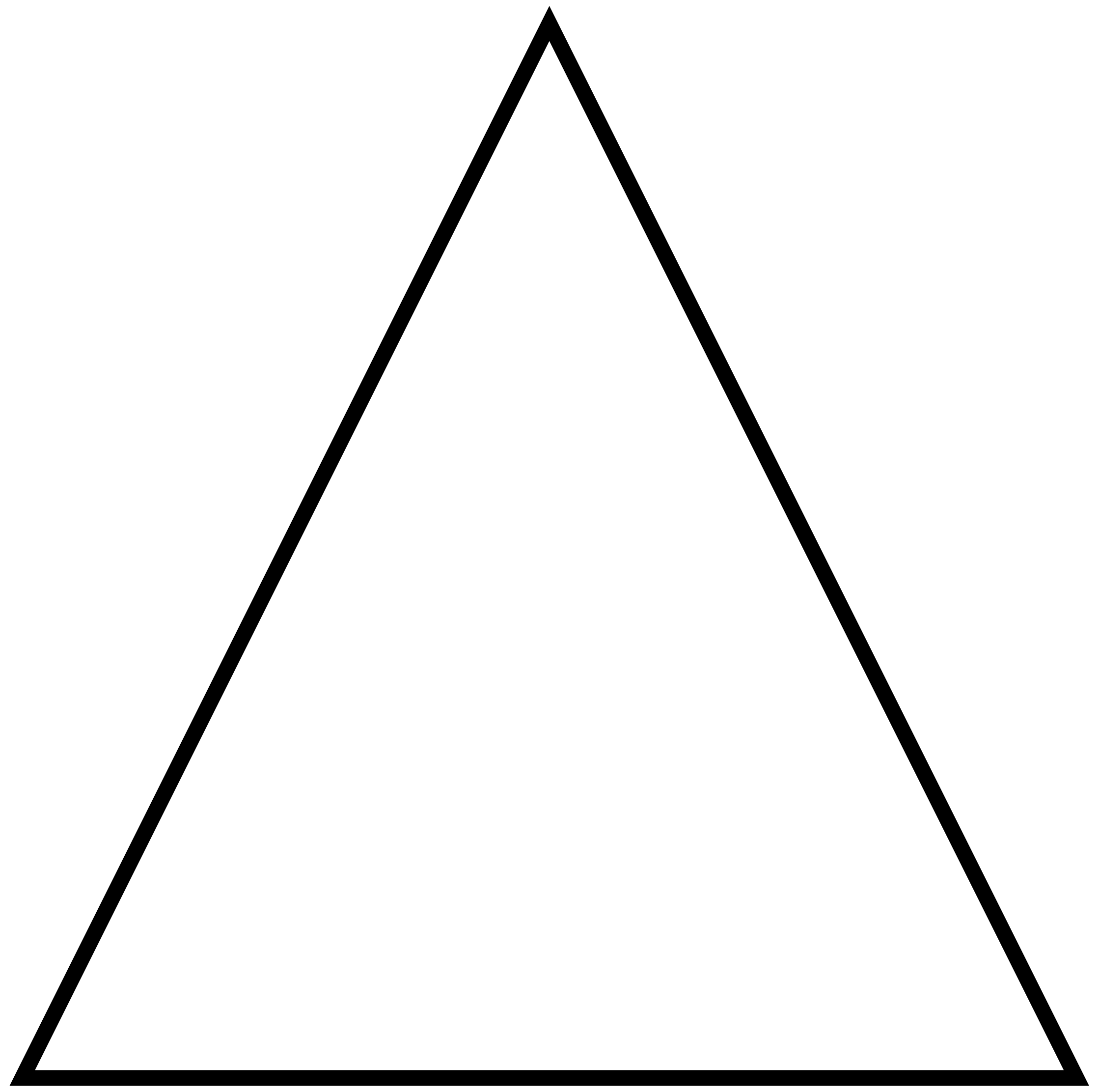


Target Geometry

Gradient Based Optimization

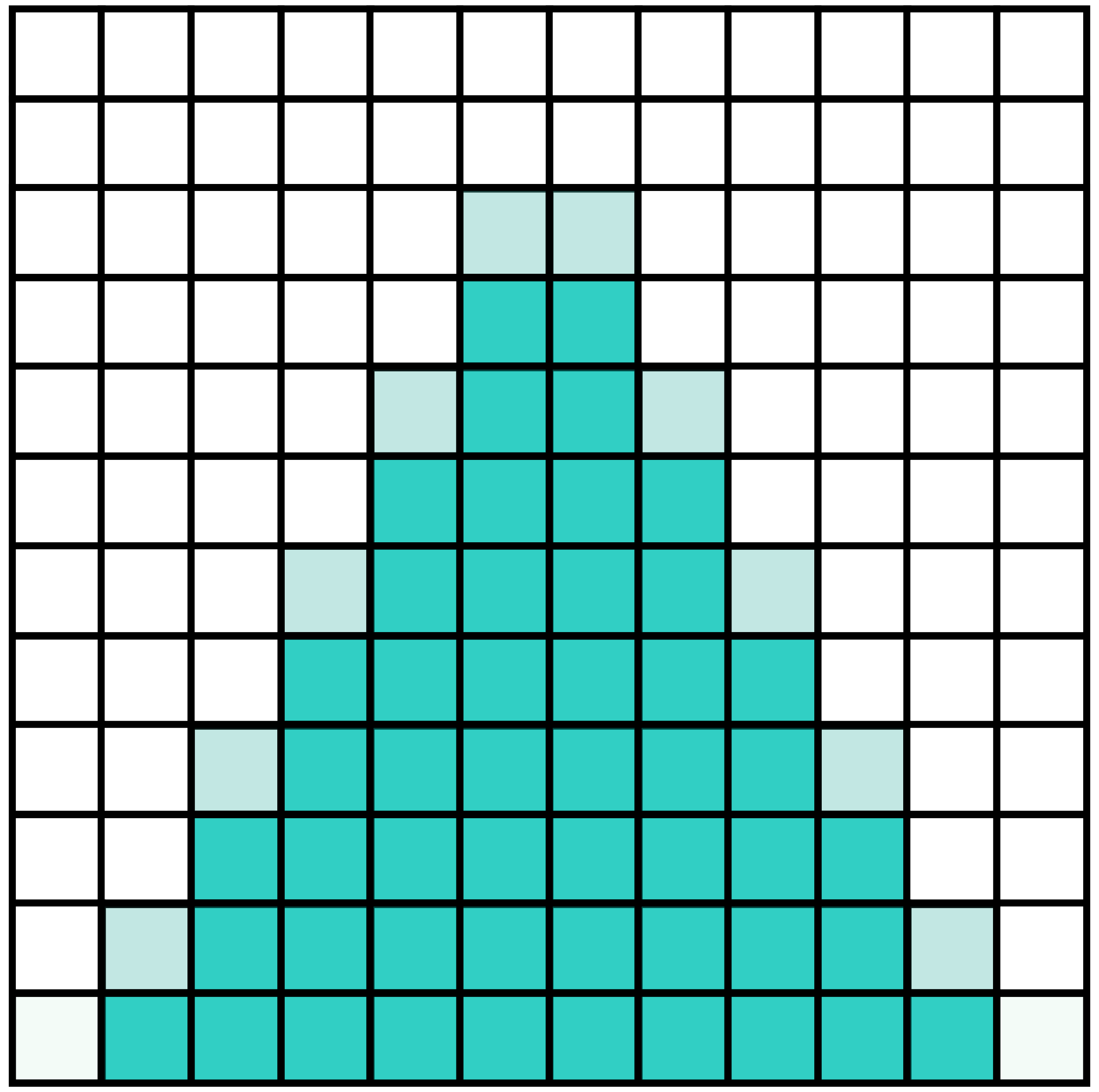


Repeat

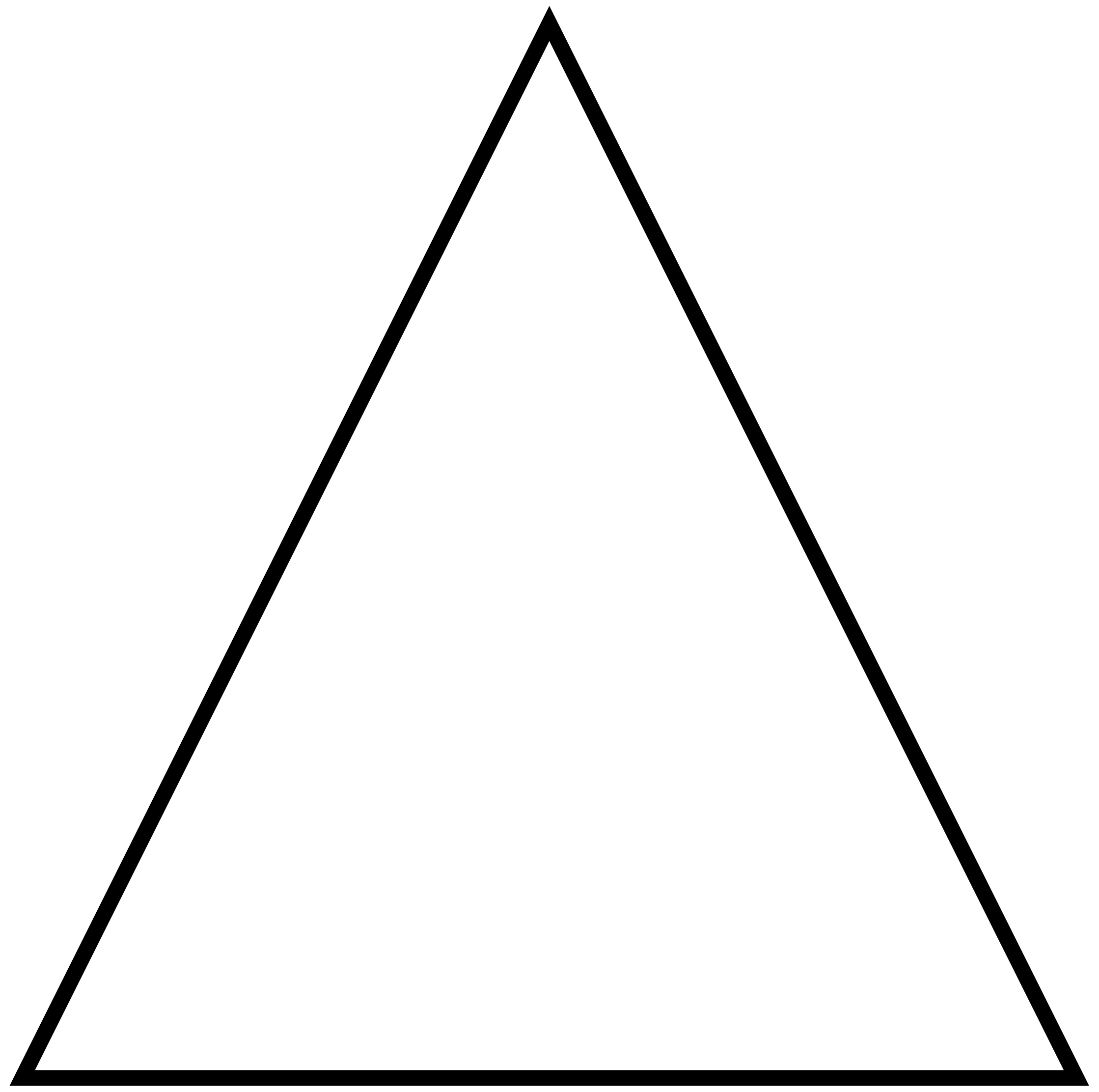


Target Geometry

Gradient Based Optimization

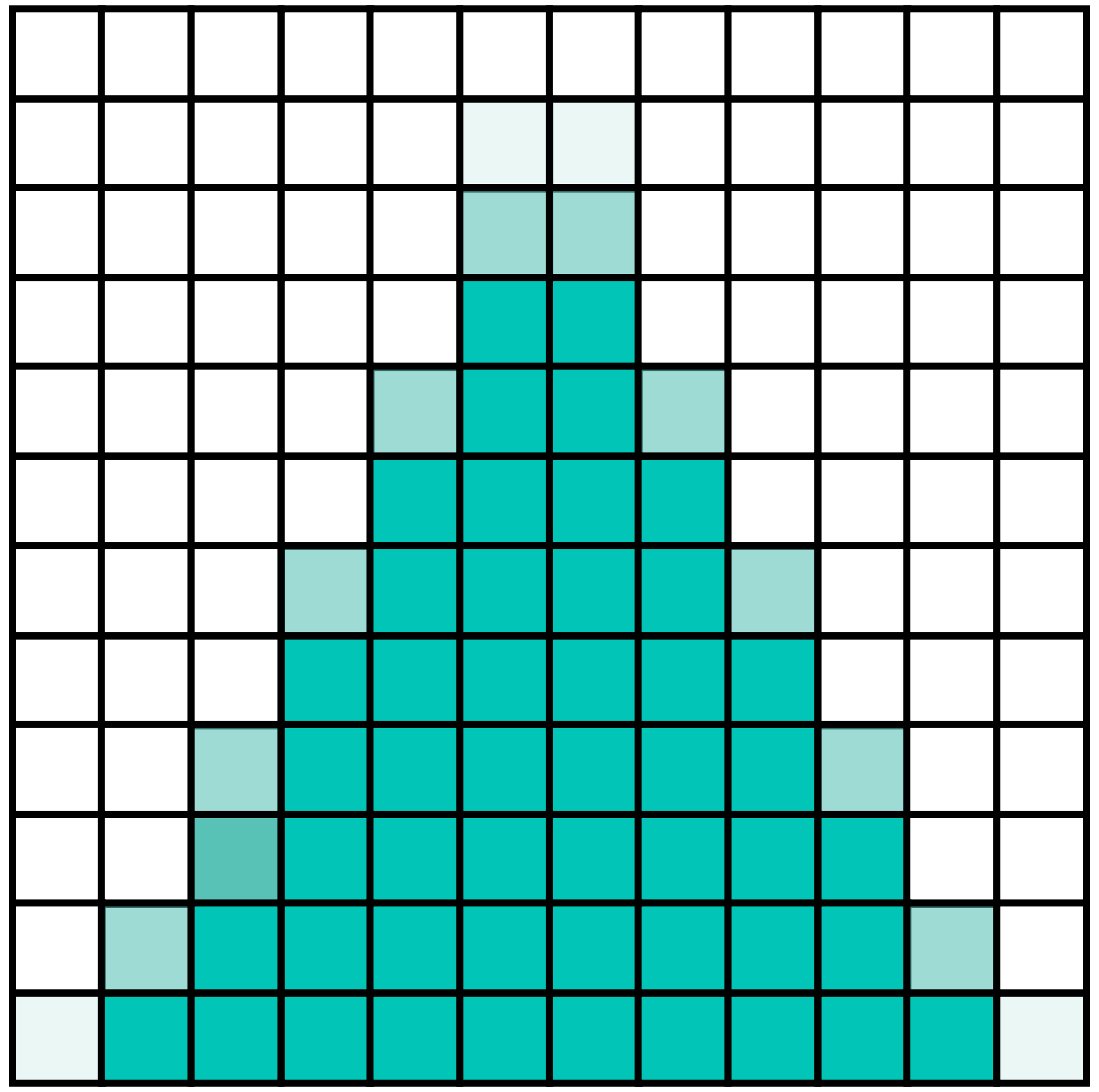


Repeat

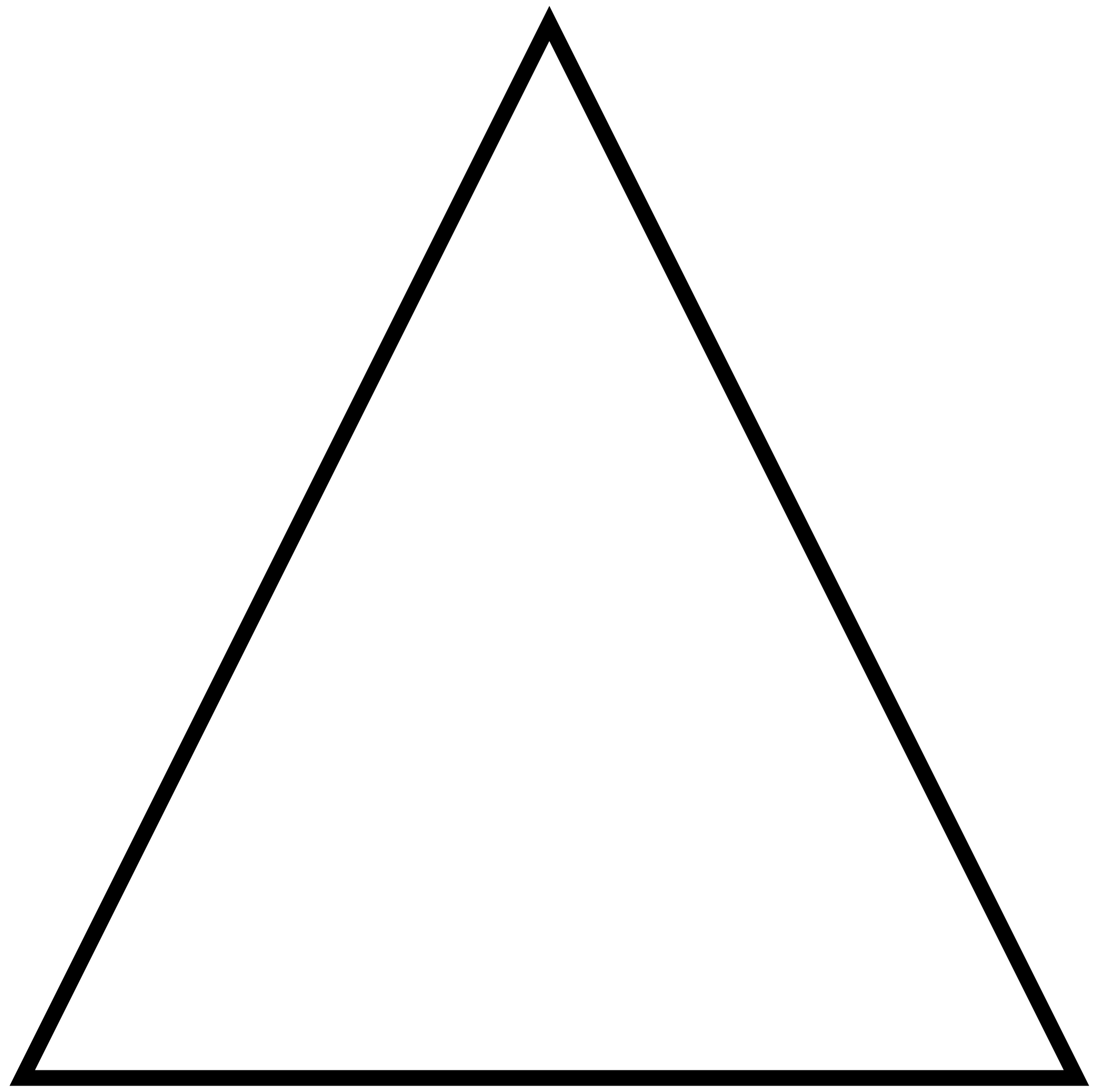


Target Geometry

Gradient Based Optimization

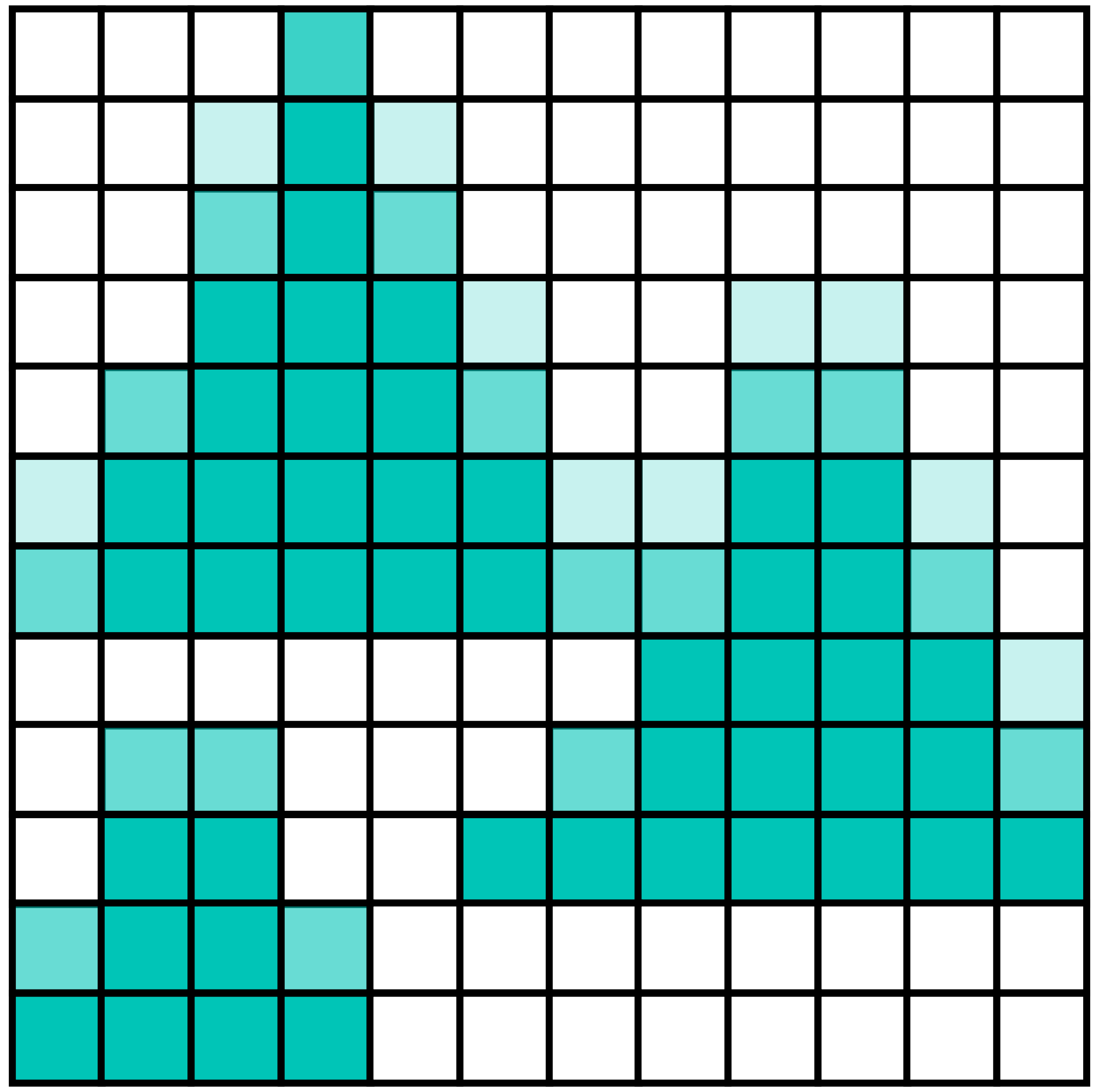


Repeat

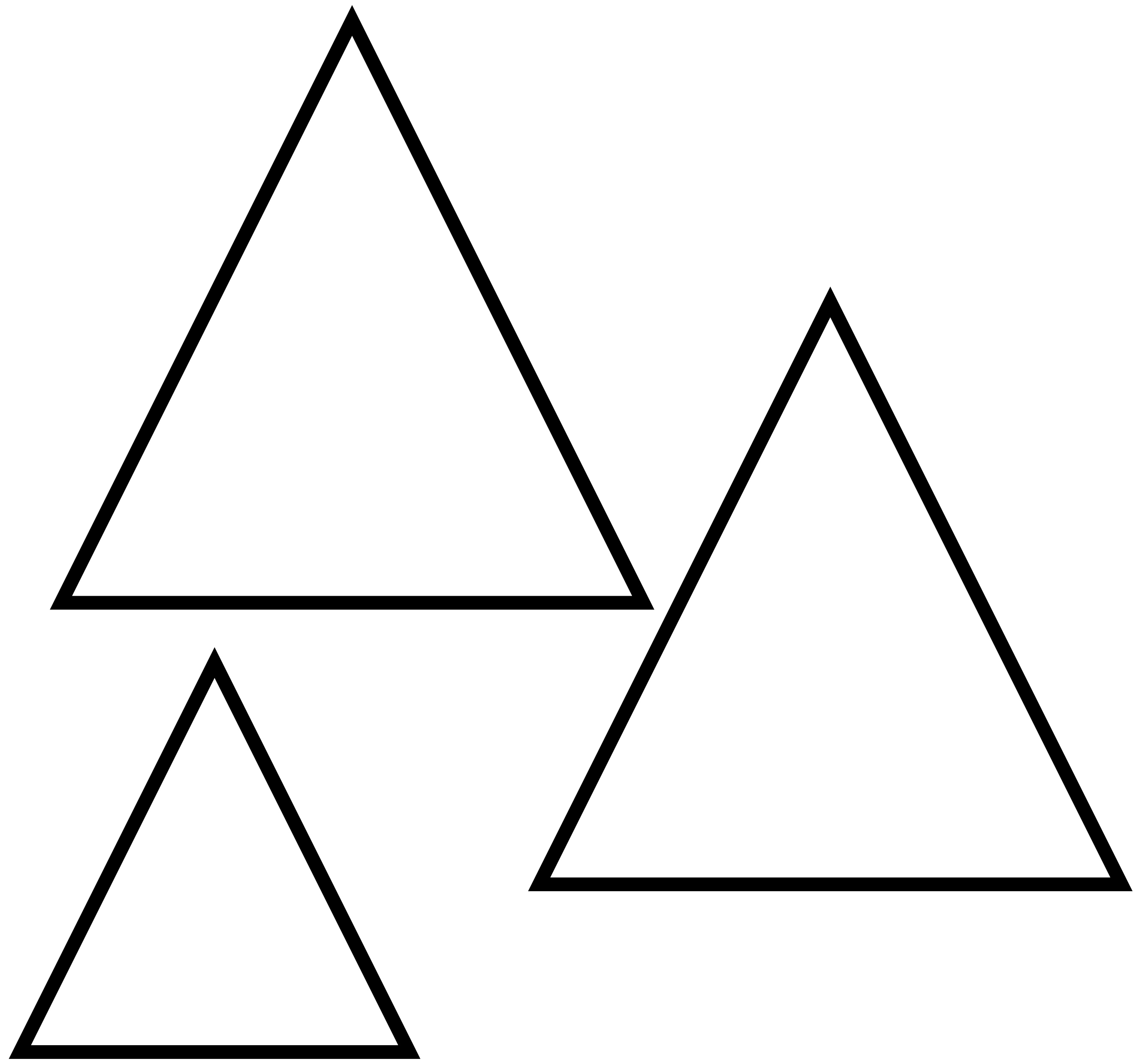


Target Geometry

Gradient Based Optimization

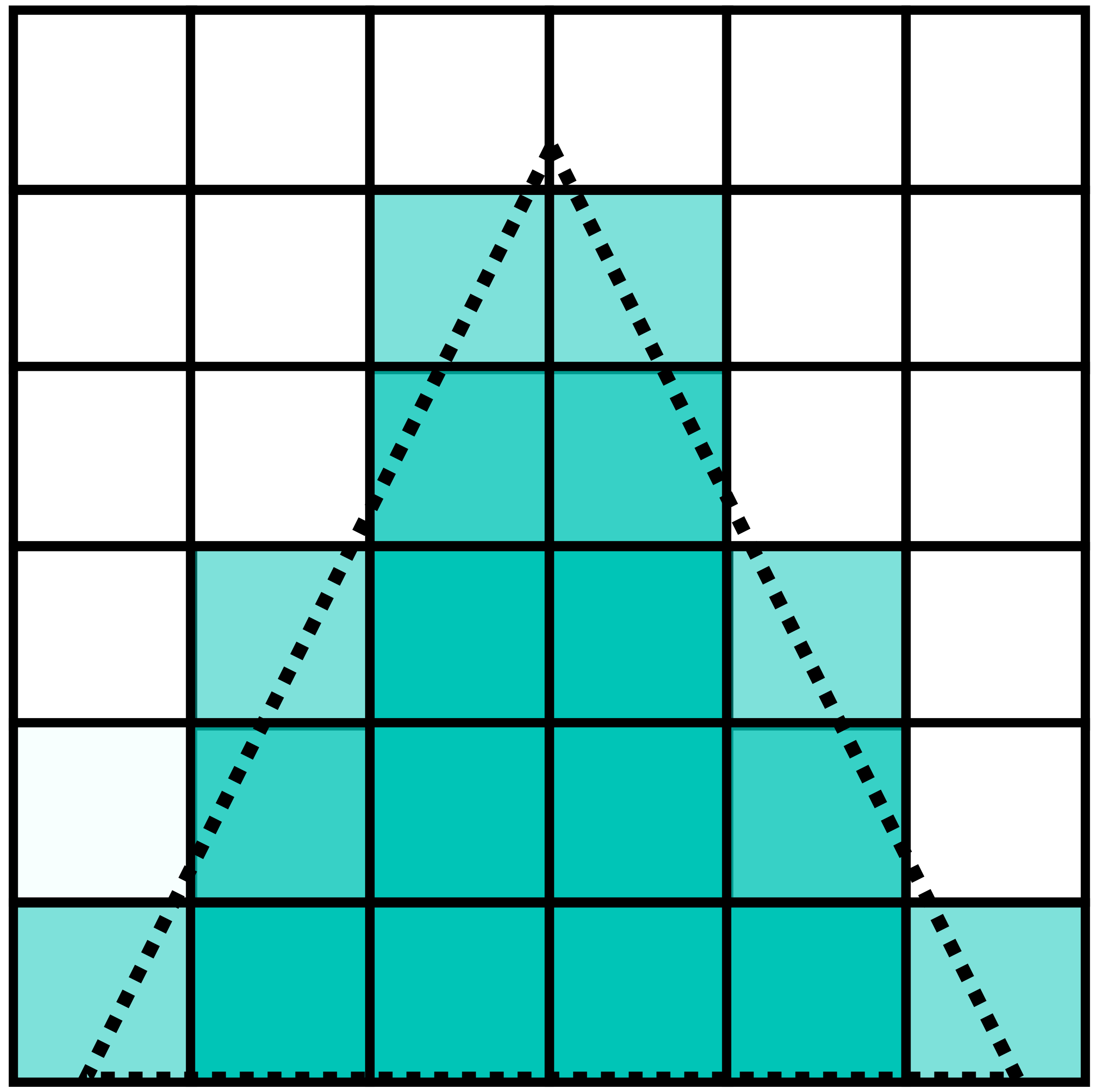


Reconstruction

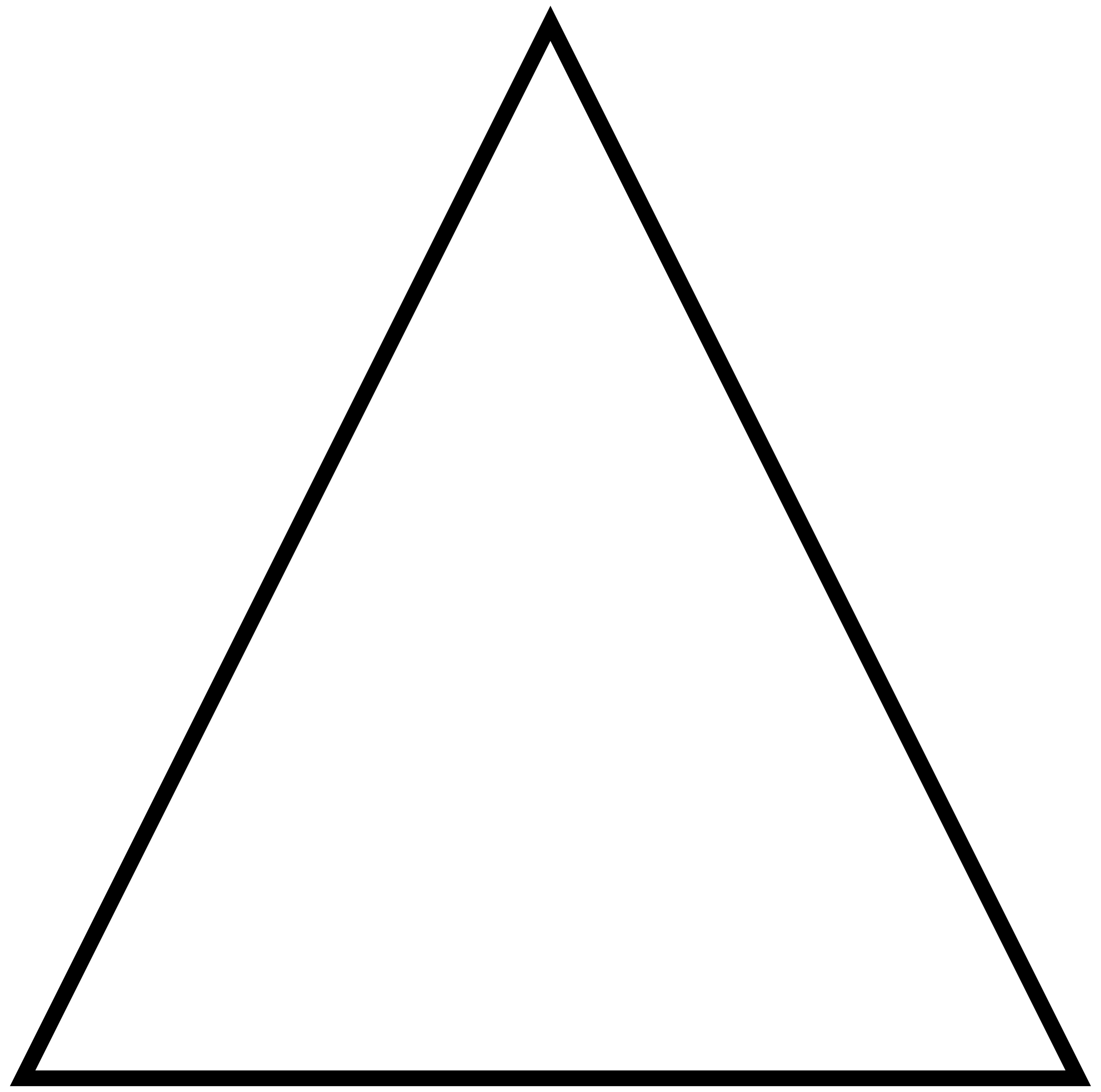


Target Geometry

Gradient Based Optimization

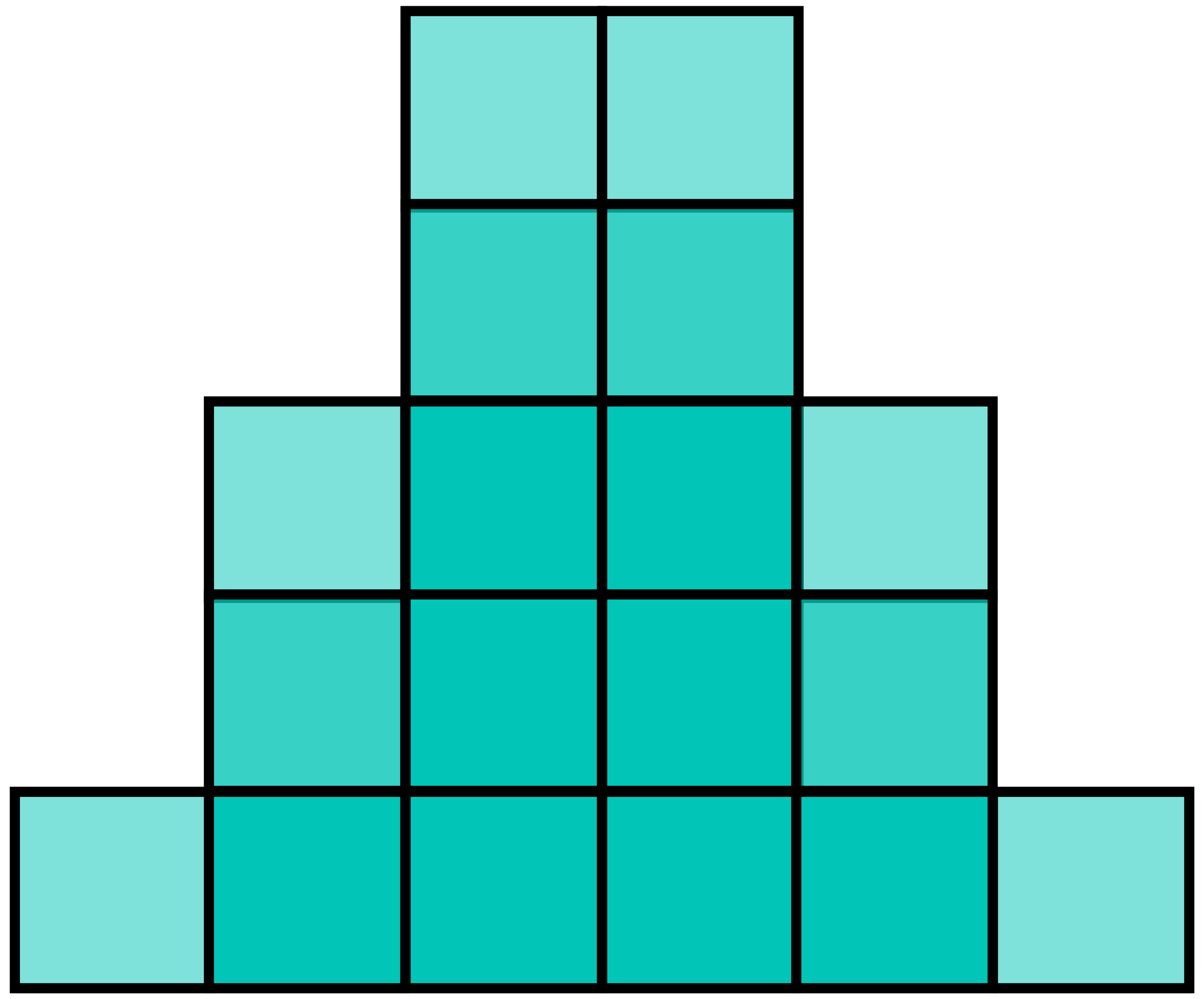


Reconstruction

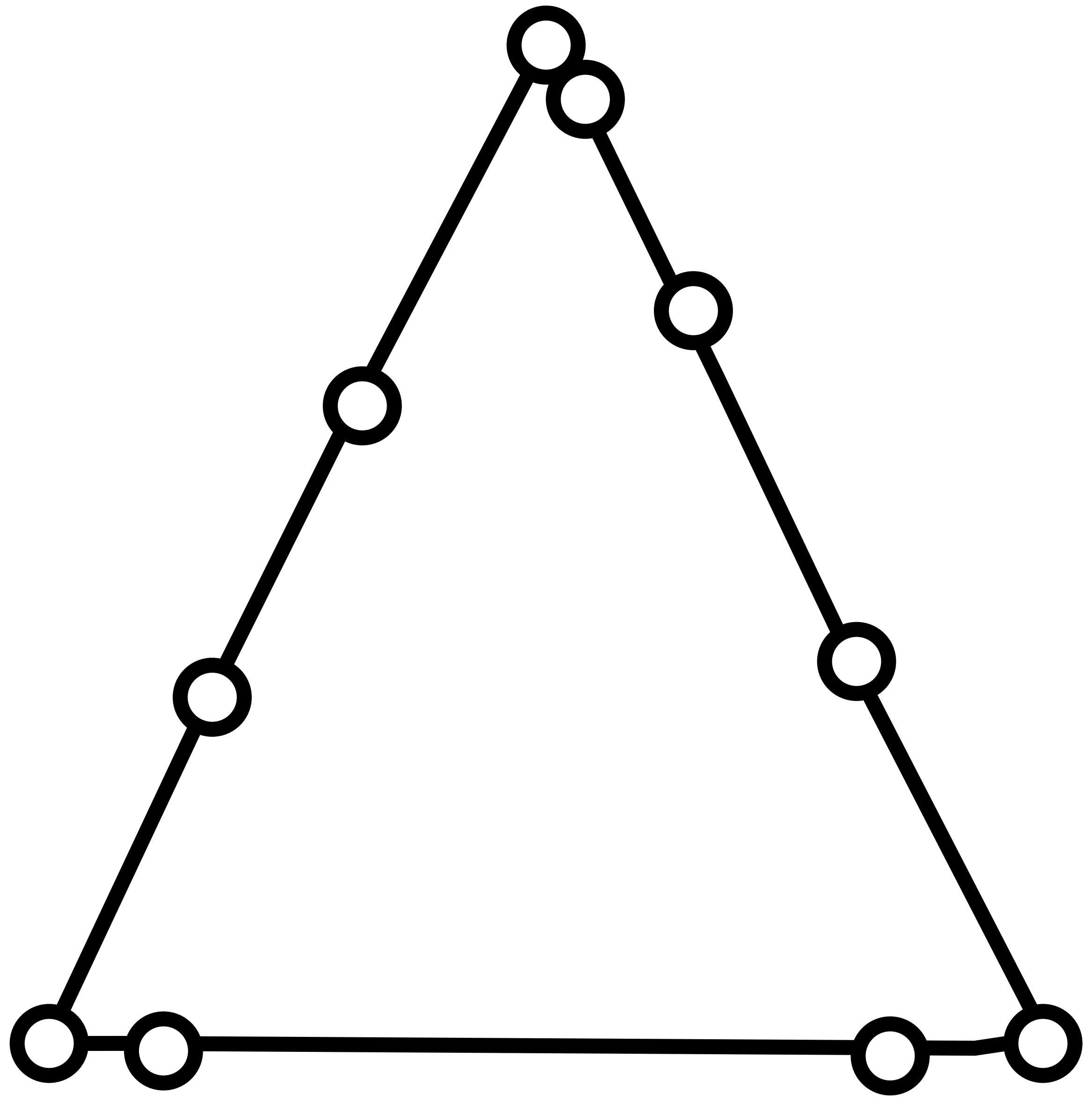


Target Geometry

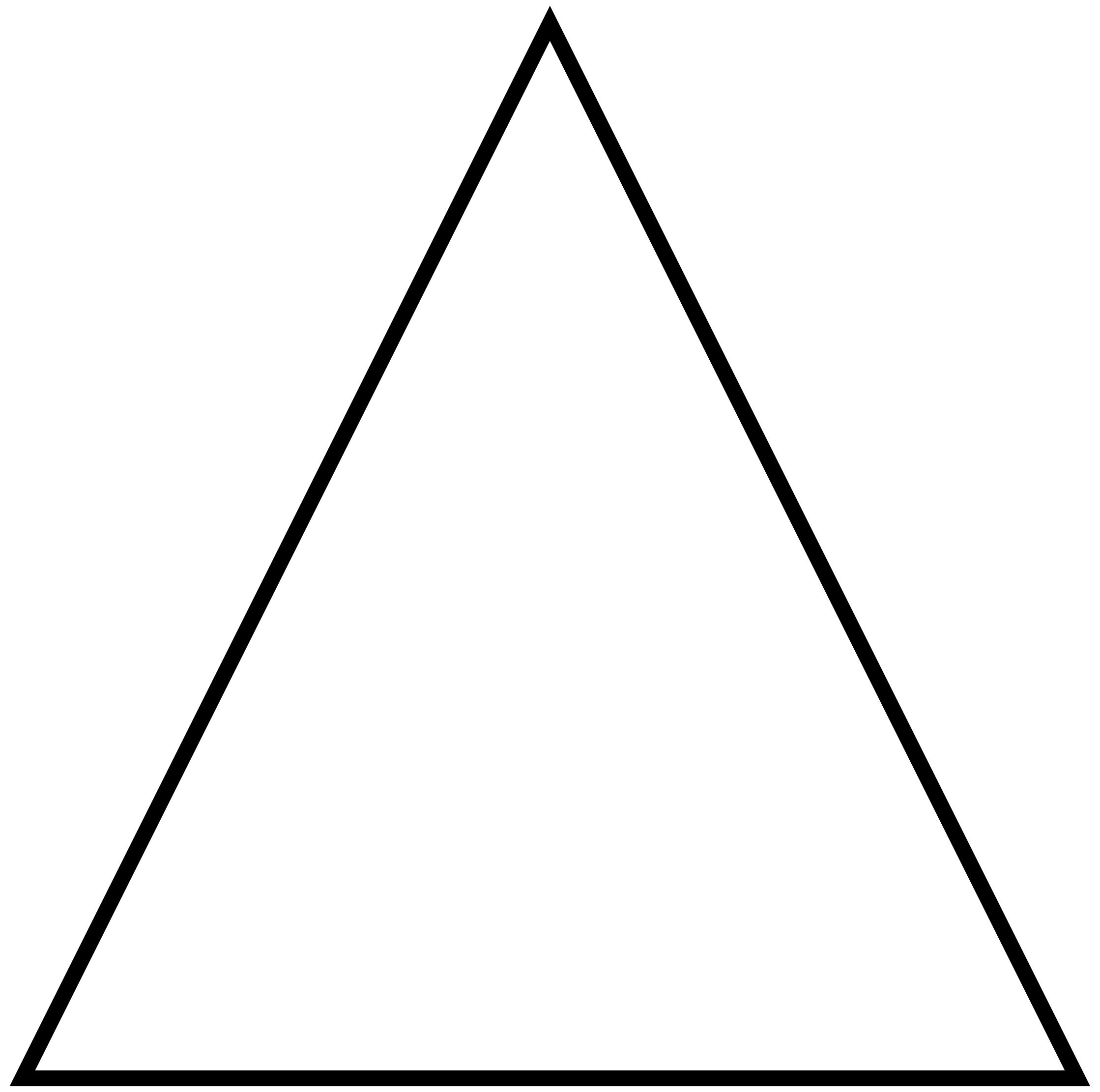
Gradient Based Optimization



Voxel

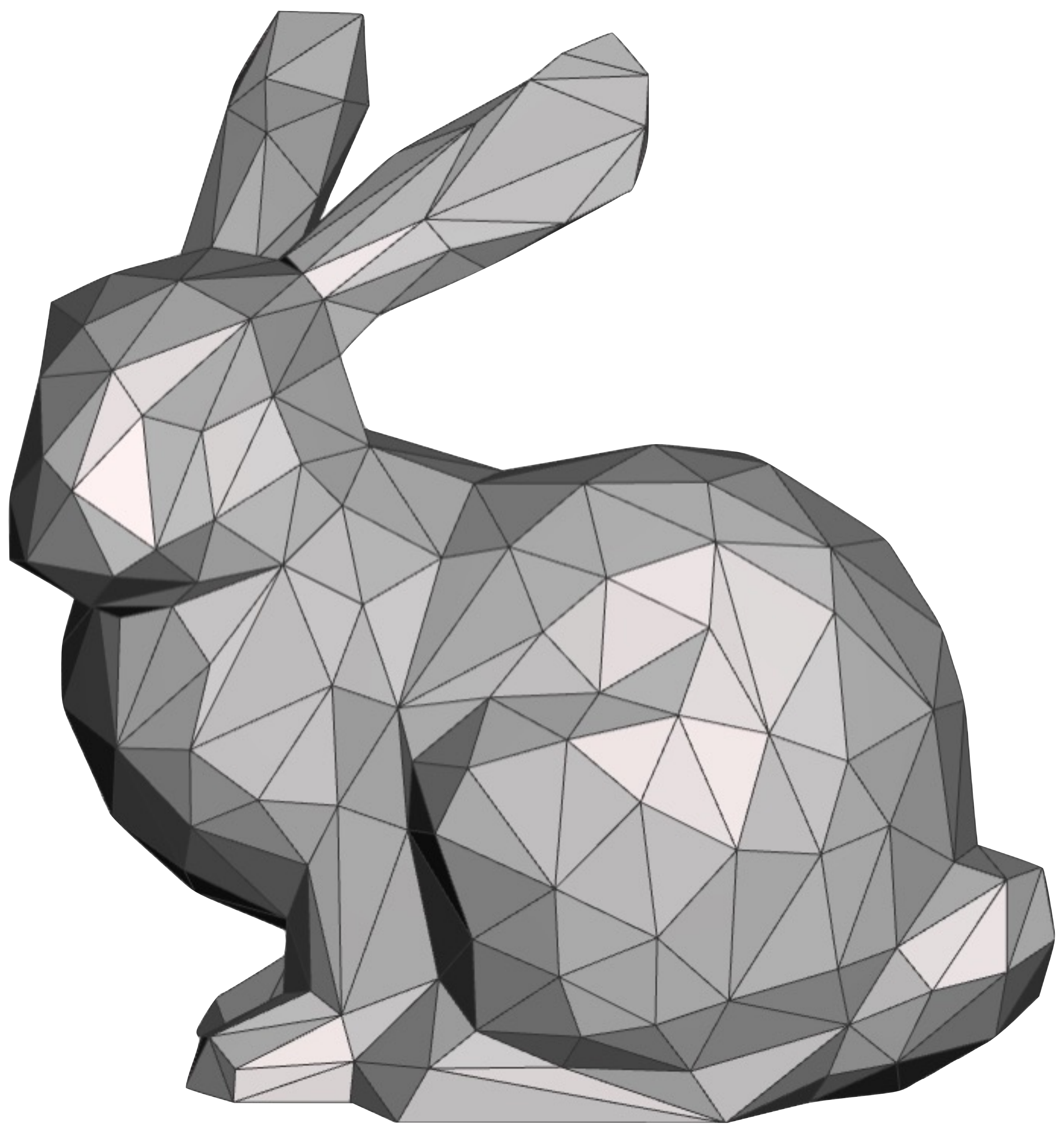


Mesh



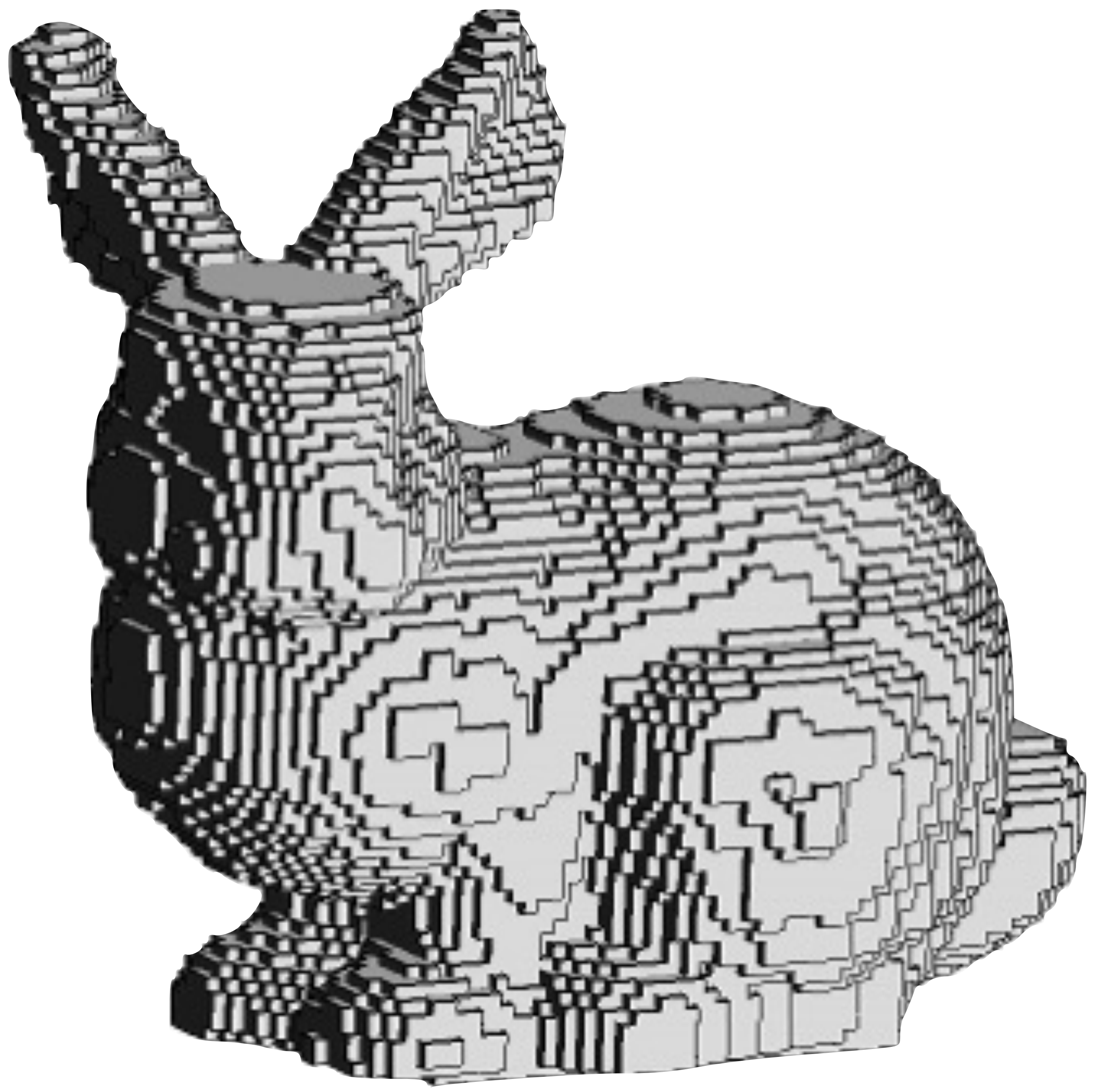
Target Geometry

Geometry Representations



Mesh Representation

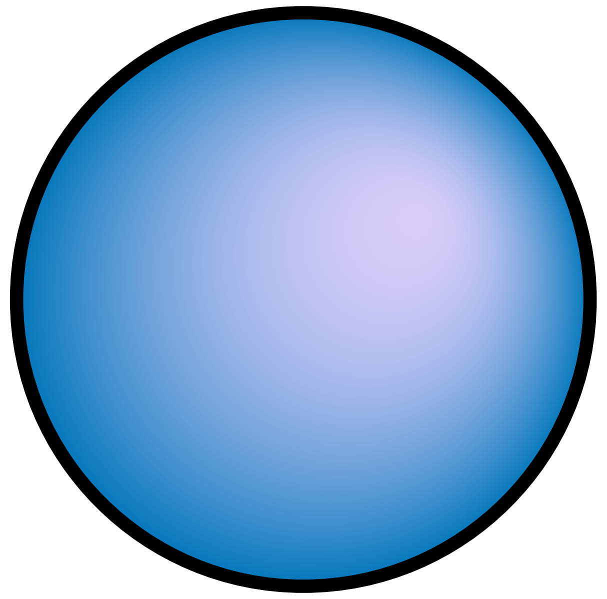
Small memory footprint
Hard to optimize



Voxel Representation

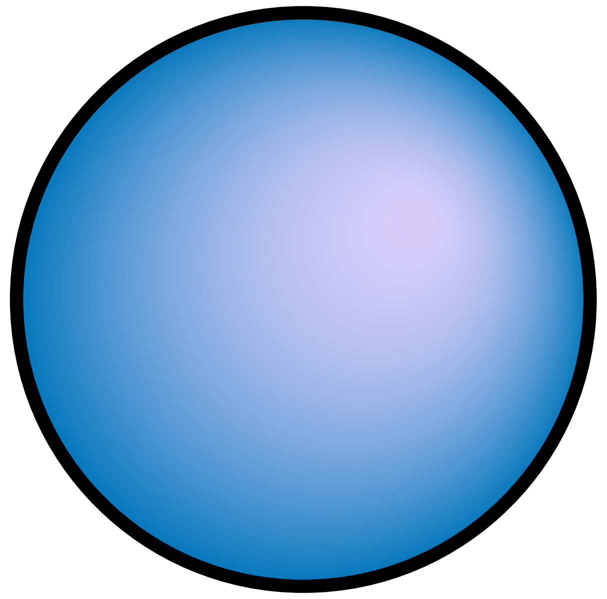
Easy to optimize
Large memory footprint

Implicit Functions

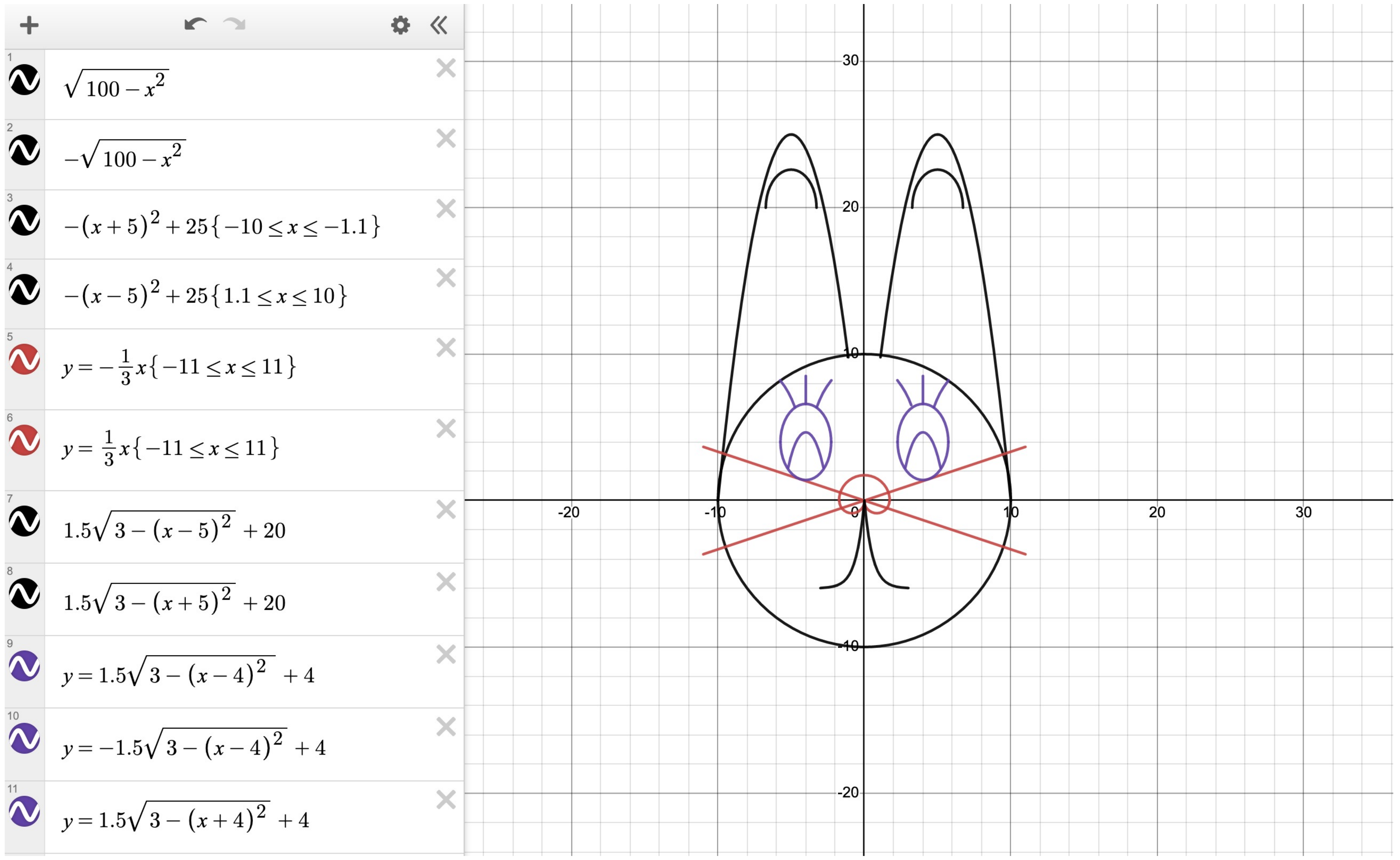


$x^2 + y^2 + z^2 = 1$

Implicit Functions

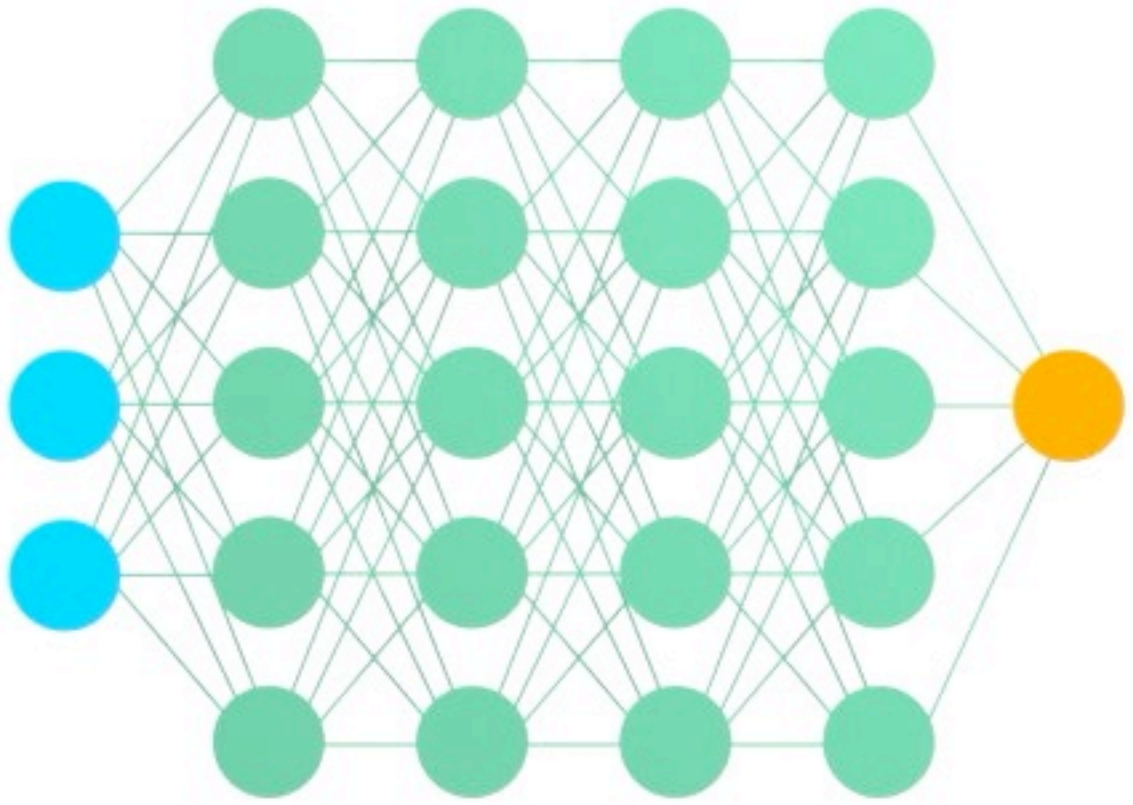


$x^2 + y^2 + z^2 = 1$



Coordinate Based Neural Network

Input
Coordinate



Value at
Coordinate

Multi Layer Perceptron
MLP

Neural networks as a continuous shape representation

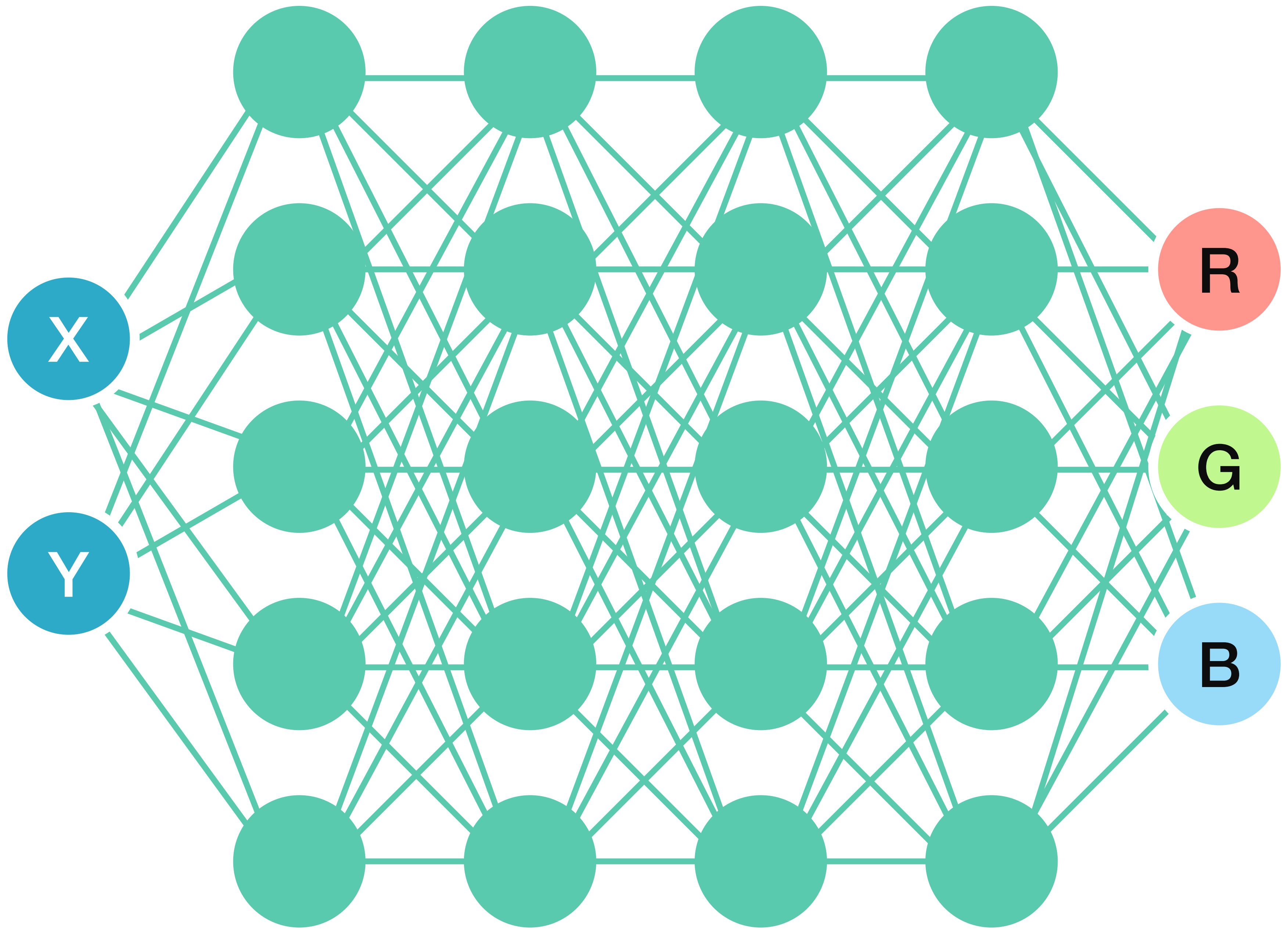
Occupancy Networks (Mescheder et al. 2019)	$(x, y, z) \rightarrow occupancy$
DeepSDF (Park et al. 2019)	$(x, y, z) \rightarrow distance$
Scene Representation Networks (Sitzmann et al. 2019)	$(x, y, z) \rightarrow$ *latent vec. (color, dist.)*
Differentiable Volumetric Rendering (Niemeyer et al. 2020)	$(x, y, z) \rightarrow$ *color, occ.*

Challenge:

- How to get MLPs to represent higher frequency functions?

Rahaman et al. 2019, Basri et al. 2020

Image Representation

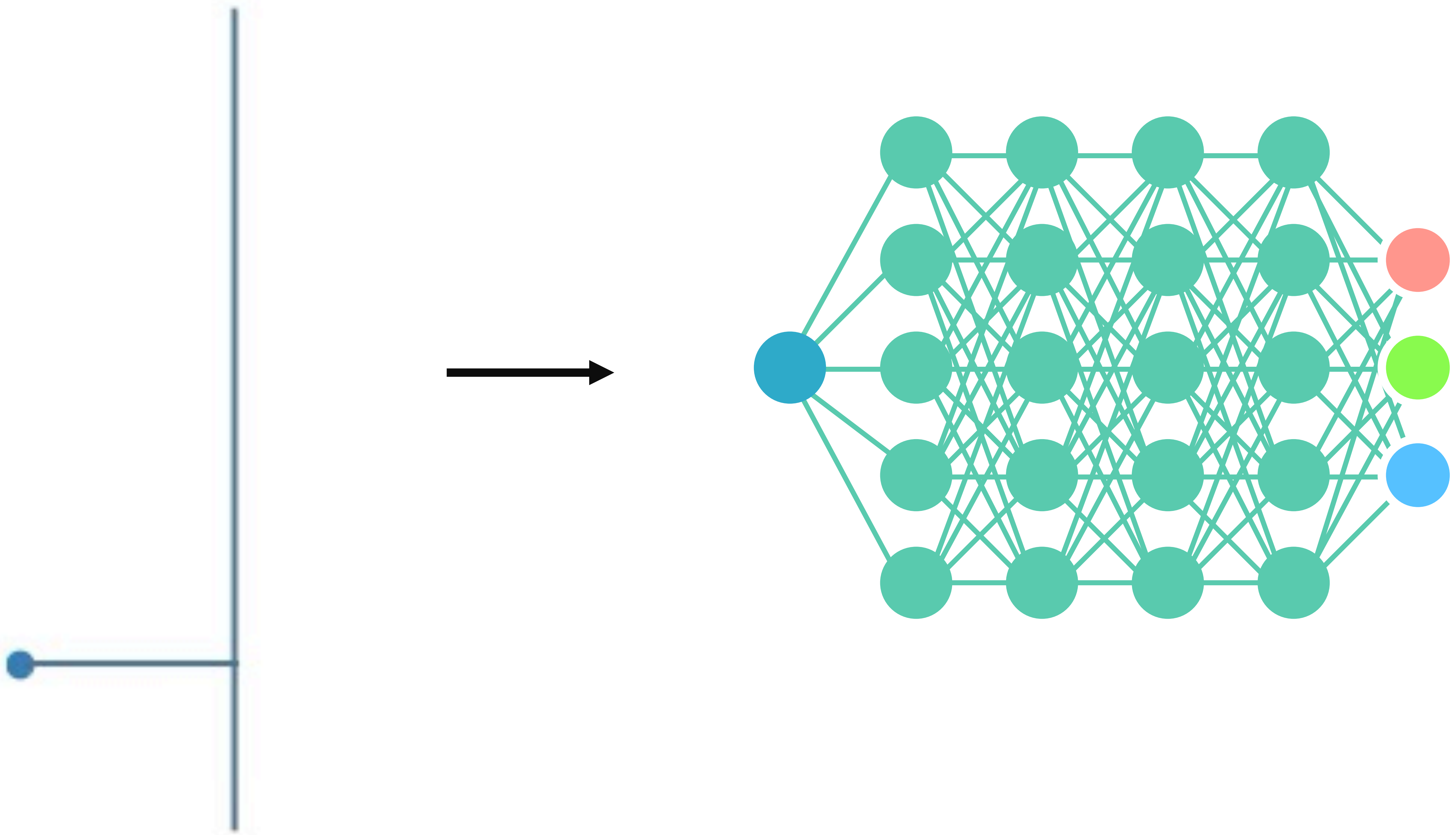


MLP output

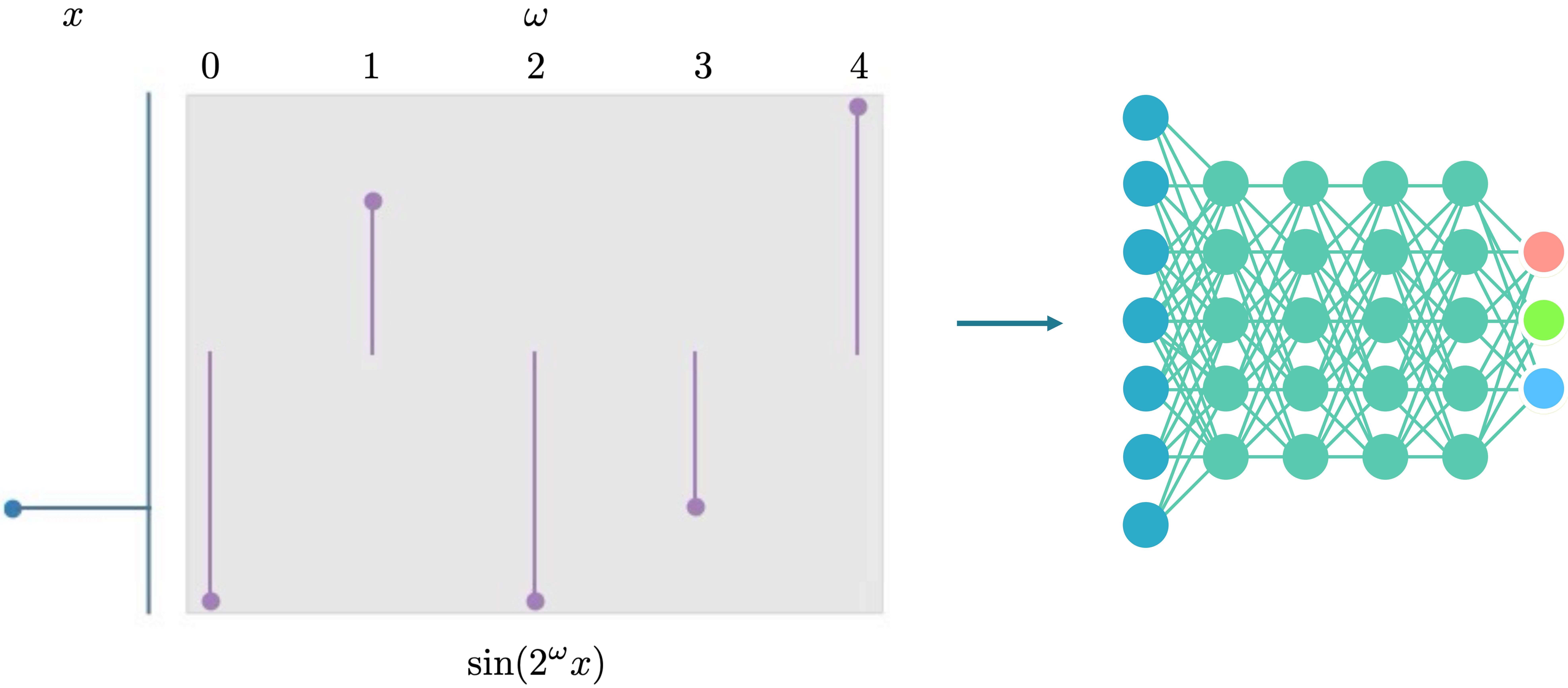
Supervision image

Standard input

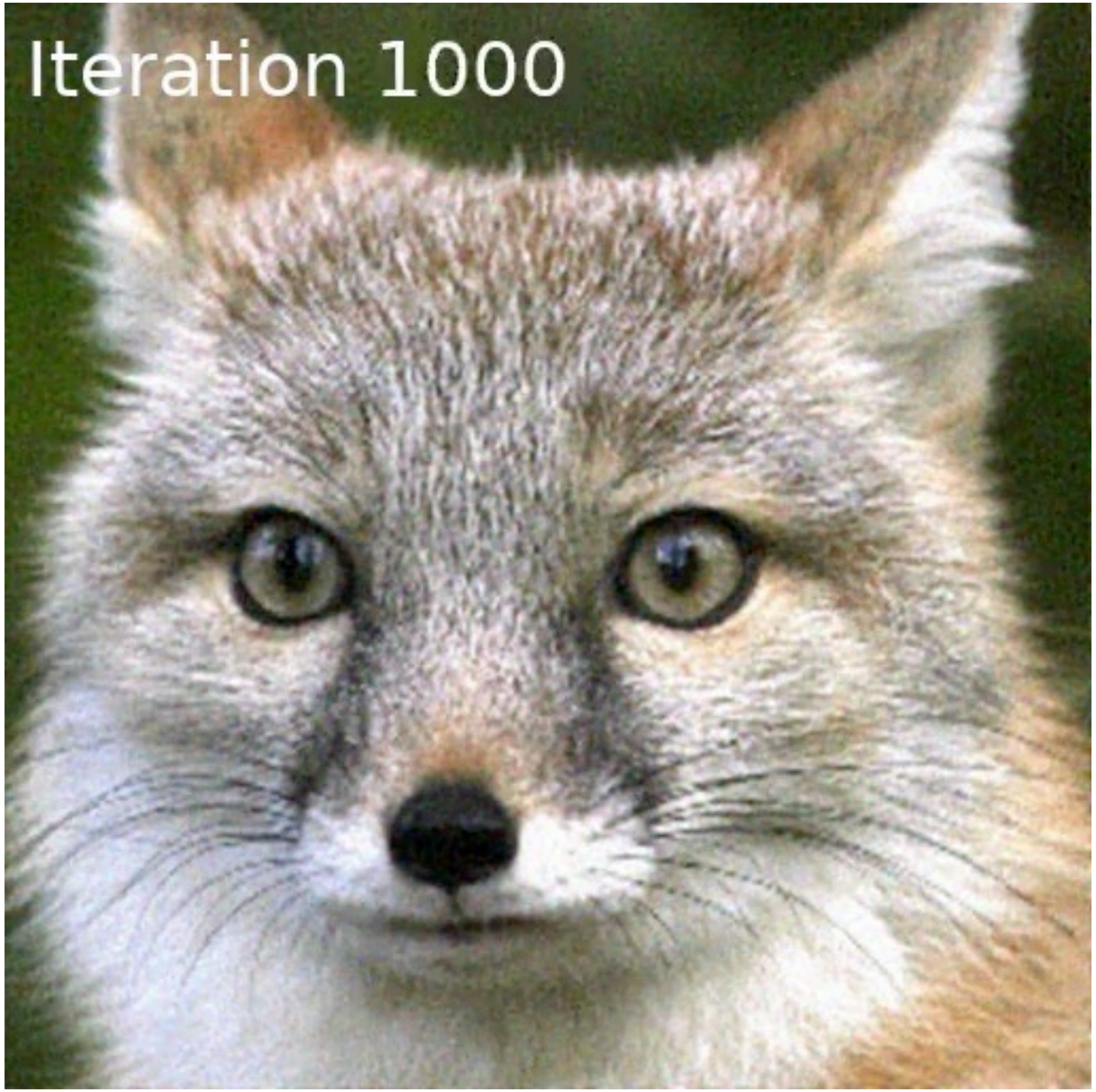
x



Standard input   Positionally Encoded input



Standard MLP



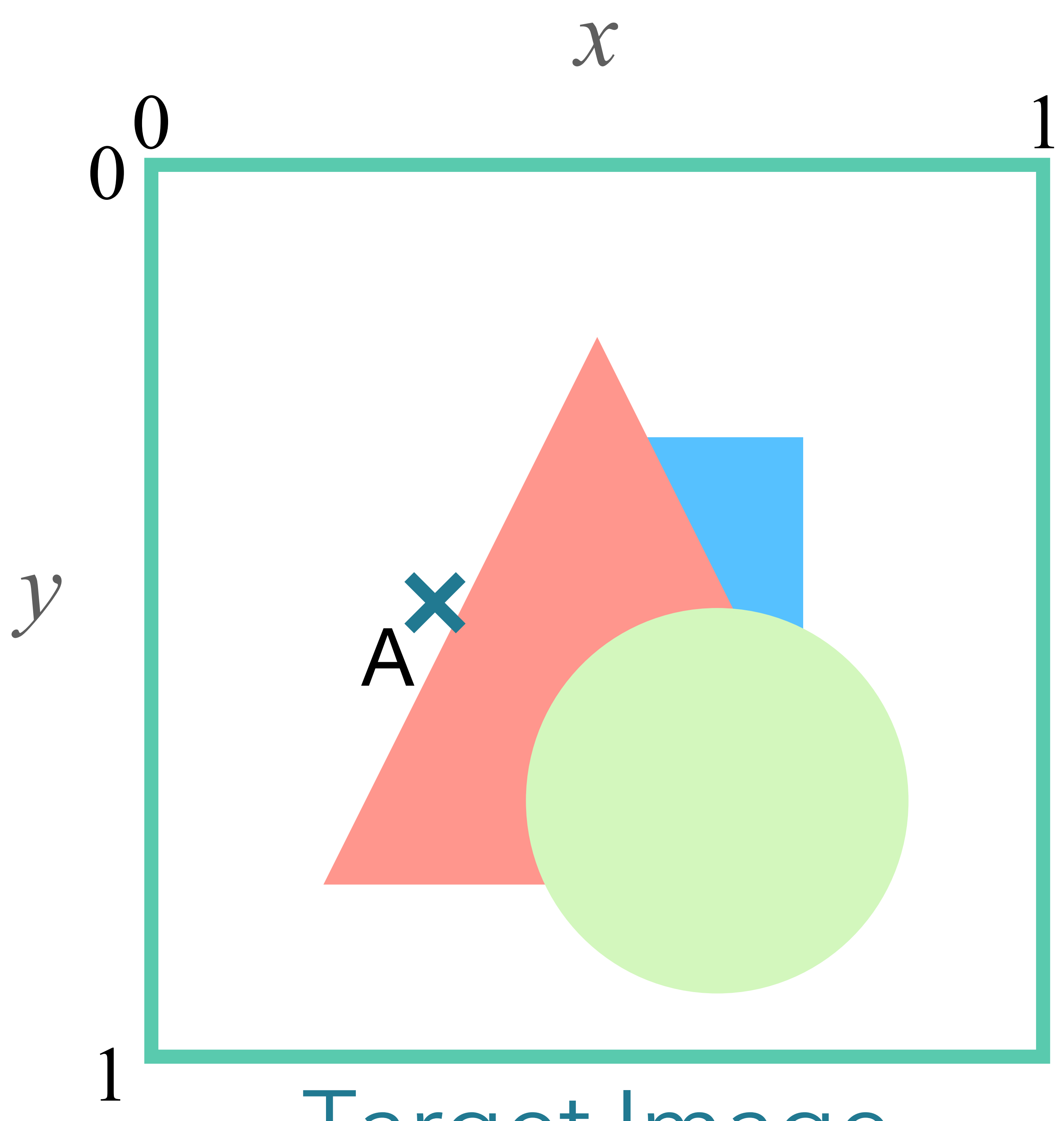
MLP with Fourier features

Why does positional encoding help?

Target Image

Why does positional encoding help?

	Input		Target
	x	y	
A	.36	.5	☐



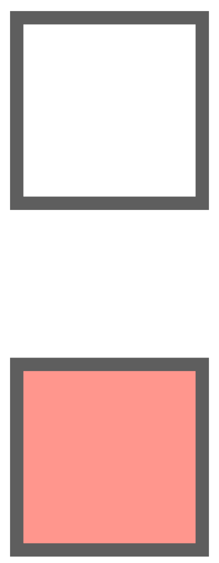
Target Image

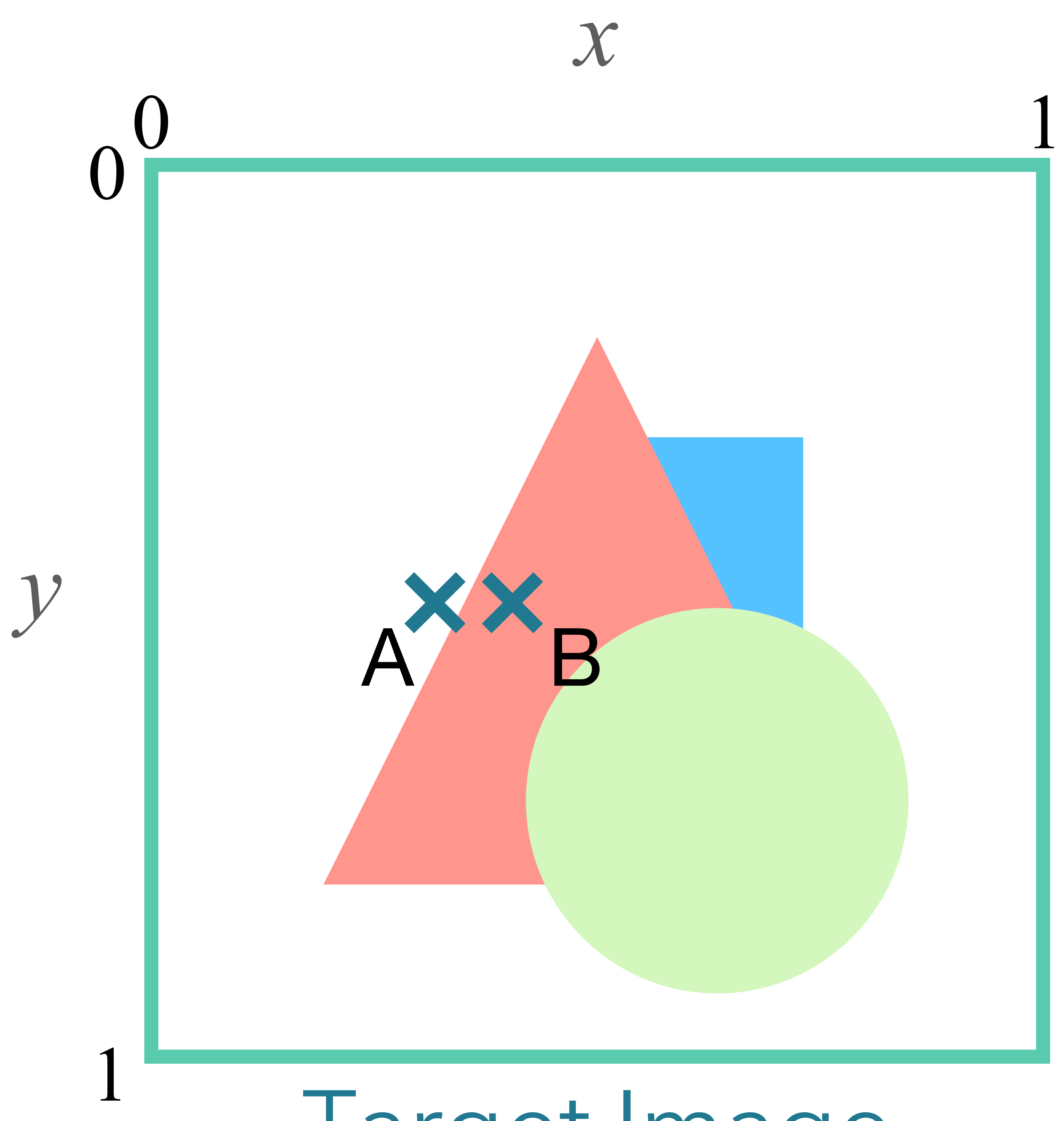
Why does positional encoding help?

Input

	x	y
A	.36	.5
B	.38	.5

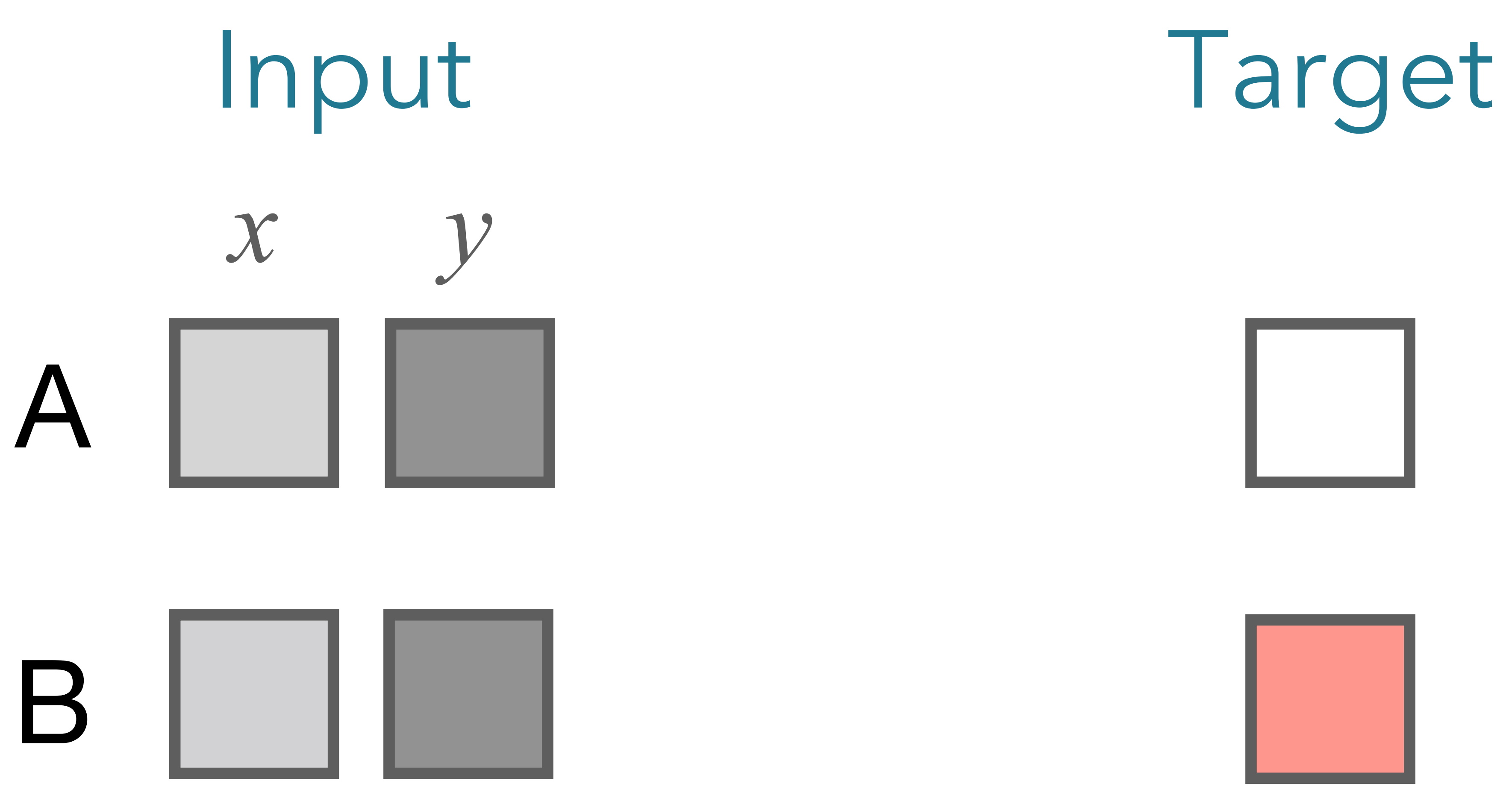
Target

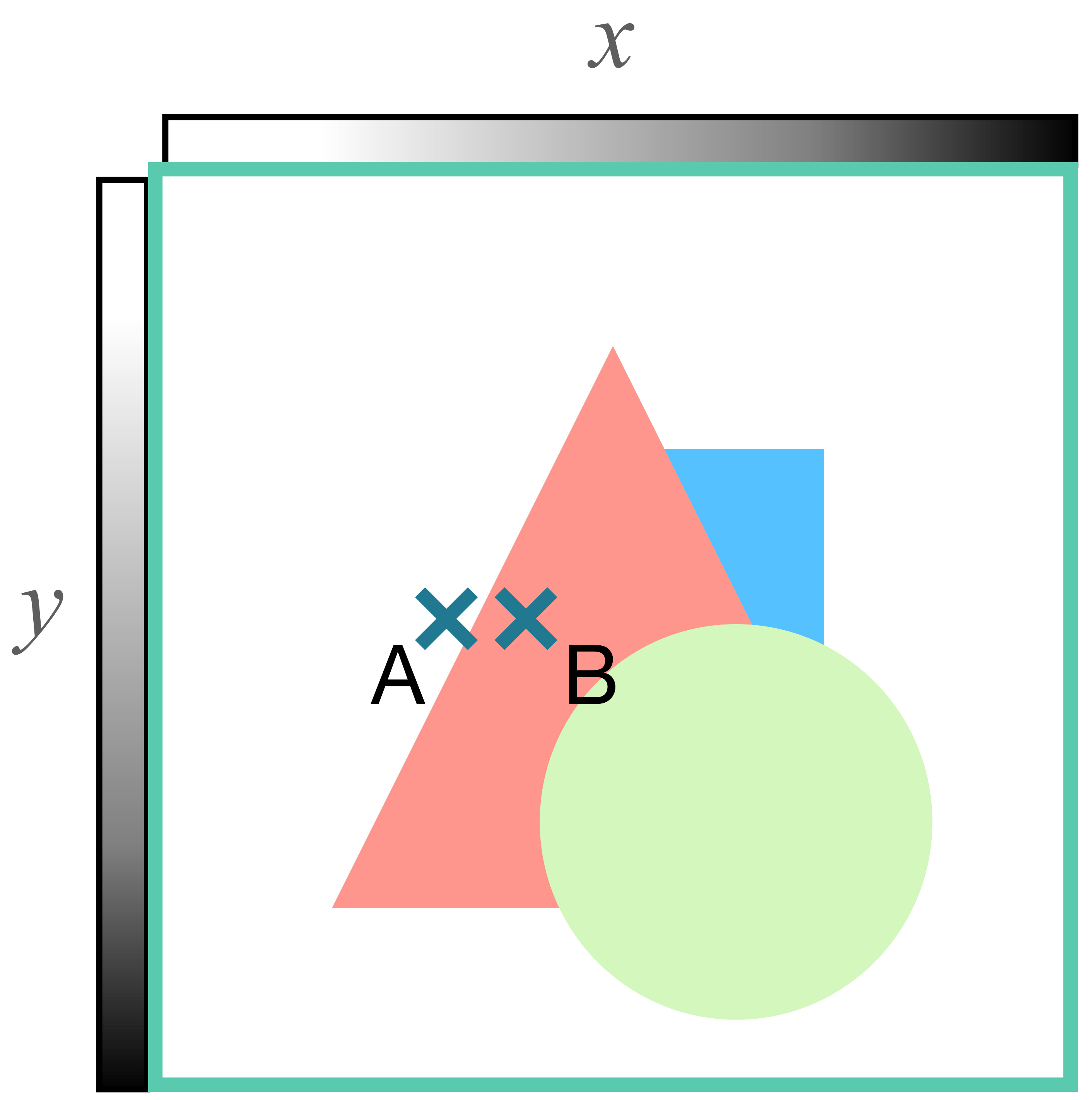




Target Image

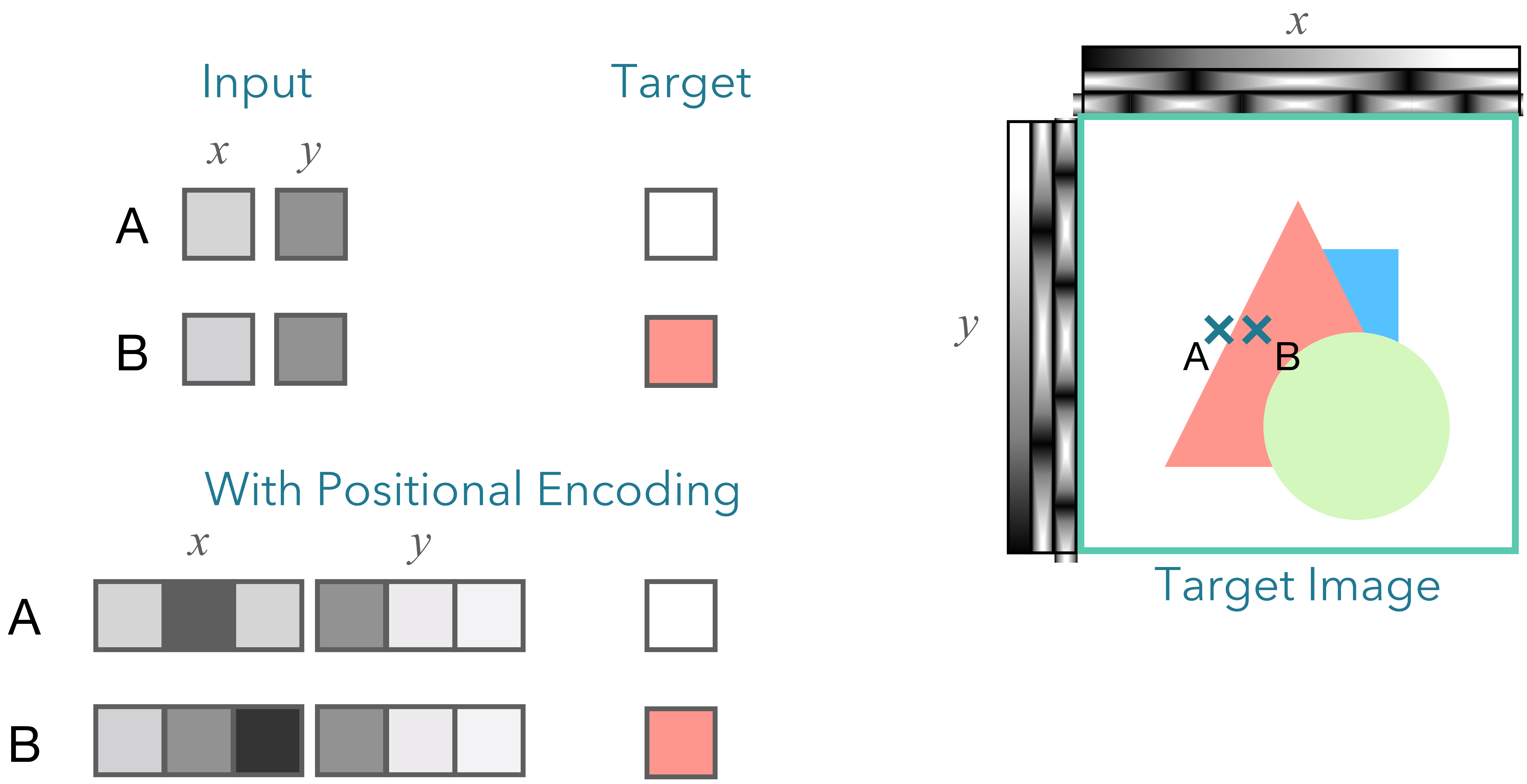
Why does positional encoding help?



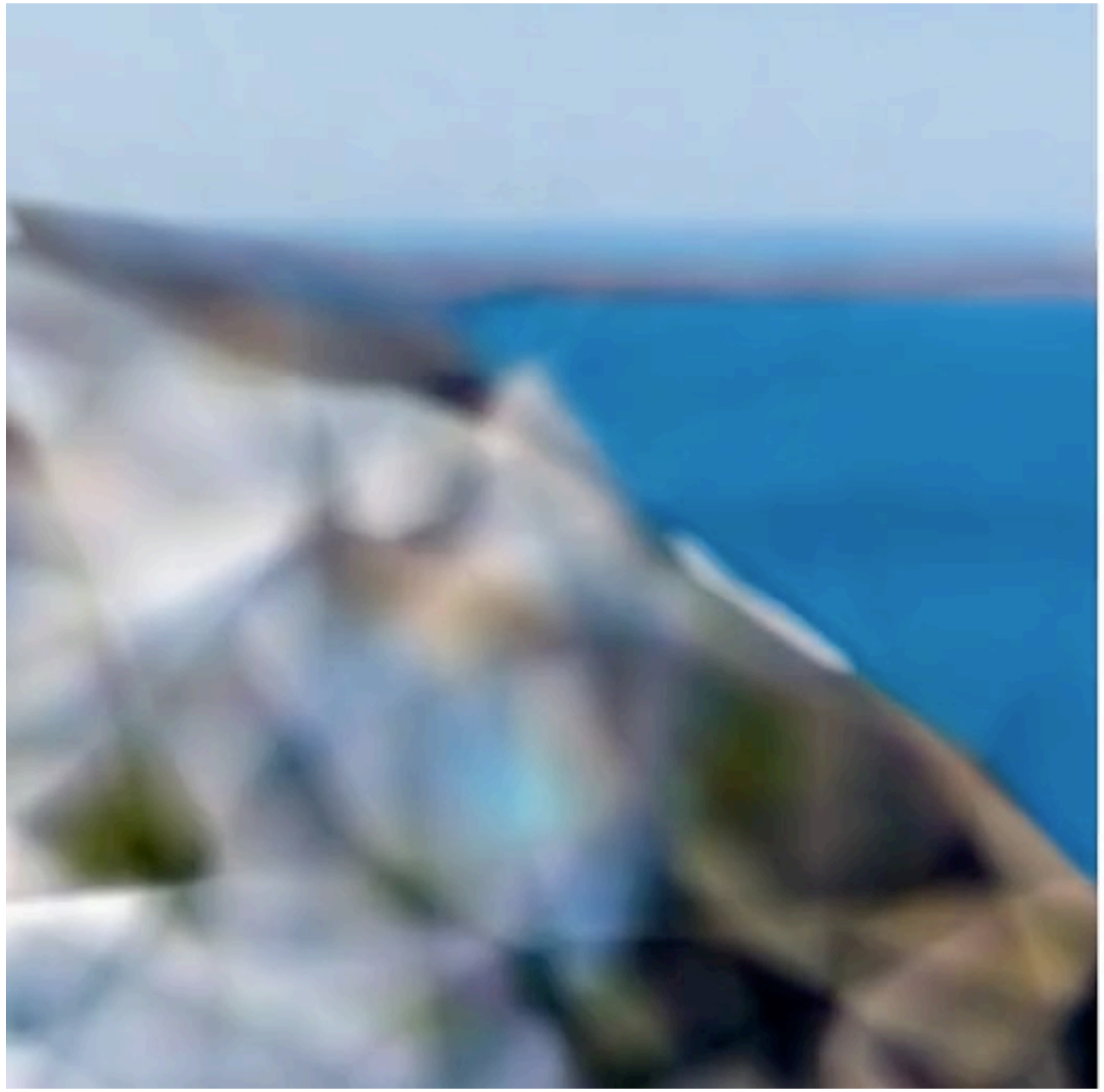


Target Image

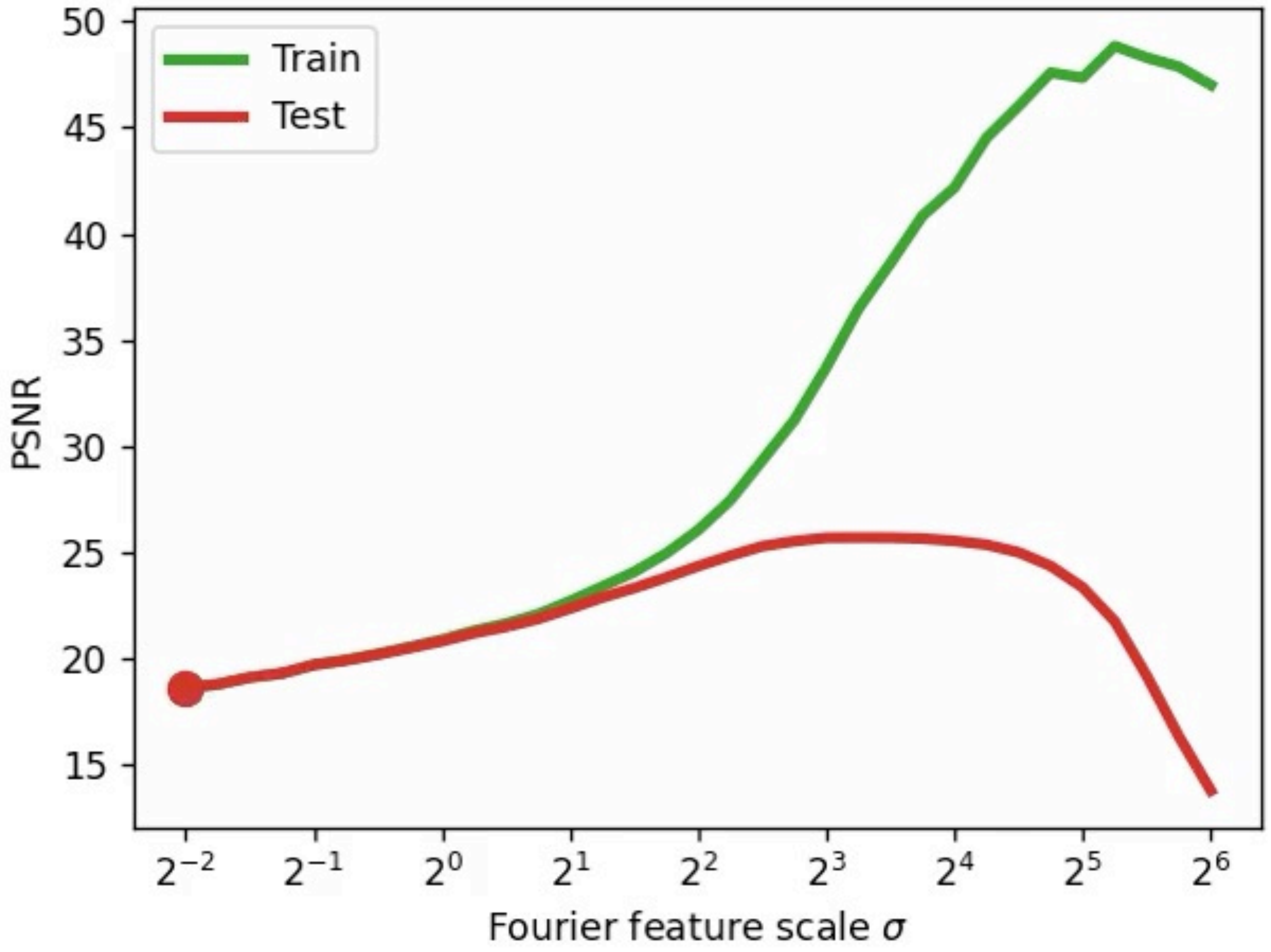
Why does positional encoding help?



Performance depends on max encoding frequency



Network output

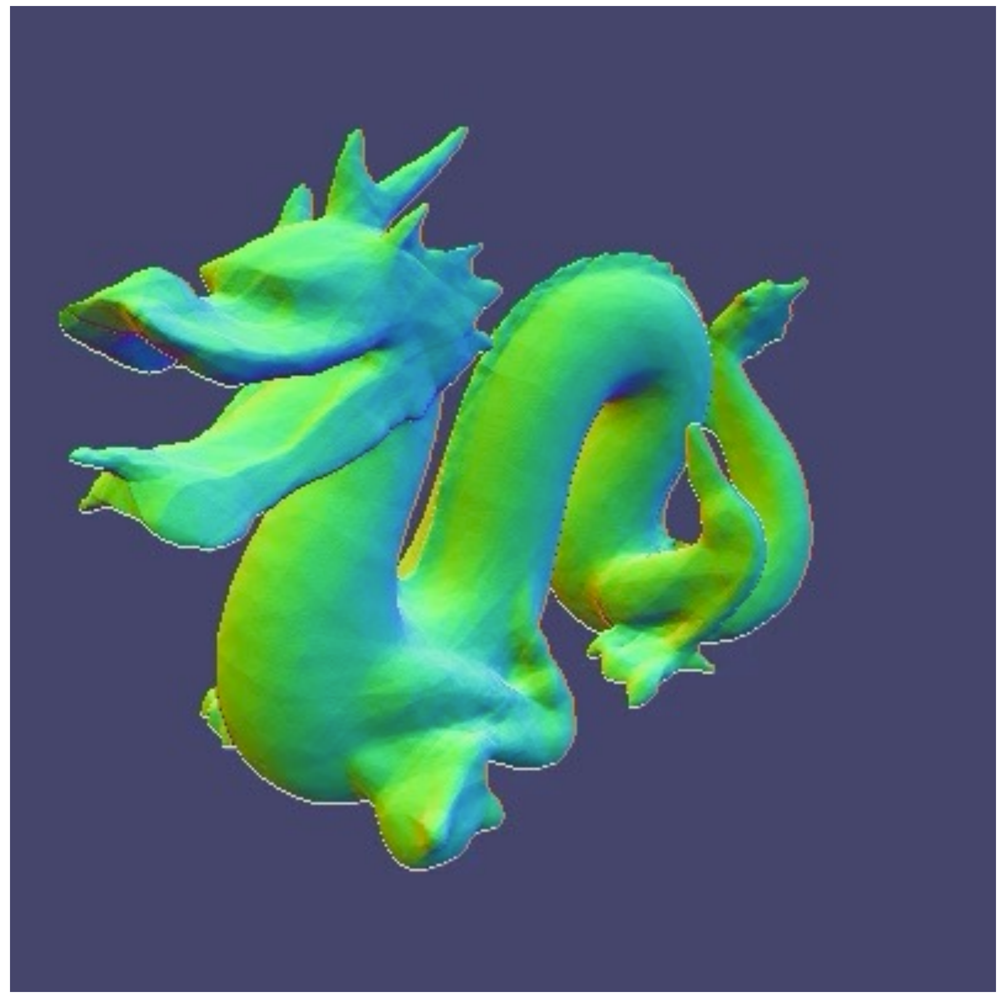


Performance vs. scale value

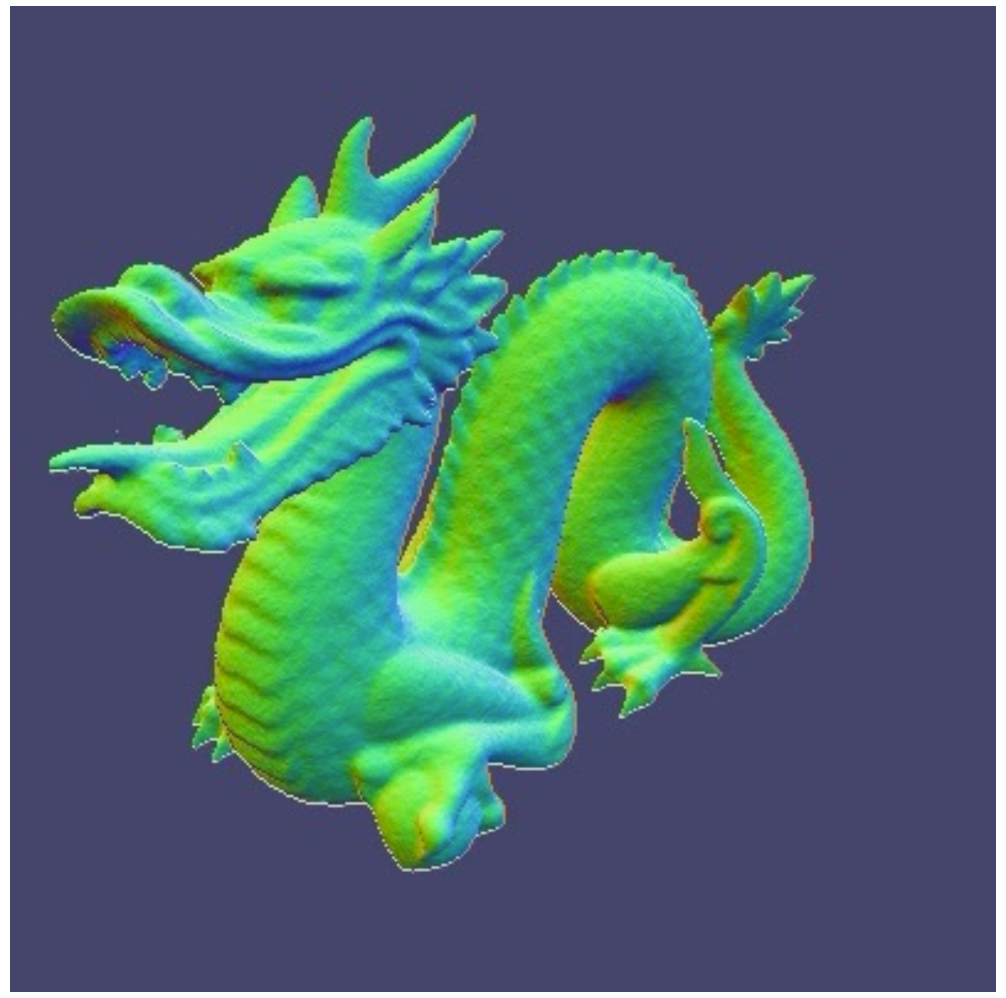
Tancik*, Srinivasan*, Mildenhall* et al. 2020, *Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains*

Coordinate-based MLPs can replace any low-dimensional array

Without Encoding



With Encoding



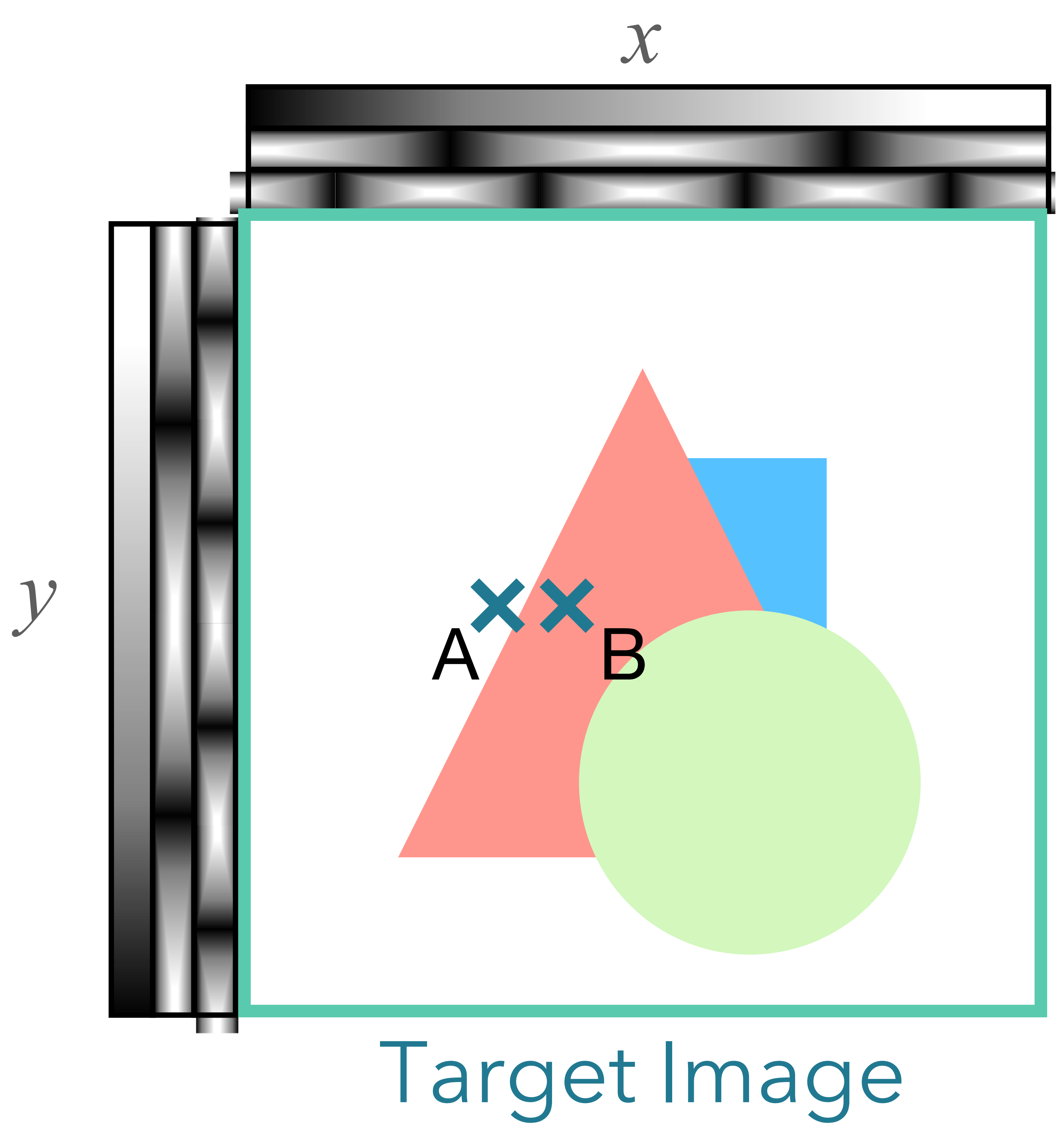
3D Shape

NeRF with and without positional encoding

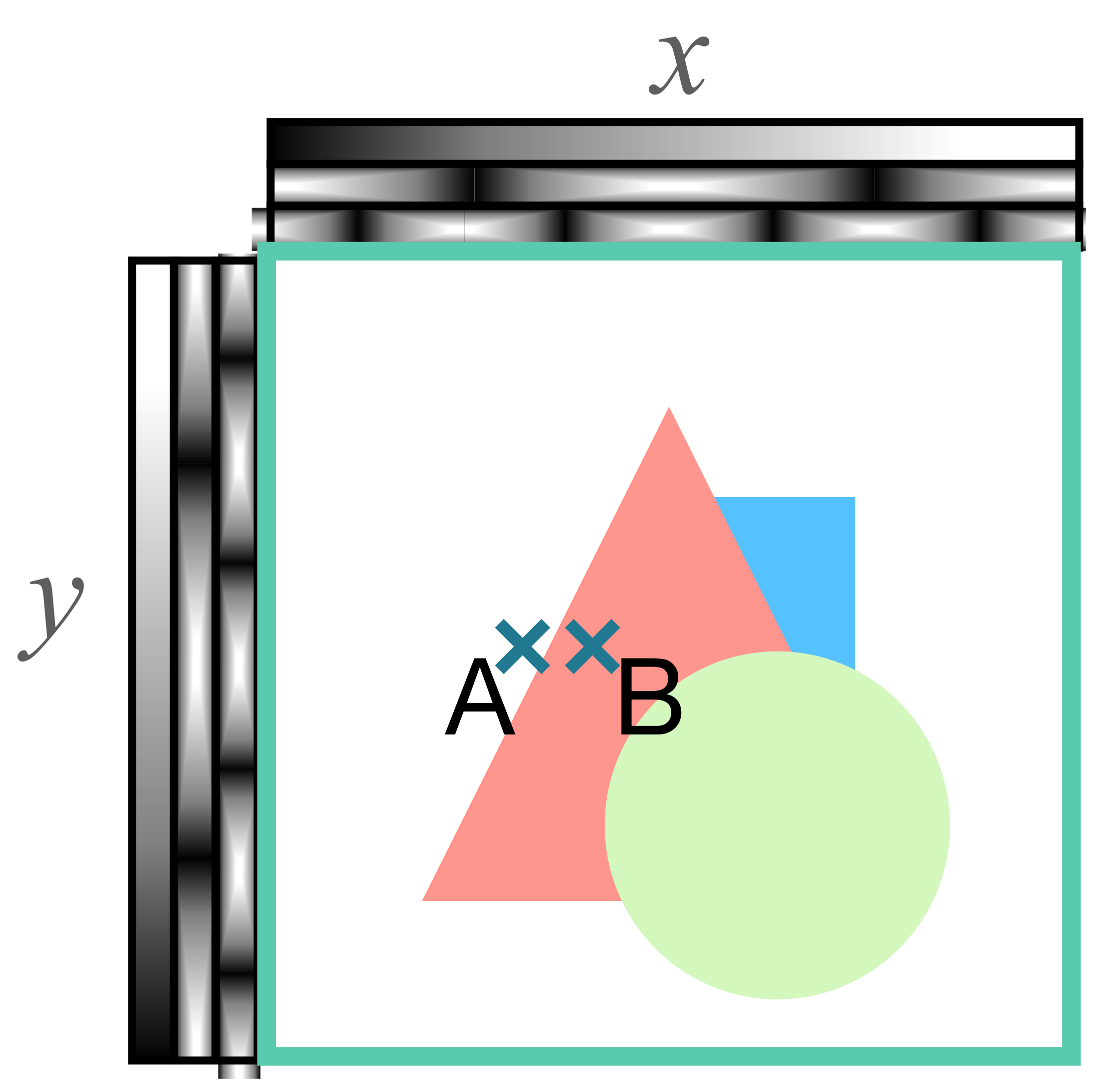
NeRF (Naive)

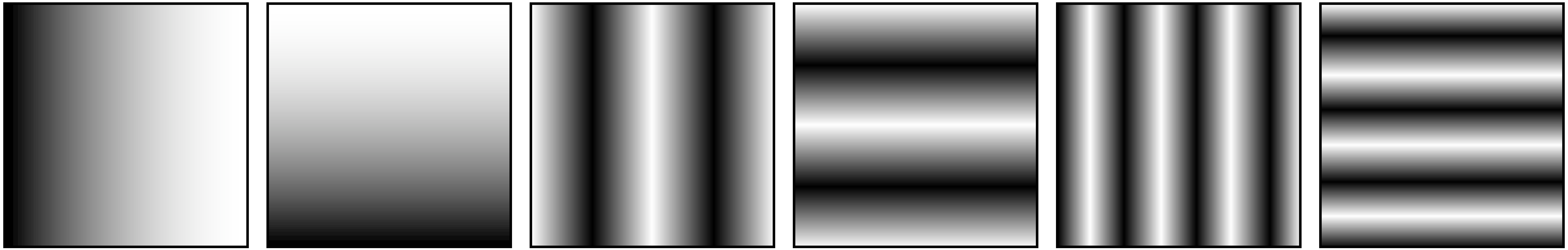
NeRF (with positional encoding)

Other Encoding Considerations

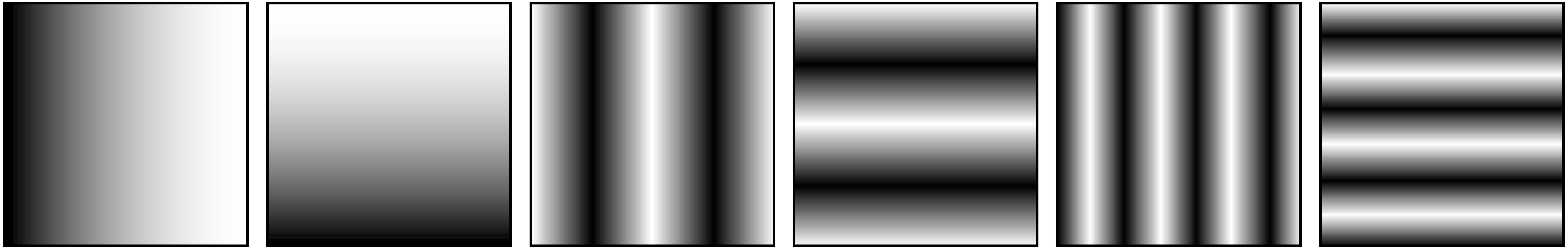


Other Encoding Considerations



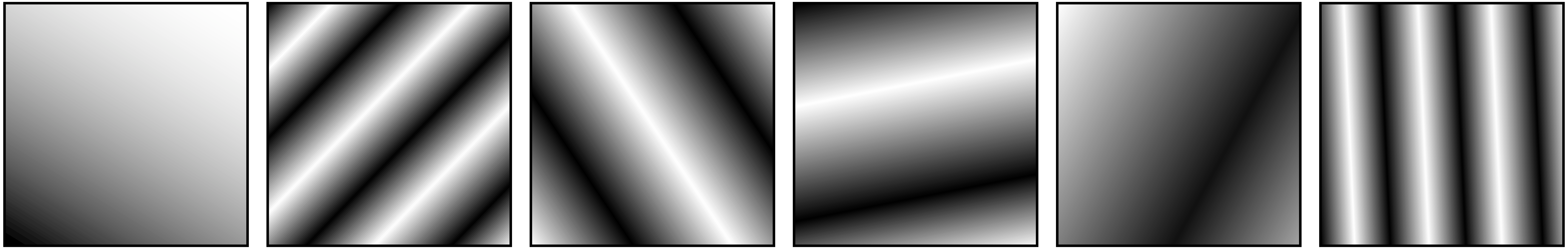


Other Encoding Considerations



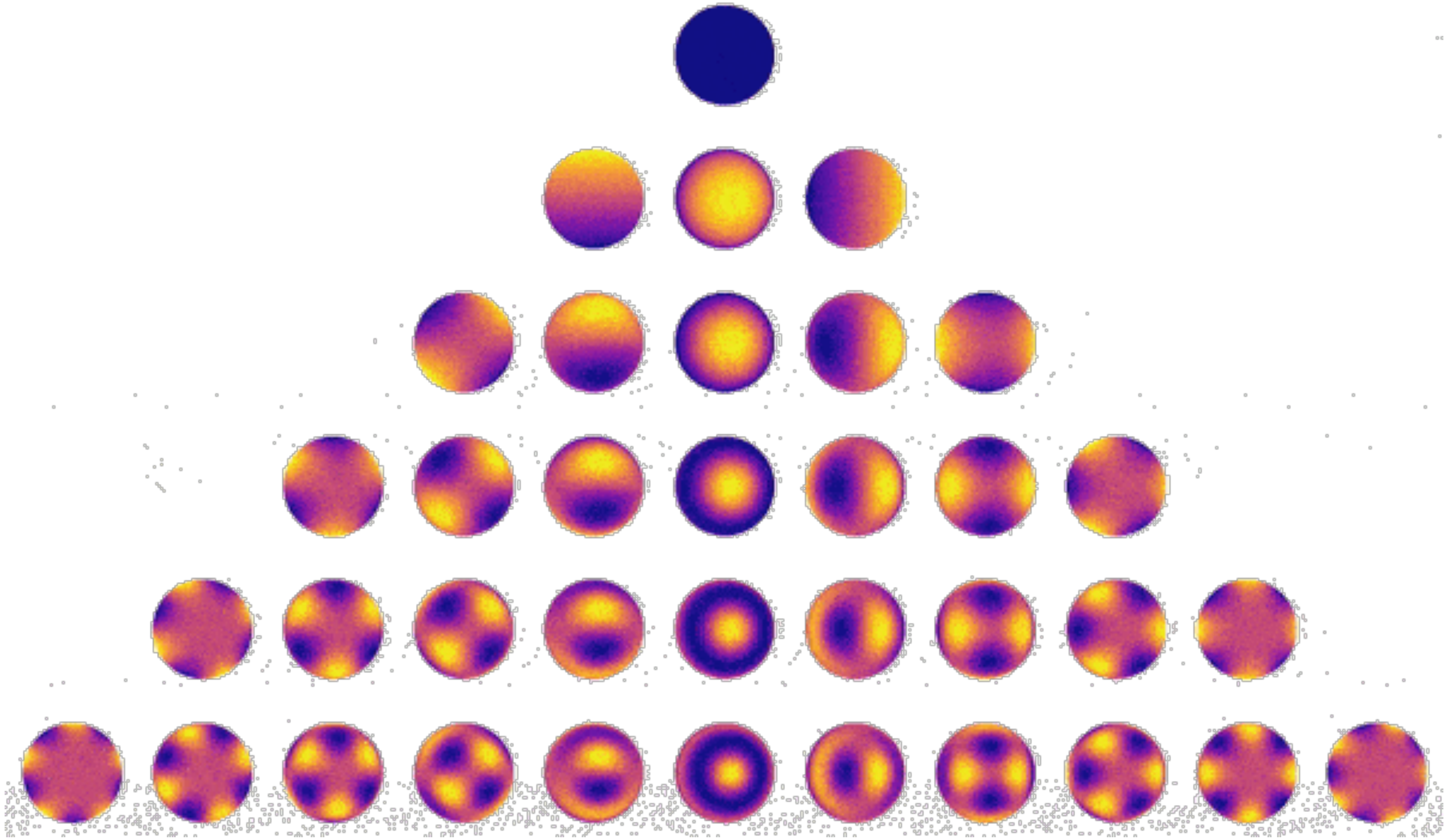
Can lead to “grid” artifacts

Other Encoding Considerations



Random Fourier Features

Other Encodings



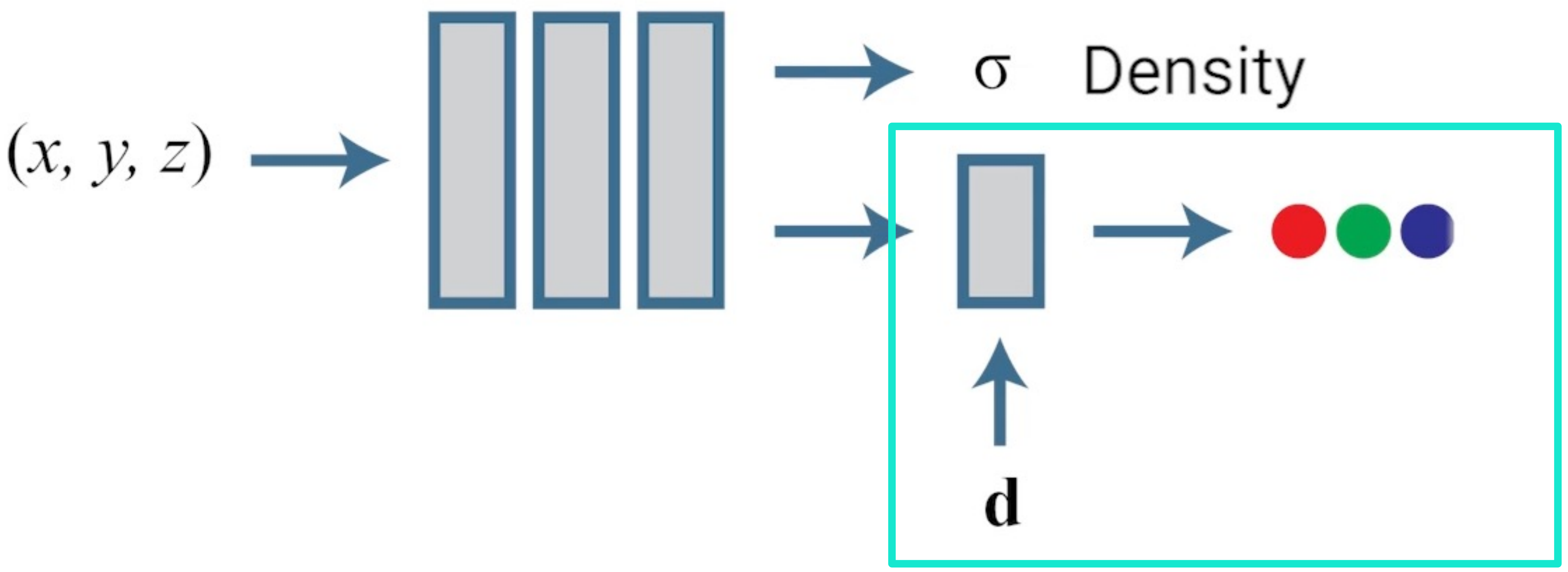
Spherical Harmonics

fourier basis on the sphere

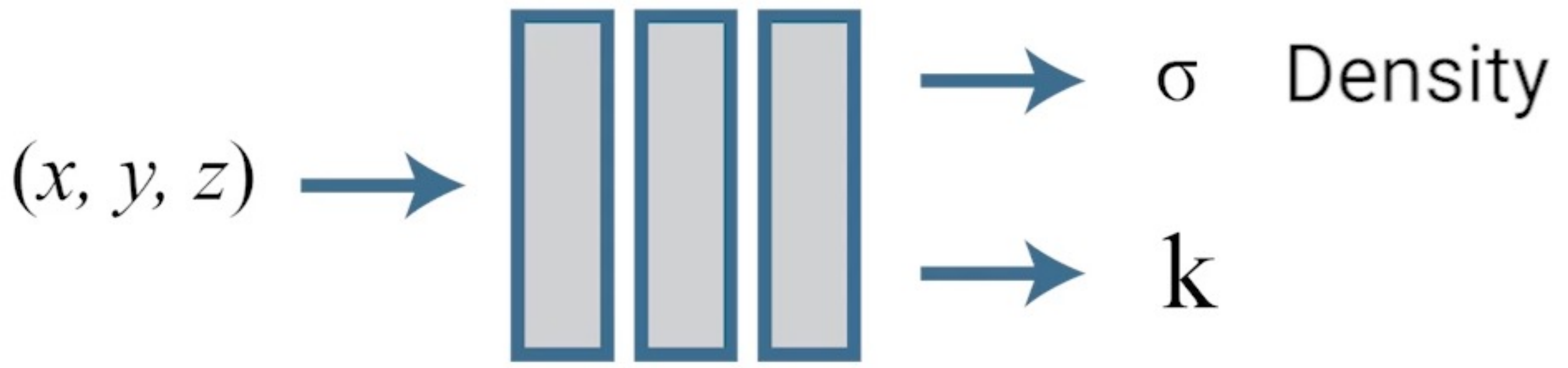
Approximate a function on the sphere with Spherical Harmonics coefficients

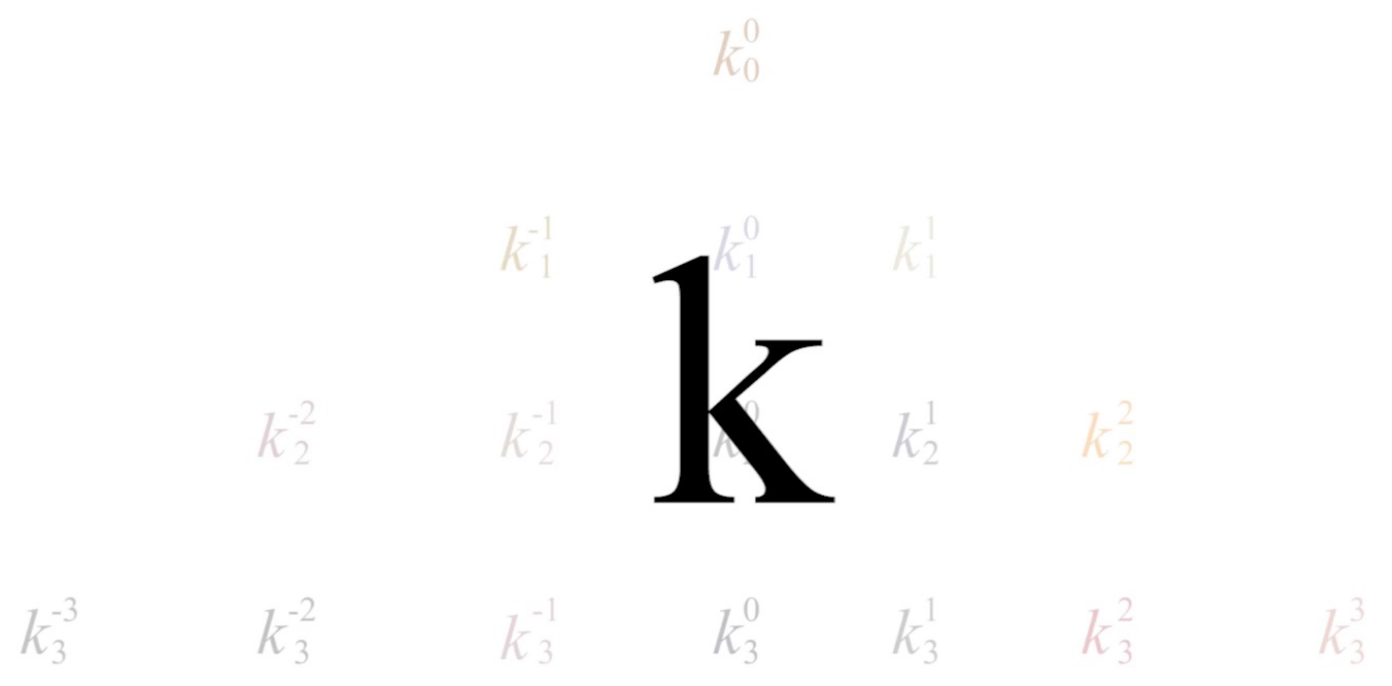
NeRF with Spherical Functions

NeRF

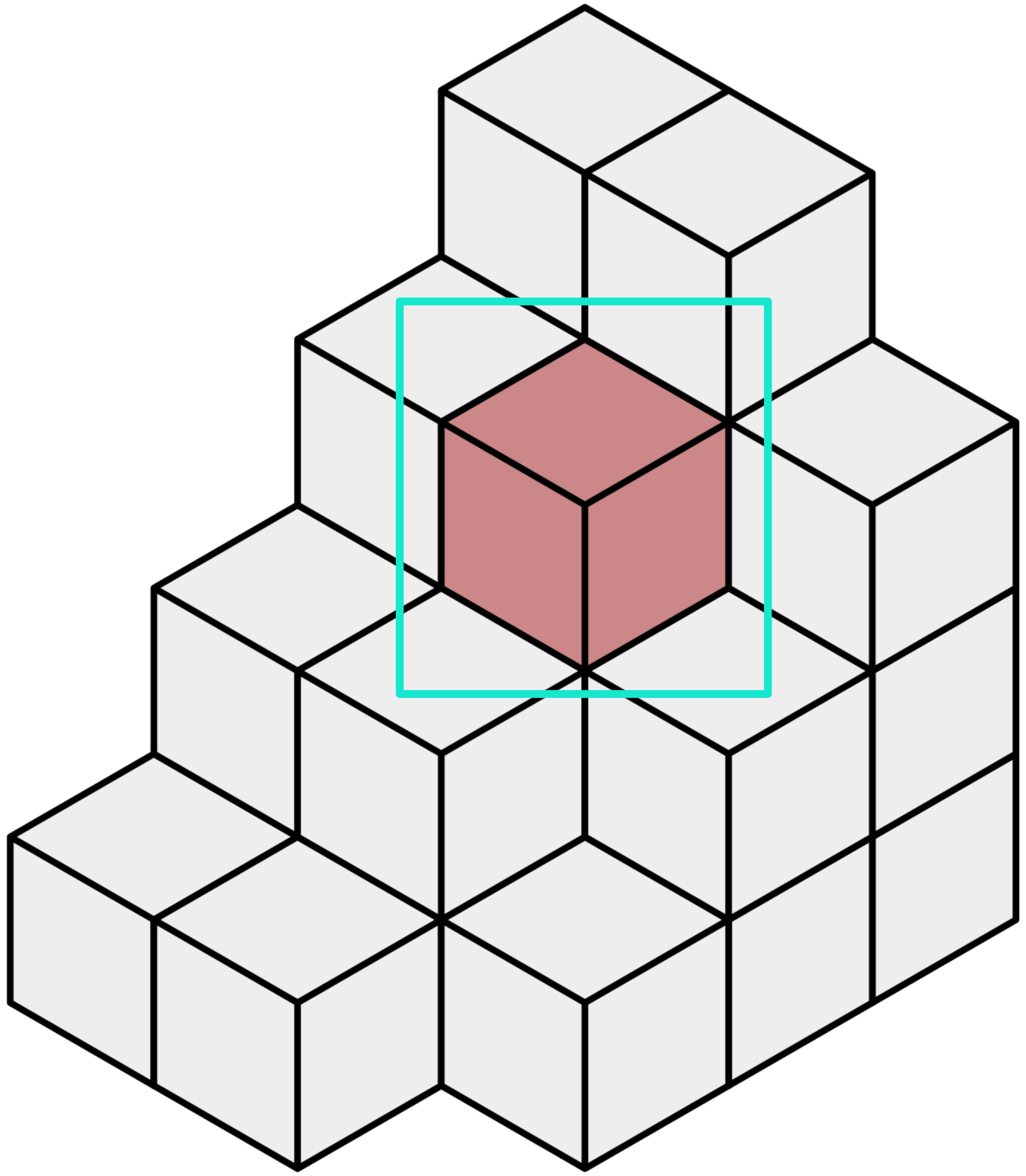


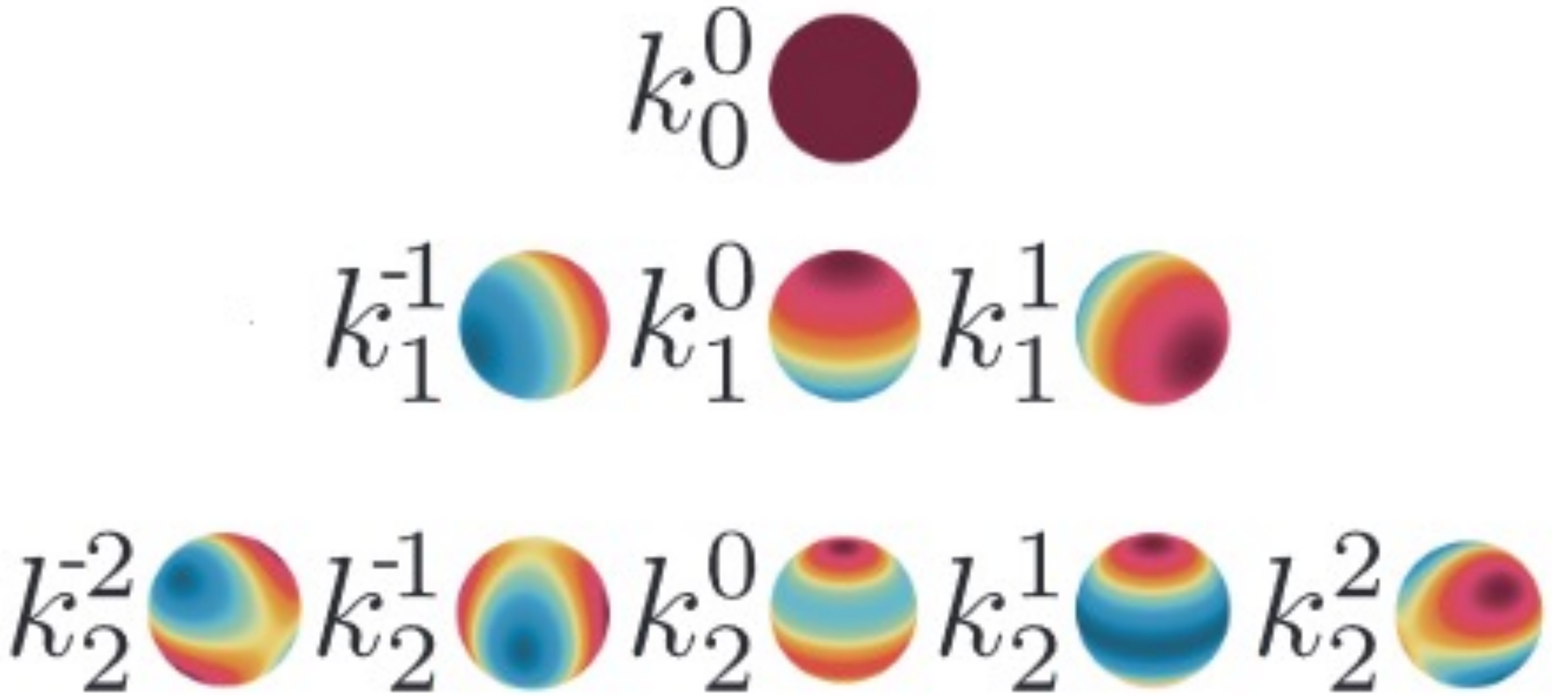
NeRF with Spherical Harmonics (NeRF-SH)





PlenOctree =
Sparse Voxels with density + SH coefficients





Skips empty regions
→ much faster rendering time!

Yu et al. PlenOctrees for Real-time Rendering of Neural Radiance Fields, ICCV 2021

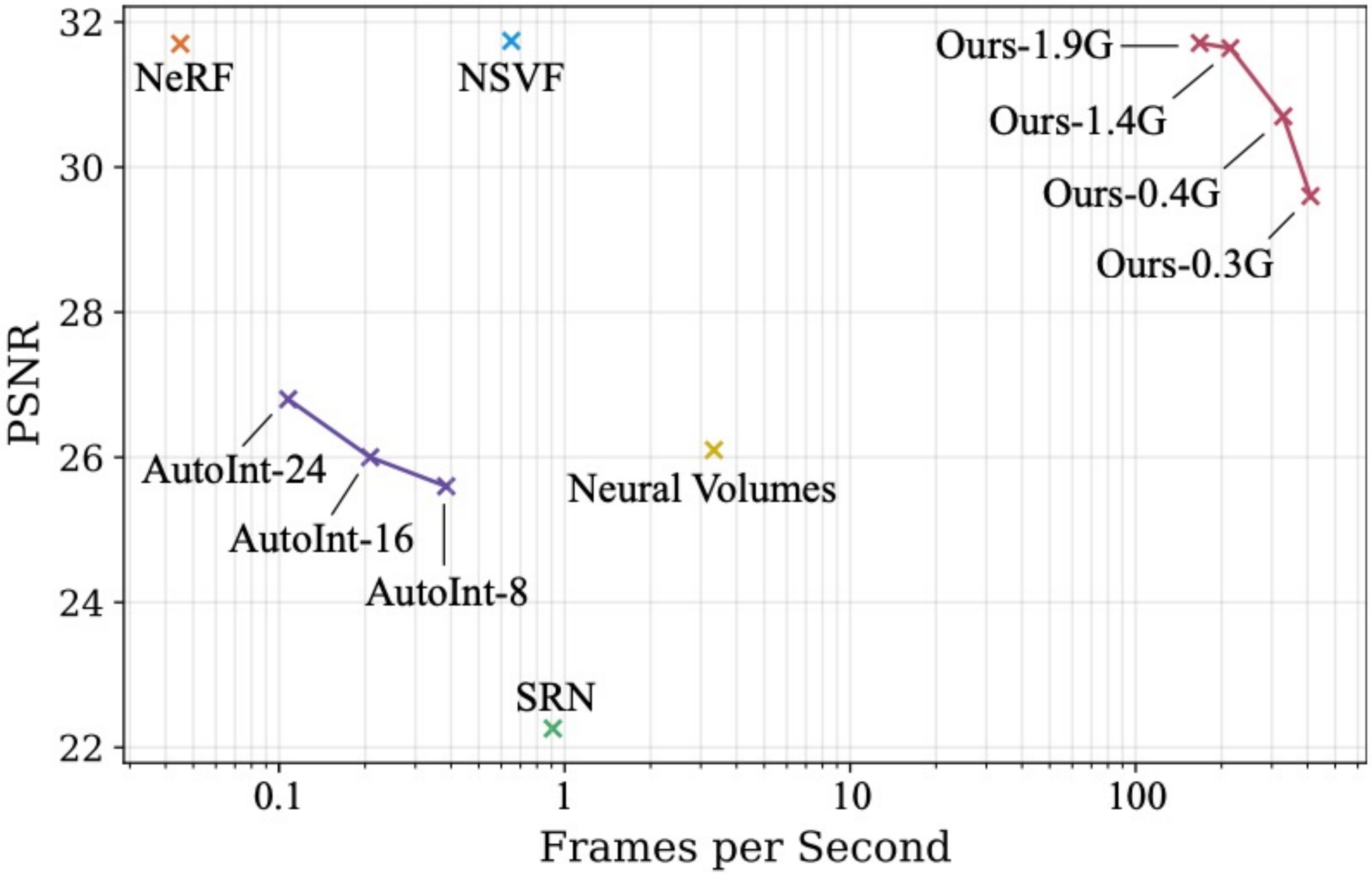
PlenOctree
54.00 FPS

More Results

NeRF
0.013 FPS

Trade-off

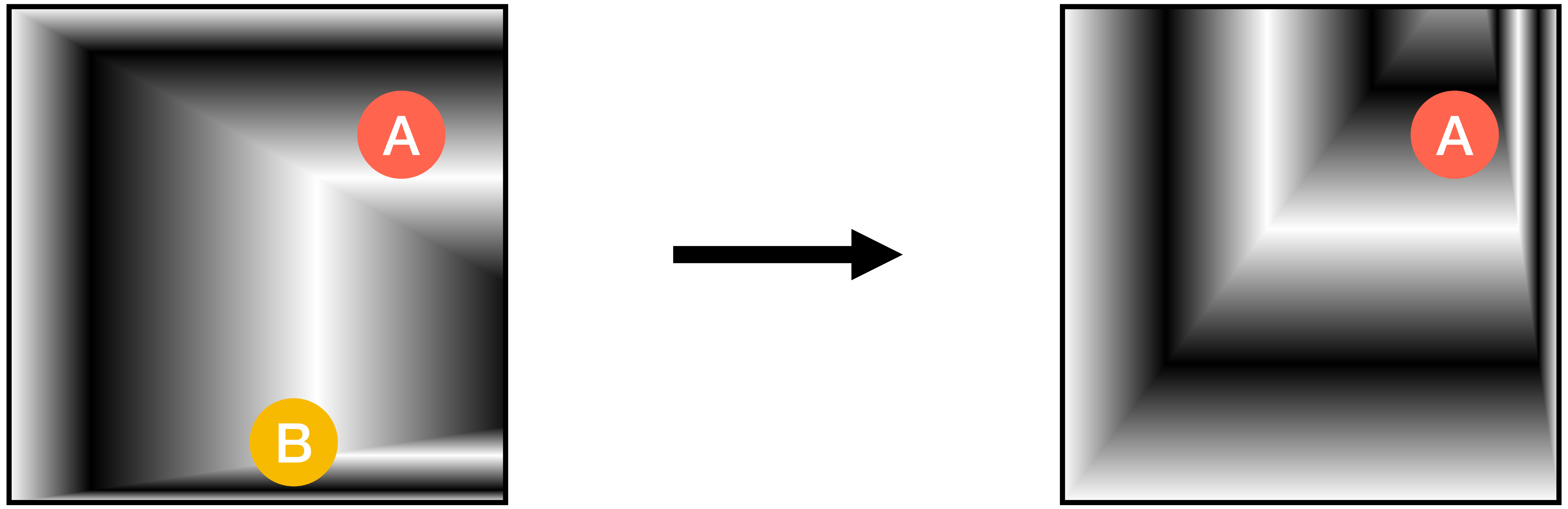
🐇 3000x faster than OG NeRF!



Can we learn the encoding?

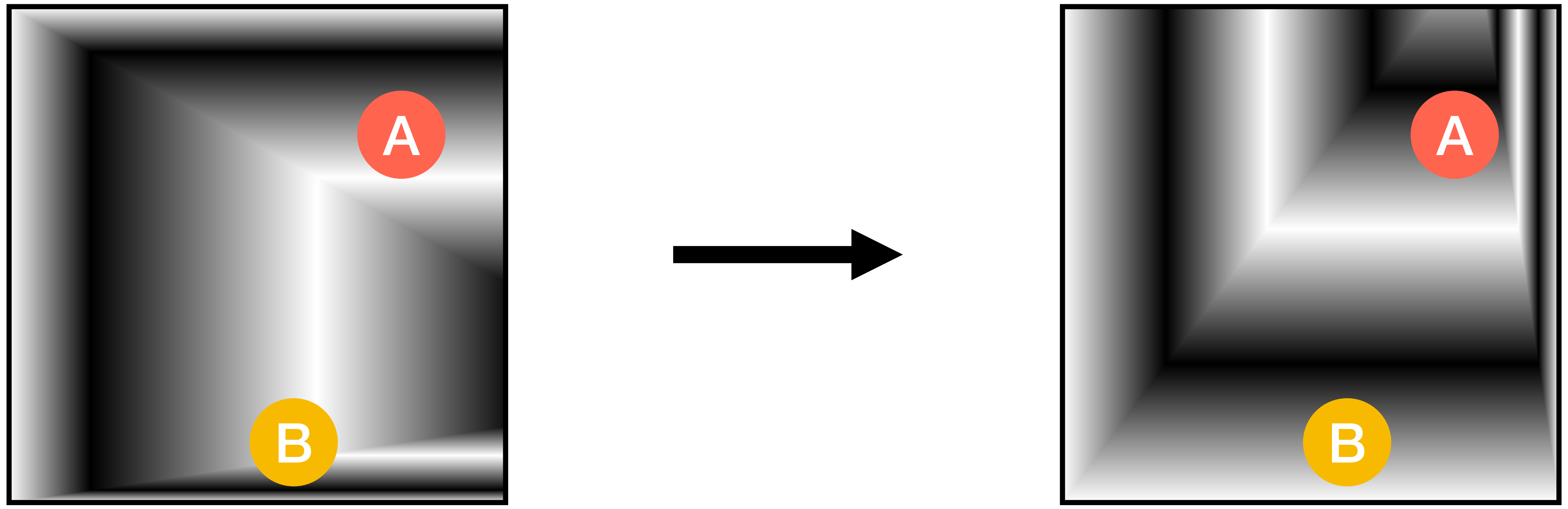
Learnable Encodings

Lets try optimizing phase and frequency s.t. A goes to



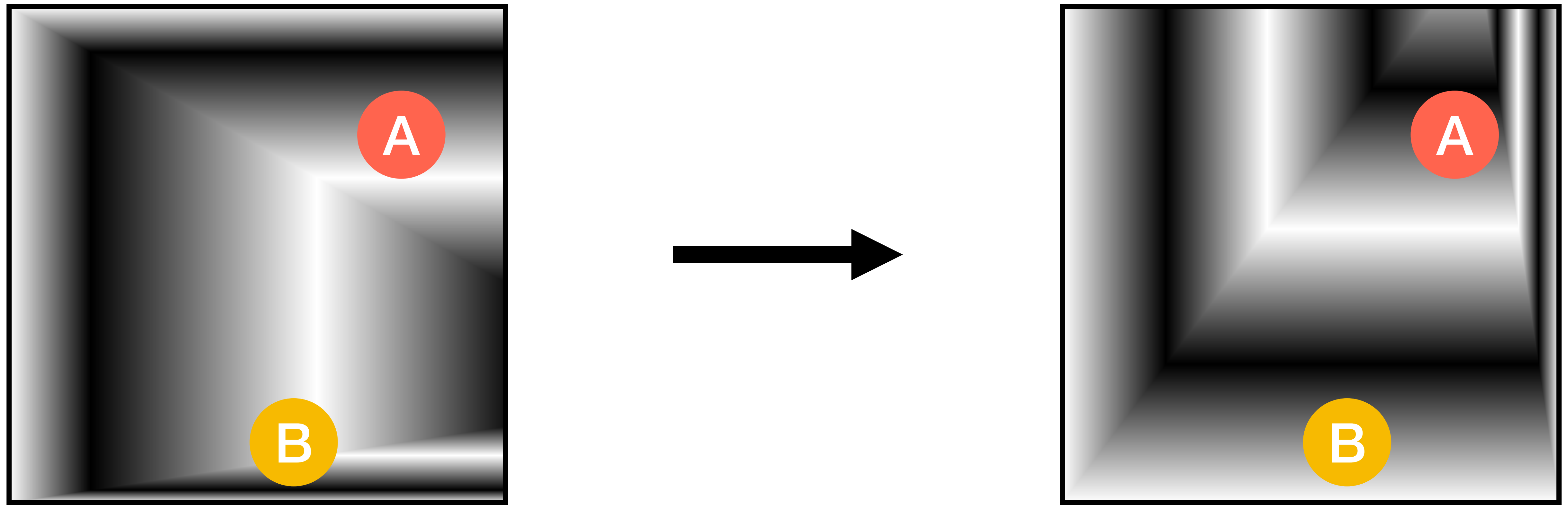
Learnable Encodings

Lets try optimizing phase and frequency s.t. A goes to



Learnable Encodings

Lets try optimizing phase and frequency s.t. A goes to



B also changes which makes optimization difficult

Desired: Learnable encoding with local extent

Desired: Learnable encoding with local extent

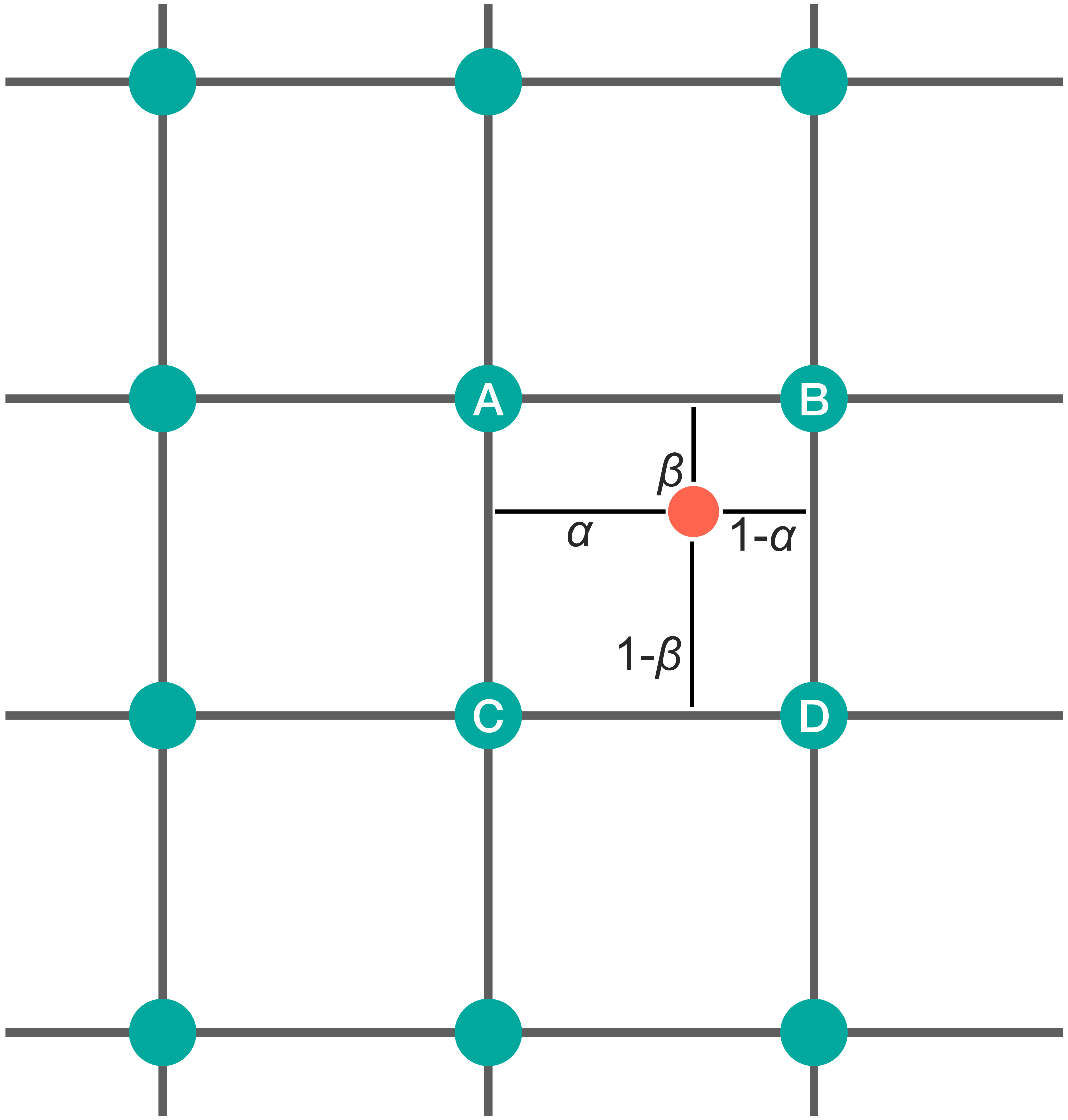
Solution: Feature Grids (i.e. voxel)

Desired: Learnable encoding with local extent

Solution: Feature Grids (i.e. voxel)

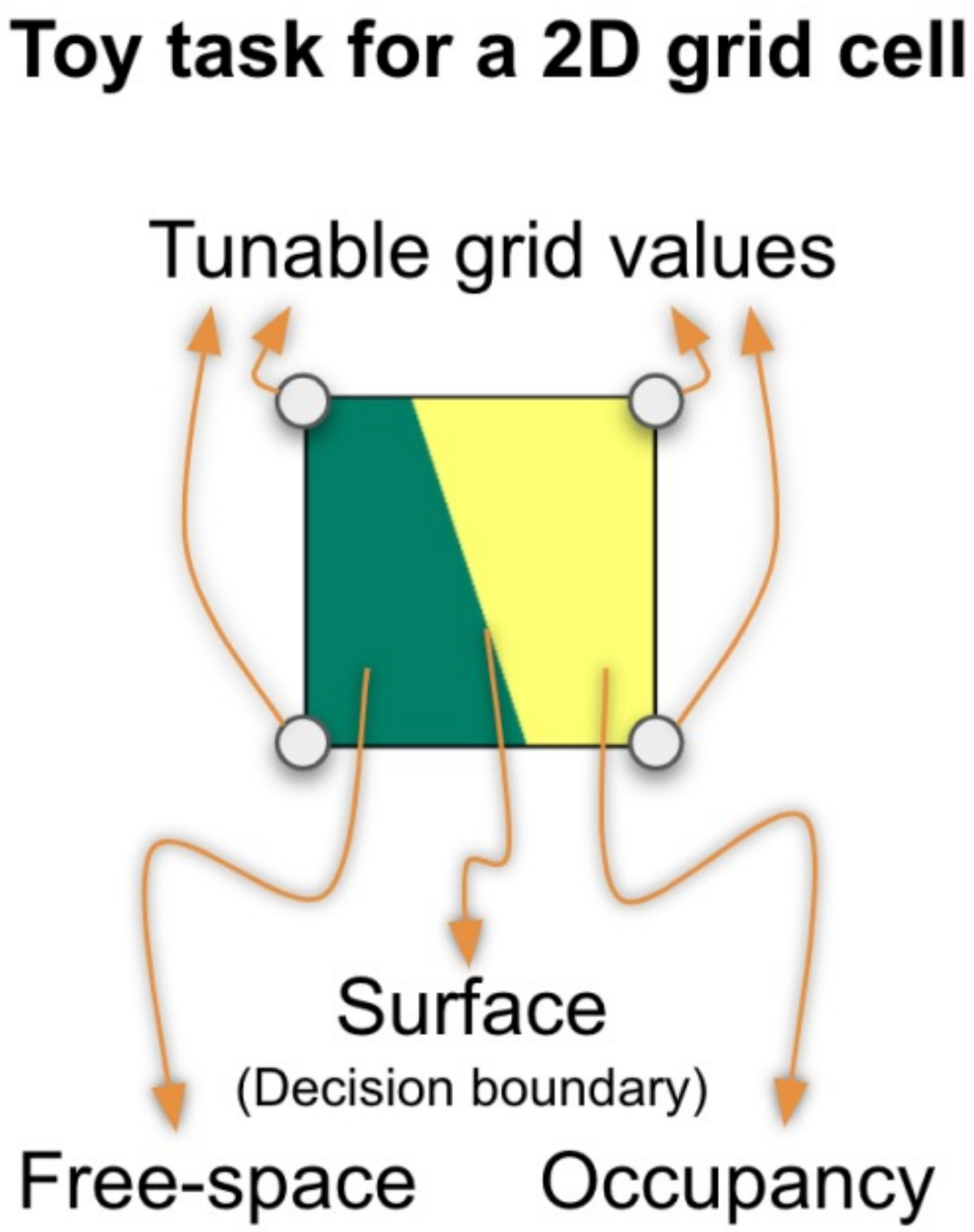
Challenge: Resolution

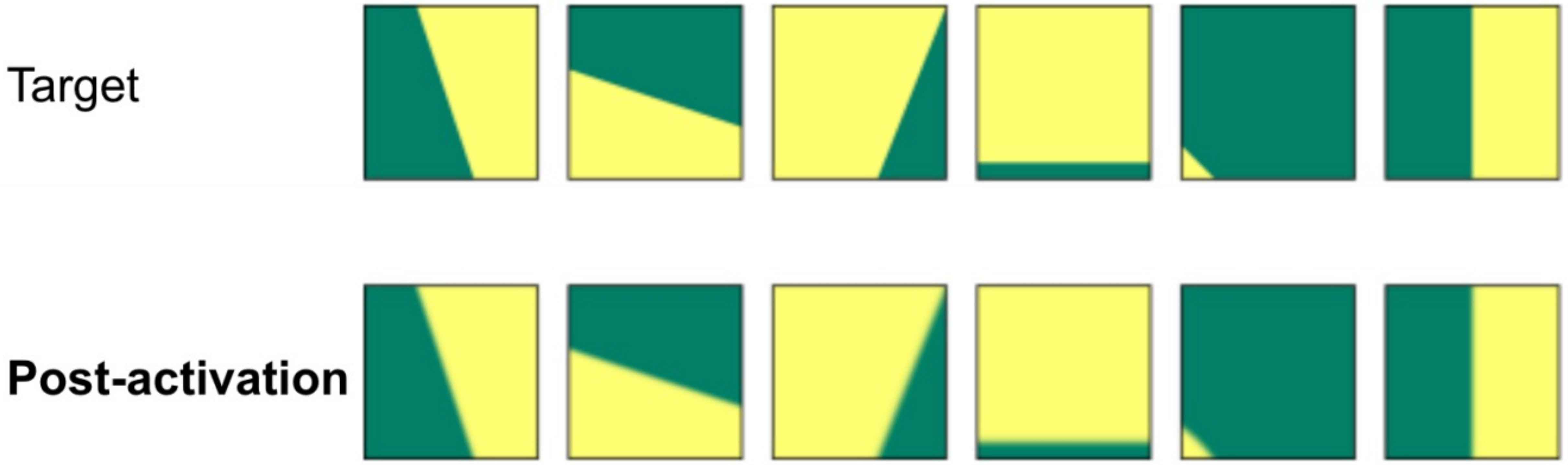
Make continuous via interpolation



$$\bullet = \beta(\alpha A + (1 - \alpha B))$$
$$+(1 - \beta)(\alpha C + (1 - \alpha D))$$

Feature grids can be effective



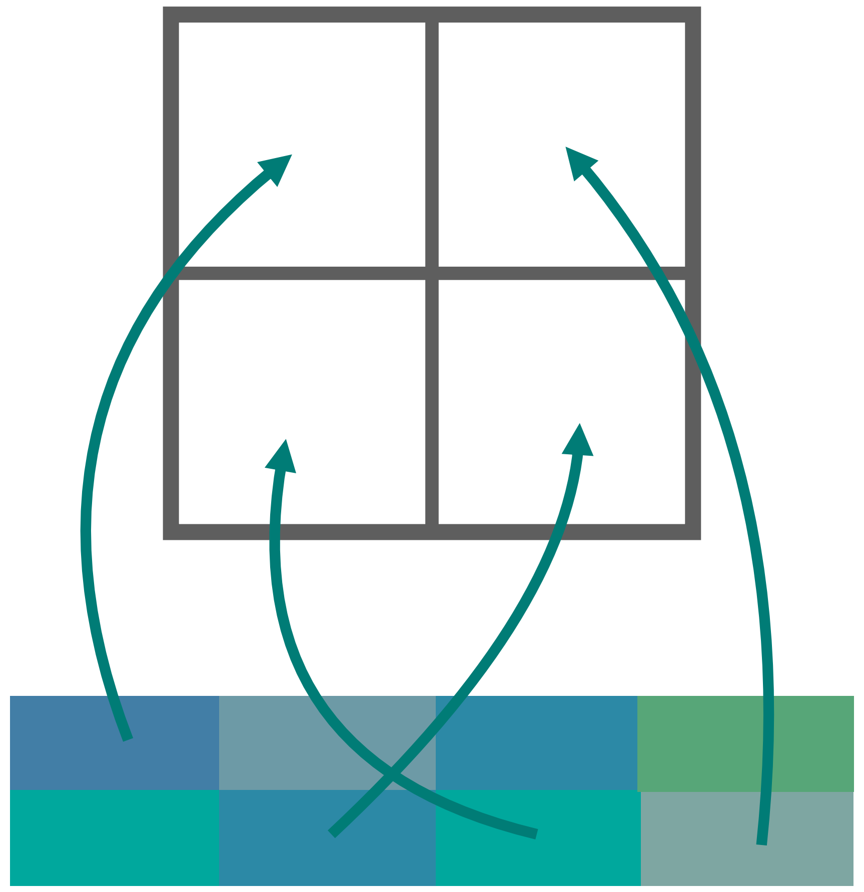


Sun et al. 2021, Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction

Compression Techniques

Sparsity

Low Rank

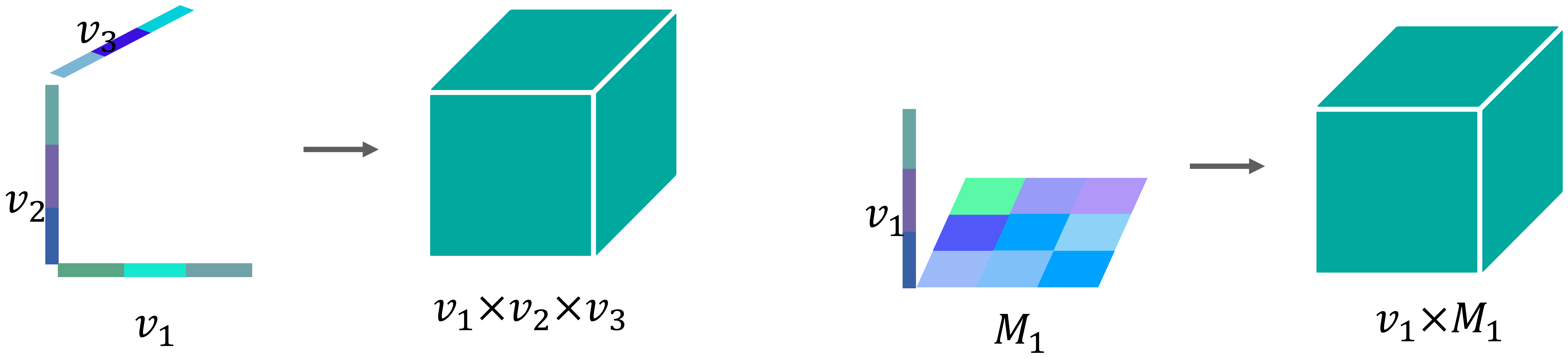


Dictionary

Sparsity

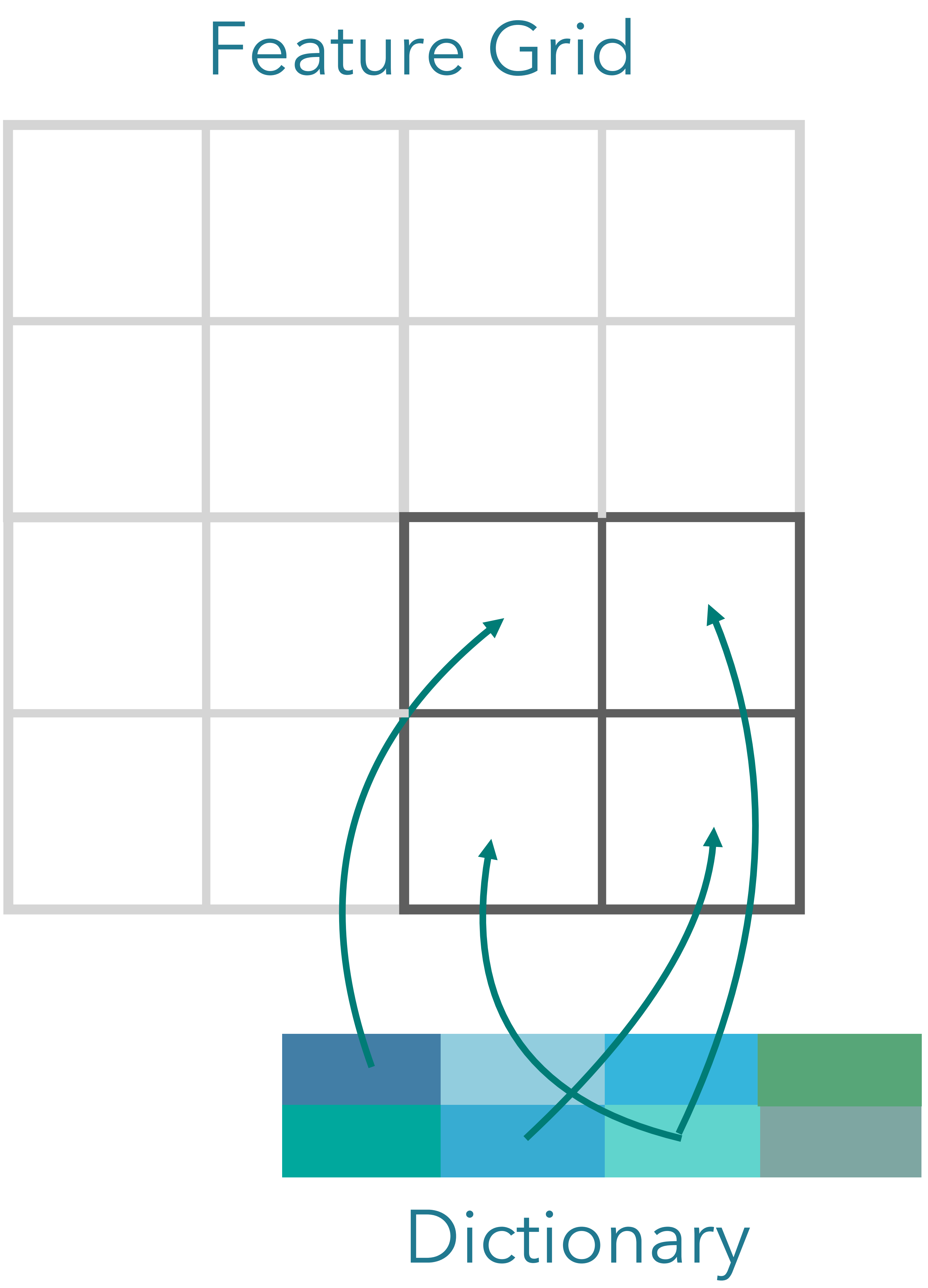
Lui et al. 2020, Neural Sparse Voxel Fields
Yu*, Friedovich-Keil* et al. 2021, Plenoxels: Radiance Fields without Neural Networks
Sun et al. 2021, Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction

Low Rank Approximations



Chen*, Xu* et al. 2022, TensoRF Tensorial Radiance Fields
Chan*, Lin*, Chan*, Nagano* et al. 2022, EG3D: Efficient Geometry-aware 3D Generative Adversarial Networks

Dictionary Methods

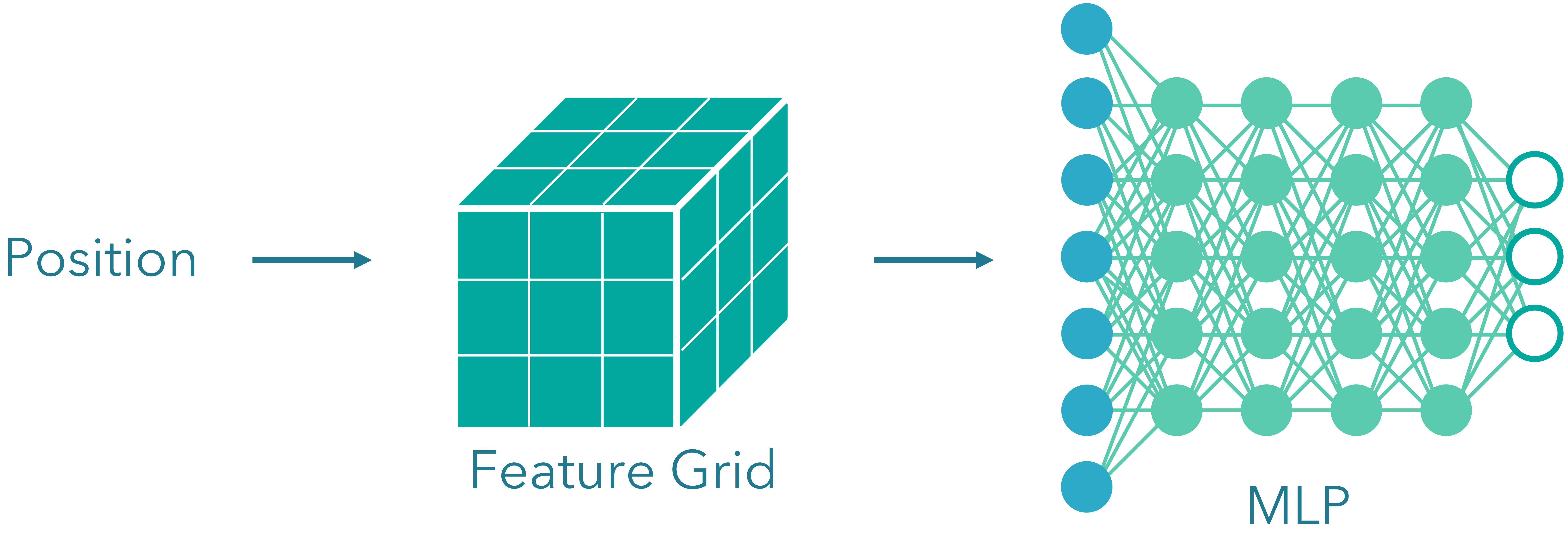


Feature Grid > Dictionary

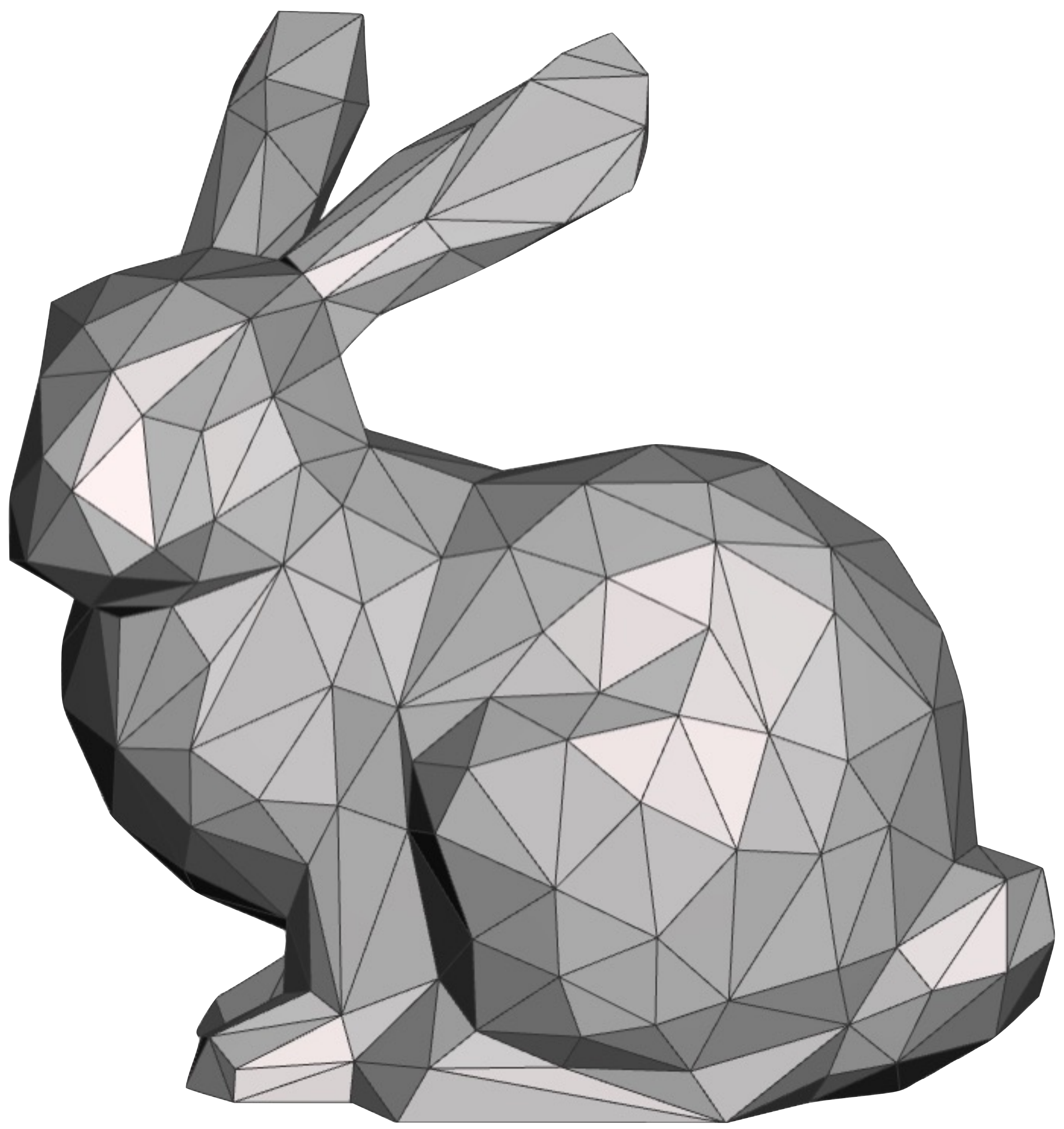
Mapping with collisions

Müller et al. 2022, Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Feature Grids as MLP Encodings

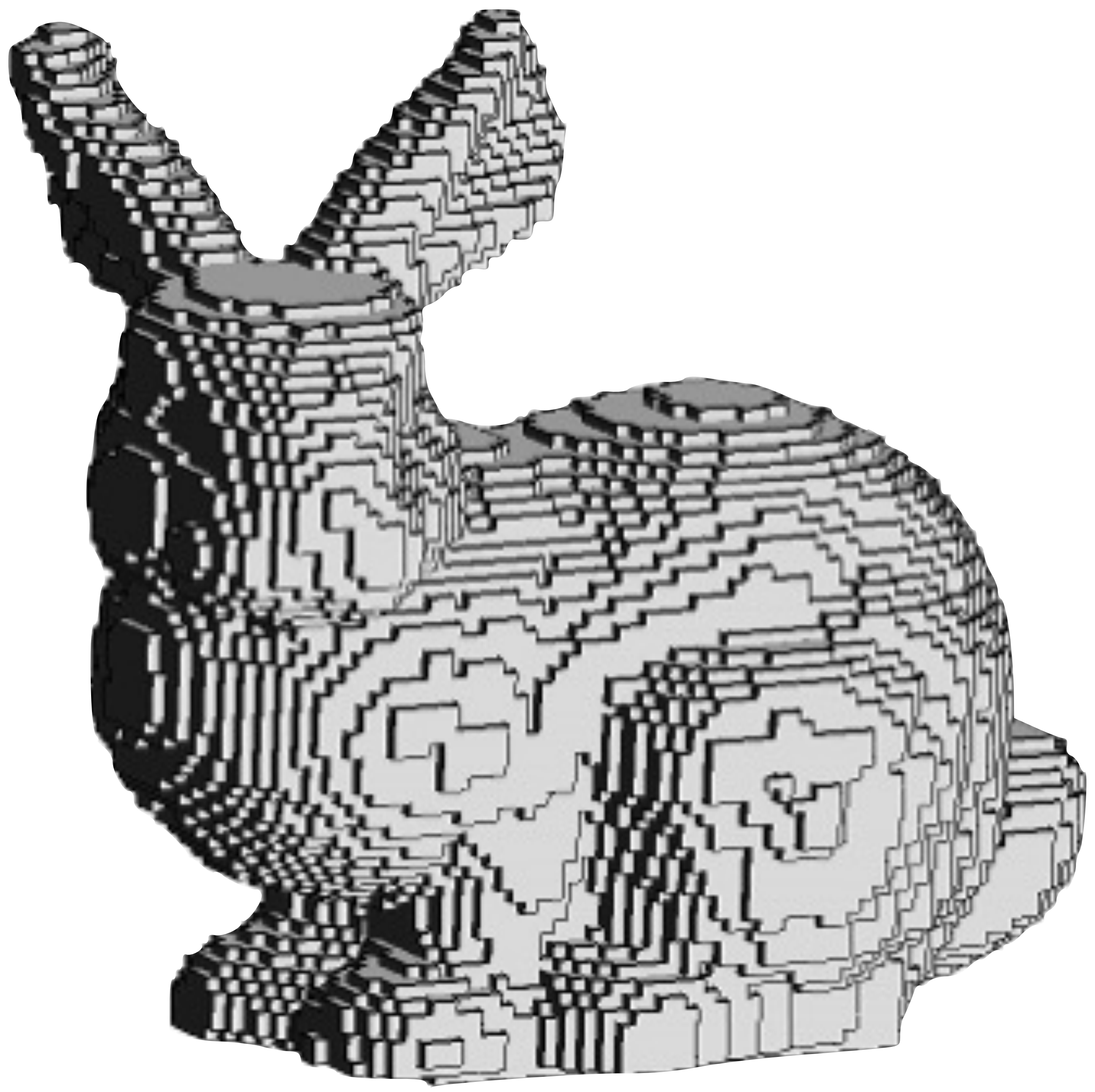


Revisiting Geometry Representations



Mesh Representation

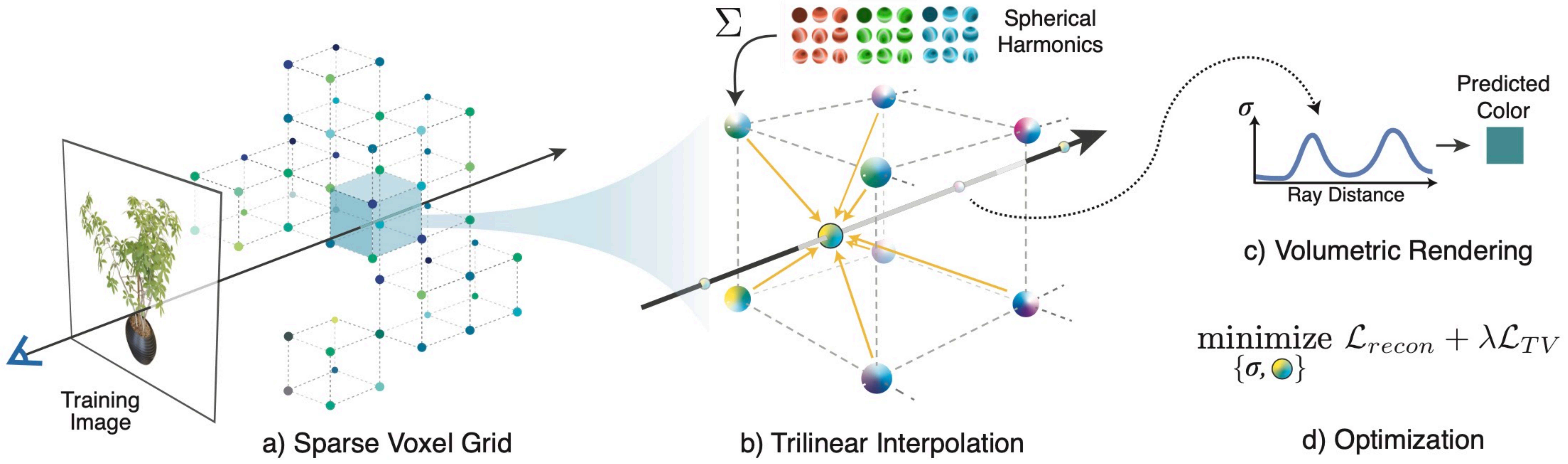
Small memory footprint
Hard to optimize



Voxel Representation

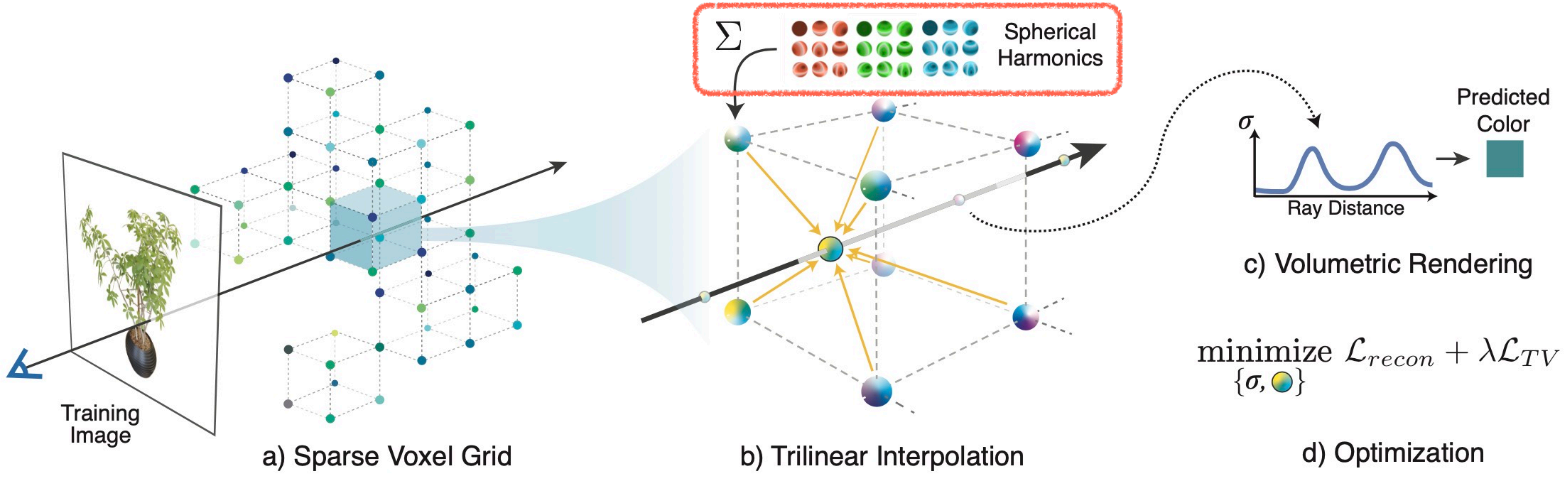
? Easy to optimize
Large memory footprint ?

MLPs are not required…



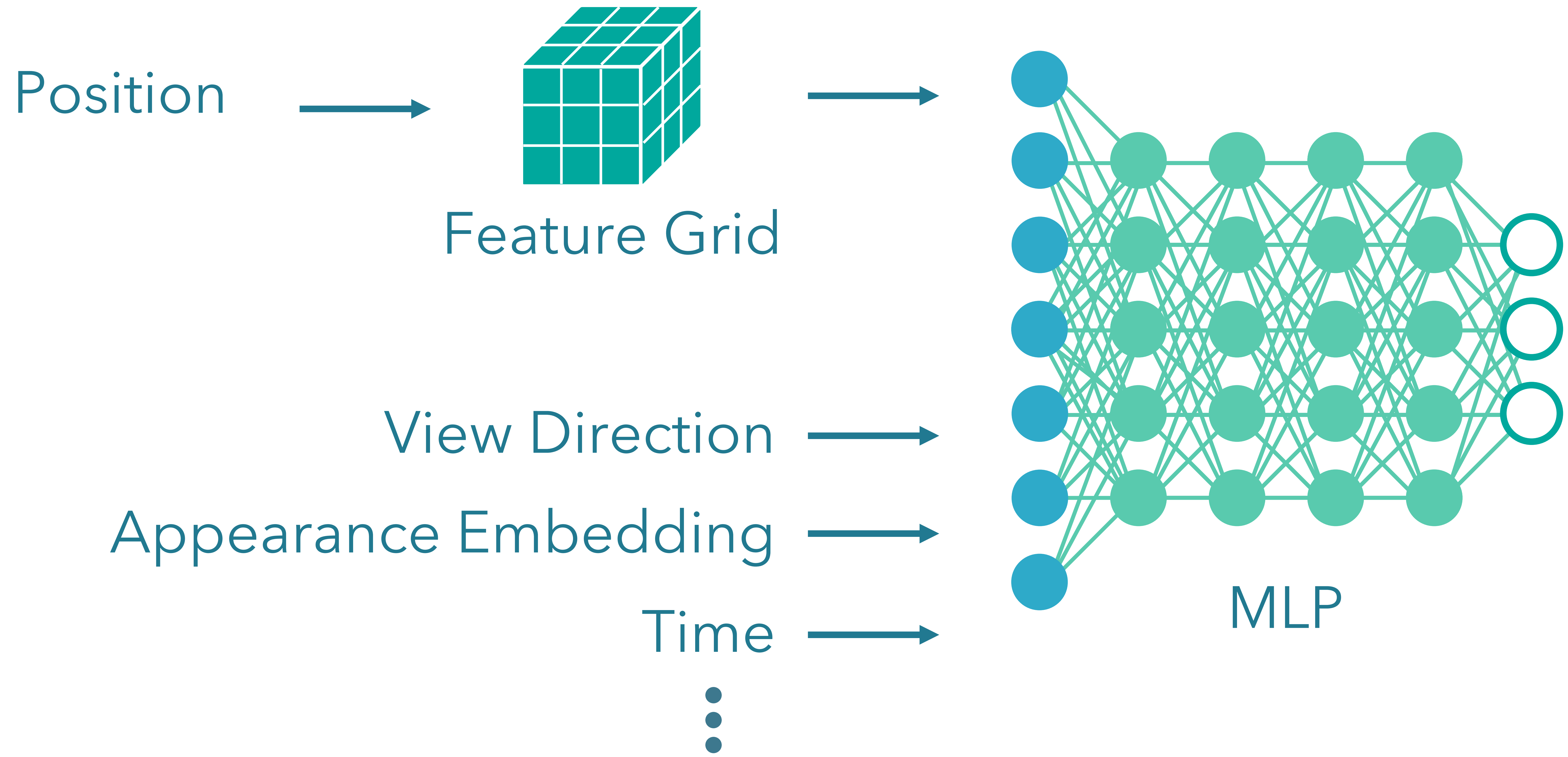
Yu*, Friedovich-Keil* et al. 2021, Plenoxels: Radiance Fields without Neural Networks

MLPs are not required…



Yu*, Friedovich-Keil* et al. 2021, Plenoxels: Radiance Fields without Neural Networks

But MLPs are convenient



Where we are

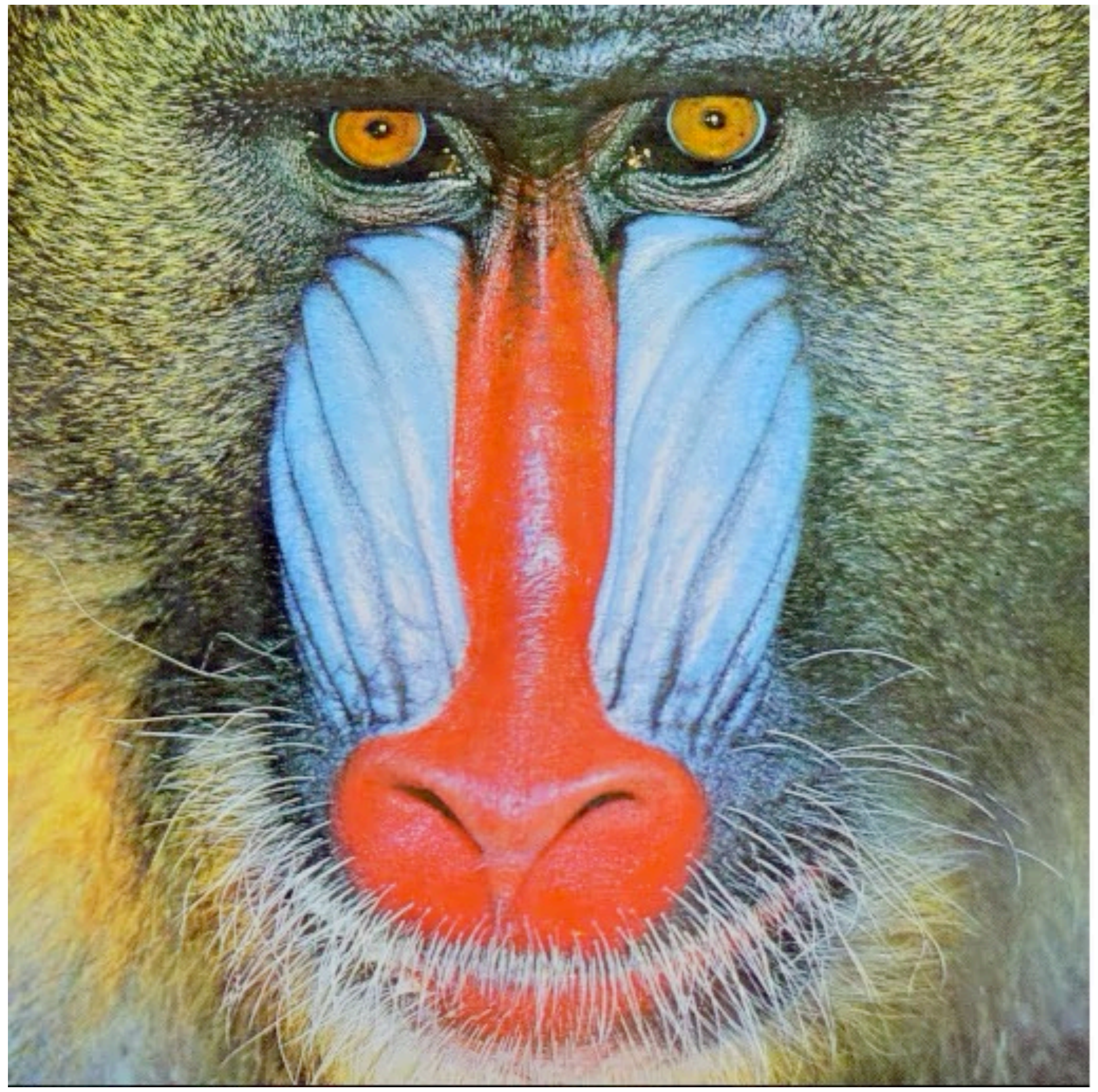
1. Birds Eye View & Background
2. Volumetric Rendering Function
3. Encoding and Representing 3D Volumes
4. **Signal Processing Considerations**
5. Challenges & Pointers

Signal Processing Considerations in NeRF

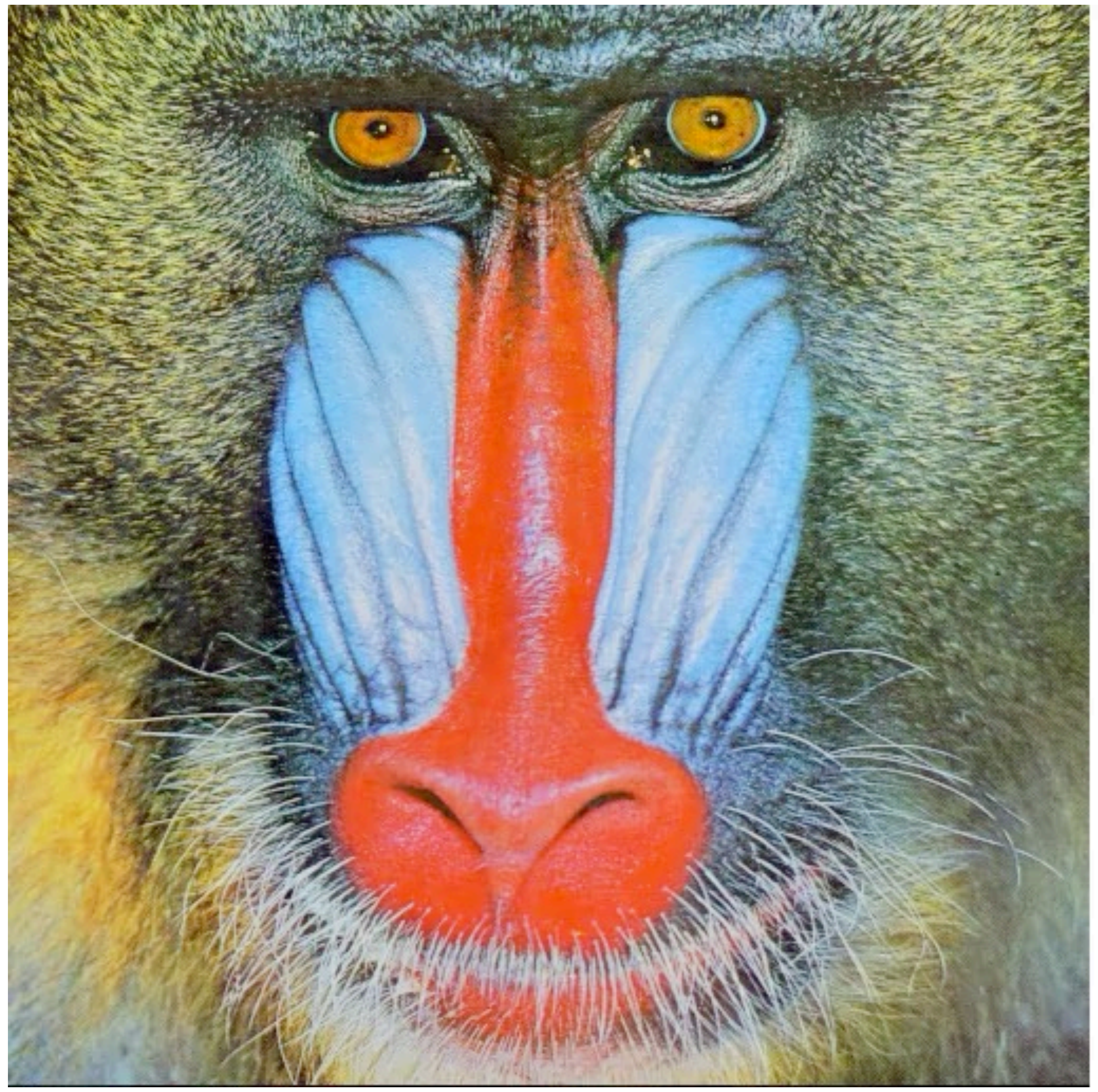
What is happening here?

Naïve (original) NeRF

Review: Aliasing in Image Processing

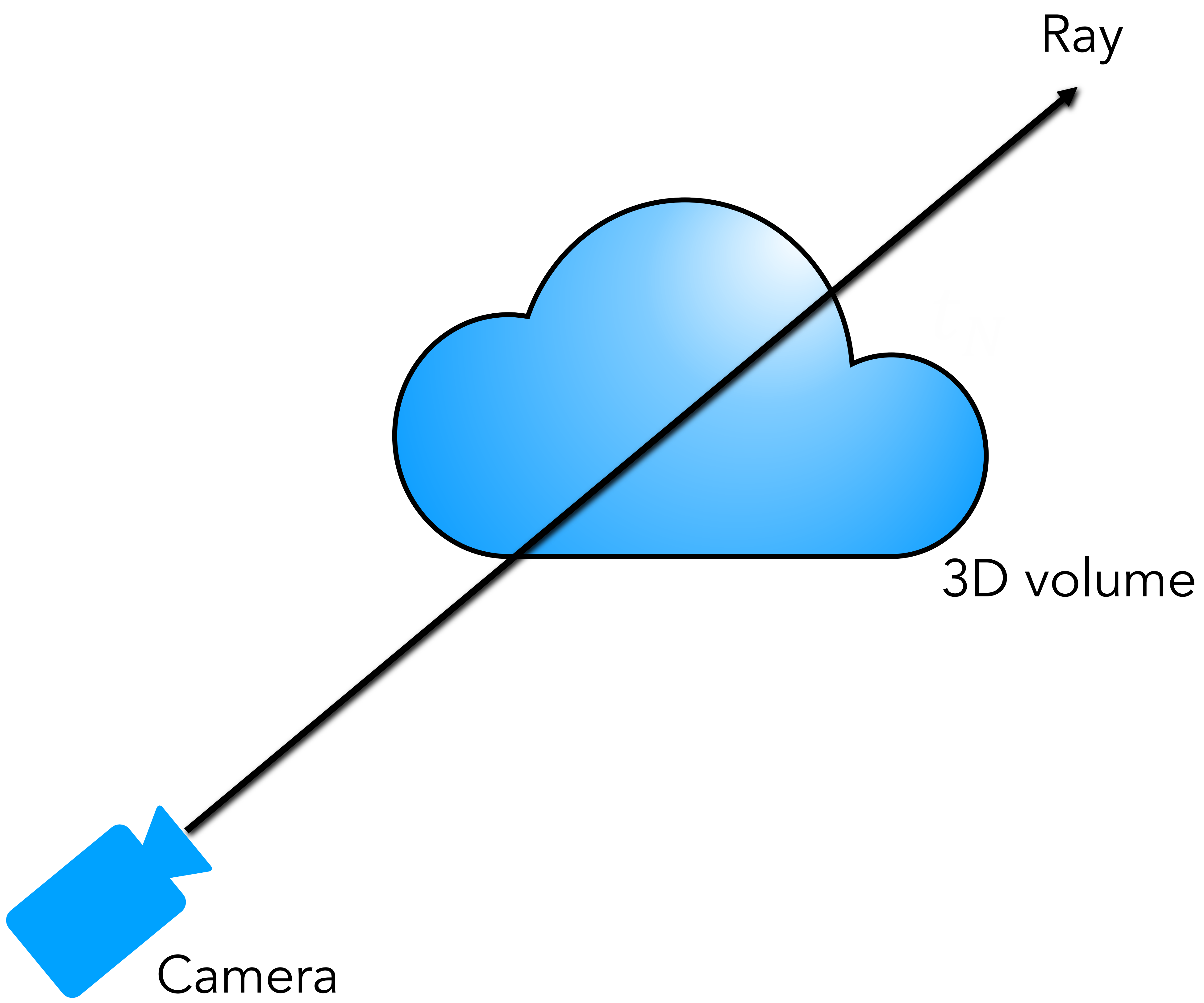


Review: Aliasing in Image Processing

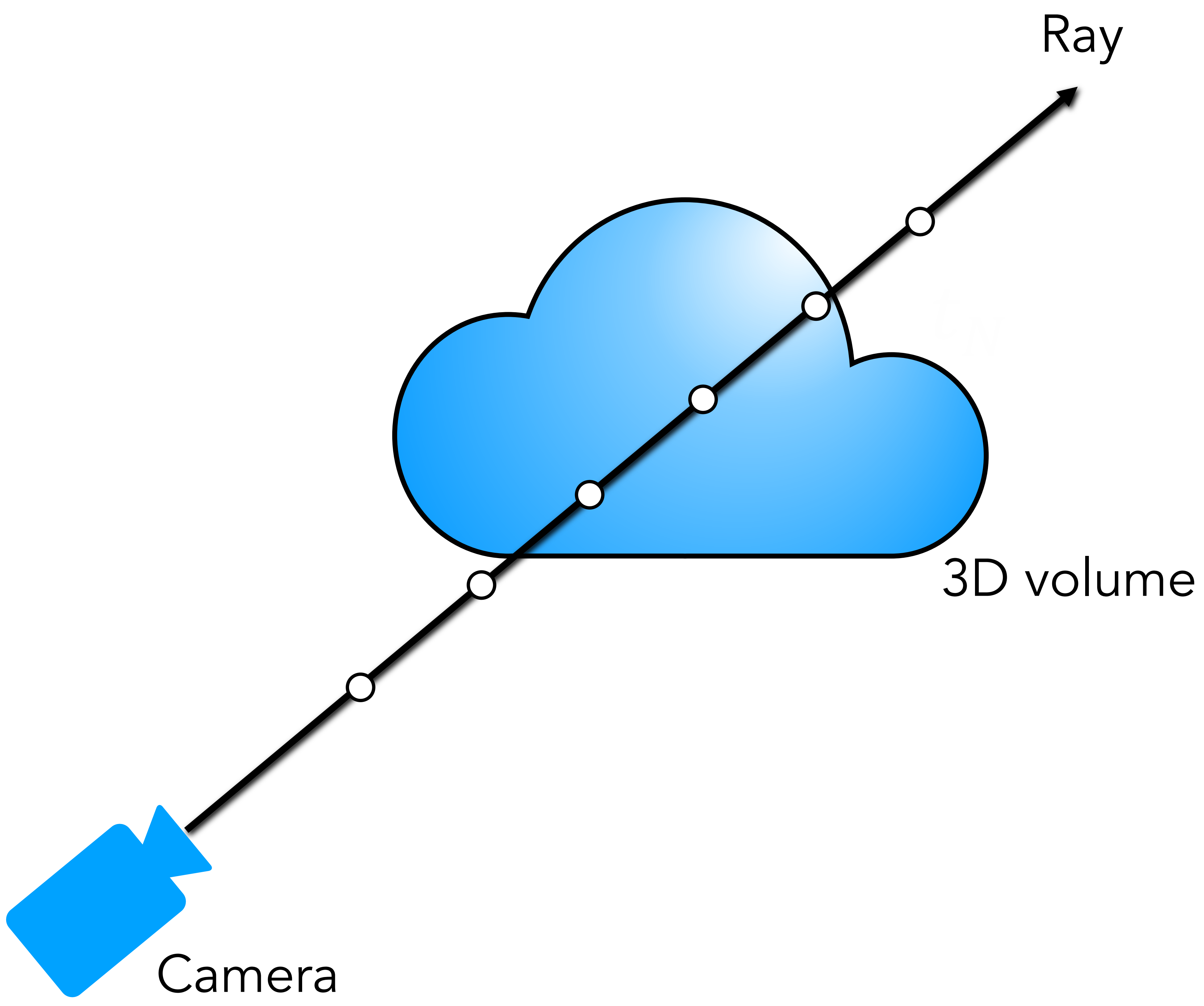


Sampling Along Rays

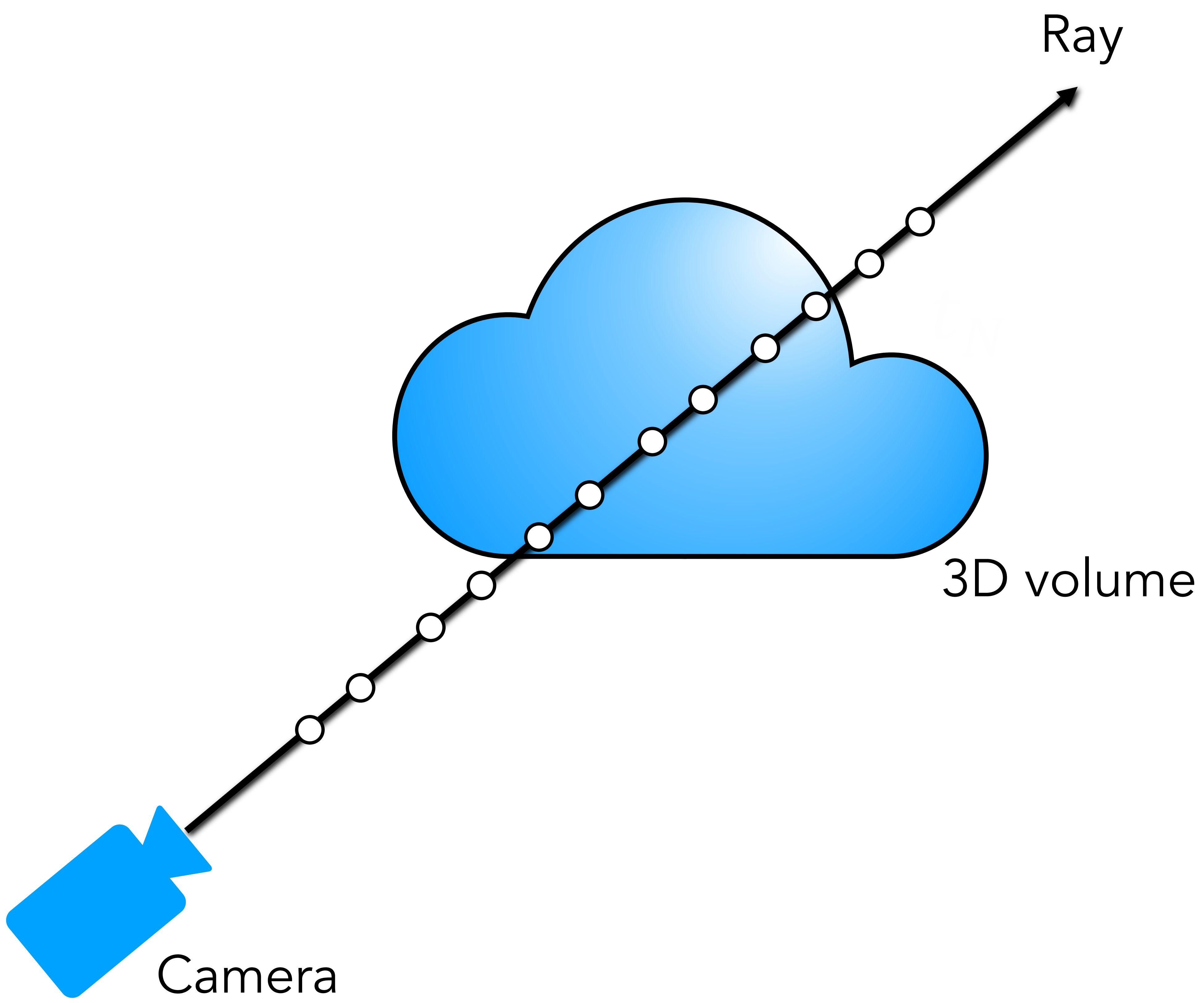
Where to place samples along rays?



How to be more efficient than dense sampling?



How to be more efficient than dense sampling?



Hierarchical Sampling vs. Acceleration Structures

Hierarchical Sampling vs. Acceleration Structures

Hierarchical Sampling

Iteratively use samples from NeRF to more efficiently sample visible scene content

Hierarchical Sampling vs. Acceleration Structures

Hierarchical Sampling

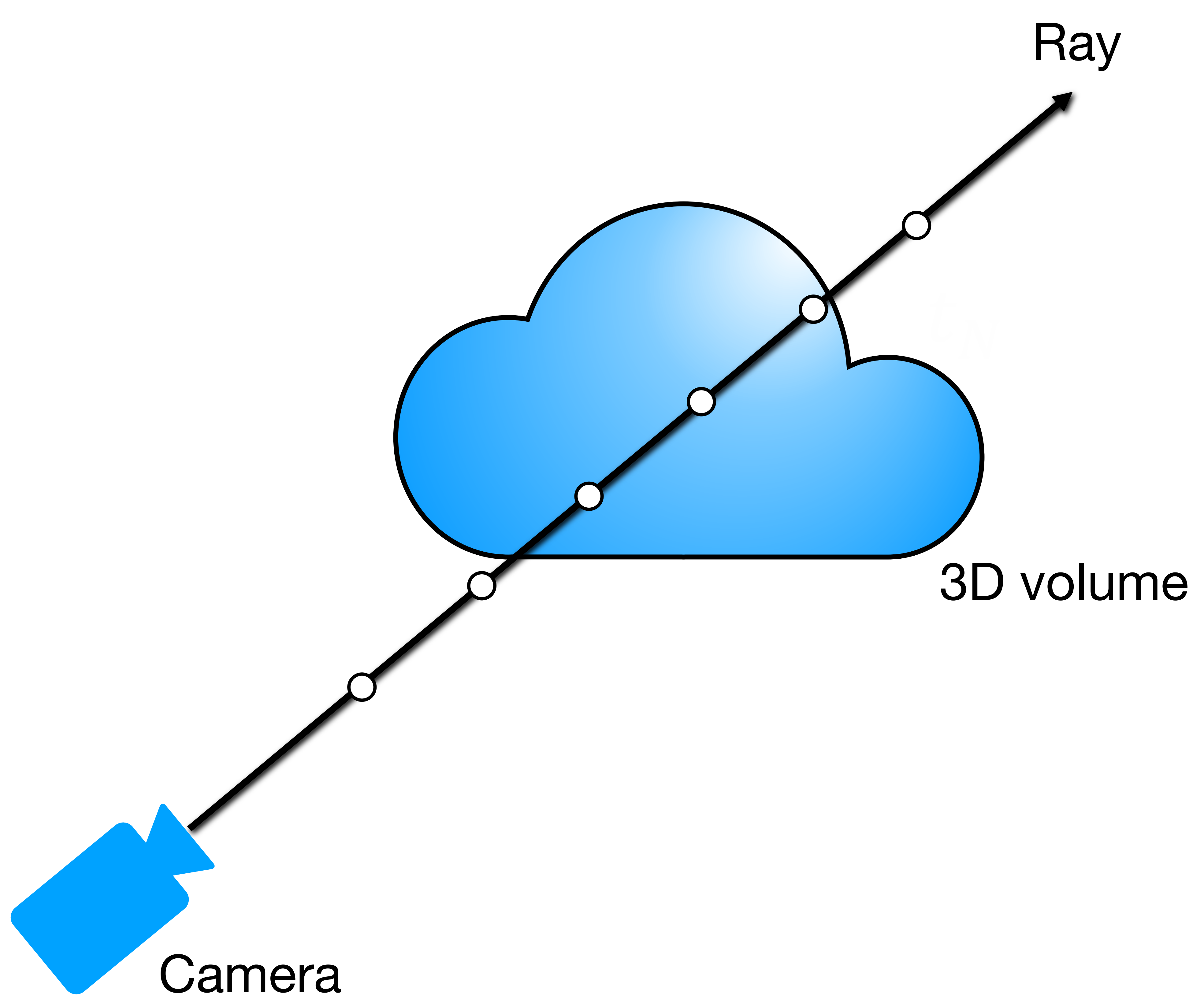
Iteratively use samples from NeRF to more efficiently sample visible scene content

Acceleration Structures

Distill/cache properties of NeRF into a structure that helps generate samples

Hierarchical ray sampling

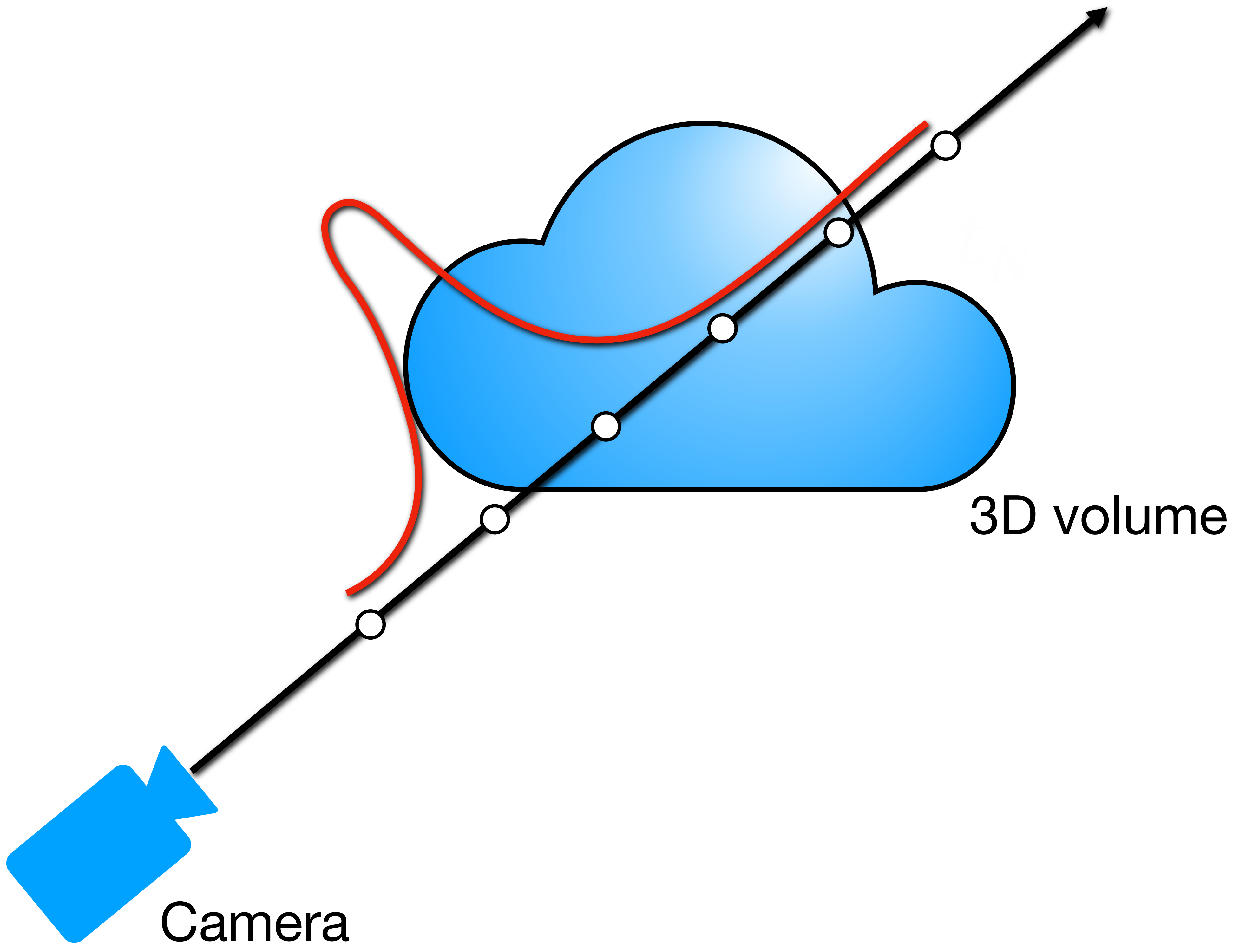
Key Idea: sample points proportionally to expected effect on final rendering



Key Idea: sample points proportionally to expected effect on final rendering

$$C \approx \sum_{i=1}^{N} \boxed{T_i \alpha_i} c_i$$

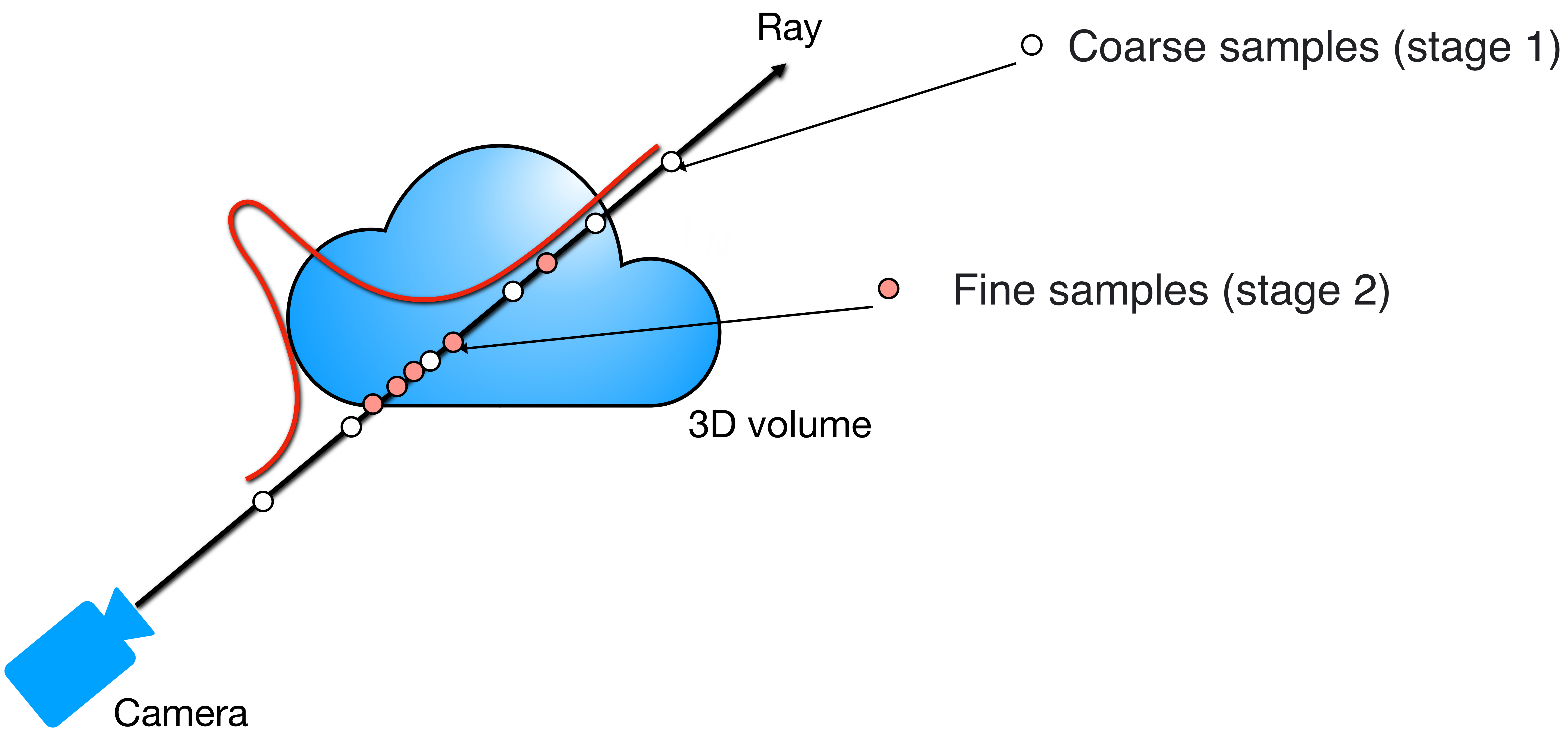
treat weights as probability distribution for new samples



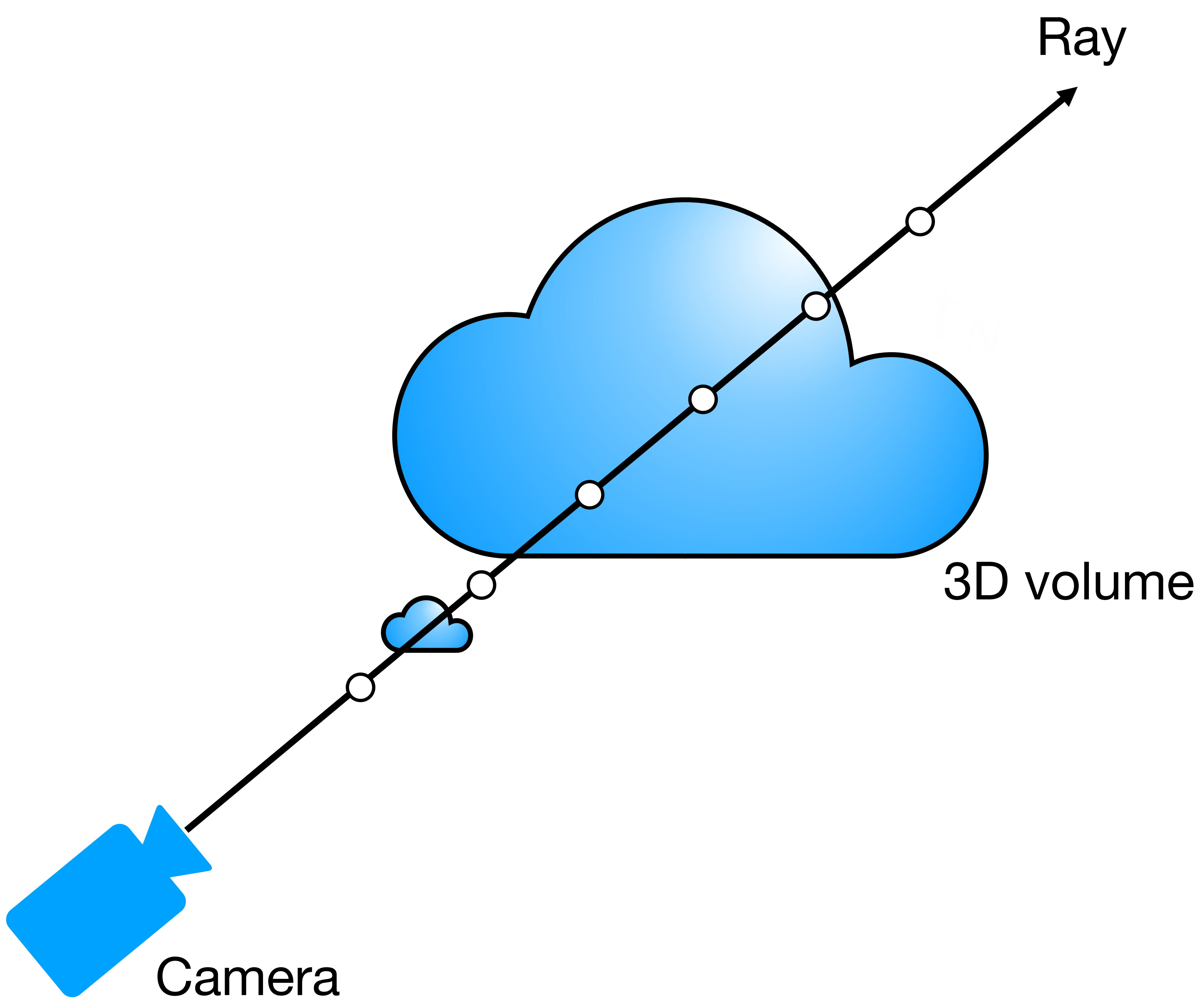
Key Idea: sample points proportionally to expected effect on final rendering

$$C \approx \sum_{i=1}^{N} \boxed{T_i \alpha_i} c_i$$

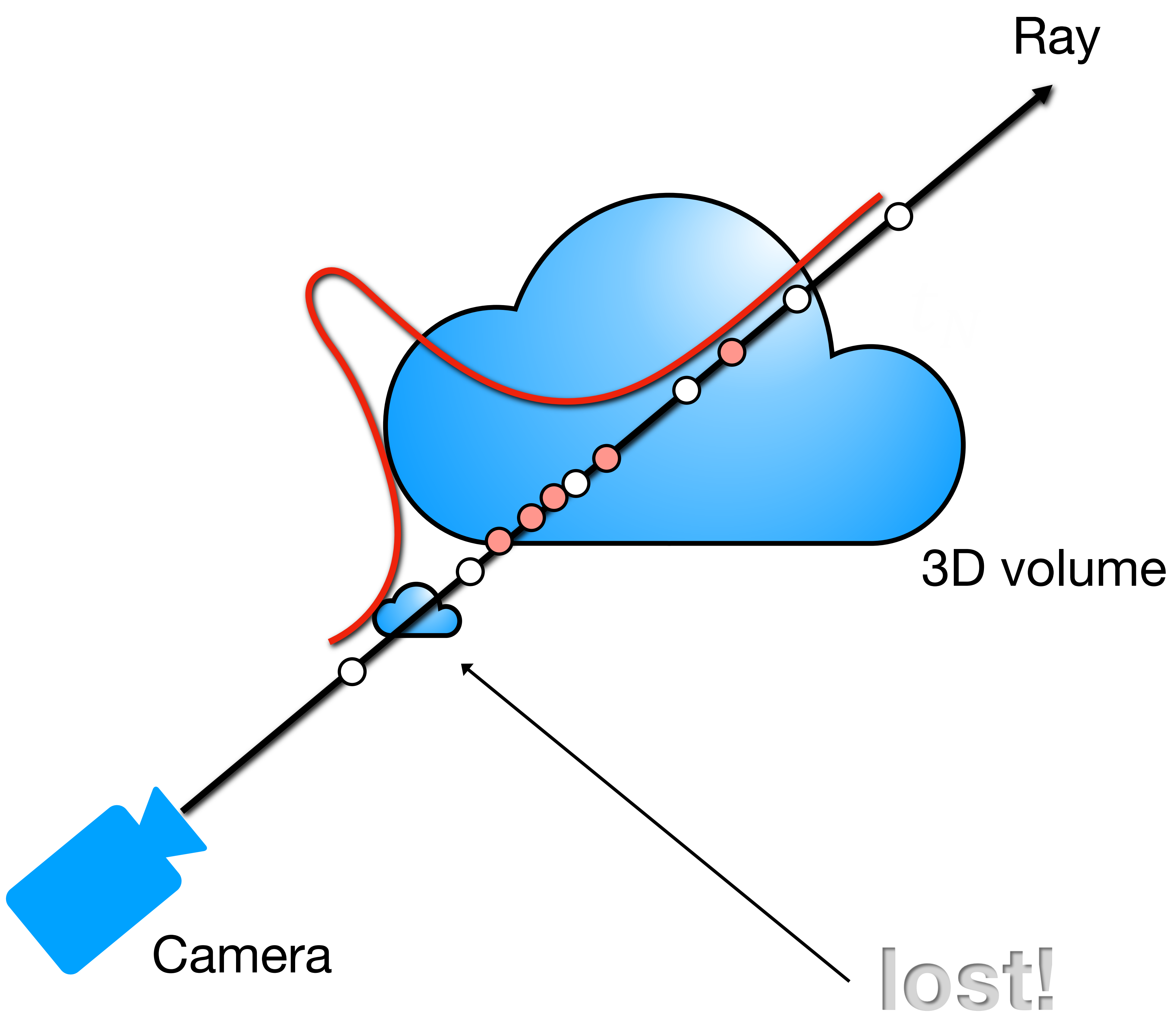
treat weights as probability distribution for new samples



What about aliasing during coarse sampling?

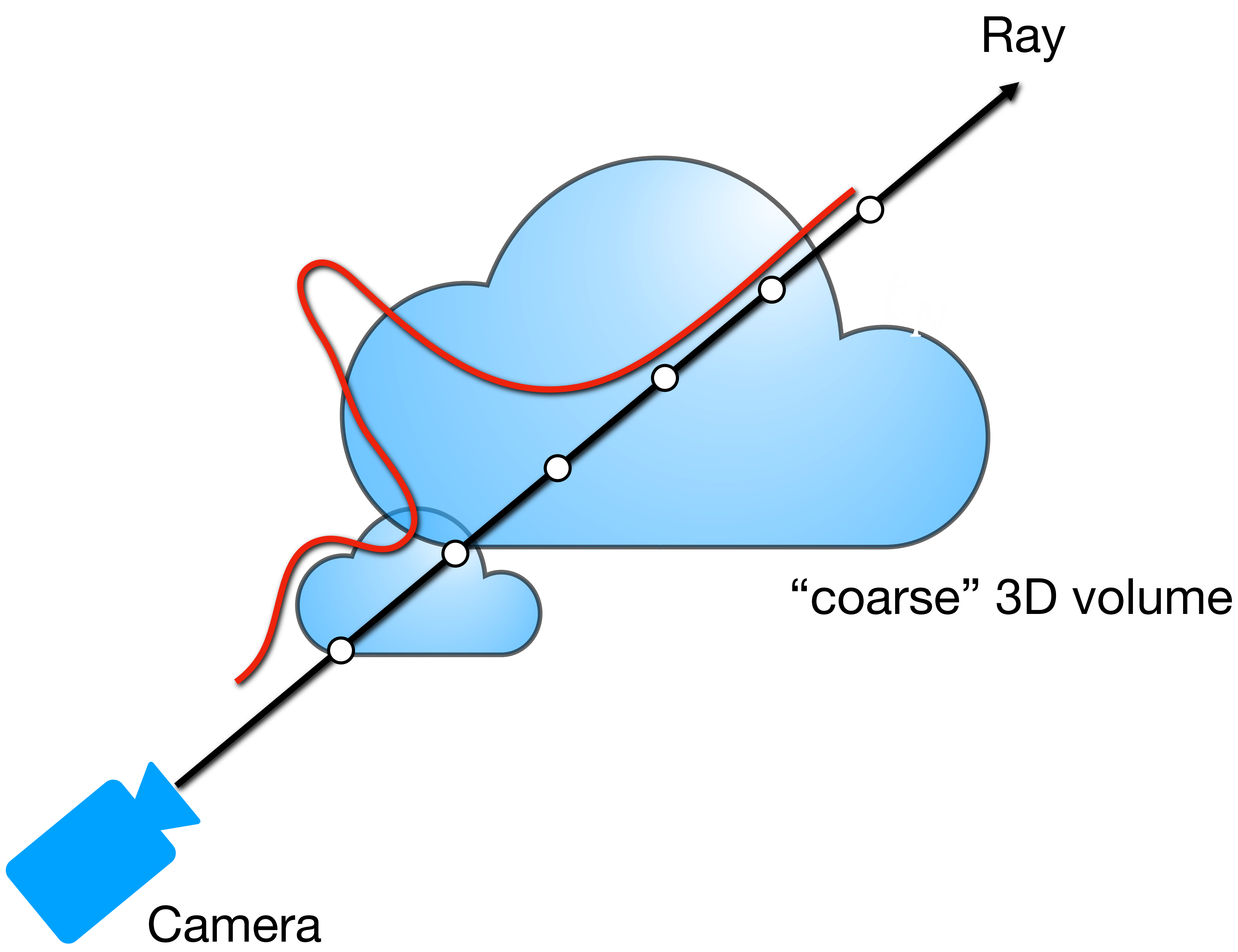


What about aliasing during coarse sampling?



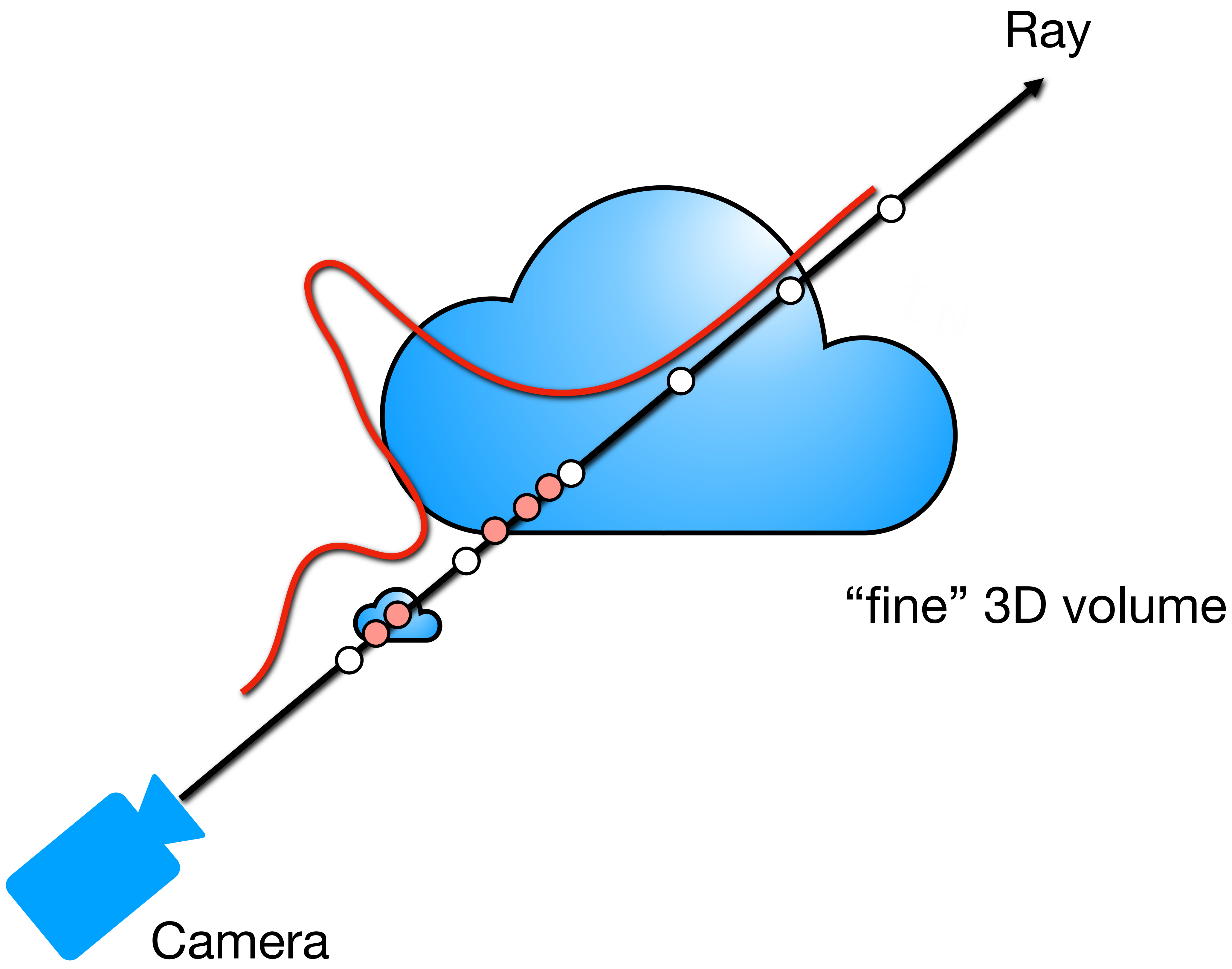
What about aliasing during coarse sampling?

Solution: train two NeRFs! —> lower resolution for first “coarse” level



What about aliasing during coarse sampling?

Solution: train two NeRFs! —> higher resolution for second “fine” level

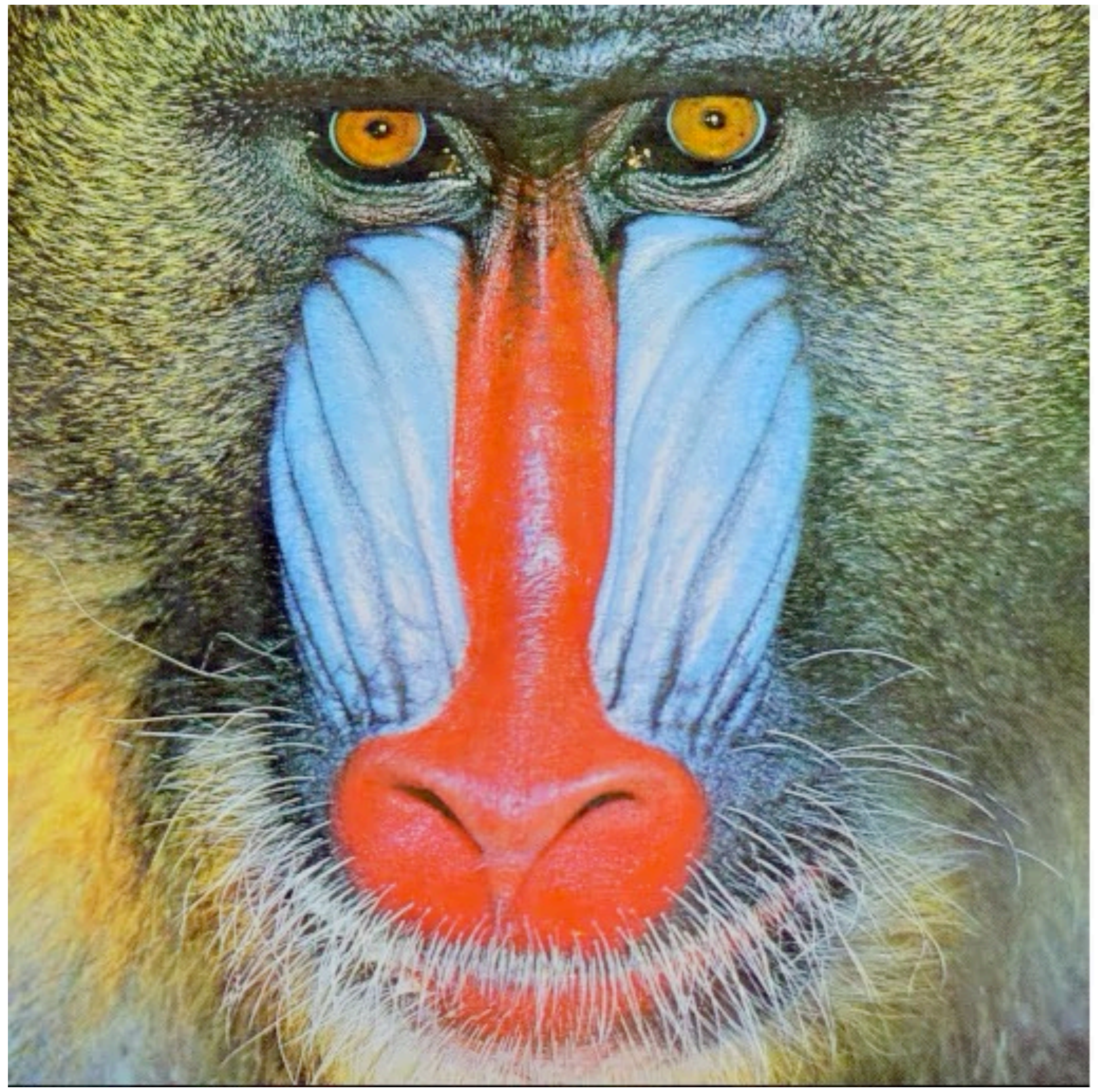


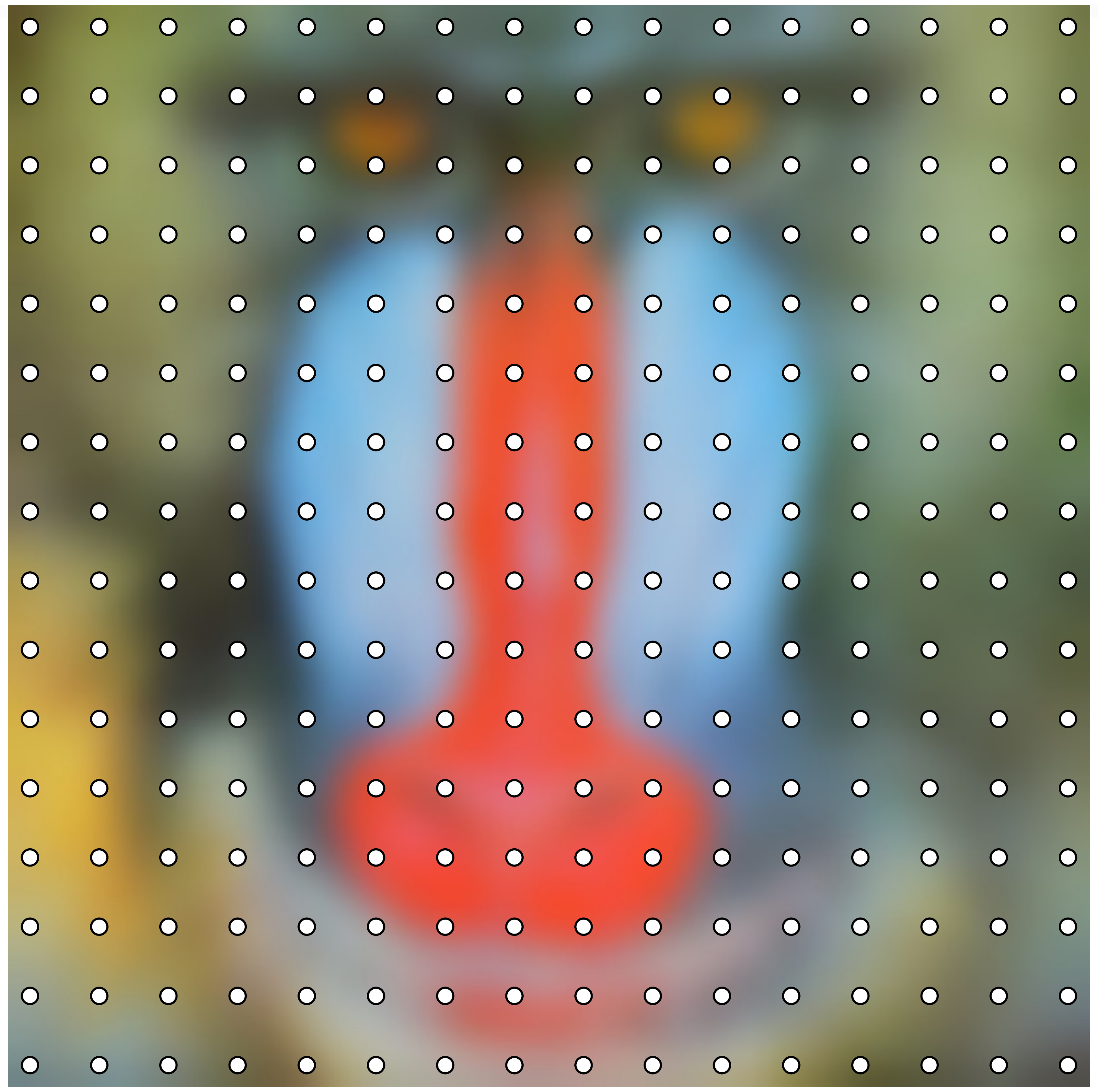
More anti-aliasing

can we avoid training two networks?

Aliasing in NeRF renderings

Recall that averaging reduces aliasing

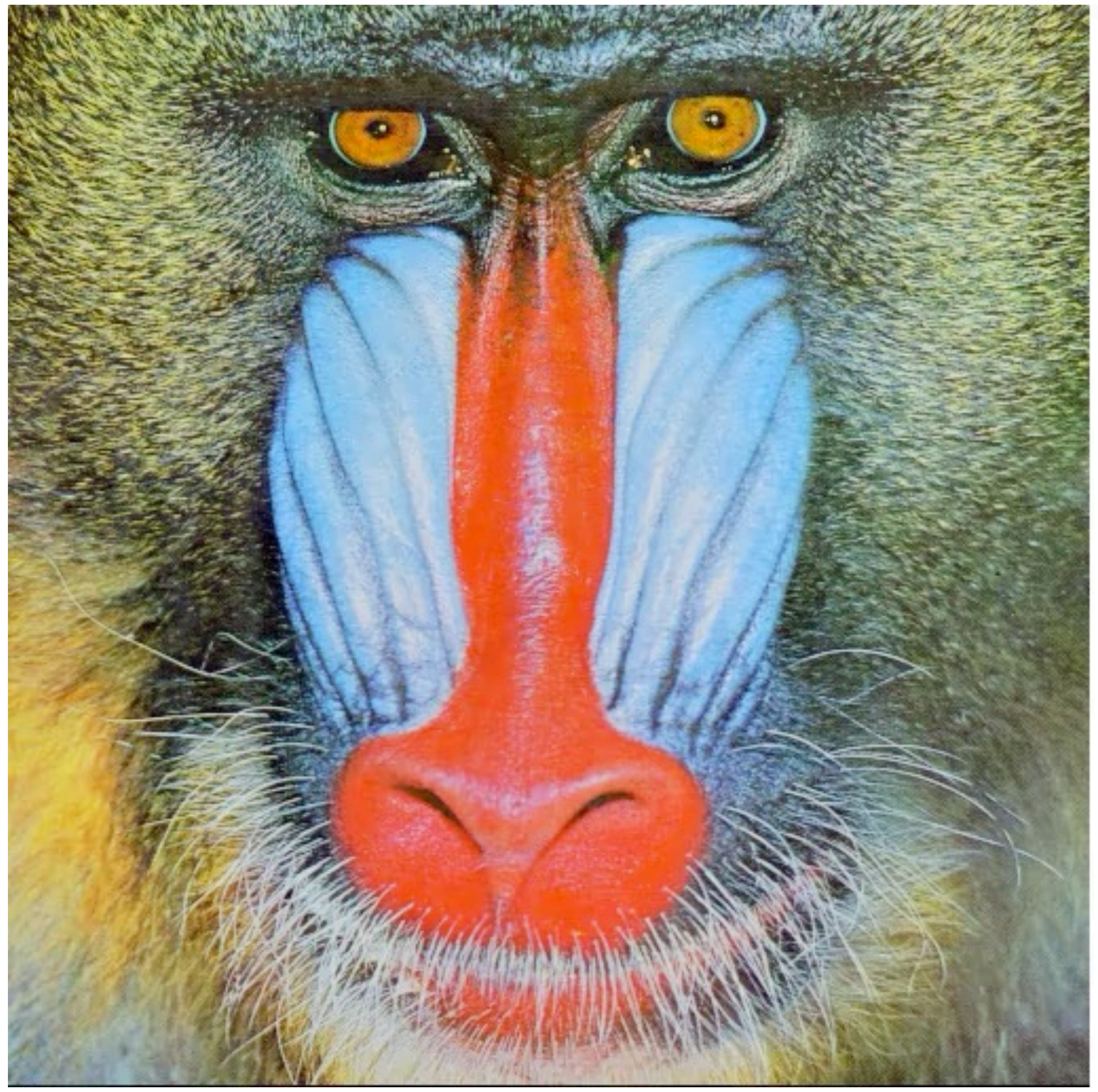




Filter

Sample

But repeatedly sampling and averaging is inefficient



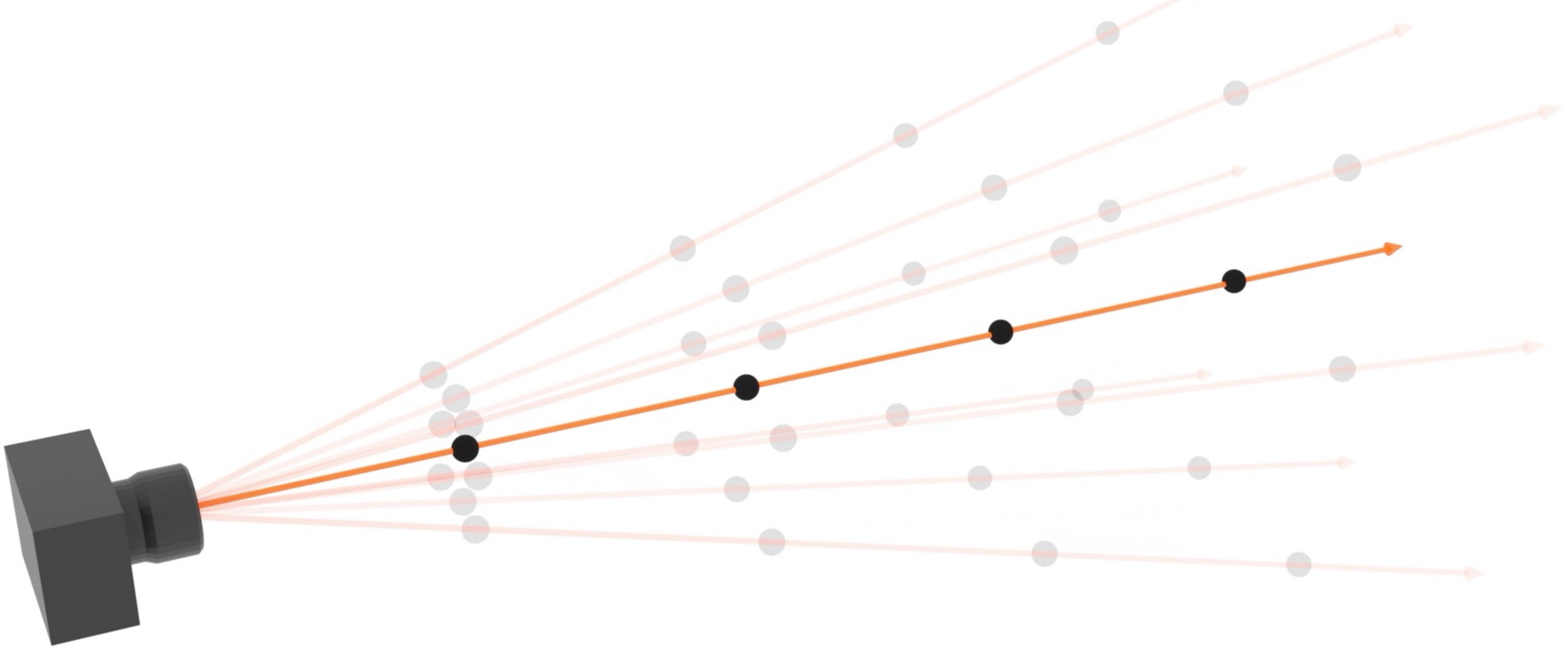
Filter

Sample

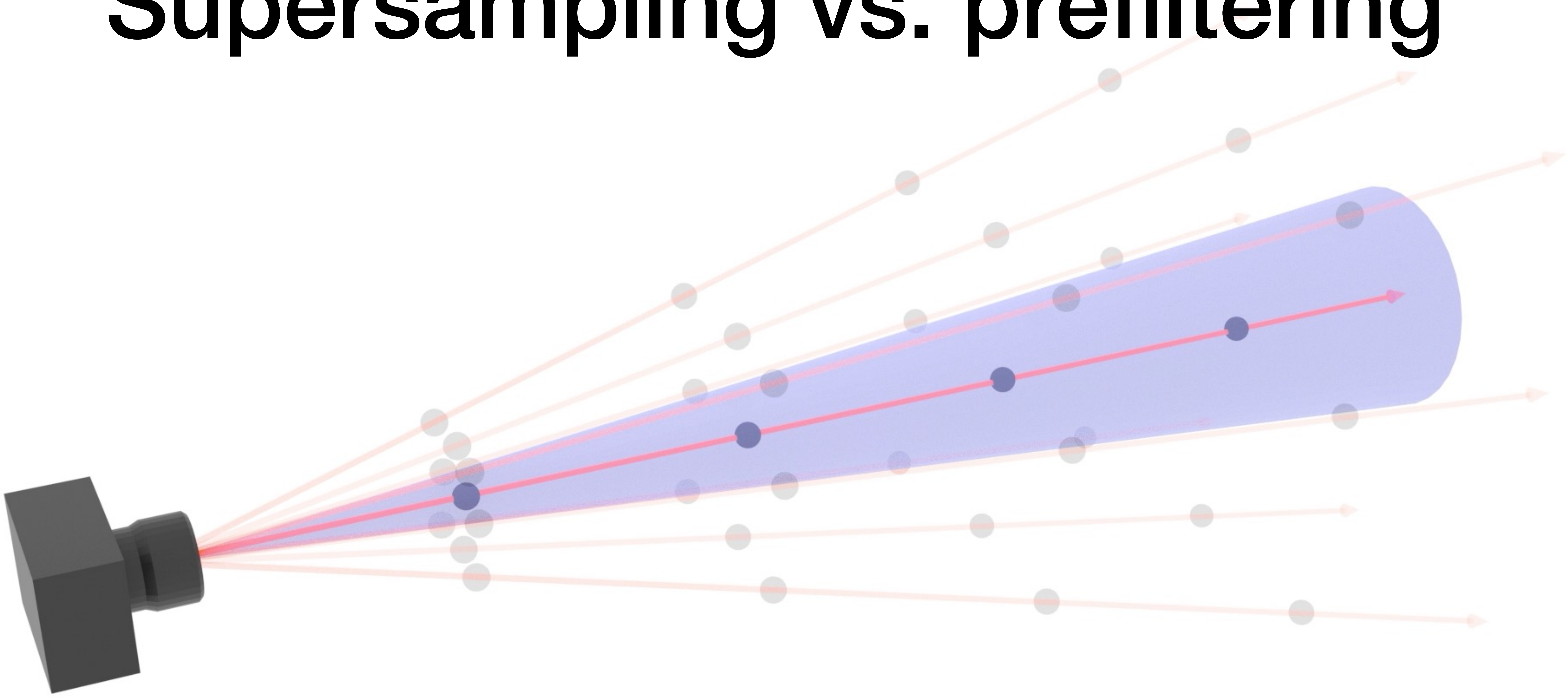
Standard solution: prefiltering with a mipmap

Pyramidal Parametrics, Lance Williams, SIGGRAPH 1983

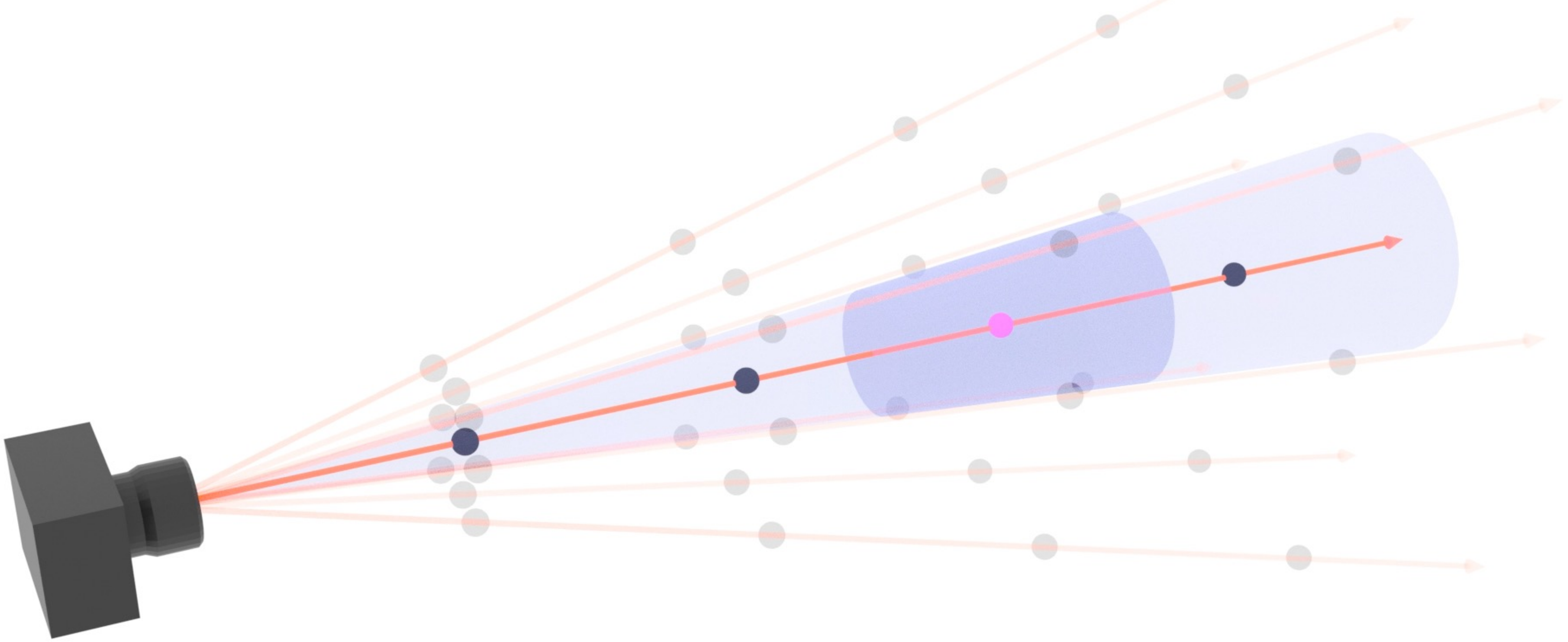
Antialiasing requires average ray color within pixel



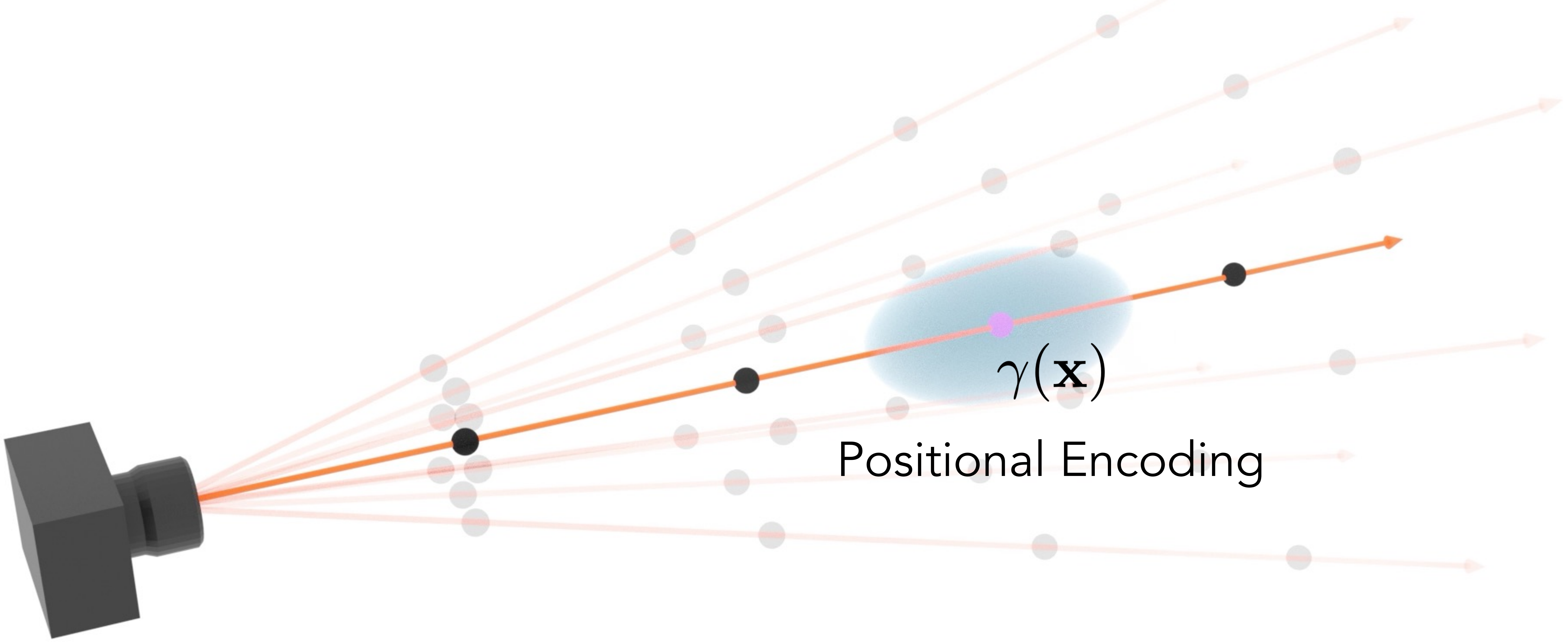
Supersampling vs. prefiltering



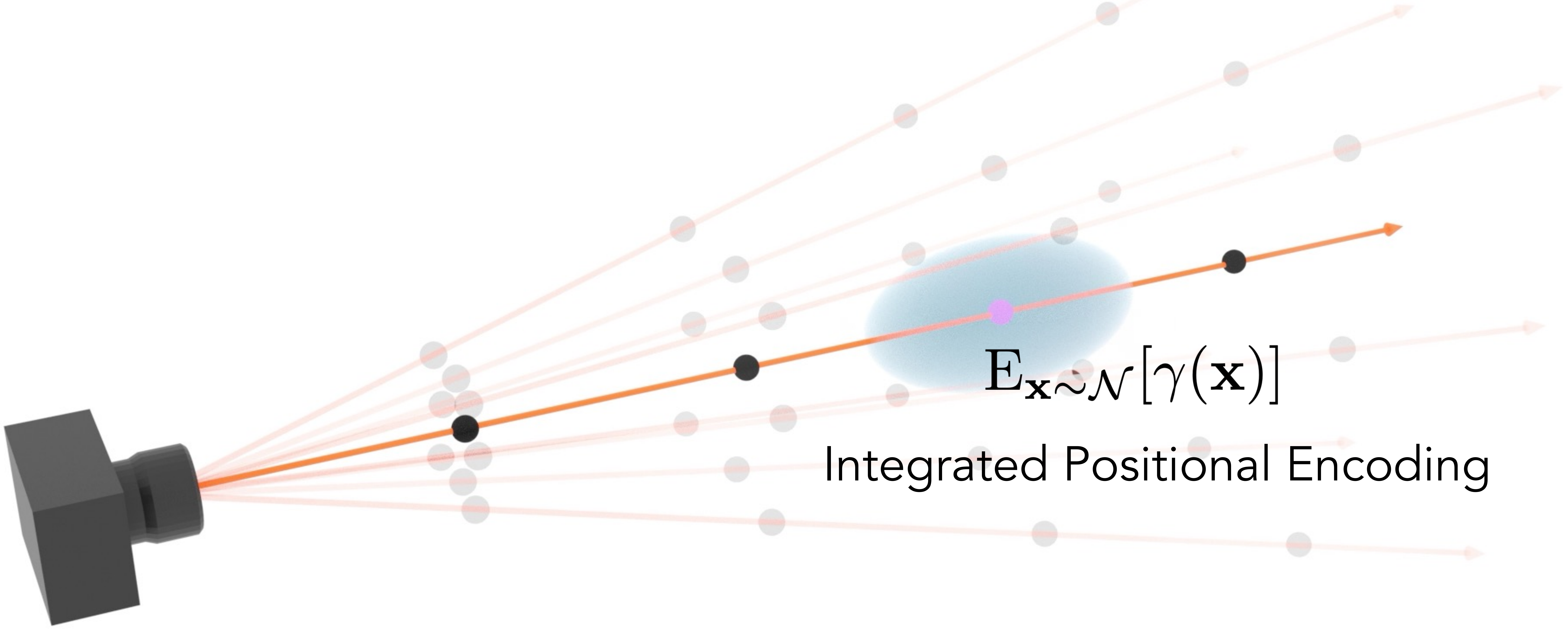
Want NeRF to represent *integrals* within frustum



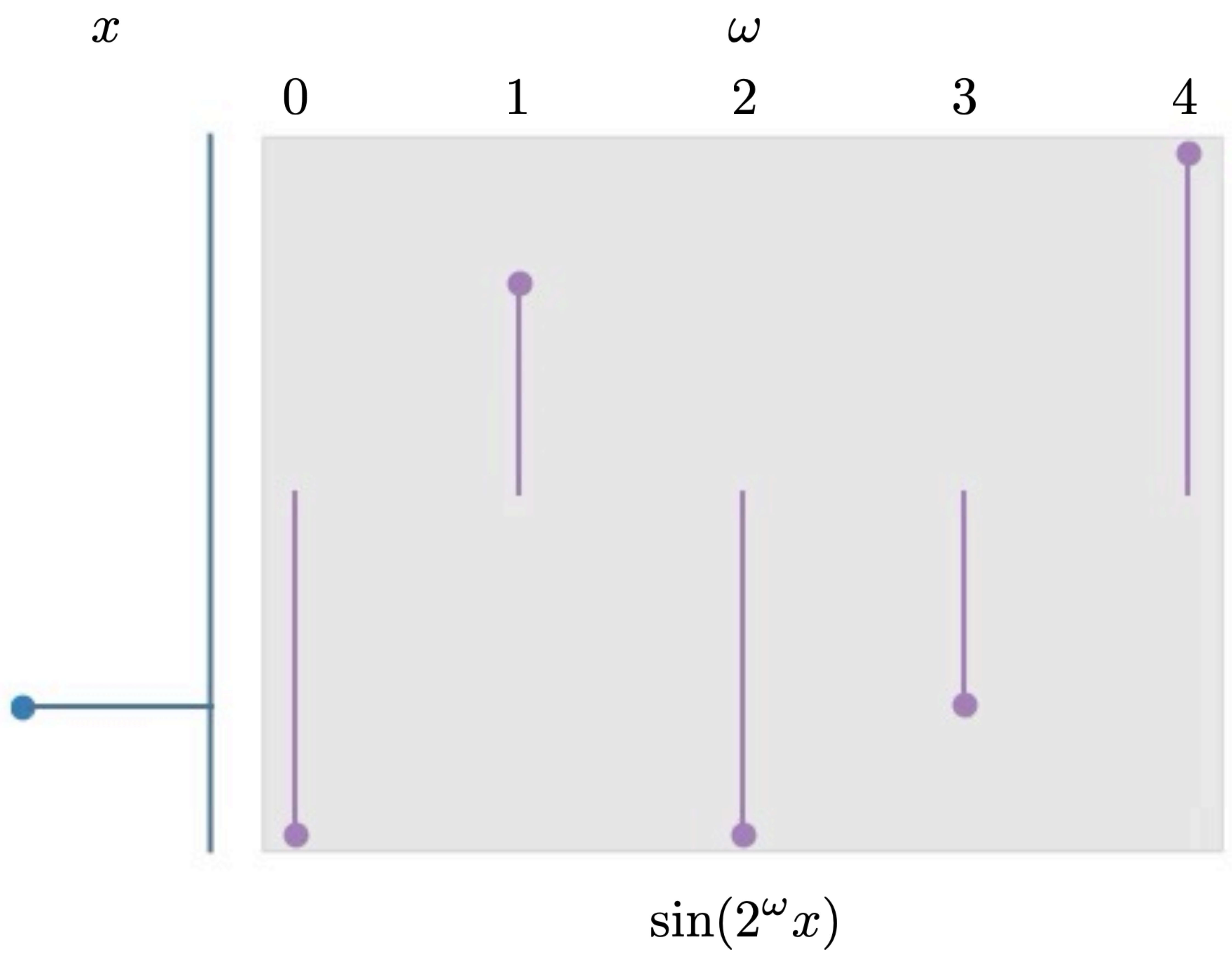
Instead of using positional encoding of a point…



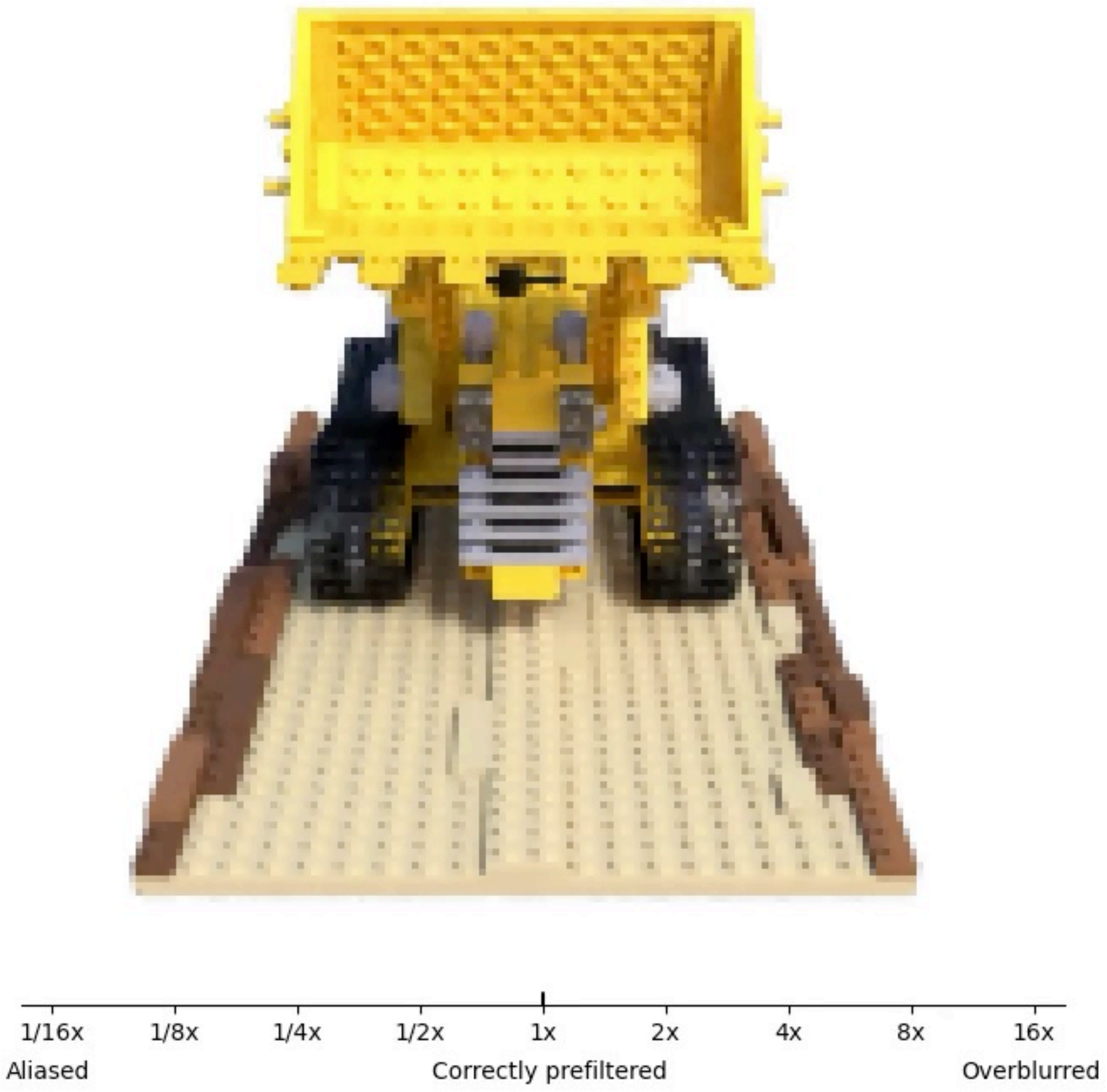
mip-NeRF uses *integrated* positional encoding



Positional Encoding

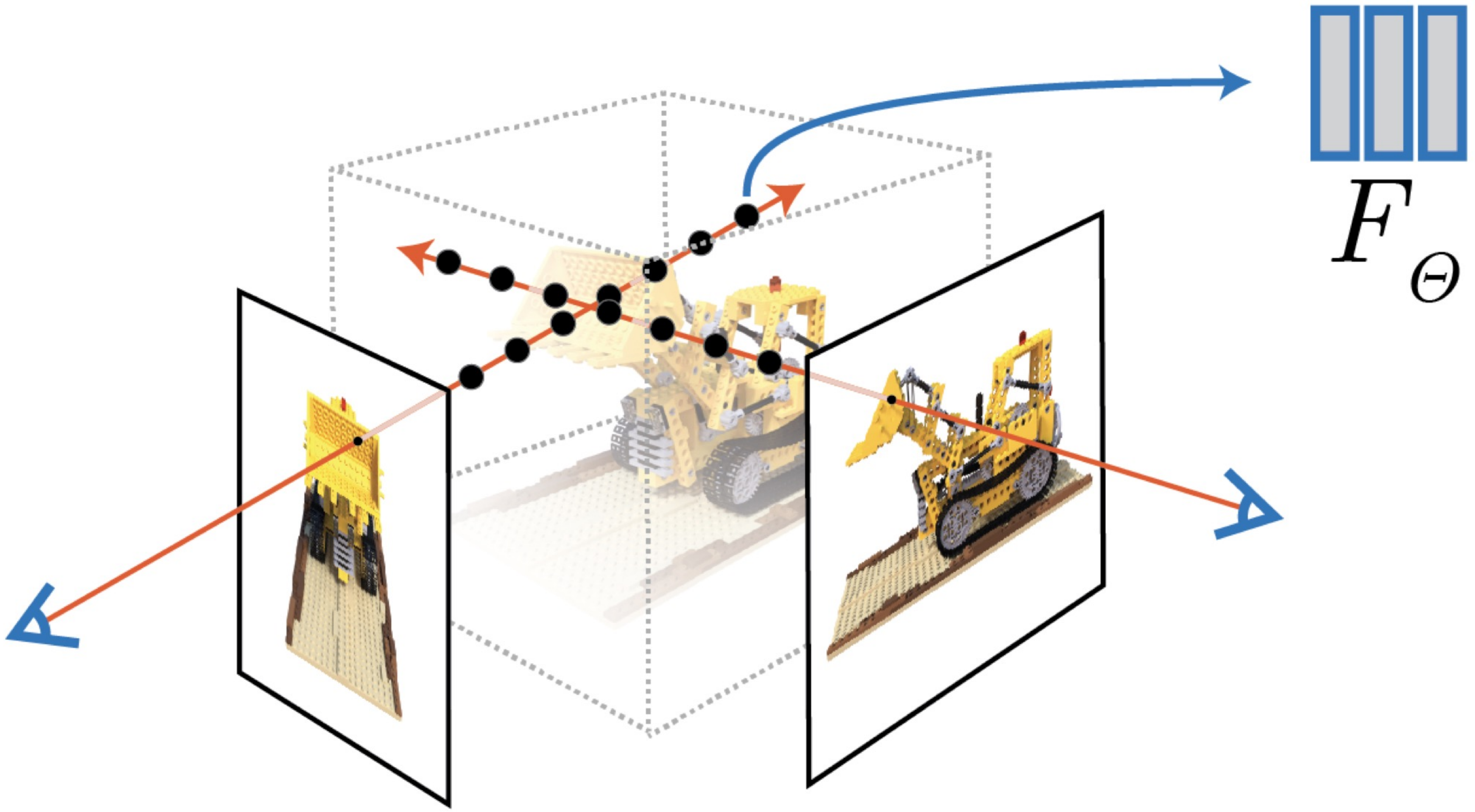


Integrated positional encoding can reasonably approximate prefiltering

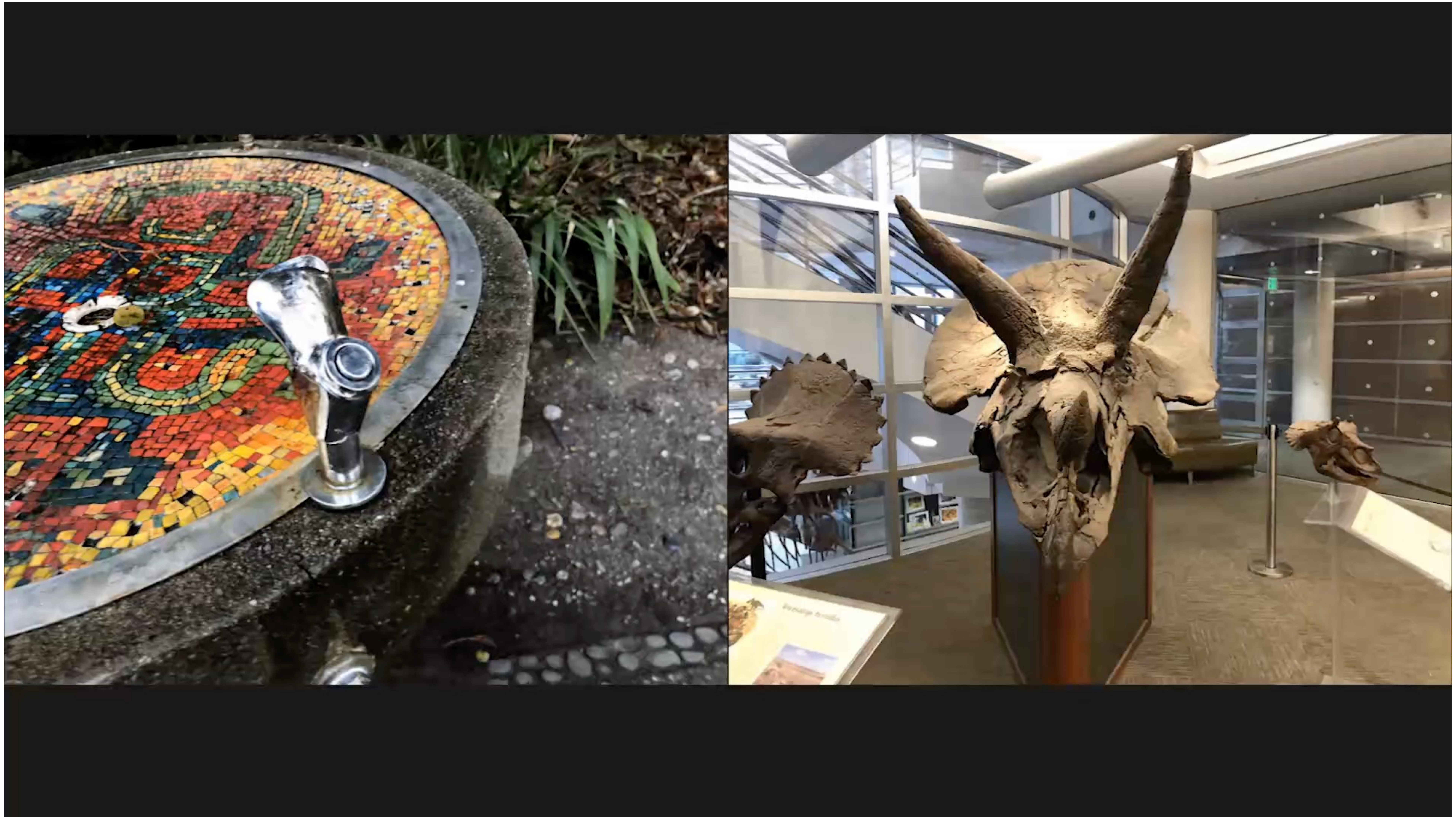


Parameterizing 3D Space

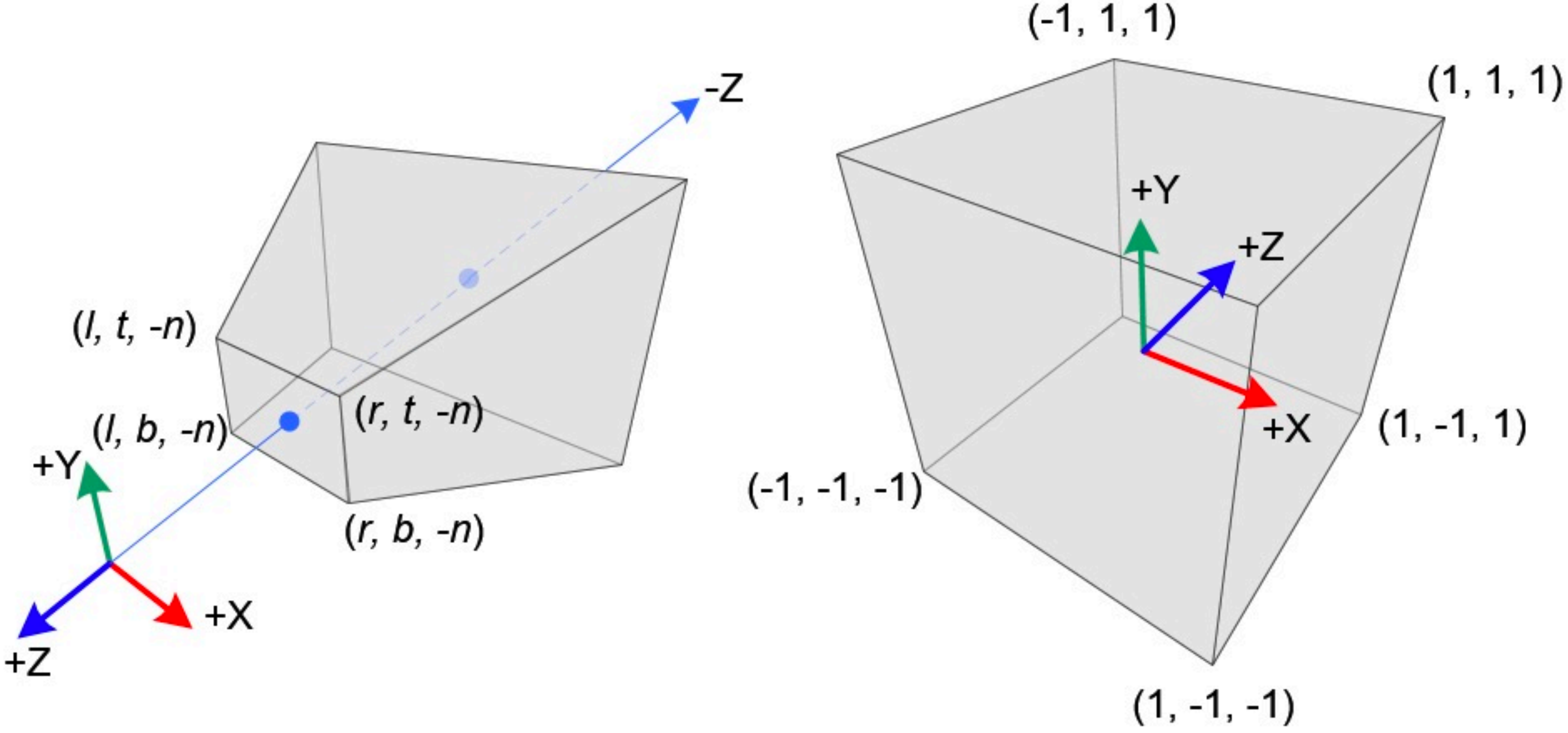
Standard coordinates for bounded volumes



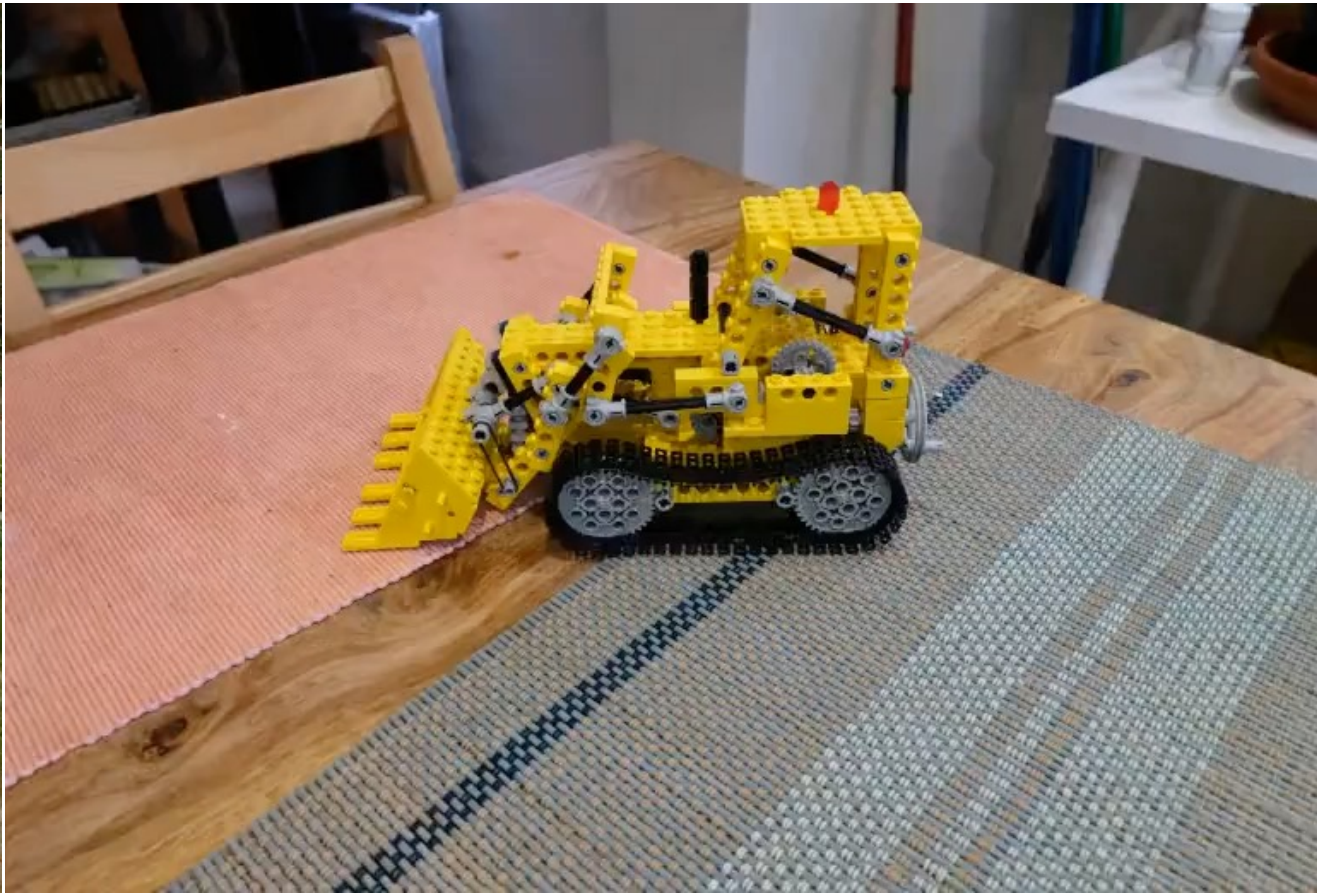
Normalized device coordinates for unbounded “forwards-facing” volumes



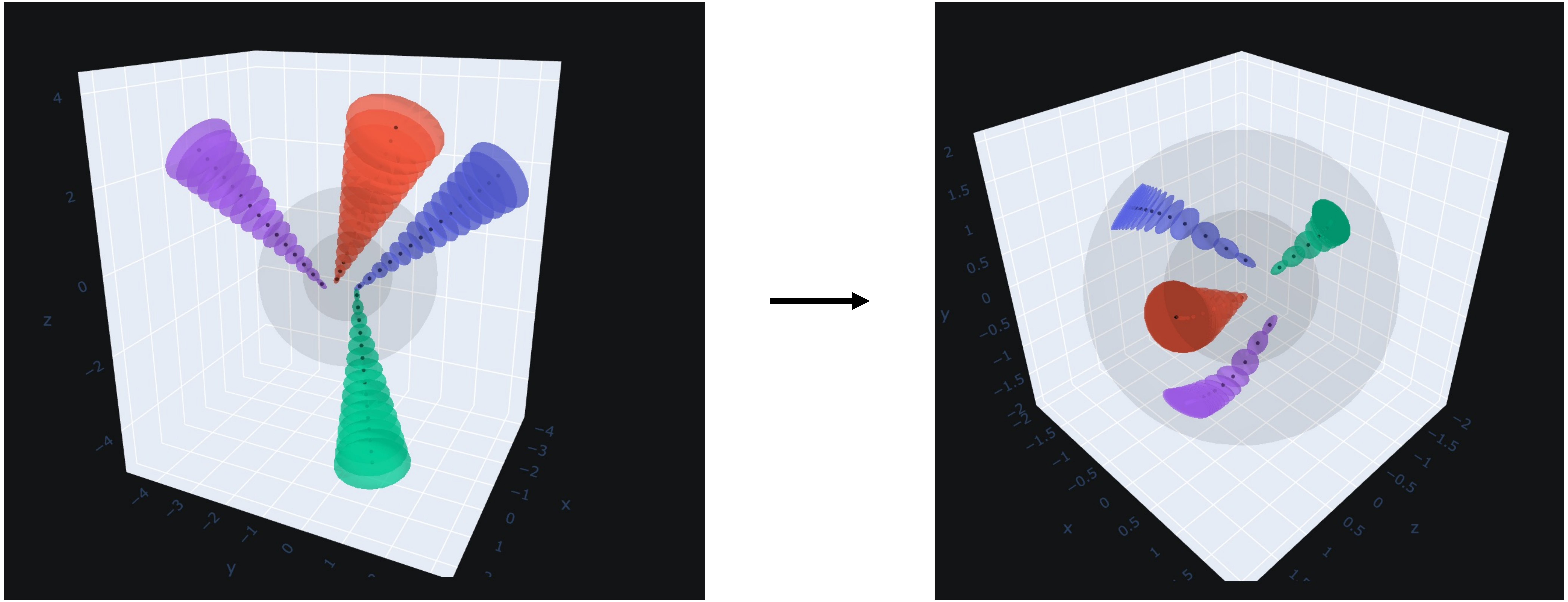
Normalized device coordinates for unbounded "forwards-facing" volumes



How to parameterize fully unbounded volumes?



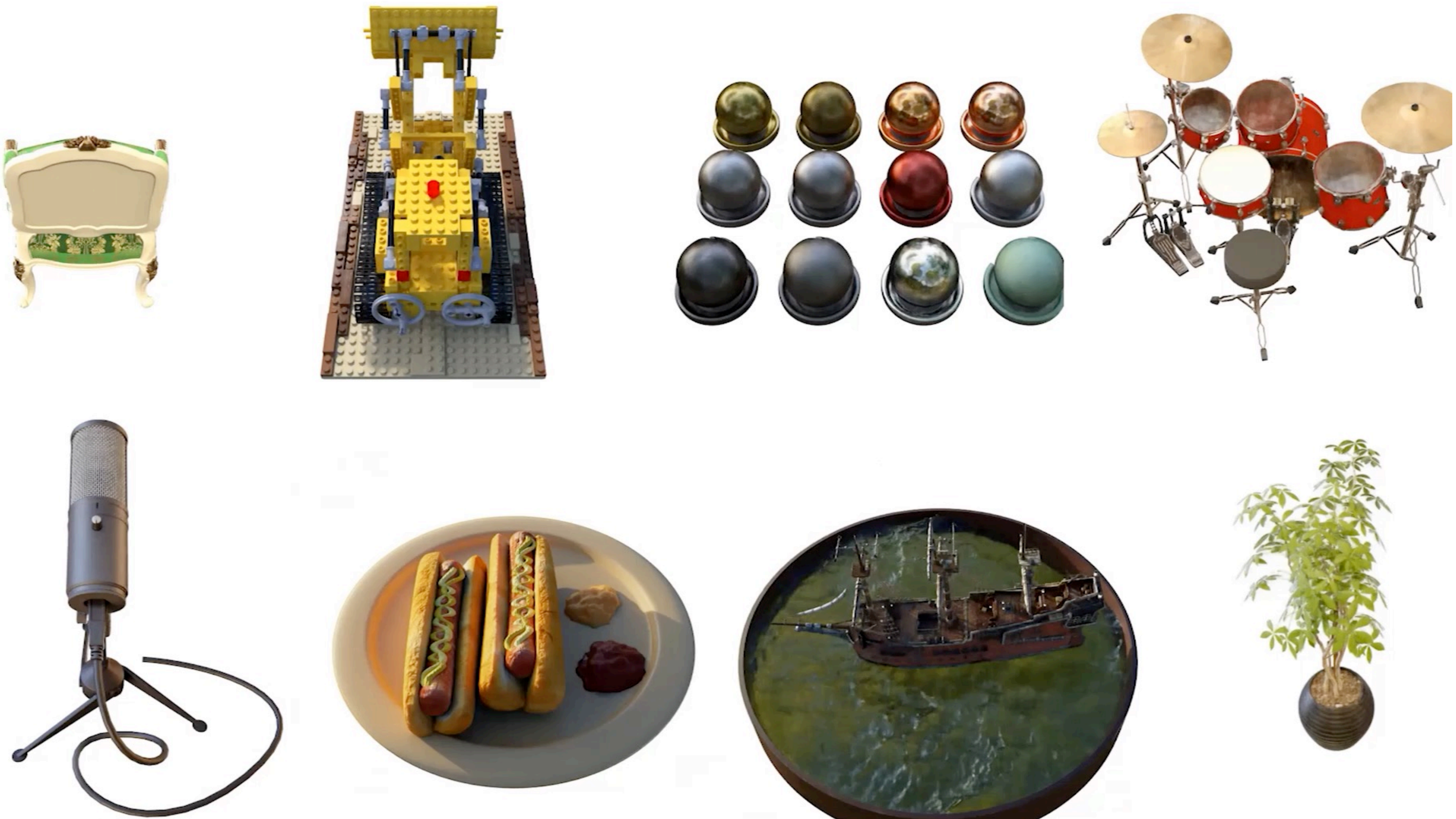
Continuous warping of space



Where we are

1. Birds Eye View & Background
2. Volumetric Rendering Function
3. Encoding and Representing 3D Volumes
4. Signal Processing Considerations
5. **Challenges & Pointers**

Caught up with the core NeRF pieces!



What are the remaining challenges?

Don’t worry there is a lot!

The neverending list of NeRF limitations (back in 2020)

- Expensive / slow to train
- Expensive / slow to render
- Sensitive to sampling strategy
- Sensitive to pose accuracy
- Assumes static lighting
- Not a mesh
- Assumes static scene
- Does not generalize between scenes

The neverending list of NeRF limitations (back in 2020)

- ~~Expensive / slow to train~~
- ~~Expensive / slow to render~~
- Sensitive to sampling strategy
- Sensitive to pose accuracy
- Assumes static lighting
- Not a mesh
- Assumes static scene
- Does not generalize between scenes

The neverending list of NeRF limitations (back in 2020)

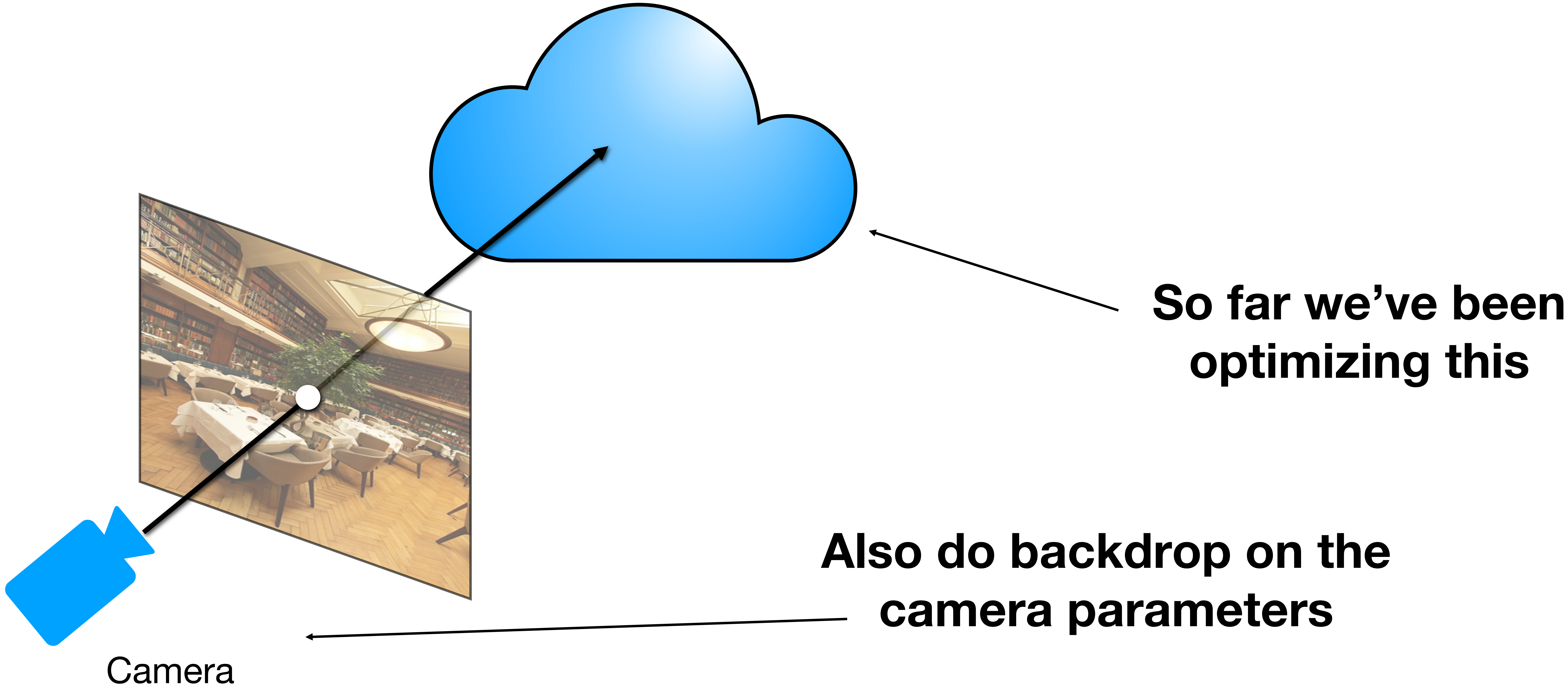
- ~~Expensive / slow to train~~
- ~~Expensive / slow to render~~
- ~~Sensitive to sampling strategy~~
- Sensitive to pose accuracy
- Assumes static lighting
- Not a mesh
- Assumes static scene
- Does not generalize between scenes

The neverending list of NeRF limitations (back in 2020)

- ~~Expensive / slow to train~~
- ~~Expensive / slow to render~~
- ~~Sensitive to sampling strategy~~
- Sensitive to pose accuracy
- Assumes static lighting
- Not a mesh
- Assumes static scene
- Does not generalize between scenes

Camera Quality

Small noise in the camera can be made robust by also optimizing the camera




No Pose Optimization
Block-NeRF

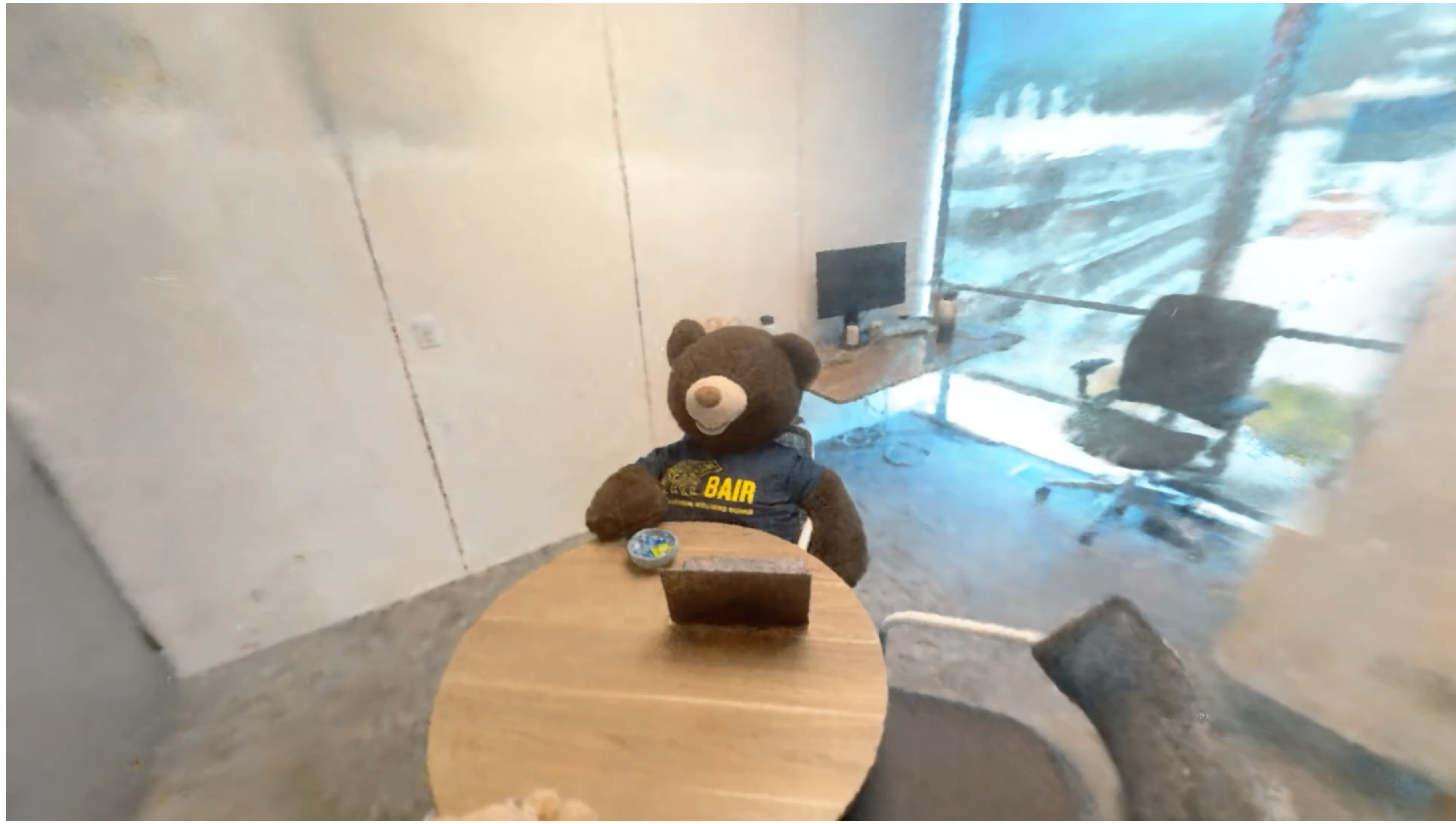
Camera Optimization

Small noise in the results can be improved

Starting from scratch is still an active area of research [Barf Lin et al. 2021, NeRF− …]

Noisy Camera from IMU/Lidar

Result with Camera Optimization



The neverending list of NeRF limitations (back in 2020)

- ~~Expensive / slow to train~~
- ~~Expensive / slow to render~~
- ~~Sensitive to sampling strategy~~
- Sensitive to pose accuracy
- Assumes static lighting
- Not a mesh
- Assumes static scene
- Does not generalize between scenes

Inverse Gr

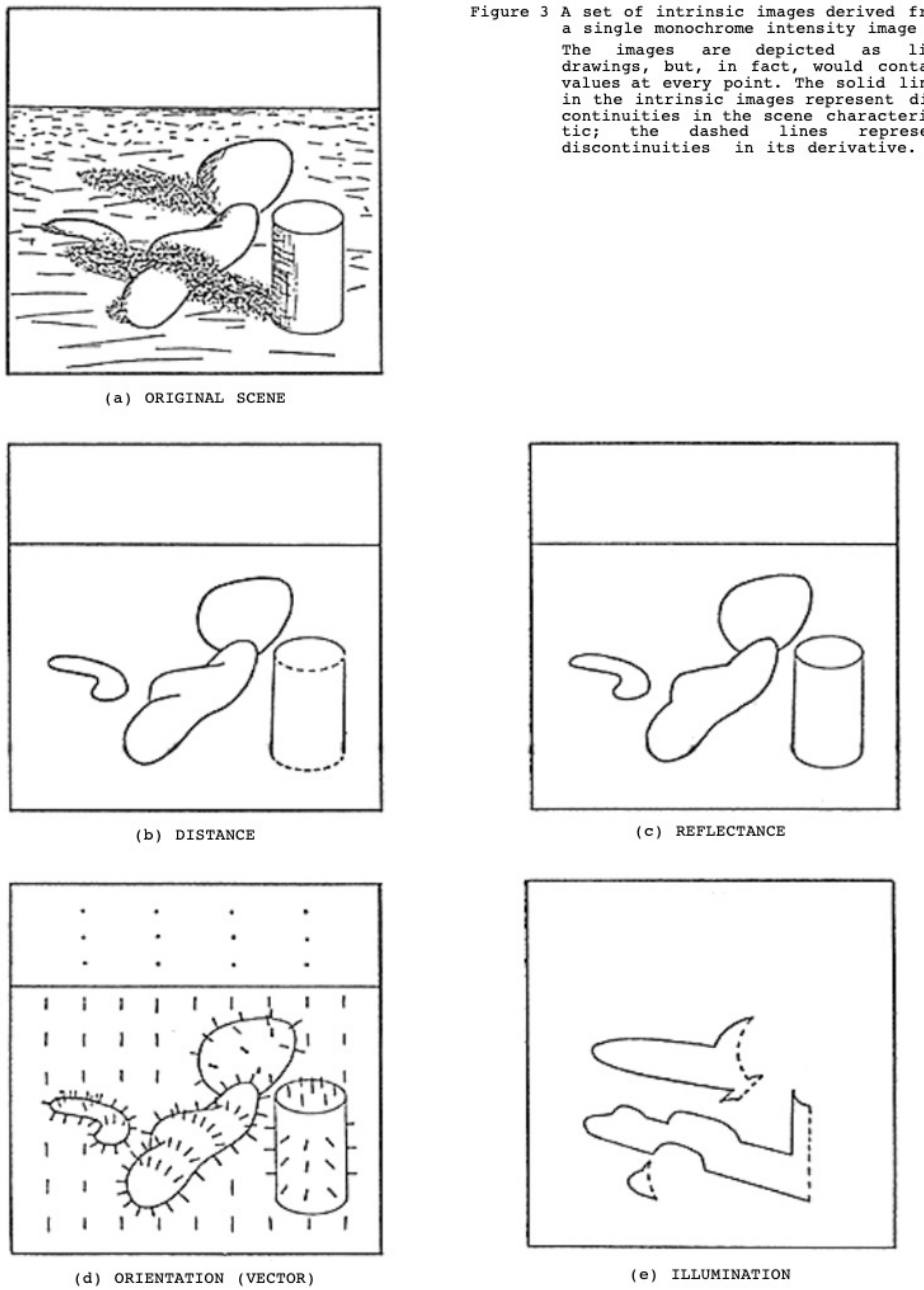


Figure 3 A set of intrinsic images derived from a single monochrome intensity image

The images are depicted as line drawings, but, in fact, would contain values at every point. The solid lines in the intrinsic images represent discontinuities in the scene characteristic; the dashed lines represent discontinuities in its derivative.

Barrow and Tennenbaum 1978

Inverse Graphics

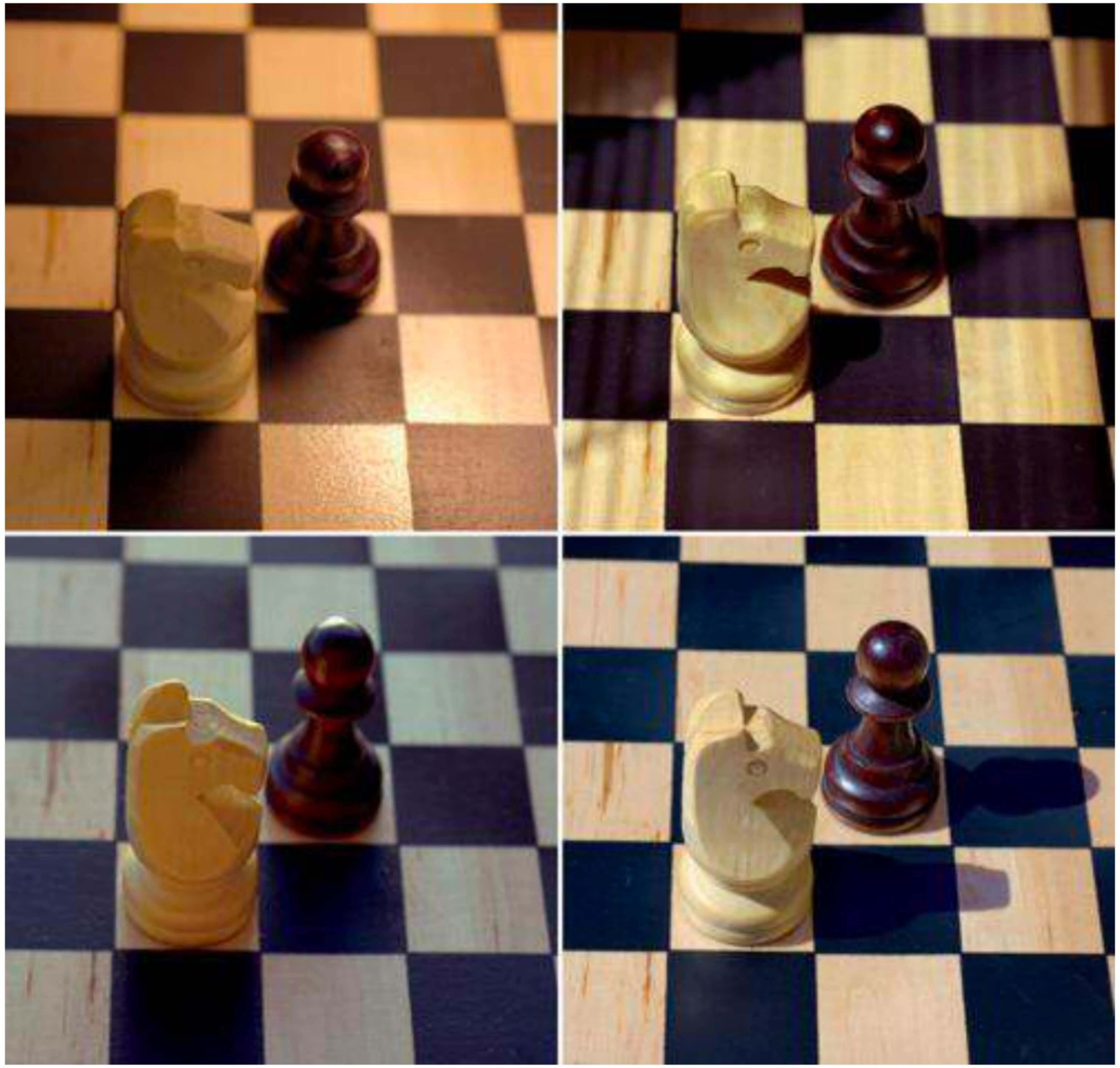
Barron and Malik, ECCV 2012

Problem with Baked Lighting

- As you now see, NeRF bakes in the lighting effects in the scene
- That’s what allows it to model the non-Lambertian effects, but it’s not always ideal

Why you want light separated

- Necessary for Relighting & Editing
  - Changing light
  - Inserting objects into another scene (with different lighting)
  - Changing material properties
  - Edit the appearance without changing light
  - …



Recall: we simplified by ignoring scattering

Absorption

http://commons.wikimedia.org

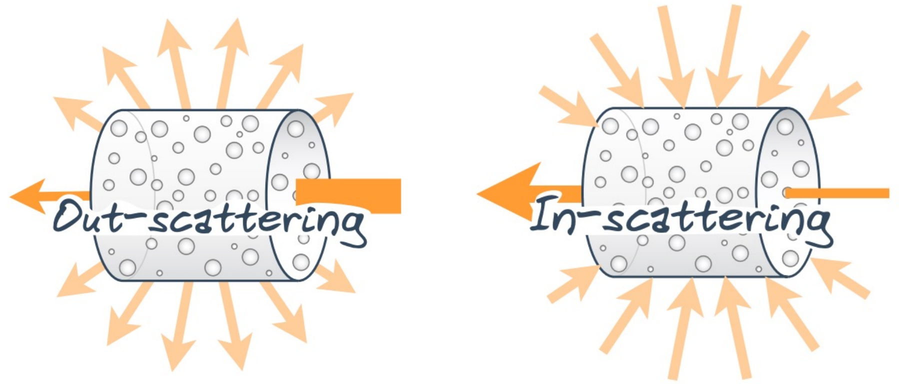
Scattering

Emission

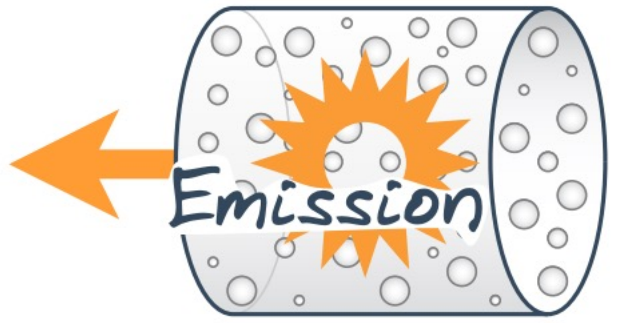
http://wikipedia.org

Slide credit: Novak et al 2018, *Monte Carlo methods for physically based volume rendering*

Absorption

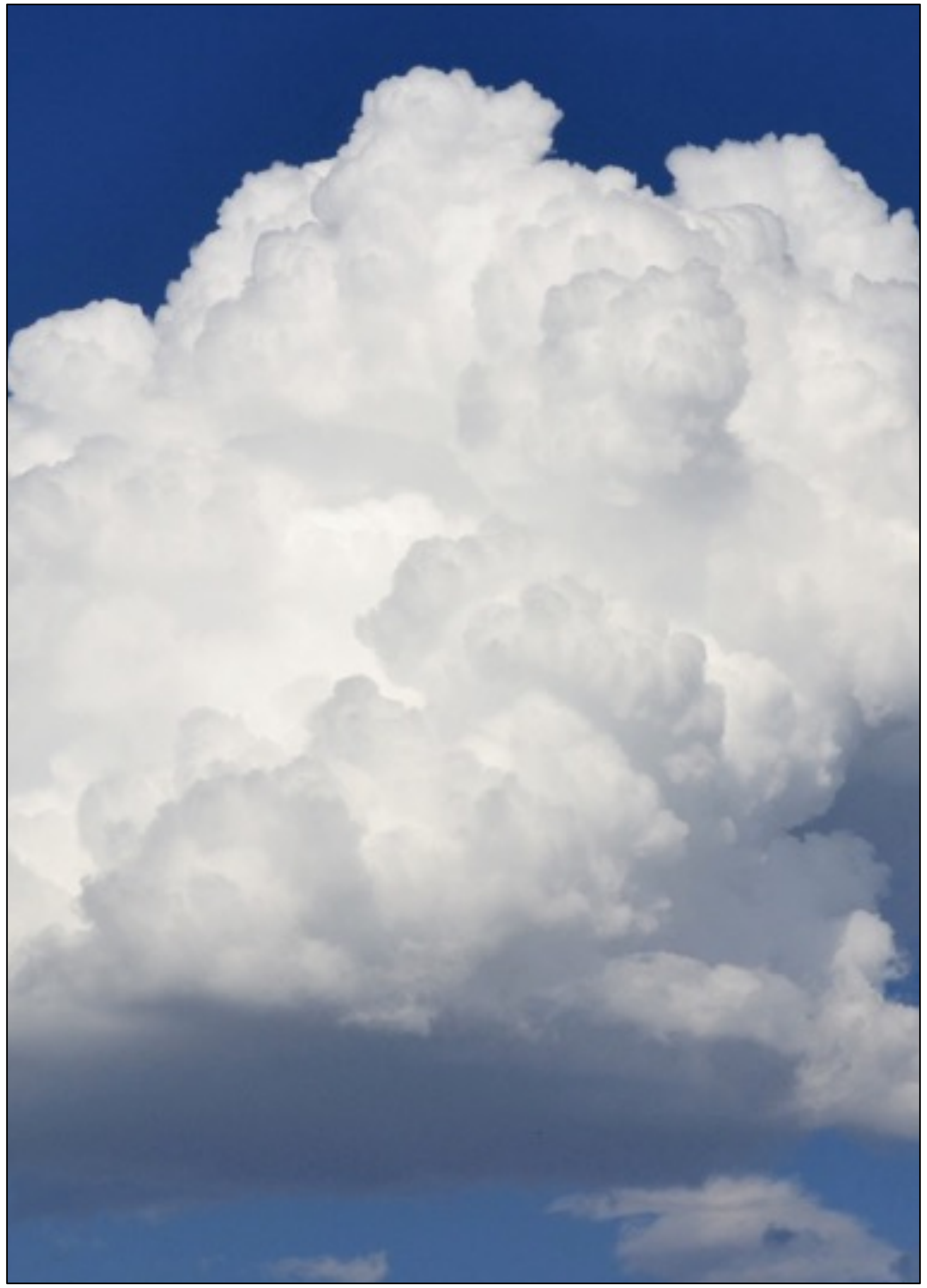


Scattering



Emission

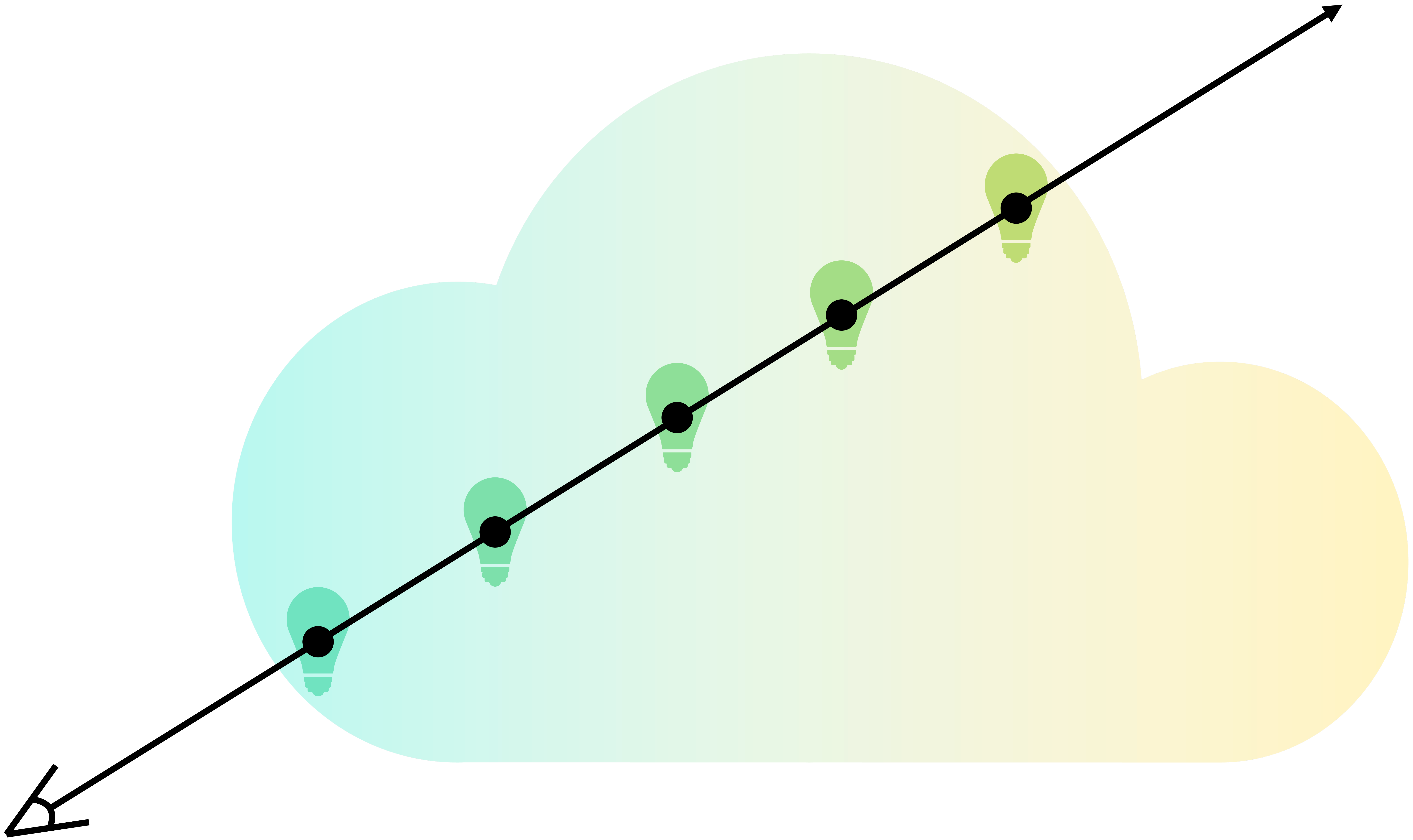
http://commons.wikimedia.org



http://wikipedia.org

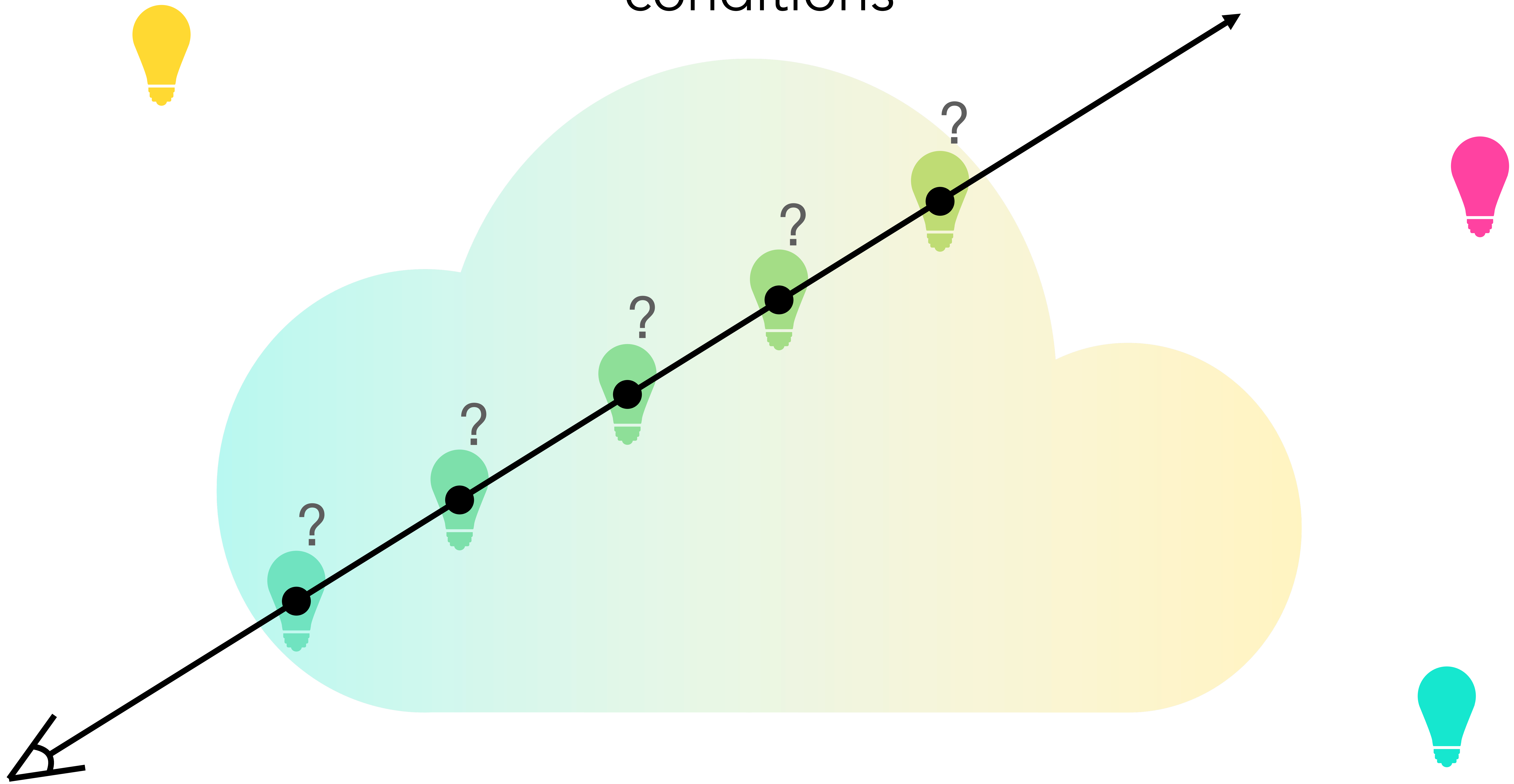
Slide credit: Novak et al 2018, *Monte Carlo methods for physically based volume rendering*

NeRF represents a volume of particles that emit light

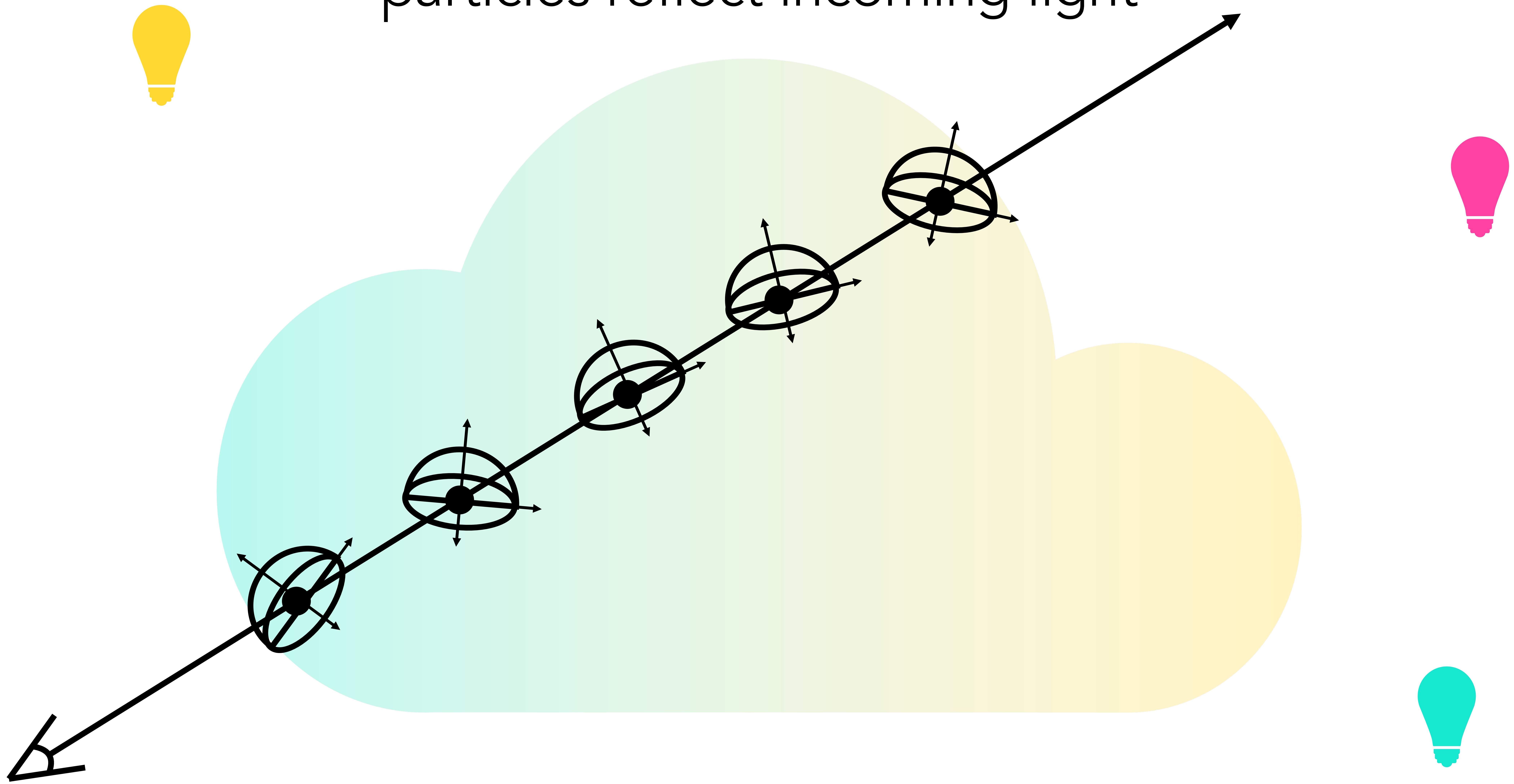


NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.
Ben Mildenhall*, Pratul Srinivasan*, Matt Tancik*, Jon Barron, Ravi Ramamoorthi, Ren Ng. ECCV 2020.

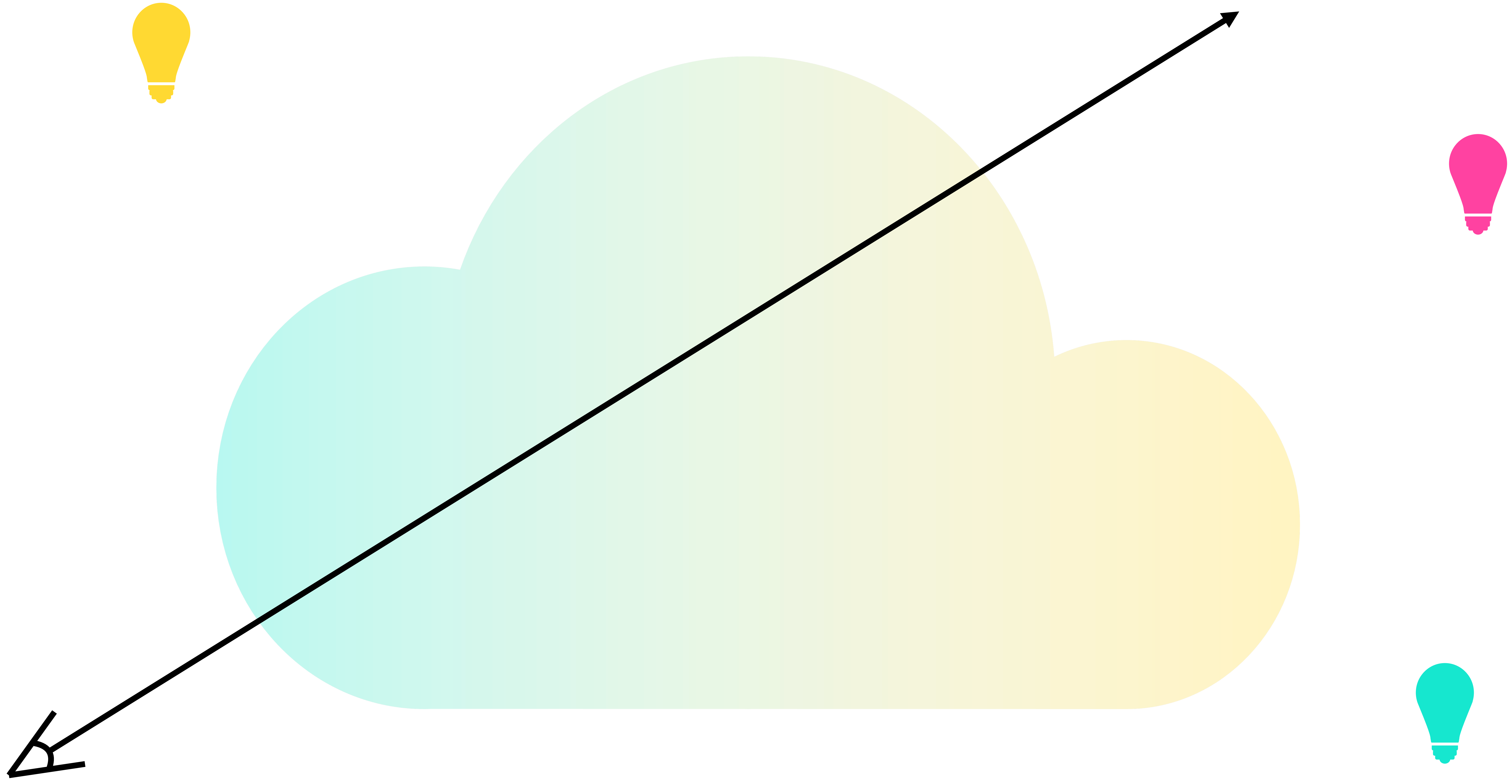
But doesn't let us simulate how light changes with new lighting conditions



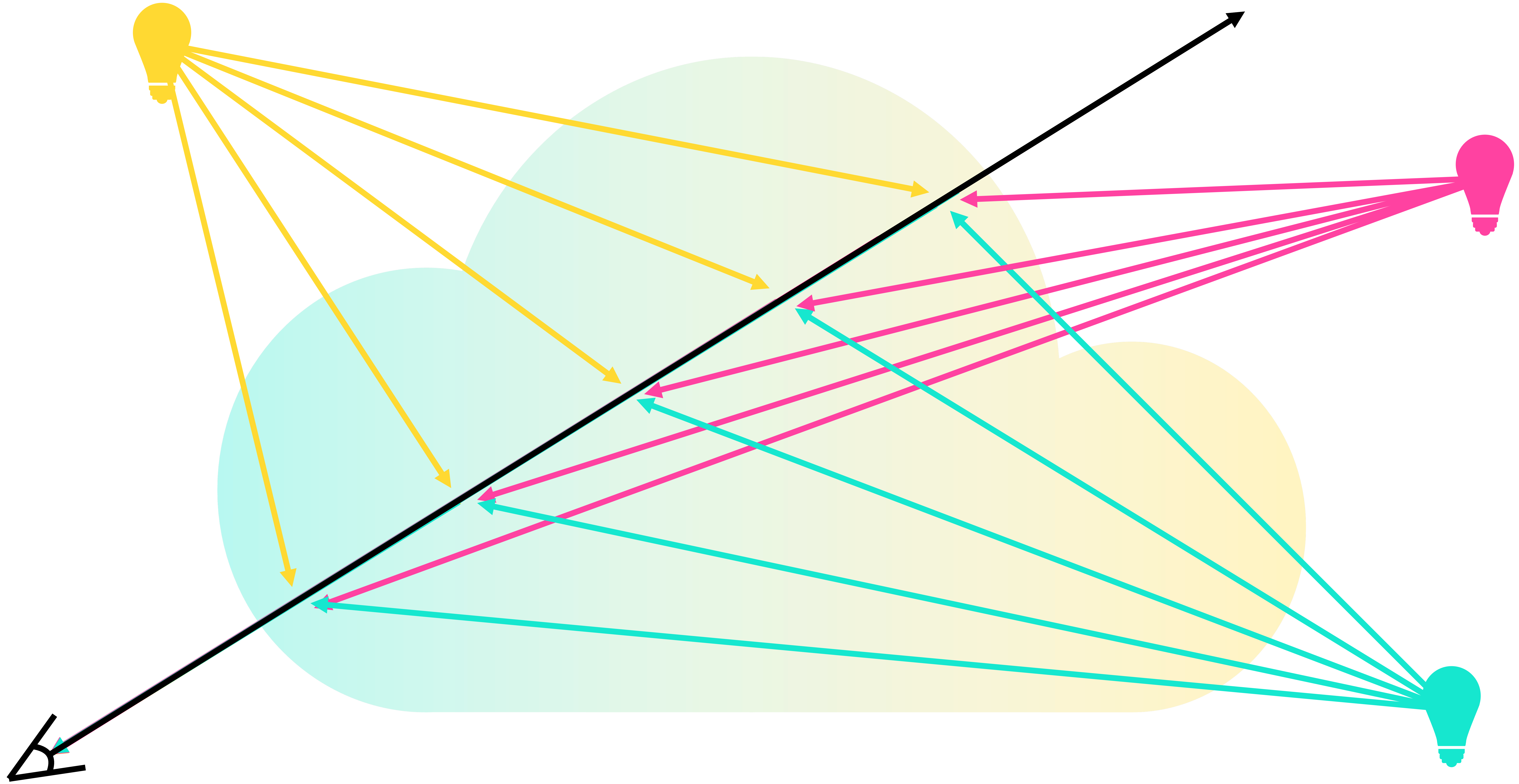
First step: replace emitted light with BRDFs that describe how particles reflect incoming light



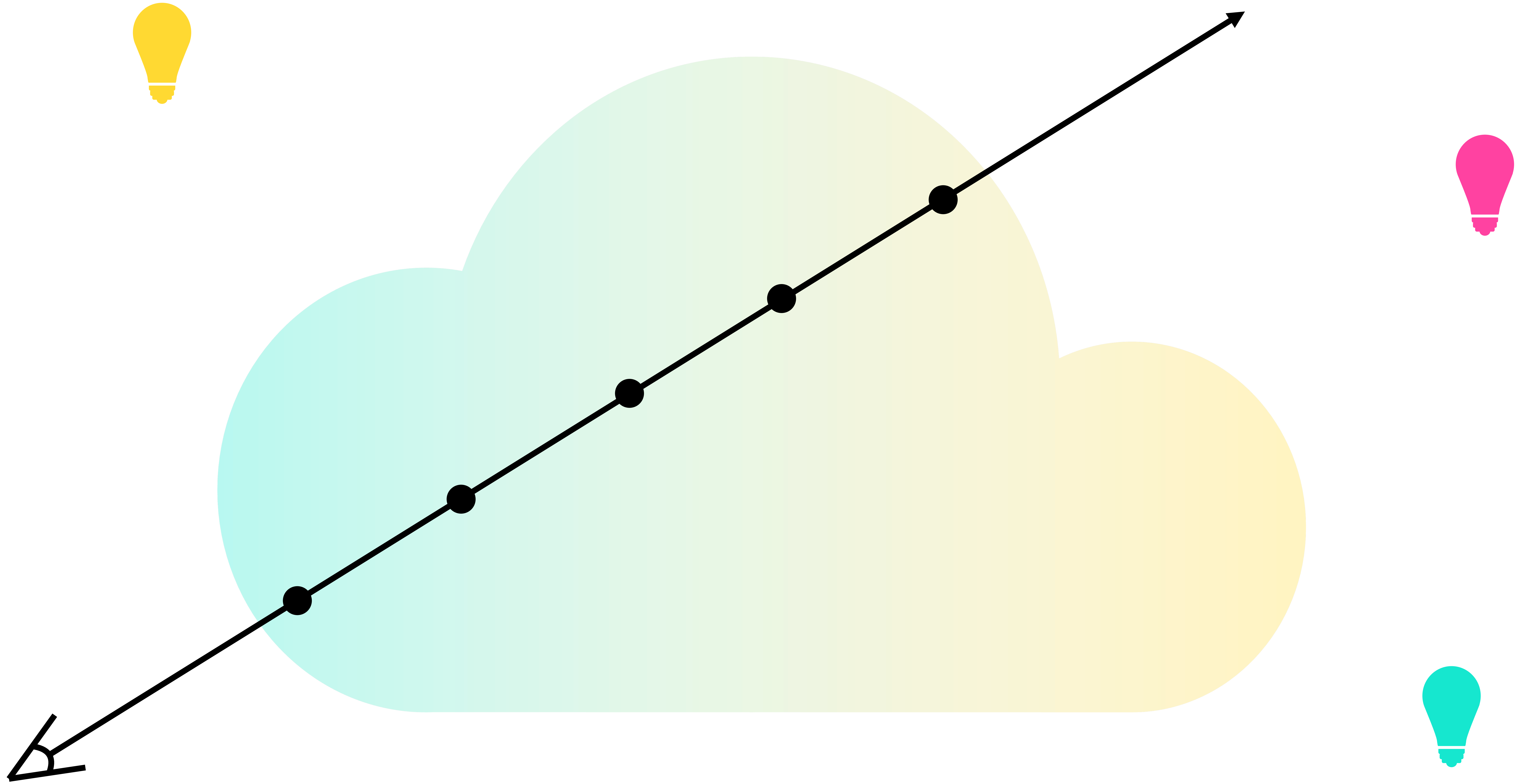
Rendering with direct lighting



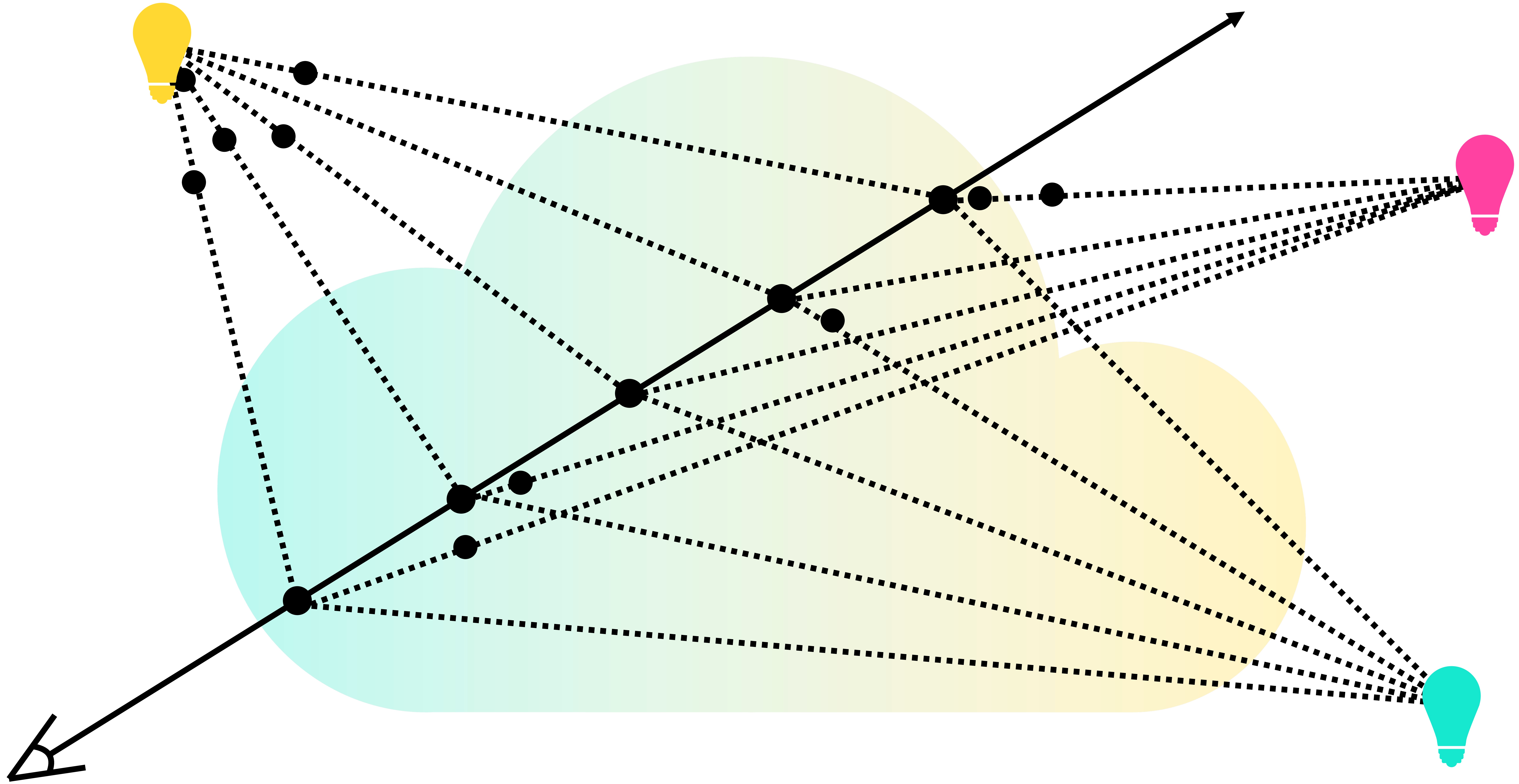
Rendering with direct lighting



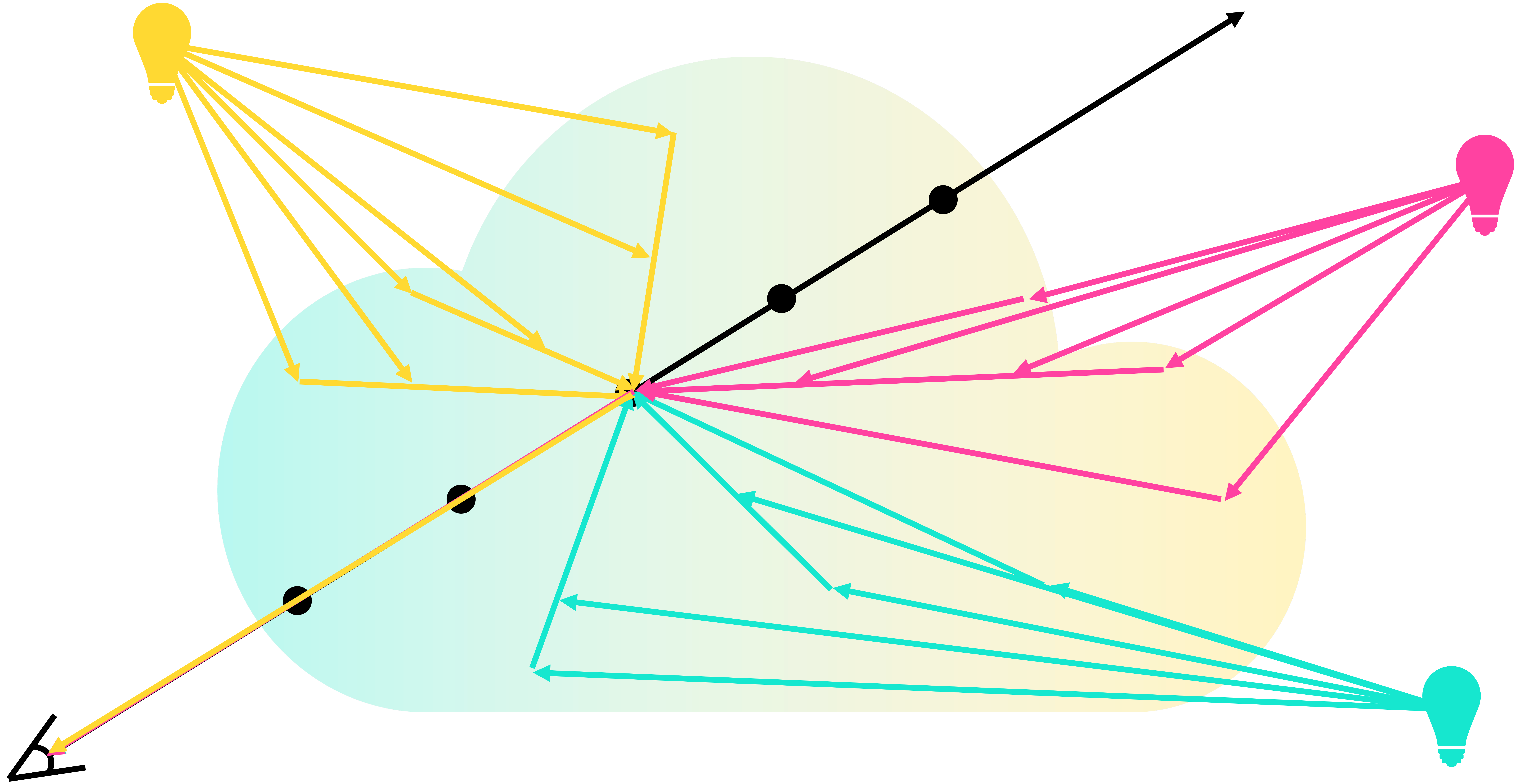
Rendering with direct lighting



Rendering with direct lighting

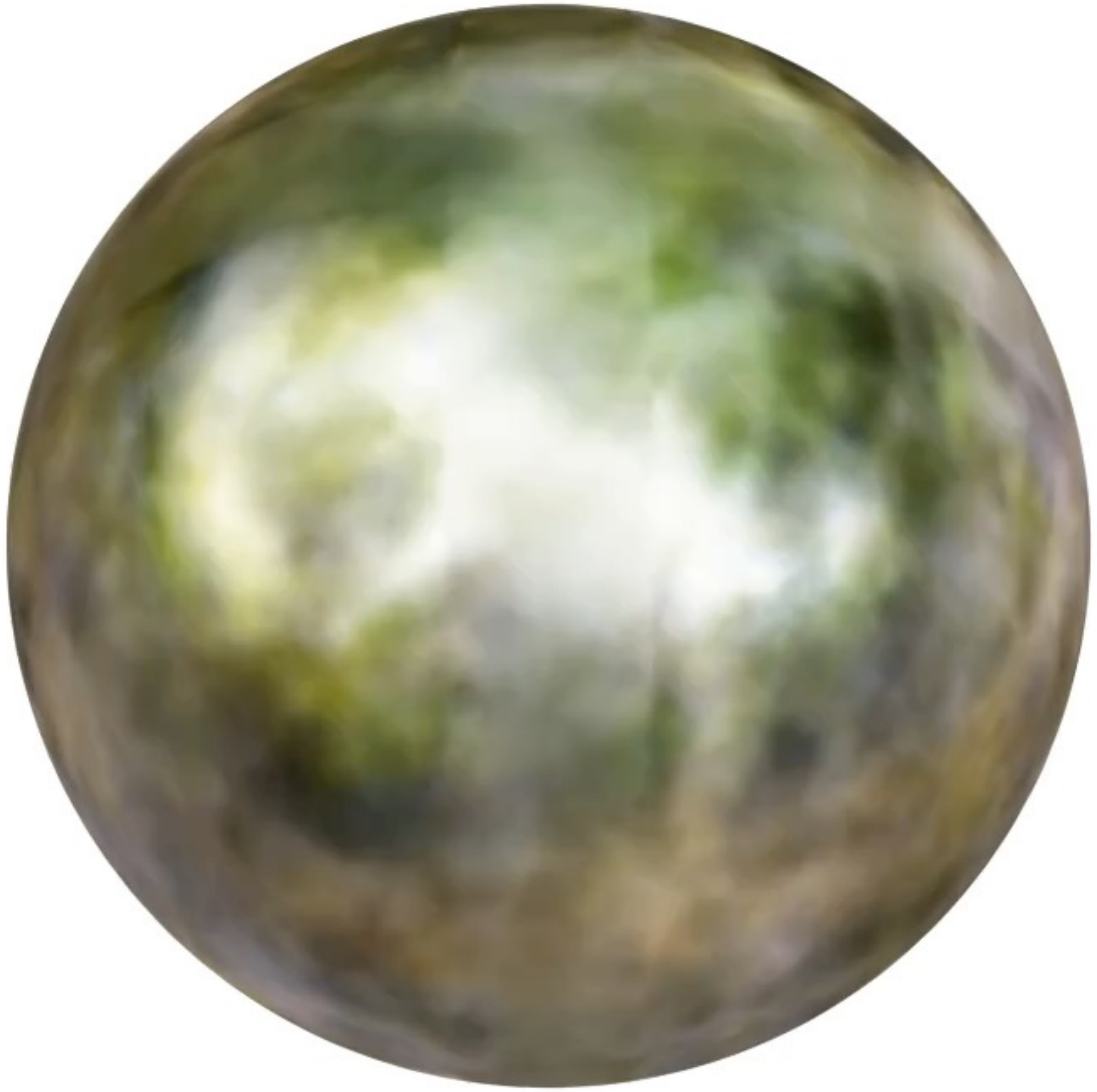


Indirect illumination is even more computationally-expensive



Modeling light can recover better surfaces

NeRF



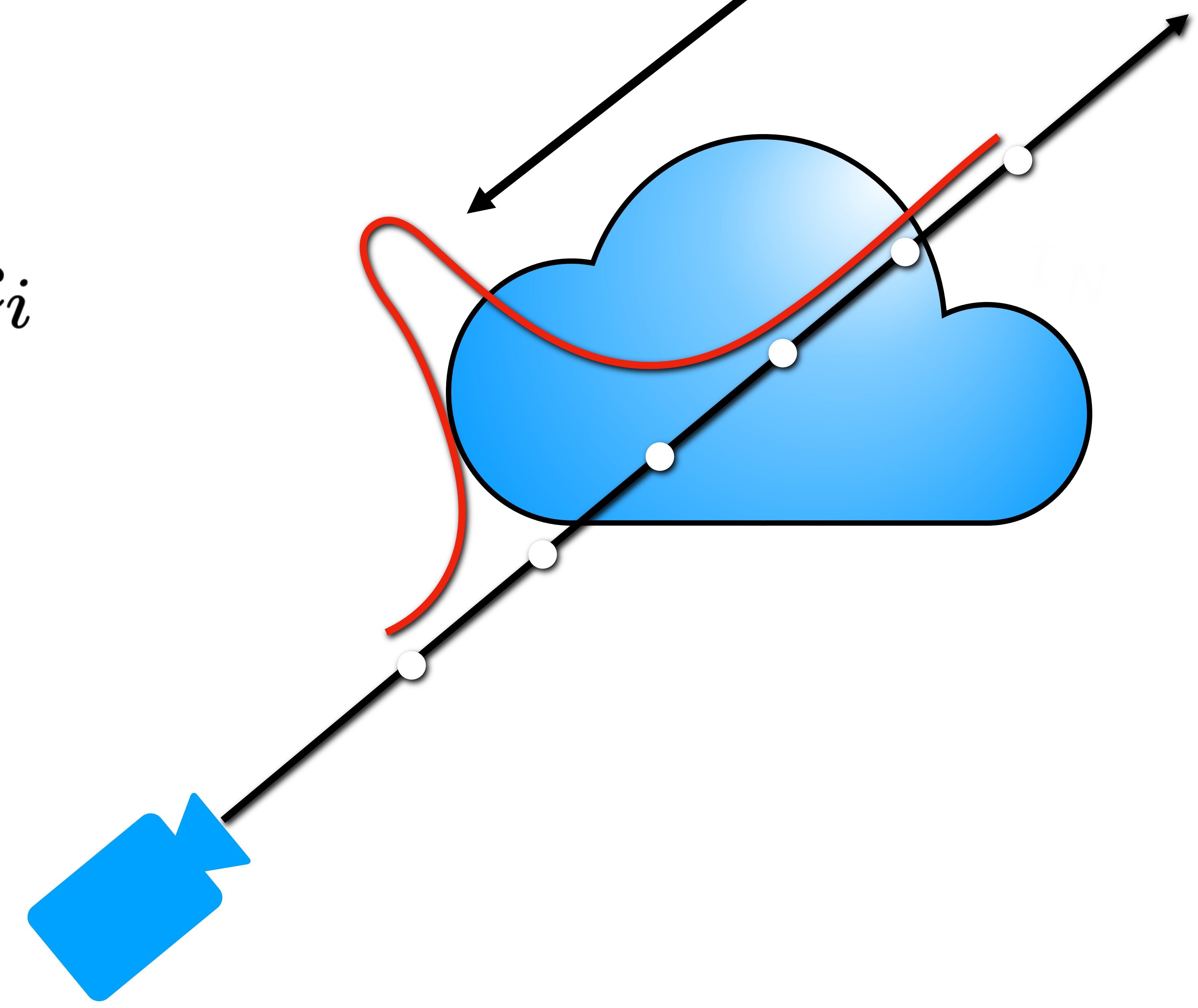
Ref-NeRF

Verbin et al. CVPR 2022

Decomposing light helps recover sharper surfaces

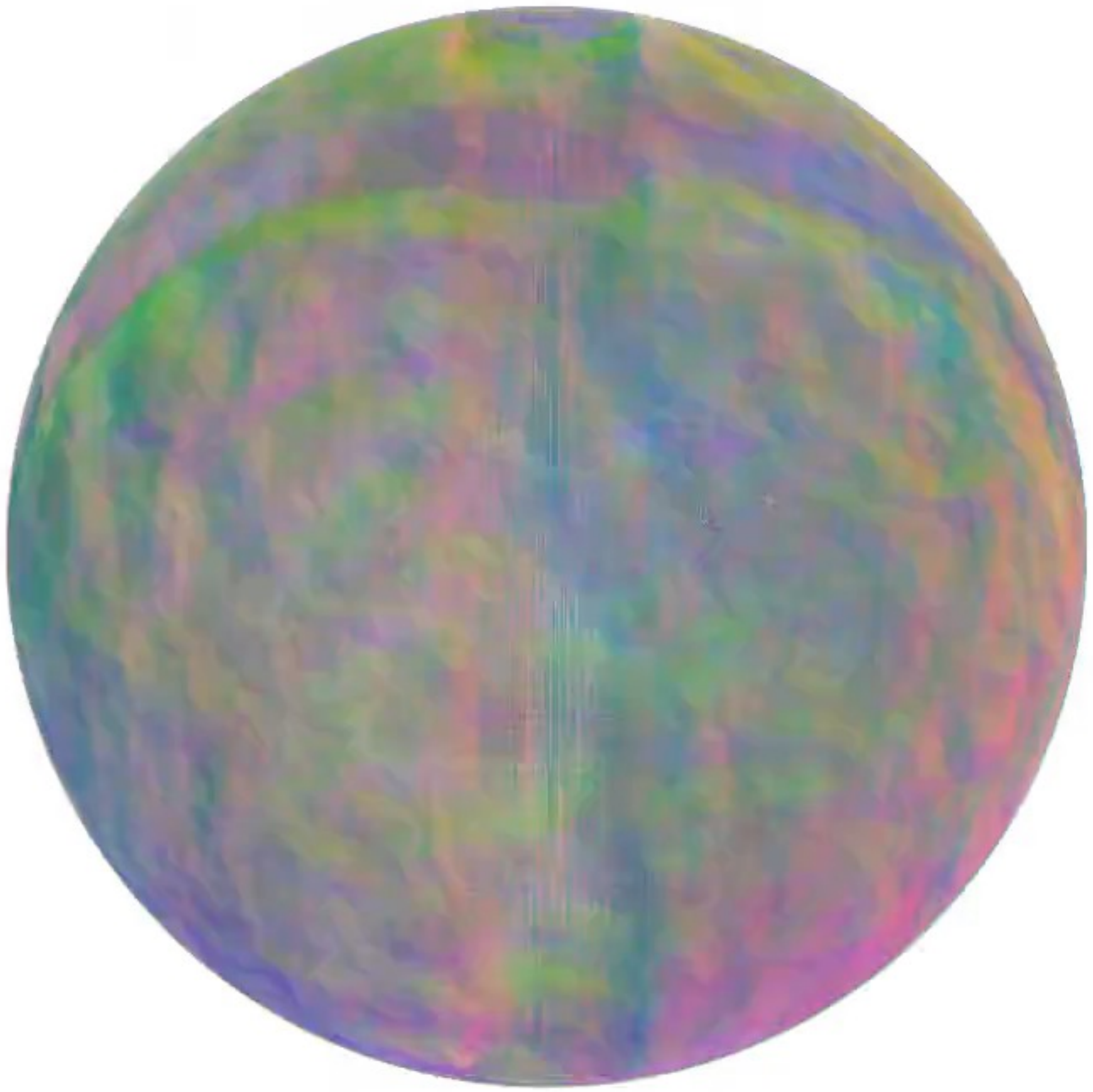
This is not what it always looks like!

$$C \approx \sum_{i=1}^{N} \boxed{T_i \alpha_i} c_i$$



Better modeling of light helps recover sharper surfaces

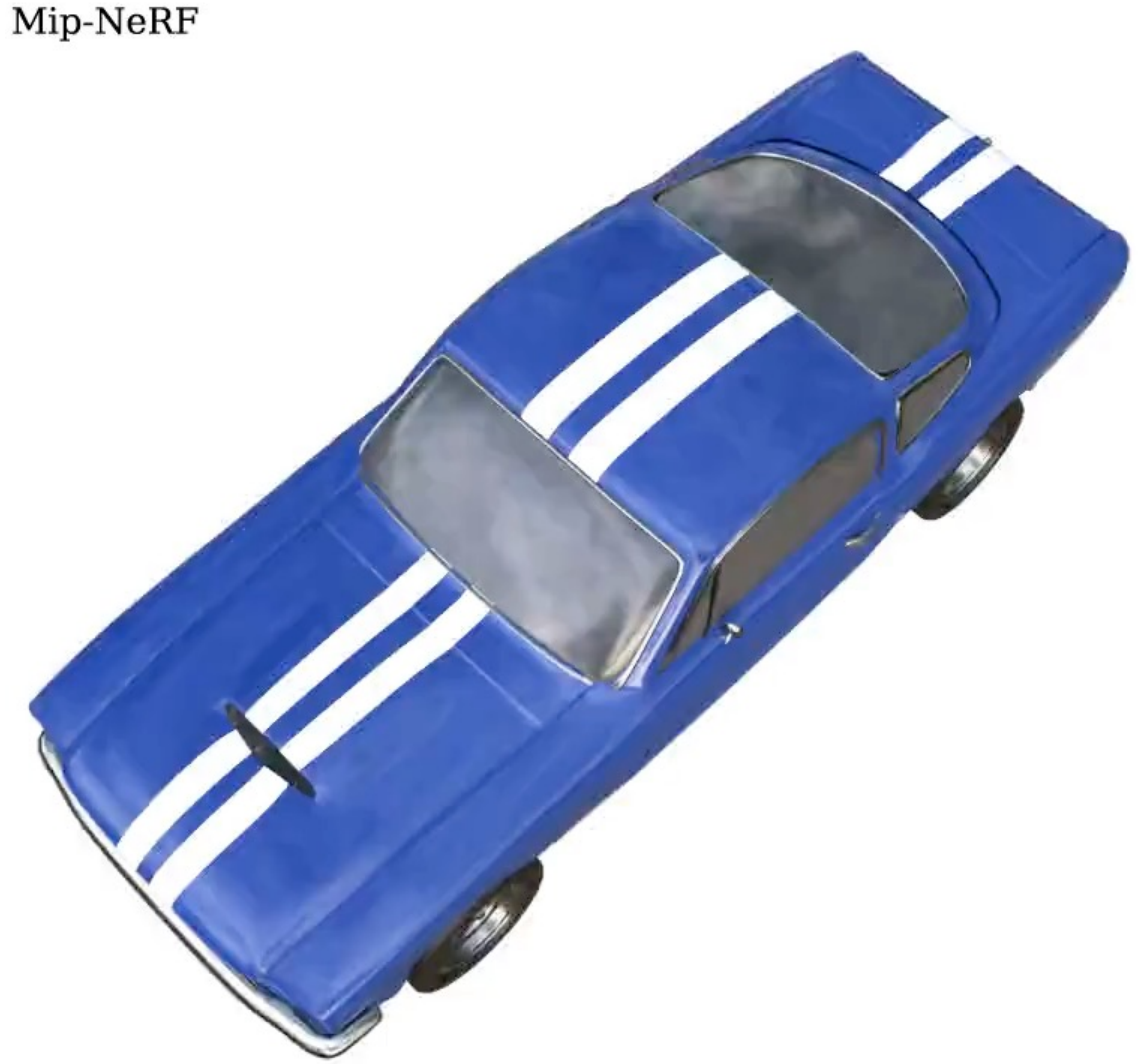
NeRF



Ref-NeRF

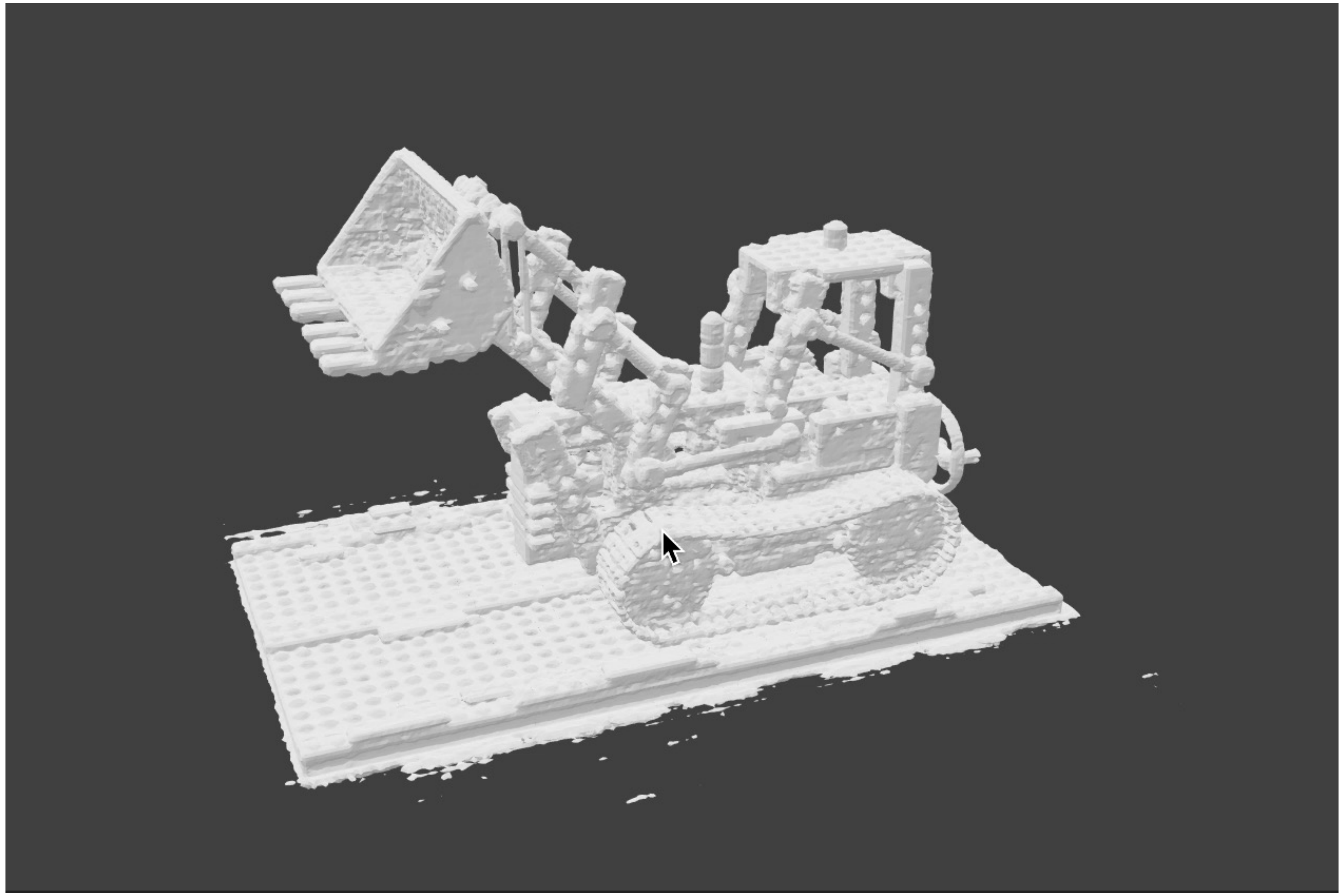
Verbin et al. CVPR 2022

Modeling light = better specularities



Editing specular and diffuse colors

Related Challenge: Extracting Surfaces



- Needed for adoption into existing gaming engines/VFX lifecycle
- Challenges: What if the density recovered isn't peaky (surface) and is not clean? What to do? What about complex scenes, how to group objects?

The Dynamic World

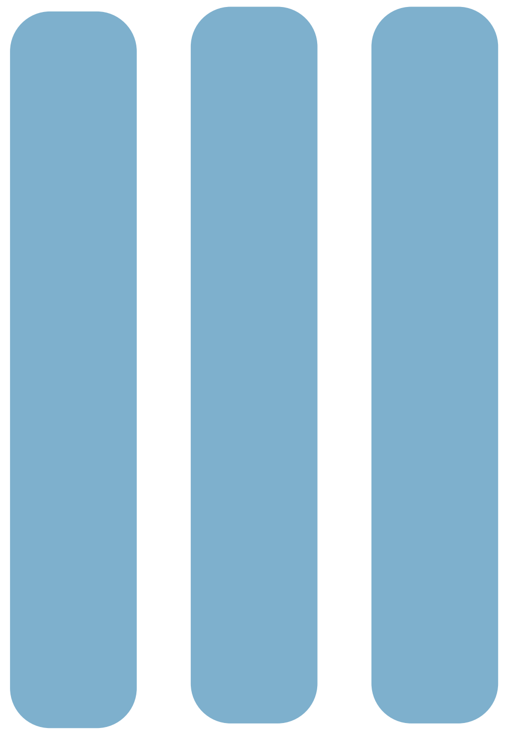
Memories of Australia –Andrew S. Hamilton

Holy grail

- Dynamic Novel View Synthesis from Monocular Camera
- Very difficult! Extremely under constrained problem

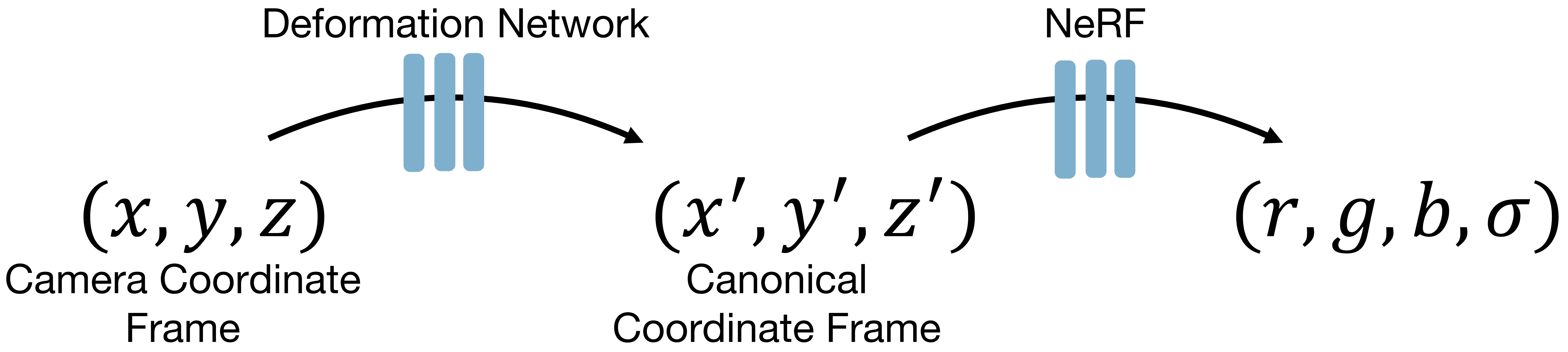
Simple baseline for adding time

$$(x, y, z, \theta, \phi, t) \rightarrow F_{\Omega} \rightarrow (r, g, b, \sigma)$$



Hard without simultaneous multiple view!

Through a deformation network



Still very under constrained

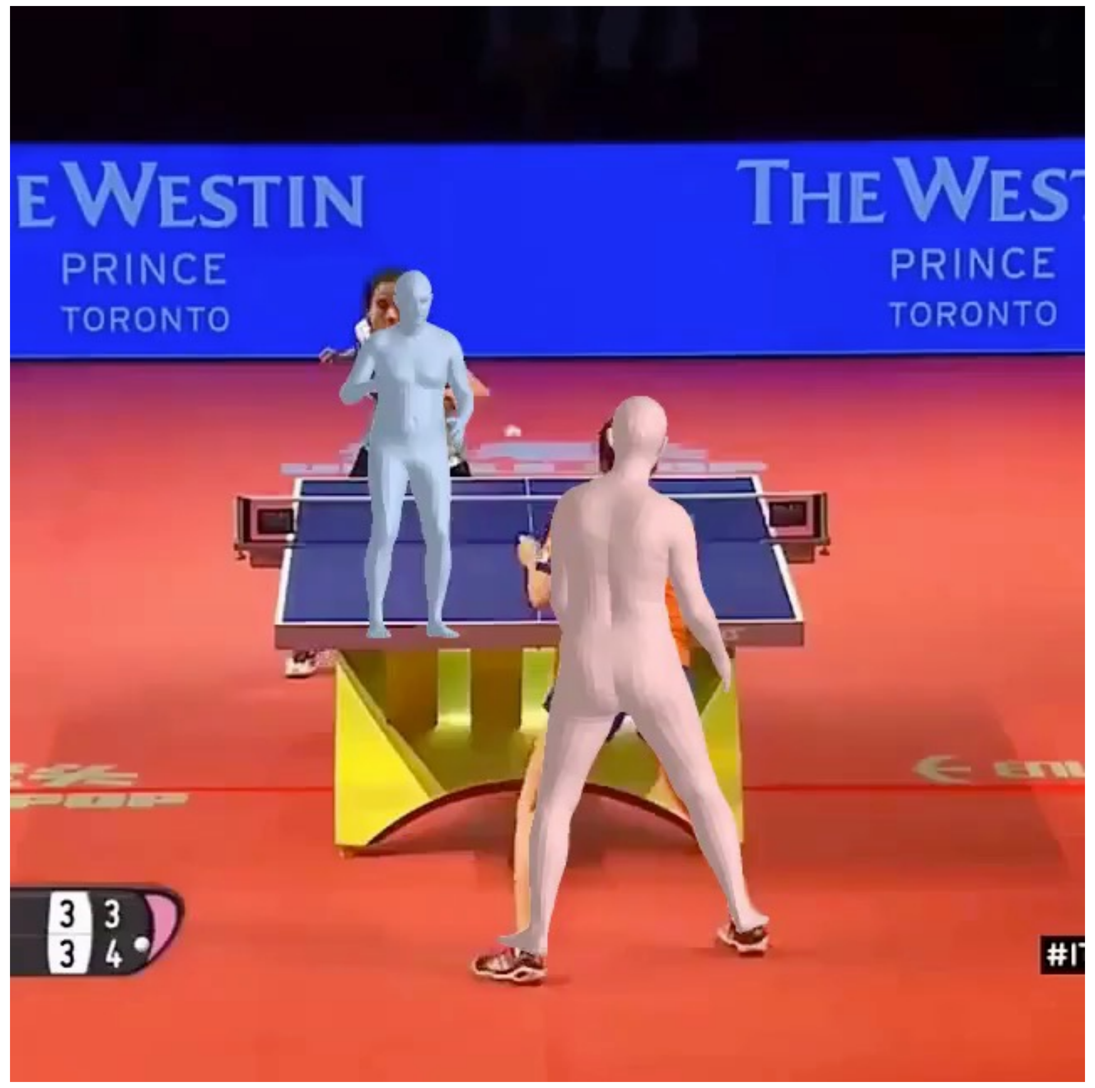
D-NeRF [Pumarola et al. CVPR 2021], Nerfies [Park et al. ICCV 2021], HyperNeRF, NR-NeRF, etc..

Dynamic View Synthesis: Monocular is hard

D-NeRF [Pumarola et al. CVPR 2021], NSFF [Li et al., CVPR 2021],HyperNeRF[Park et al. SIGASia 2021]…..

- But performance on in-the-wild monocular capture still far [Gao et al. NeurIPS 2022]

What if we knew how they deform?



HMMR, Kanazawa et al.
CVPR 2019

HumanNeRF Weng et al. CVPR 2022

Other kinds of dynamic changes

Appearance Changes

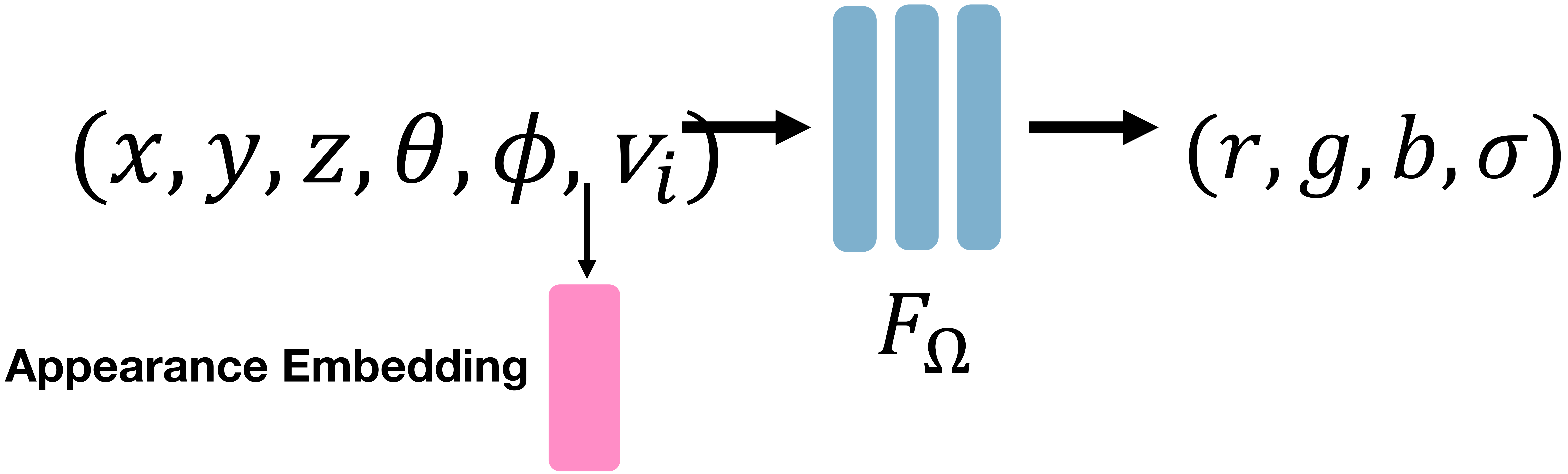
Exposure differences

Lighting changes (day, night)..

Clouds passing by..

Nerf-W [Martin-Brualla et al. CVPR 2021]

Appearance Embedding: Pretty Robust Solution



N-dim vector

Optimized *per* image: “Auto-Decoding”

ie GLO: Generative Latent Optimization [Bojanowski et al. ICML 2018]

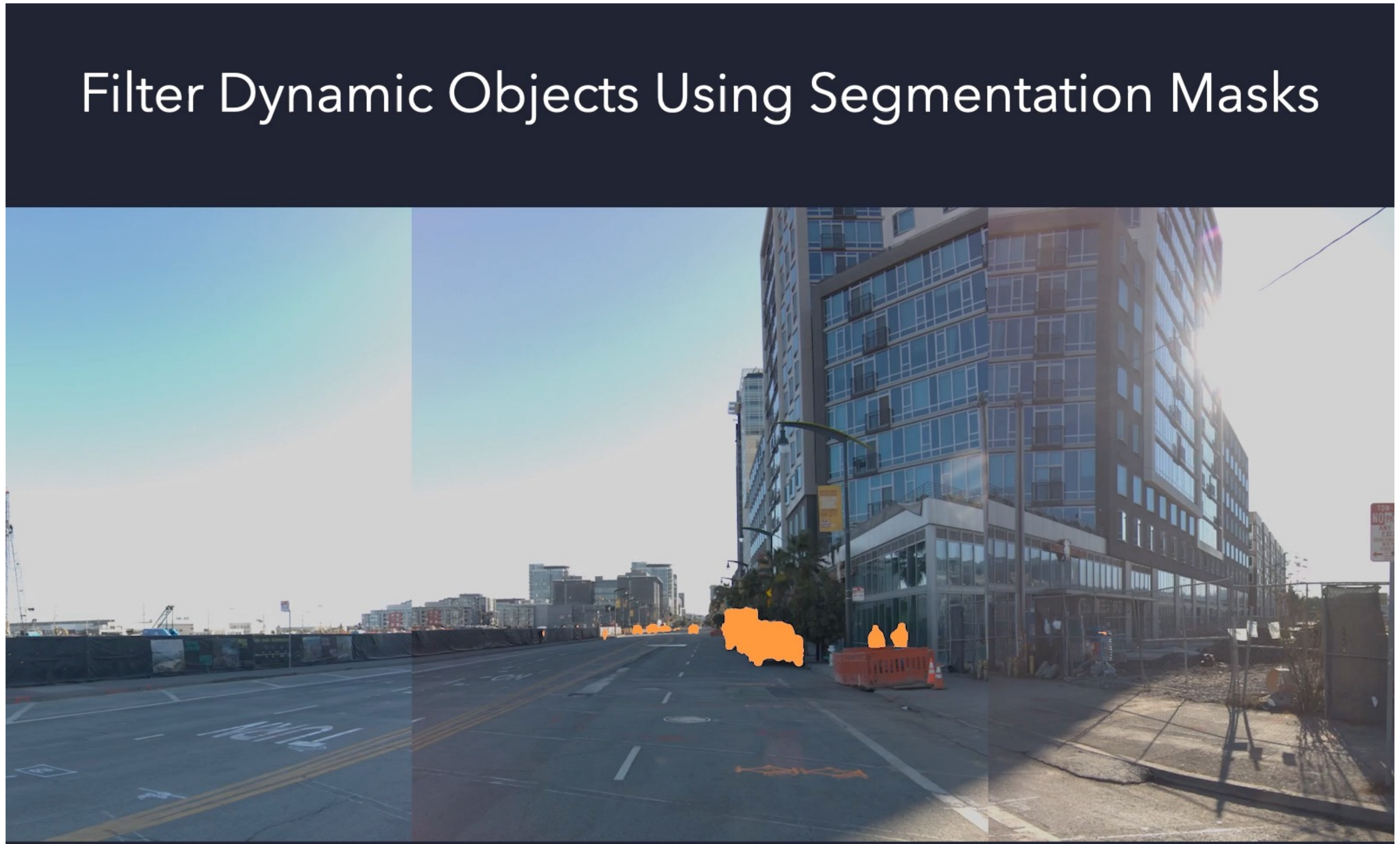
Nerf-W [Martin-Brualla et al. CVPR 2021]

Appearance Changes

Appearance Encoding is Effective

Transient objects

- Happens all the time! People moving around, interacting with the world
- Difficult! Problem of Grouping
  - how do you know which part is connected or
  - Can use two NeRFs, one global, one per-image, but this often leads to degenerate solutions
- Current solution: Ignore (mask out)



Why is dynamic scenes hard?

- Unless you have a light dome
- Essentially you only have a single-view

Building & Reusing Prior Knowledge

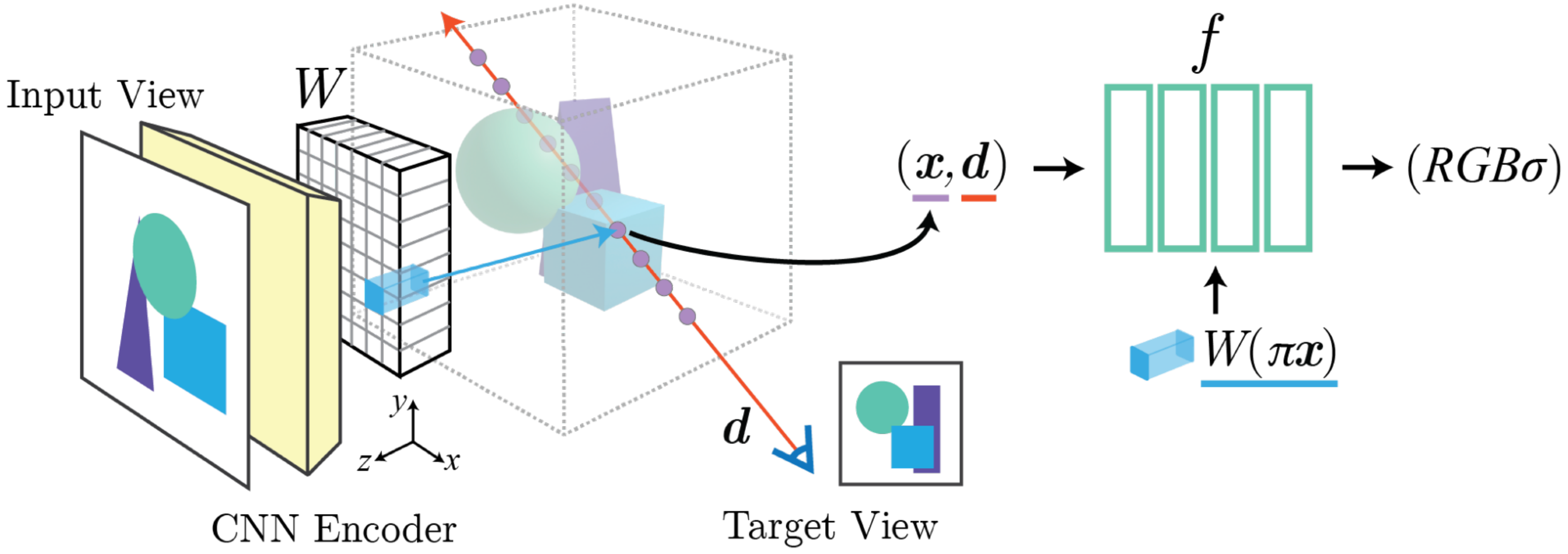
Machine Learning

NeRF is per-scene optimization

- We need lots of images to get good view synthesis!!
- Also there’s no knowledge reused from prior scene reconstructions
- How to bring learning in the picture?

Few-shot NeRF

- One-shot (single-view): pixelNeRF [Yu et al. CVPR'19]



- Few-shot (3~10 views): pixelNeRF,I BRNet [Wang et al. CVPR'21], MVSNet [Chen et al. ICCV'21], etc…
- Challenging for predicting completely unseen real scenes

- How to deal with the multi-modal nature of the problem??

IBRNet

Data is the bottleneck

- Large-scale Real-World Multi-view Data is hard to collect: CO3D [Reizenstein ICCV 2021]

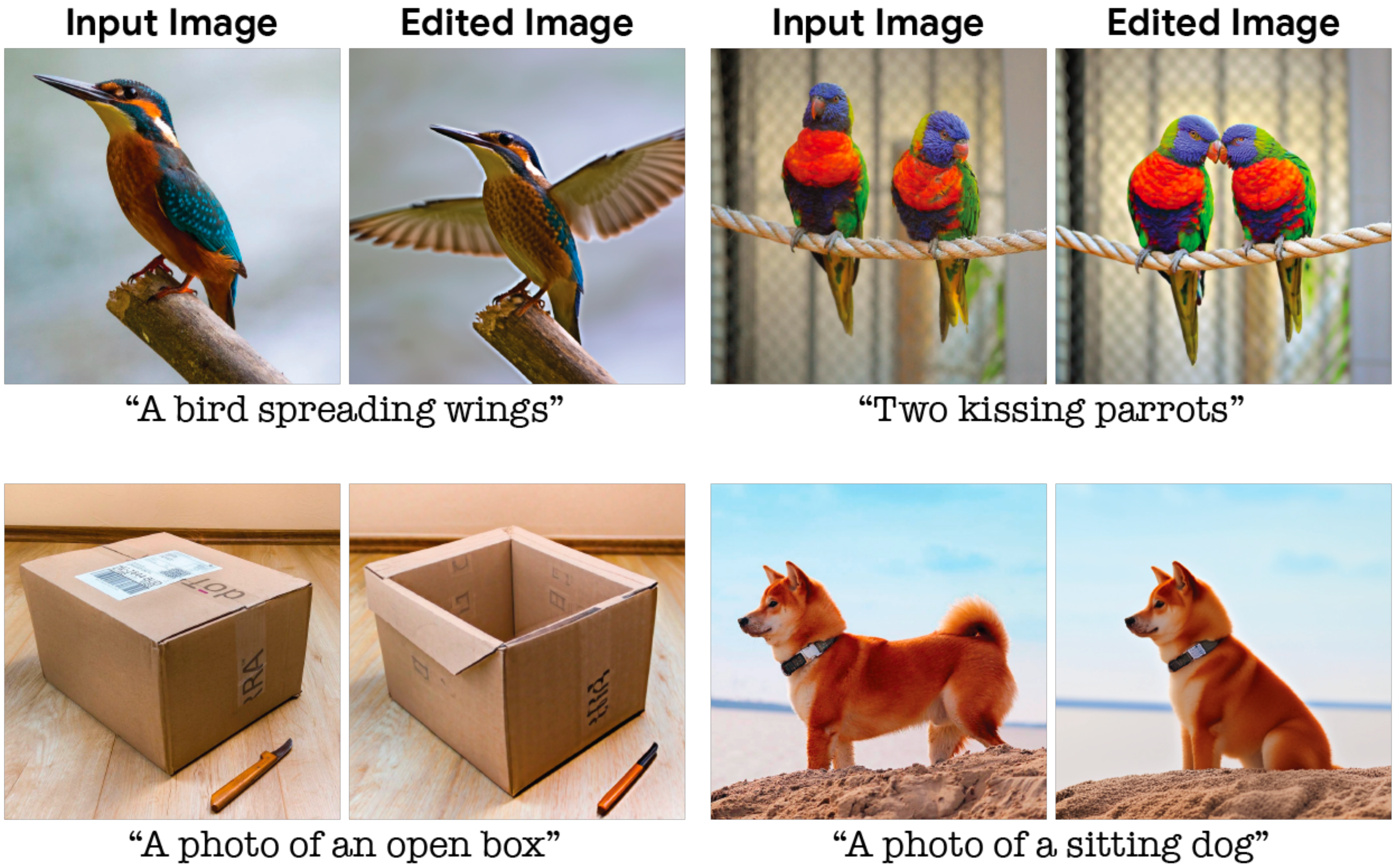
- A lot to learn from other single-view 3D prediction models:

Gkioxari et al. CVPR 2022

Generating NeRFs from 2D Generative Models

DreamFusion
[Poole et al.
arXiv 2022]

Enabling specific edits



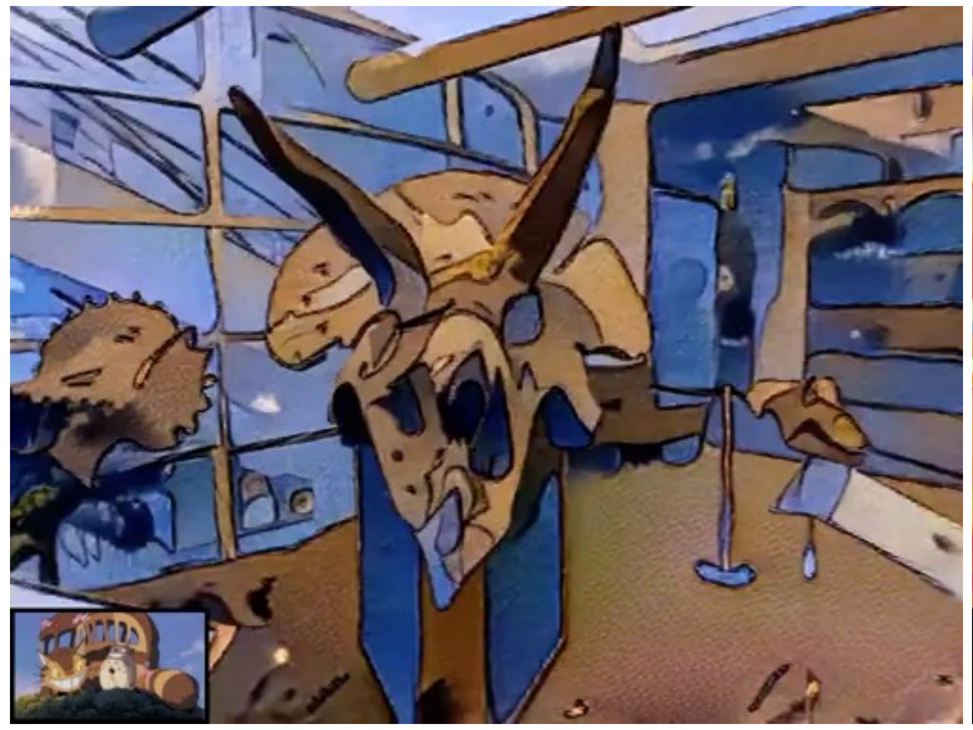
https://imagic-editing.github.io/

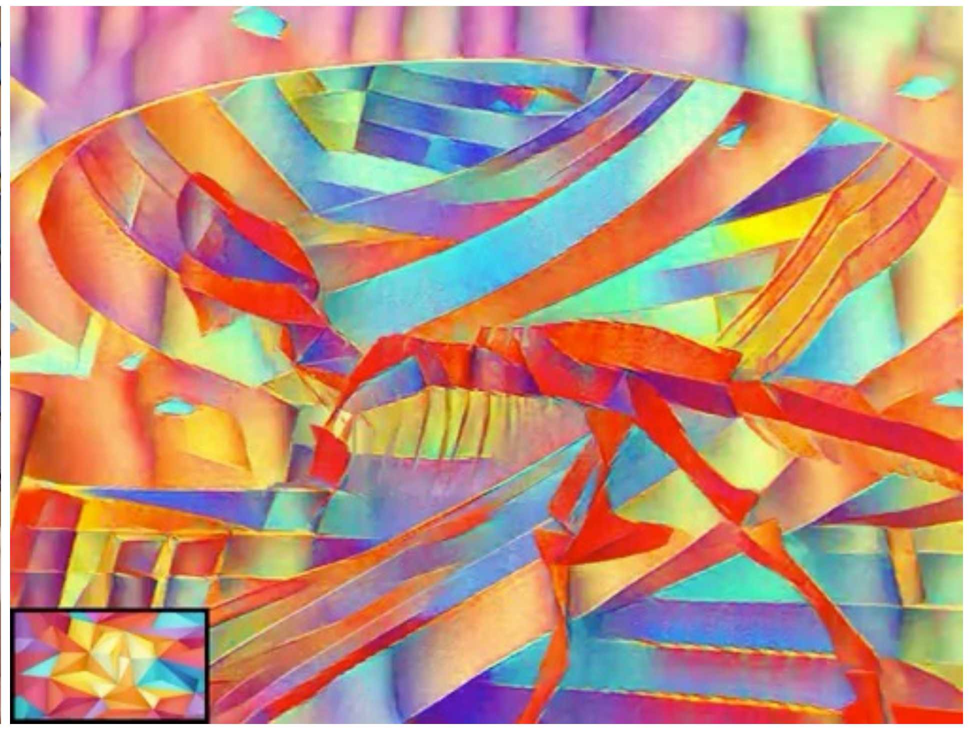
Kawar and Zada et al. Arxiv 2022

Semantic Editing

ClipNeRF Wang et al. CVPR 2022, Feature Field Distillation - Kobayashi et al. NeurIPS 2022 … Many more papers here!!

Manipulating captured scenes





Artistic Radiance Fields Zhang et al. ECCV 2022

nerfstudio

Matthew Tancik*, Ethan Weber*, Evonne Ng*, Ruilong Li, Brent Yi,
Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi,
Abhik Ahuja, David McAllister, Angjoo Kanazawa

+14 additional Github collaborators

Workflow

Step 1: Capture Data

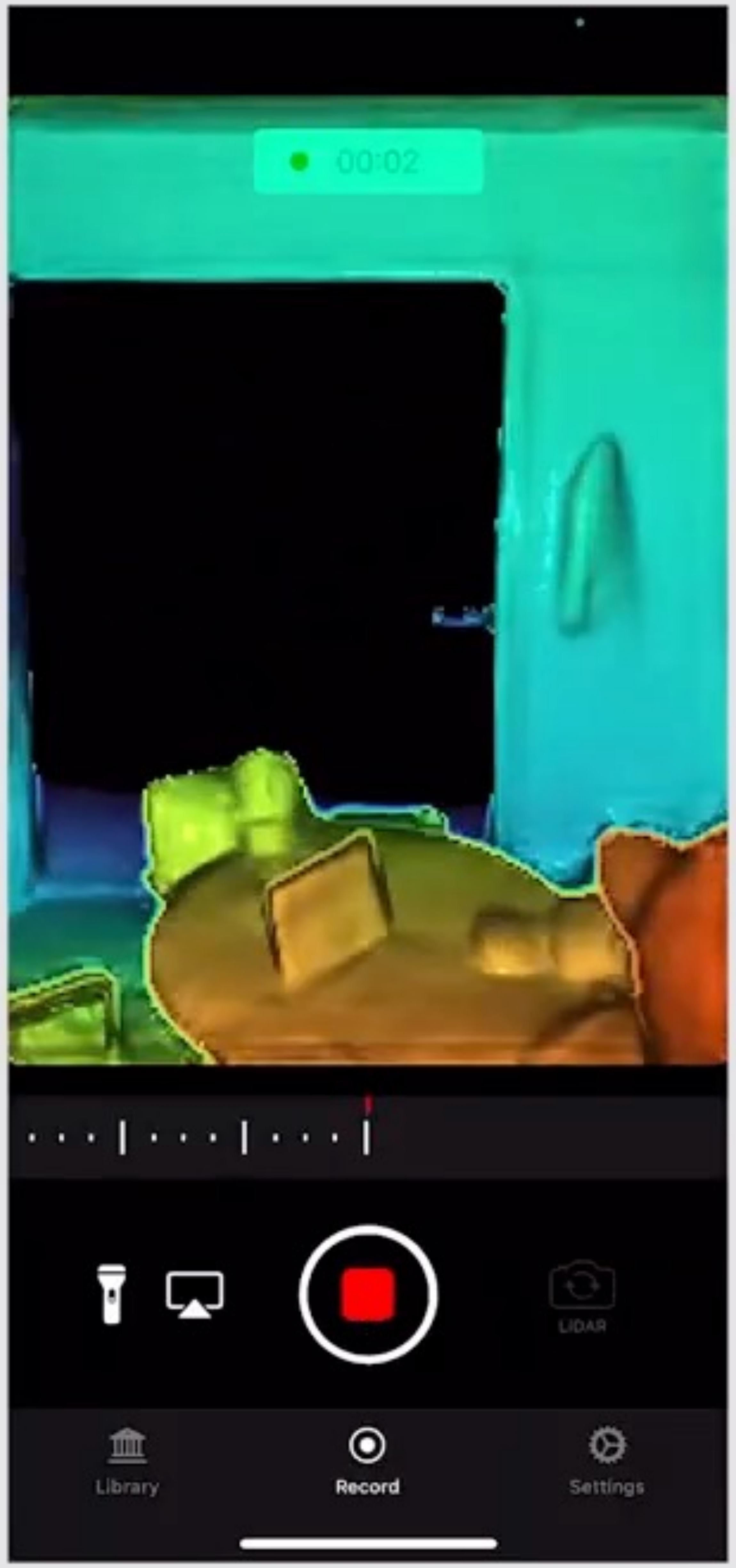
Step 1: Capture Data

- Maximize view coverage
  - Try to get at least 5 views per point in scene.
  - Wide angle / Fisheye lenses work well
- Minimize motion blur
  - Noise is an OK tradeoff
- Minimize dynamic objects

Step 2: Recover Camera Poses

COLMAP scripts

Step 2: Recover Camera Poses



COLMAP Alternative
Record3D

Step 3: Optimize NeRF!

```
(nerfstudio) :~/nerfstudio$ ll data/nerfstudio/desolation/
total 93541
drwxr-xr-x 7 evonneng users        9 Oct  5 00:58 ./
drwxr-xr-x 9 evonneng users        9 Oct  5 00:58 ../
drwxr-xr-x 2 evonneng users        3 Oct  5 00:58 camera_paths/
drwxr-xr-x 8 evonneng users        9 Oct  5 01:00 colmap/
drwxr-xr-x 2 evonneng users      209 Oct  5 00:58 images/
drwxr-xr-x 2 evonneng users      209 Oct  5 00:58 images_2/
drwxr-xr-x 2 evonneng users      209 Oct  5 00:58 images_4/
-rw-r--r-- 1 evonneng users 95638275 Oct  5 00:58 IMG_8981.MOV
-rw-r--r-- 1 evonneng users   177295 Oct  5 00:58 transforms.json
(nerfstudio) :~/nerfstudio$ ns-train nerfacto --data data/nerfstudio/desolation/ --viewer.websocket-port 7
```


GETTING STARTED
GITHUB
DOCUMENTATION
nerfstudio
VIEWPORT
RENDER VIEW
RESUME TRAINING
Show Scene
Show Images
Refresh Page
Resolution: 640x1024px
Time Allocation: 100% spent on viewer
Server Connected
Render Connected
CONTROLS
RENDER
SCENE
LOAD PATH
EXPORT PATH
Height
1080
Width
1920
FOV
50
Seconds
4
FPS
24
ADD CAMERA
Smoothness
0.00
0
1
2
3
CAMERA 0
0
1
2

Step 4: Render

```
(nerfstudio) :~/nerfstudio$
```

Goals of nerfstudio

- Modular Framework
- Open, Evolving Framework
- Reference Source

Modularity

Encoders

- Positional Encoding
- Fourier Features
- Hash Encoding
- Spherical Harmonics
- Matrix Decomposition

Samplers

- Uniform
- Occupancy
- PDF
- Proposal
- Spacing Fn

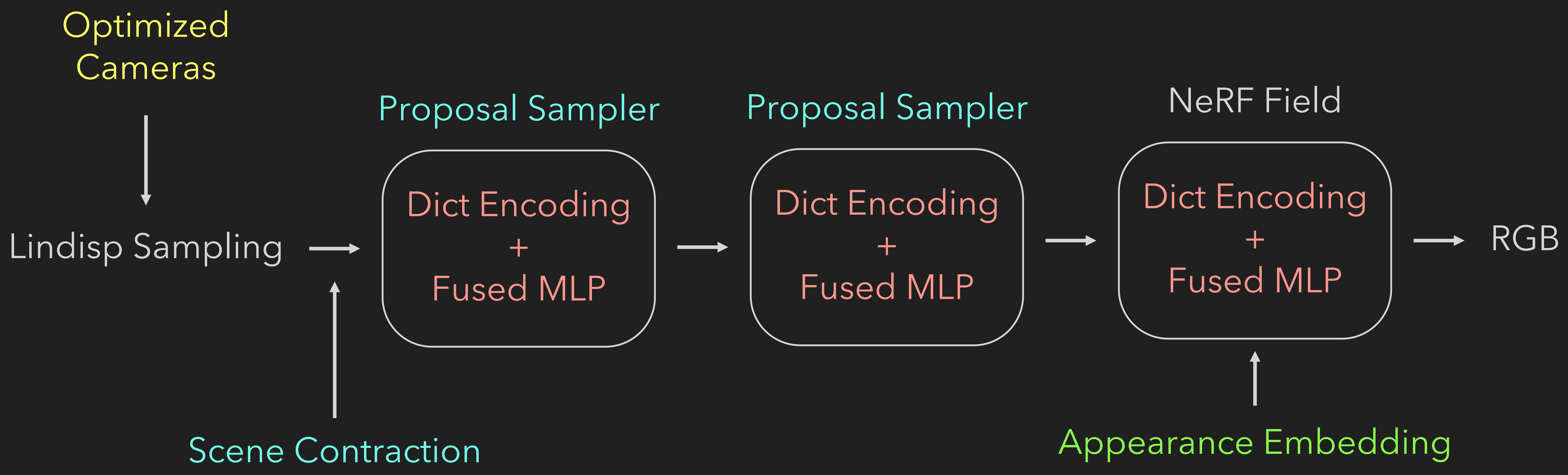
Fields

- Fused MLP
- Voxel Grid

Renderers

- RGB
- RGB-SH
- Depth
- Accumulation

Case study: Nerfacto Model



Current Model

Case study: Nerfacto Model

Since Release:

- 1.5x Faster training
- 3x Less Memory
- Improved Quality

Before

After

Spatial Distortions

If you are trying to reconstruct an object floating in an empty void, you can stop reading. However if you are trying to reconstruct a scene or object from images, you may wish to consider adding a spatial distortion.

When rendering a target view of a scene, the camera will emit a camera ray for each pixel and query the scene at points along this ray. We can choose where to query these points using different samplers. These samplers have some notion of *bounds* that define where the ray should start and terminate. If you know that everything in your scenes exists within some predefined bounds (ie. a cube that a room fits in) then the sampler will properly sample the entire space. If however the scene is unbounded (ie. an outdoor scene) defining where to stop sampling is challenging. One option to increase the far sampling distance to a large value (ie. 1km). Alternatively we can warp the space into a fixed volume. Below are supported distortions.

Scene Contraction

Contract unbounded space into a ball of radius 2. This contraction was proposed in MipNeRF-360. Samples within the unit ball are not modified, whereas sample outside the unit ball are contracted to fit within the ball of radius 2.

We use the following contraction equation:

$$f(x) = \begin{cases} x & ||x|| \leq 1 \\ (2 - \frac{1}{||x||})(\frac{x}{||x||}) & ||x|| > 1 \end{cases}$$

Below we visualize a ray before and after scene contraction. Visualized are 95% confidence intervals for the multivariate Gaussians for each sample location (this guide explains why the samples are represented by Gaussians). We are also visualizing both a unit sphere and a radius 2 sphere.




nerfstudio

docs.nerf.studio

Discord