Fast Rendering of Neural Radiance Fields

Lingjie Liu

Background

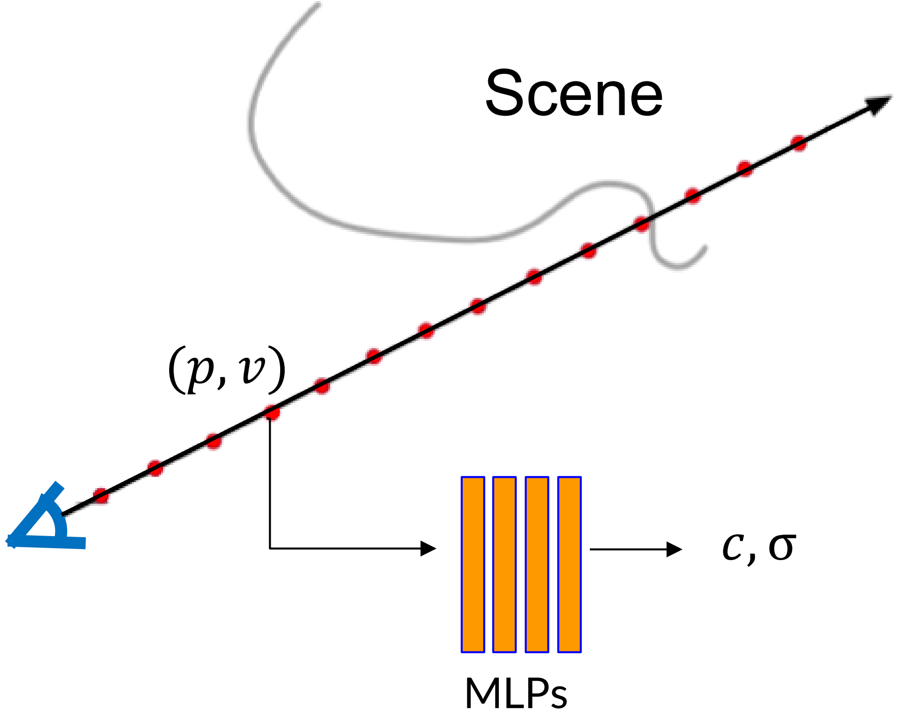


Illustration of volume rendering in NeRF

Background

- NeRF (Mildenhall et al. 2020)

Rendering speed: 100 s/frame

Image resolution: 1920x1080

To render an image at 1920x1080 pixels, how many calls of the MLPs are needed?

(1920x1080) x 192 = 398, 131, 200

It takes about 100 seconds to render such an image using an NVIDIA V100 GPU

Background

- NeRF (Mildenhall et al. 2020)

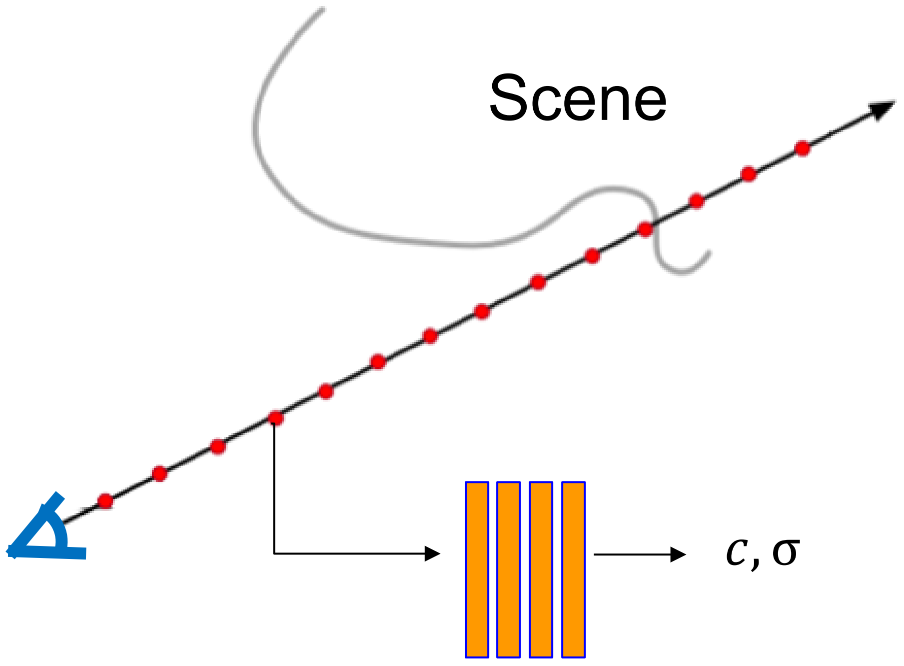


Illustration of volume rendering in NeRF

To render an image at 1920x1080 pixels, how many calls of the MLPs are needed?

(1920x1080) x 192 = 398, 131, 200

It takes about 100 seconds to render such an image using an NVIDIA V100 GPU

Two possible ideas to accelerate the rendering process:

1. Reduce sampling points.
2. Reduce the runtime for one pass.

Neural Sparse Voxel Fields

Lingjie Liu*, Jiatao Gu*, Kyaw Zaw Lin, Tat-Seng Chua, Christian Theobalt (* equal contribution)

NeurIPS 2020 Spotlight

Our Method -- Neural Sparse Voxel Fields **(NSVF)**

- Avoid sampling points in empty space as much as possible.
- Neural Sparse Voxel Fields (NSVF), a hybrid scene representation for fast and high-quality free-viewpoint rendering.

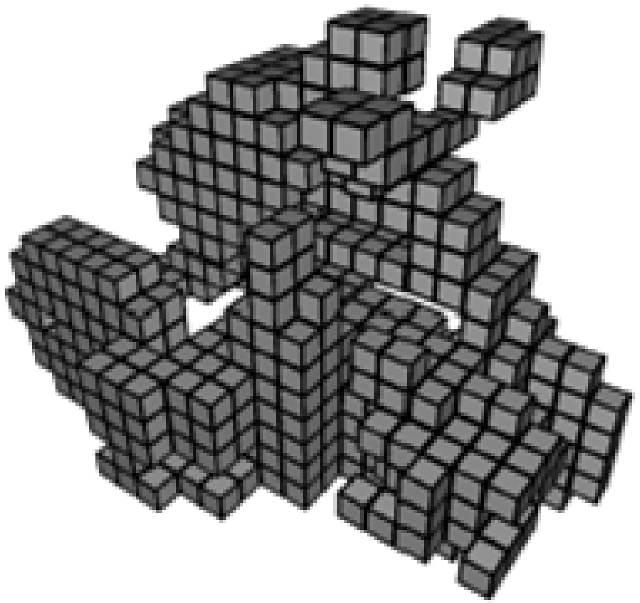
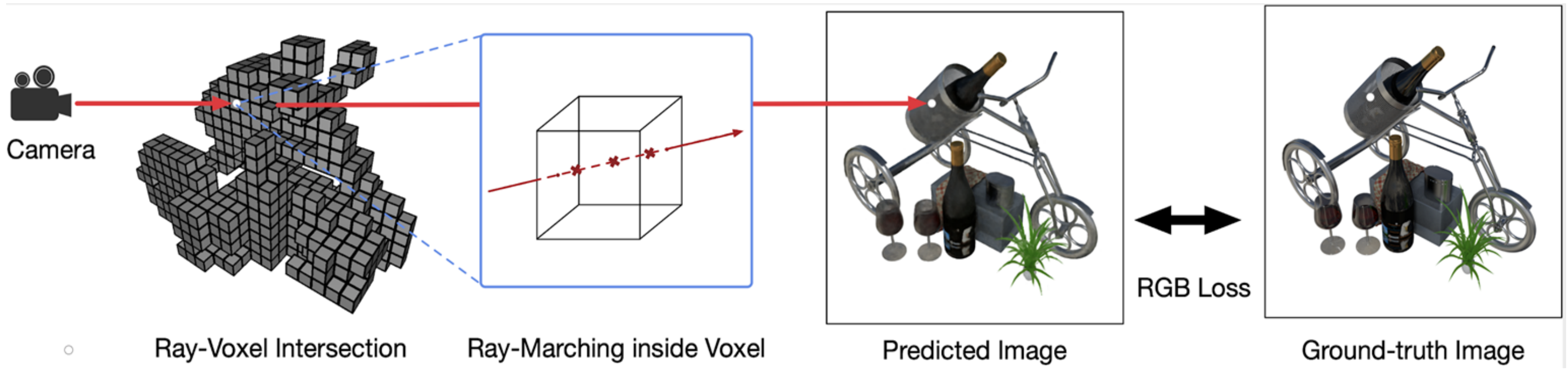


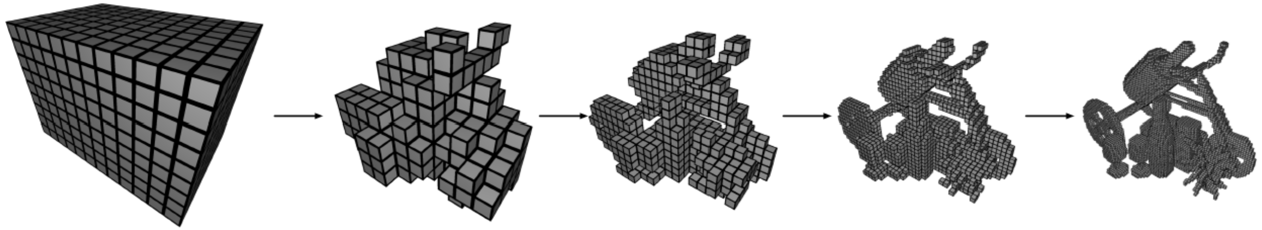
Illustration of NSVF

Our Method (NSVF)

- Scene Representation - Neural Sparse Voxel Fields (NSVF).
- Volume Rendering with NSVF.



- Progressive Learning: we train NSVF progressively with the differentiable volume rendering operation from a set of posed 2D images.



Scene Representation - NSVF

The scene is modeled as a set of voxel-bounded implicit functions:

$$F_\theta(\boldsymbol{p}, \boldsymbol{v})$$

The relevant non-empty parts of a scene are contained within a set of sparse bounding voxels:

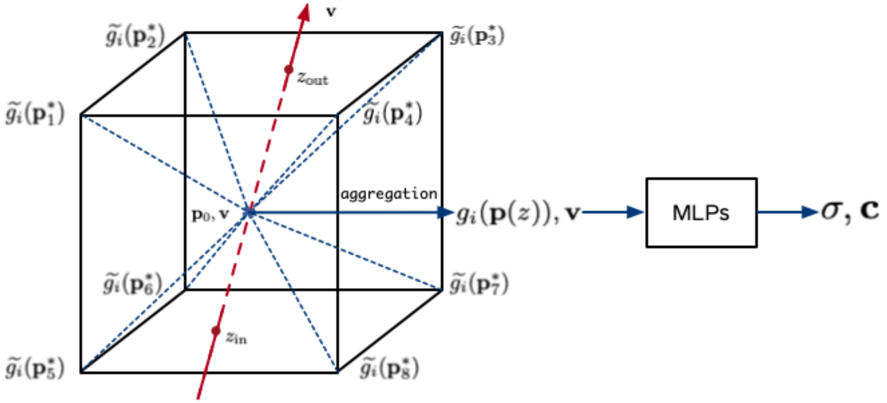
$$\mathcal{V} = \{V_1 \dots V_K\}$$

Scene Representation - NSVF

A voxel-bounded implicit field

- For a given point **p** inside voxel Vi, the voxel-bounded implicit field is defined as: F_θ^i

$$F_\theta^i : (\underbrace{\boldsymbol{g}_i(\boldsymbol{p})}_{\text{voxel embedding}}, \underbrace{\boldsymbol{v}}_{\text{ray direction}}) \rightarrow (\underbrace{\boldsymbol{c}}_{\text{color}}, \underbrace{\sigma}_{\text{density}}), \forall \boldsymbol{p} \in V_i,$$



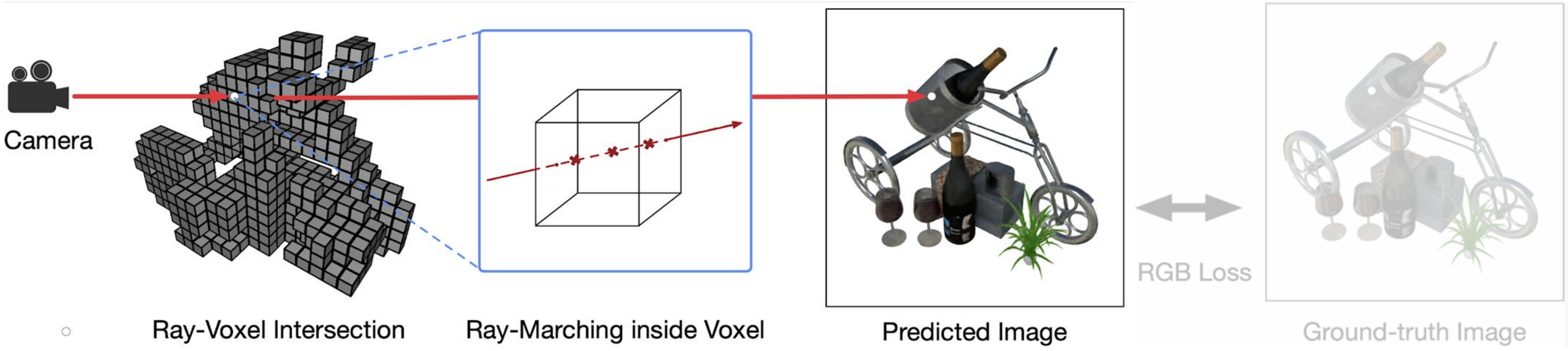
- Voxel embedding is defined as:

$$g_i(\boldsymbol{p}) = \underbrace{\boxed{\zeta}}_{\textit{Positional encoding}}\Big(\overbrace{\boxed{\chi}}^{\textit{Trilinear interpolation}}\big(\widetilde{g}_i(\boldsymbol{p}_1^*), \ldots, \overbrace{\boxed{\widetilde{g}_i}}^{\textit{Voxel features (e.g. learnable voxel embeddings)}}(\boldsymbol{p}_8^*)\big)\Big)$$

Volume Rendering with NSVF

Rendering NSVF is fast because it avoids sampling points in the empty space.

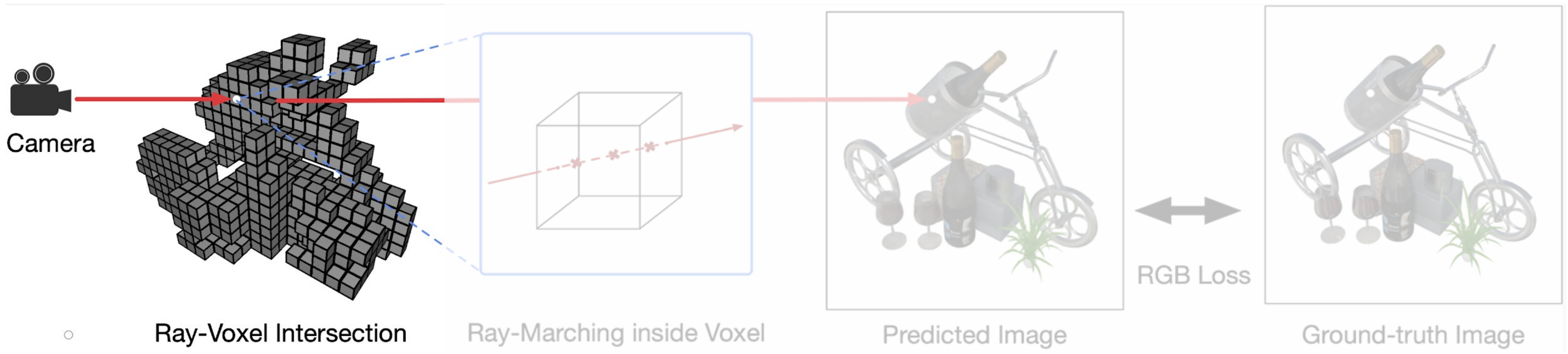
- Ray-voxel Intersection.
- Ray marching inside voxels.



Volume Rendering with NSVF

Ray-voxel Intersection

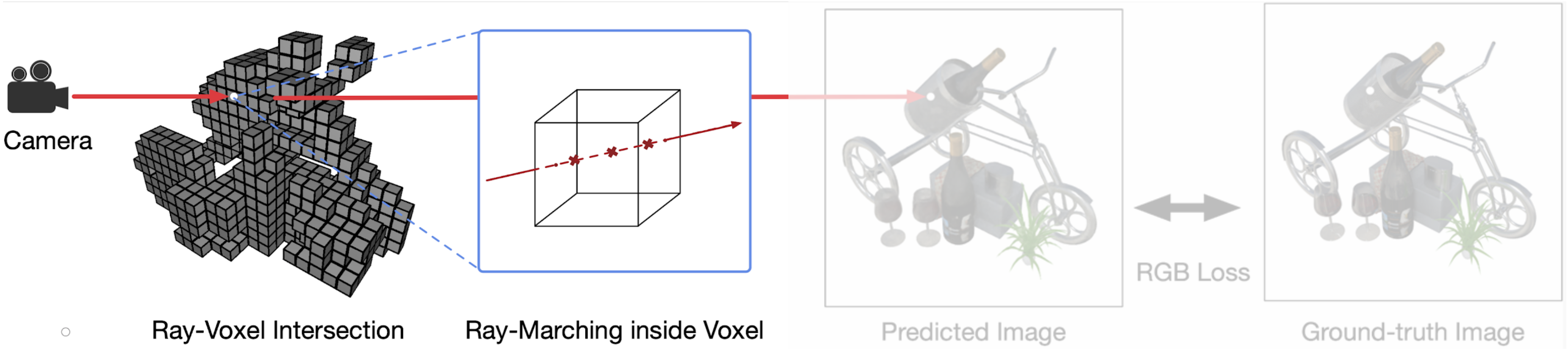
- Apply Axis Aligned Bounding Box (AABB) intersection test [Haines, 1989] for each ray.
- AABB is very efficient for NSVF, handling millions of ray-voxel intersections in real time.



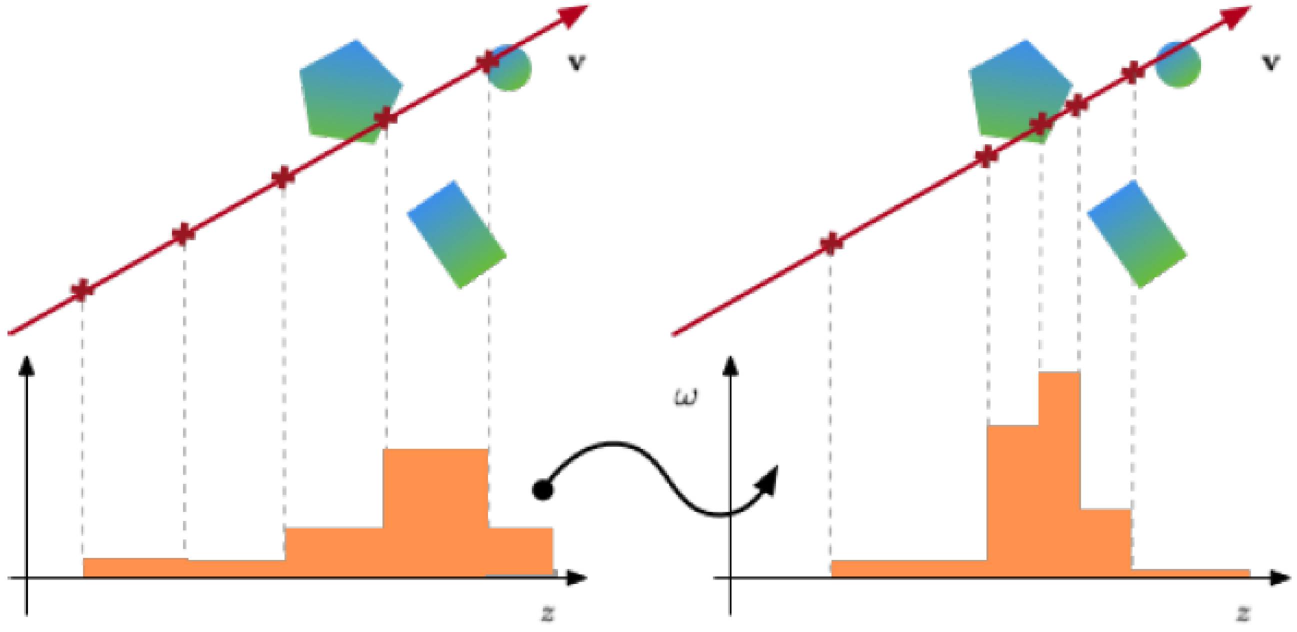
Volume Rendering with NSVF

Ray Marching inside Voxels

- Uniformly sample points along the ray inside each intersected voxel, and evaluate NSVF to get the color and density of each sampled point.



Volume Rendering with NSVF



Coarse sampling

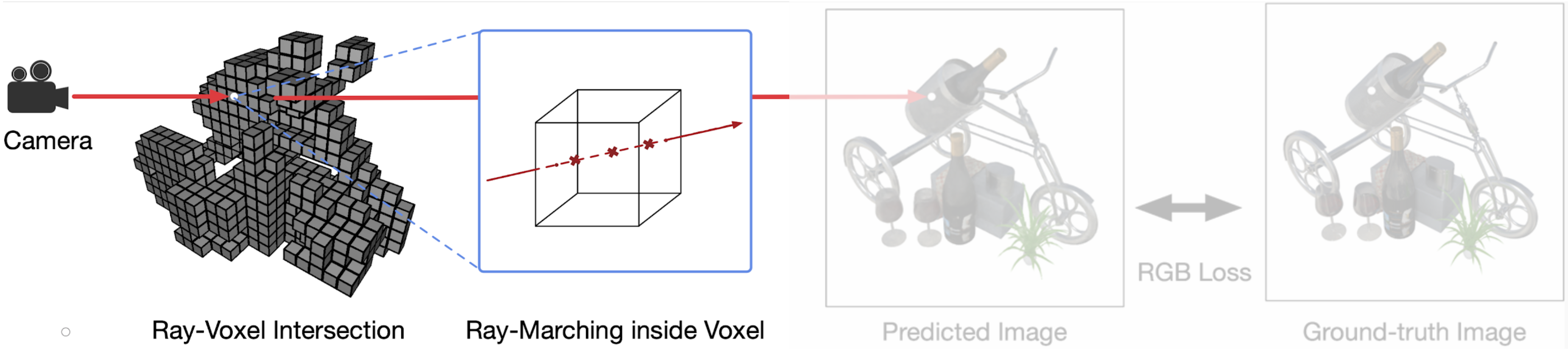
Importance sampling

NeRF’s sampling method

Volume Rendering with NSVF

Early Termination

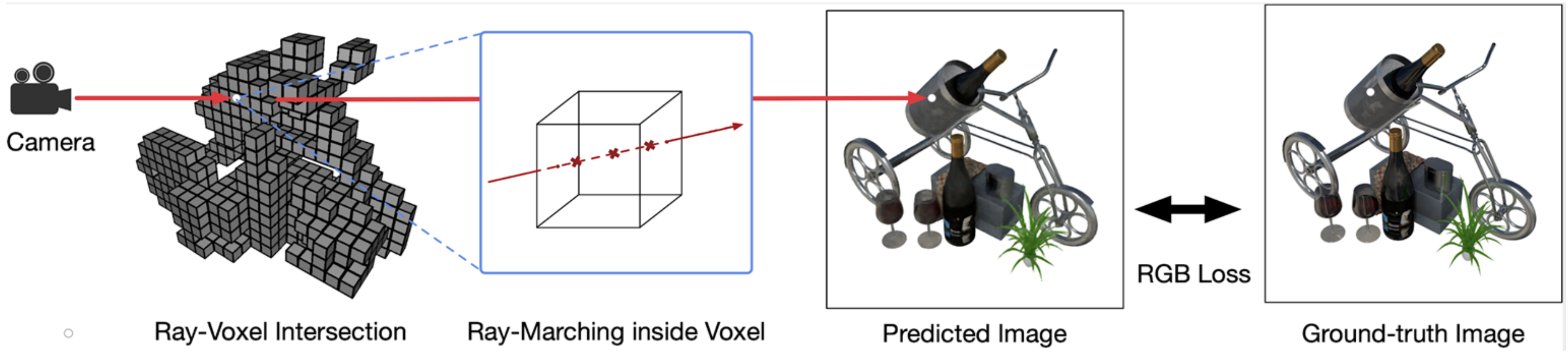
- Avoid taking unnecessary accumulation steps behind the surface;
- Stop evaluating points earlier when the accumulated densities close to 1



Progressive Learning

- Because our rendering process is differentiable, the model can be trained end-to-end with 2D posed images as input for supervision.

$$\mathcal{L} = \sum_{(\boldsymbol{p}_0,\boldsymbol{v})\in R} \| \underbrace{\boldsymbol{C}(\boldsymbol{p}_0,\boldsymbol{v})}_{\text{Predicted color}} - \underbrace{\boldsymbol{C}^*(\boldsymbol{p}_0,\boldsymbol{v})}_{\text{Ground truth color}} \|_2^2 + \lambda \cdot \Omega\left(A(\boldsymbol{p}_0,\boldsymbol{v})\right)$$



Progressive Learning

A progressive training strategy to learn NSVF from coarse to fine

- Voxel Initialization
- Self-Pruning
- Progressive Training

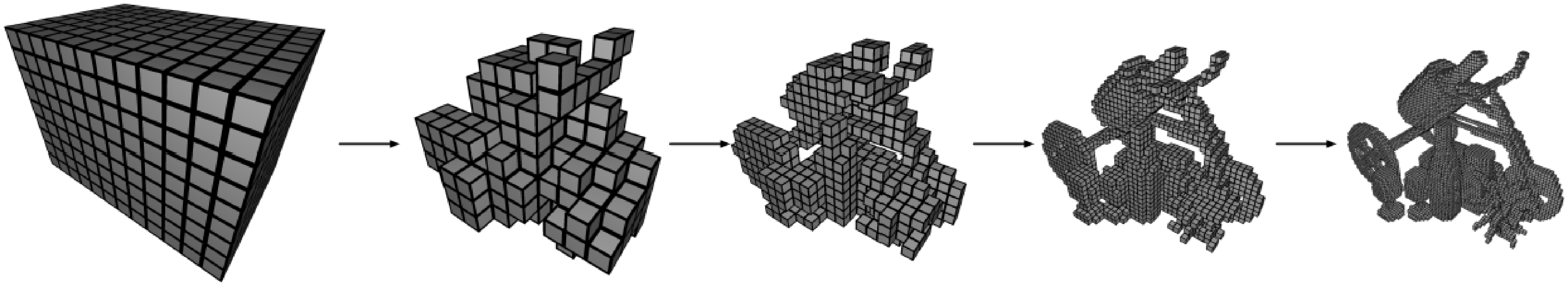
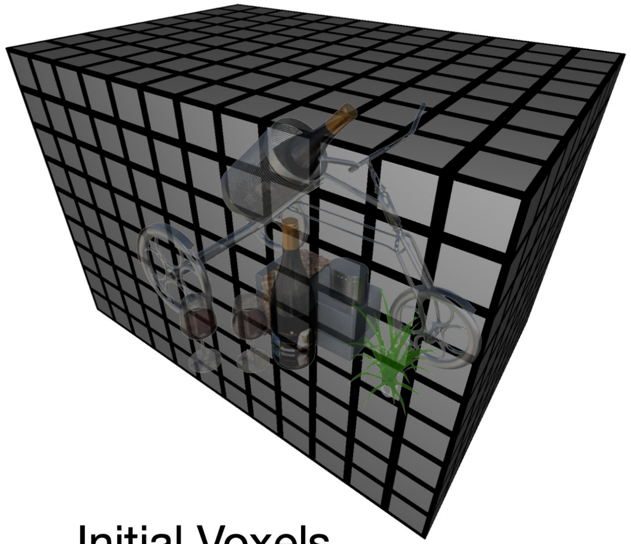


Illustration of self-pruning and progressive training

Progressive Learning

Voxel Initialization

- The initial bounding box encloses the whole scene with sufficient margin. We eventually subdivide the bounding box into ~1000 voxels.
- If a coarse geometry is available, the initial voxels can also be initialized by voxelizing the coarse geometry.



Initial Voxels

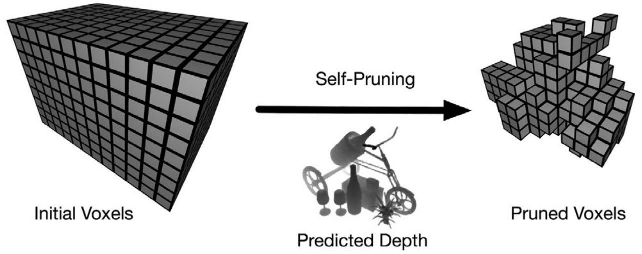
Progressive Learning

Self-Pruning

- We improve rendering efficiency by pruning “empty” voxels.
  - Determine whether a voxel is empty or not by checking the maximum predicted density from sampled points inside the voxel.

$$V_i \text{ is pruned } \textbf{if} \min_{j=1\ldots G} \exp(-\boxed{\sigma}(g_i(\boldsymbol{p}_j))) > \gamma, \quad \boldsymbol{p}_j \in V_i, V_i \in \mathcal{V},$$

density



  - Since this pruning process does not rely on other processing modules or input cues, we call it “*self-pruning*”.

Progressive Learning

Progressive Training

- Self-pruning enables us to progressively allocate our resources.
- Progressive training:
  - Halve the size of voxels → Split each voxel into 8 sub-voxels.
  - Halve the size of ray marching steps.
  - The feature representations of the new vertices are initialized via trilinear interpolation of feature representations at the original eight voxel vertices.

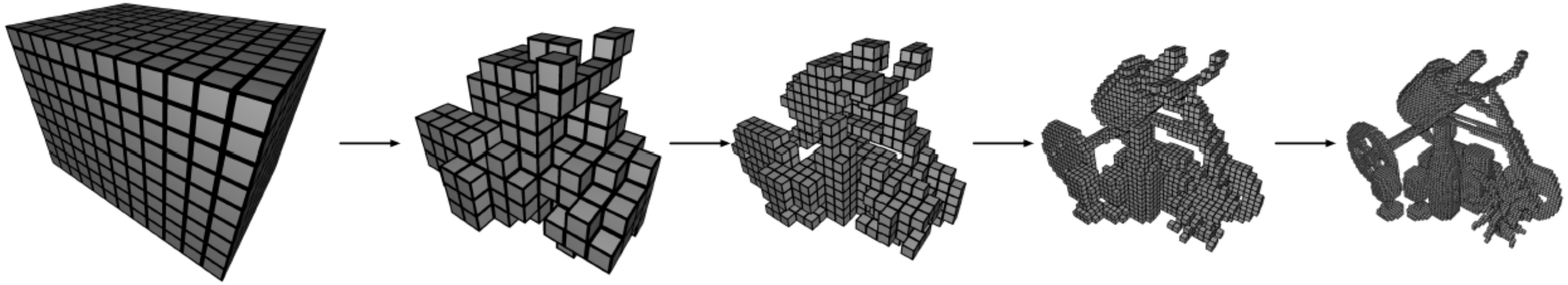


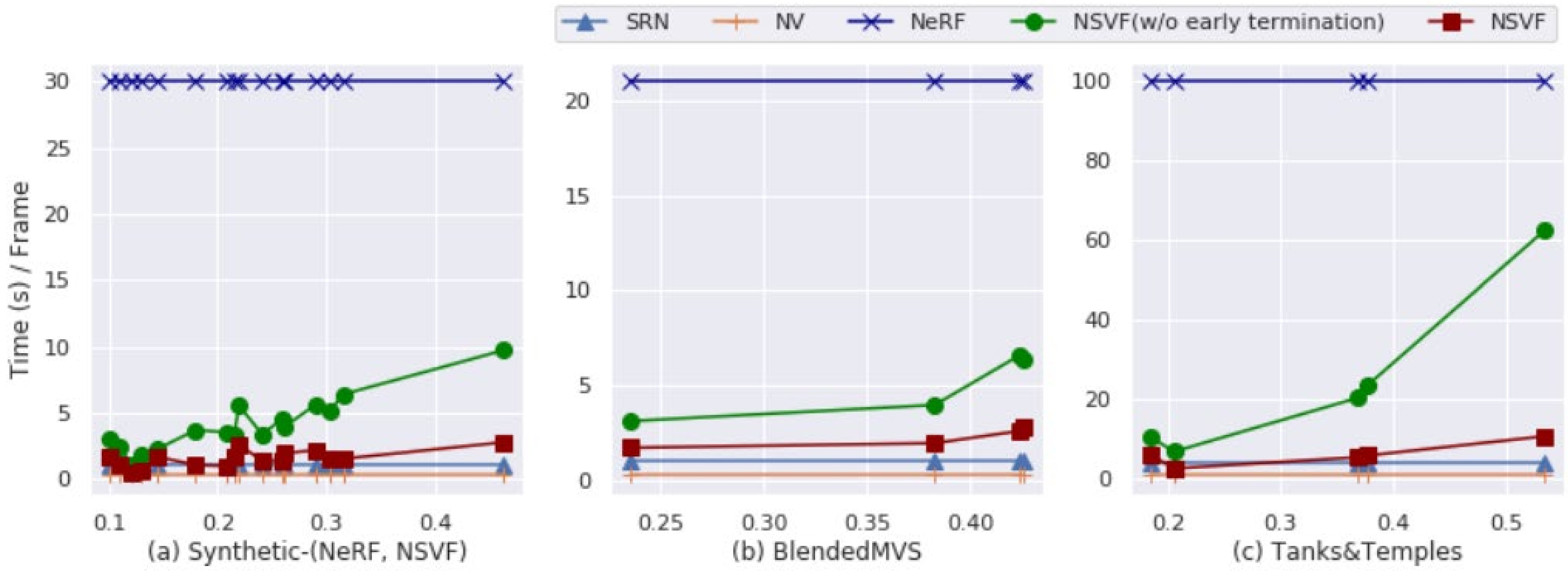
Illustration of self-pruning and progressive training

Results

Models	Synthetic-NeRF PSNR	Synthetic-NeRF SSIM	Synthetic-NeRF LPIPS	Synthetic-NSVF PSNR	Synthetic-NSVF SSIM	Synthetic-NSVF LPIPS	BlendedMVS PSNR	BlendedMVS SSIM	BlendedMVS LPIPS	Tanks and Temples PSNR	Tanks and Temples SSIM	Tanks and Temples LPIPS
SRN	22.26	0.846	0.170	24.33	0.882	0.141	20.51	0.770	0.294	24.10	0.847	0.251
NV	26.05	0.893	0.160	25.83	0.892	0.124	23.03	0.793	0.243	23.70	0.834	0.260
NeRF	31.01	0.947	0.081	30.81	0.952	0.043	24.15	0.828	0.192	25.78	0.864	0.198
NSVF^0	**31.75**	**0.954**	0.048	**35.18**	**0.979**	**0.015**	26.89	**0.898**	0.114	**28.48**	**0.901**	0.155
NSVF	31.74	0.953	**0.047**	35.13	**0.979**	**0.015**	**26.90**	**0.898**	**0.113**	28.40	0.900	**0.153**

*NSVF^0 is without early termination
*NSVF is executed with early termination ($\varepsilon = 0.01$)

Results



x-axis: foreground to background ratio
y-axis: rendering time in second

*NSVF^0 is without early termination **(Green curve)**
*NSVF is executed with early termination $(\varepsilon = 0.01)$ **(Red curve)**

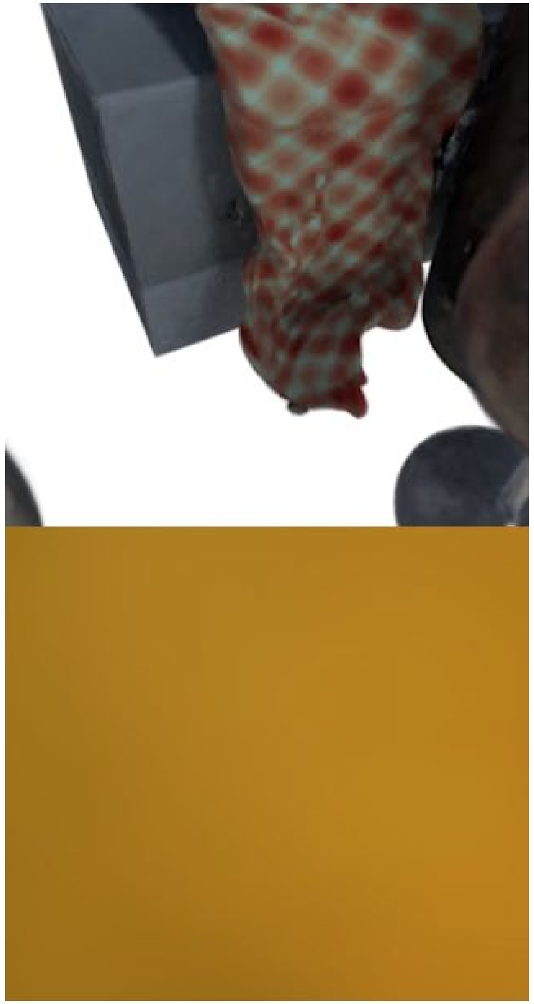
Results

Robot (From SyntheticNSVF dataset)

NeRF (Mildenhall et al. 2020)
(Rendering speed: 30 s/frame)

Ours (NSVF)
(Rendering speed: 0.6 s/frame)

Zoom-in & Zoom-out Effects



Rendering of Dynamic Scenes

Normals of NSVF result

NSVF

(Input sequence from Fraunhofer Heinrich Hertz Institute)

Rendering of Large-scale Indoor Scenes

(Input sequence and 3D mesh from ScanNet [Dai et al. 2017])

Scene Editing and Composition

Interactive editing

NSVF

Main Limitation

- Real-time performance
  - Although our method is typically 10x faster than Nerf, it is still far from real time performance.
  - NeRF 0.06 FPS v.s. NSVF 1.1 FPS v.s. Real-time Rendering >25 FPS

Real-time NeRF Rendering

Caching the Network Outputs with a Sparse Voxel Octree.

- The key idea is to use caching to trade memory for computational efficiency at inference time.

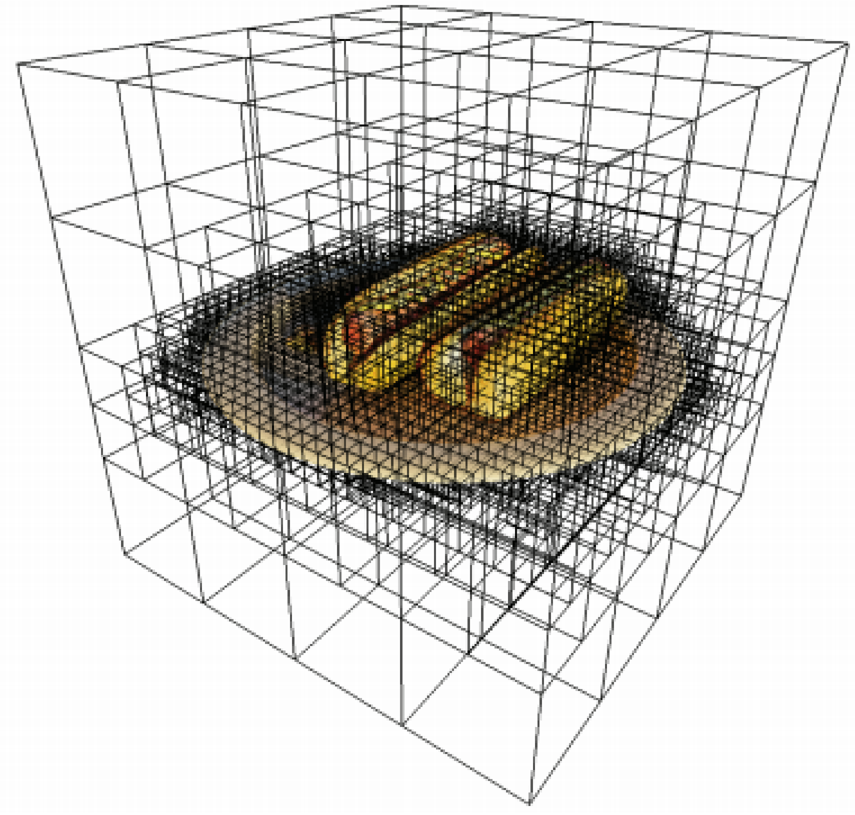


Image from [Yu et al., 2021]

Caching the Network Outputs with a Sparse Voxel Octree.

- The key idea is to use caching to trade memory for computational efficiency at inference time.

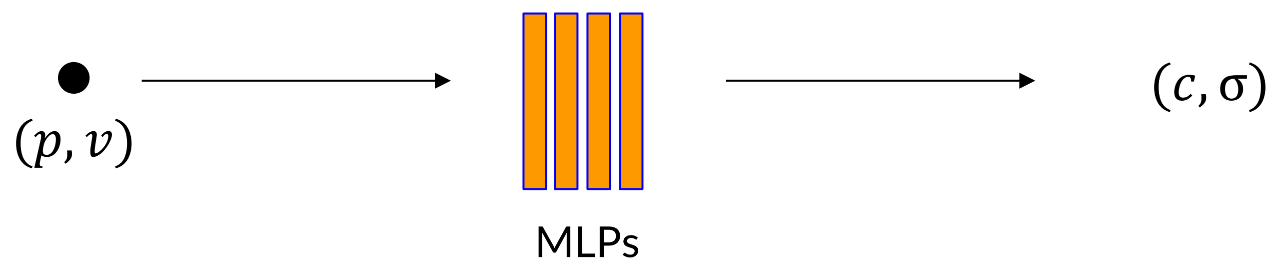
- There are three related papers:
- PlenOctrees for Real-time Rendering of Neural Radiance Fields, Yu et al., Arxiv 2021 ~200FPS
- FastNeRF: High-Fidelity Neural Rendering at 200FPS, Garbin et al., Arxiv 2021 ~200FPS
- Baking Neural Radiance Fields for Real-Time View Synthesis, Hedman et al., Arxiv 2021 ~84FPS

Caching the Network Outputs with a Sparse Voxel Octree.

- The key idea is to use caching to trade memory for computational efficiency at inference time.

- Specifically,

  (1) Train a NeRF-like network to predict density and color for each sampled point.



Caching the Network Outputs with a Sparse Voxel Octree.

- The key idea is to use caching to trade memory for computational efficiency at inference time.

- Specifically,

  (1) Train a NeRF-like network to predict density and color for each sampled point.

  (2) After training, extract the volumetric content and represent it using a sparse voxel Octree.

  (3) Precompute the network outputs for each octree leaf.

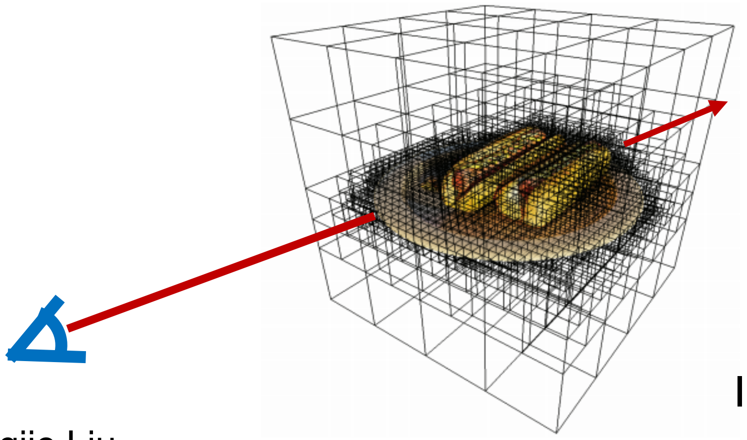


Image from [Yu et al., 2021]

Using Multiple Shallow Networks

- A single high-capacity MLP for representing the entire scene can be replaced with thousands of small MLPs for the decomposed parts of the scene.
- KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs, Reiser et al., Arxiv 2021 ~13 FPS

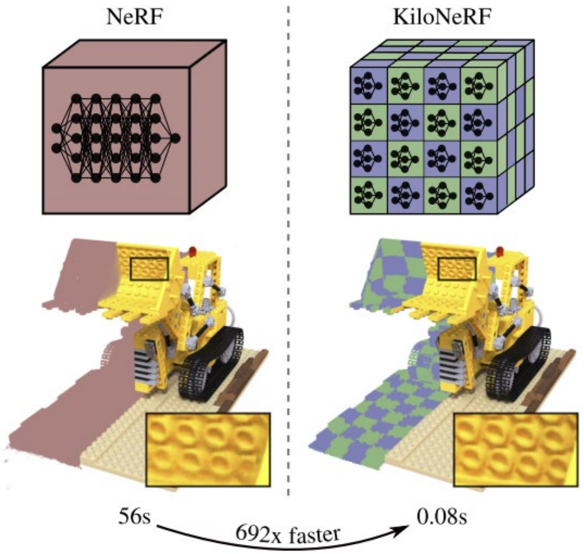


Image from [Reiser et al., 2021]

Using Multiple Shallow Networks

- A single high-capacity MLP for representing the entire scene can be replaced with thousands of small MLPs for the decomposed parts of the scene.
- The similar idea is also used in:

  DeRF: Decomposed Radiance Fields, Rebain et al., CVPR 2021 ~0.18 FPS

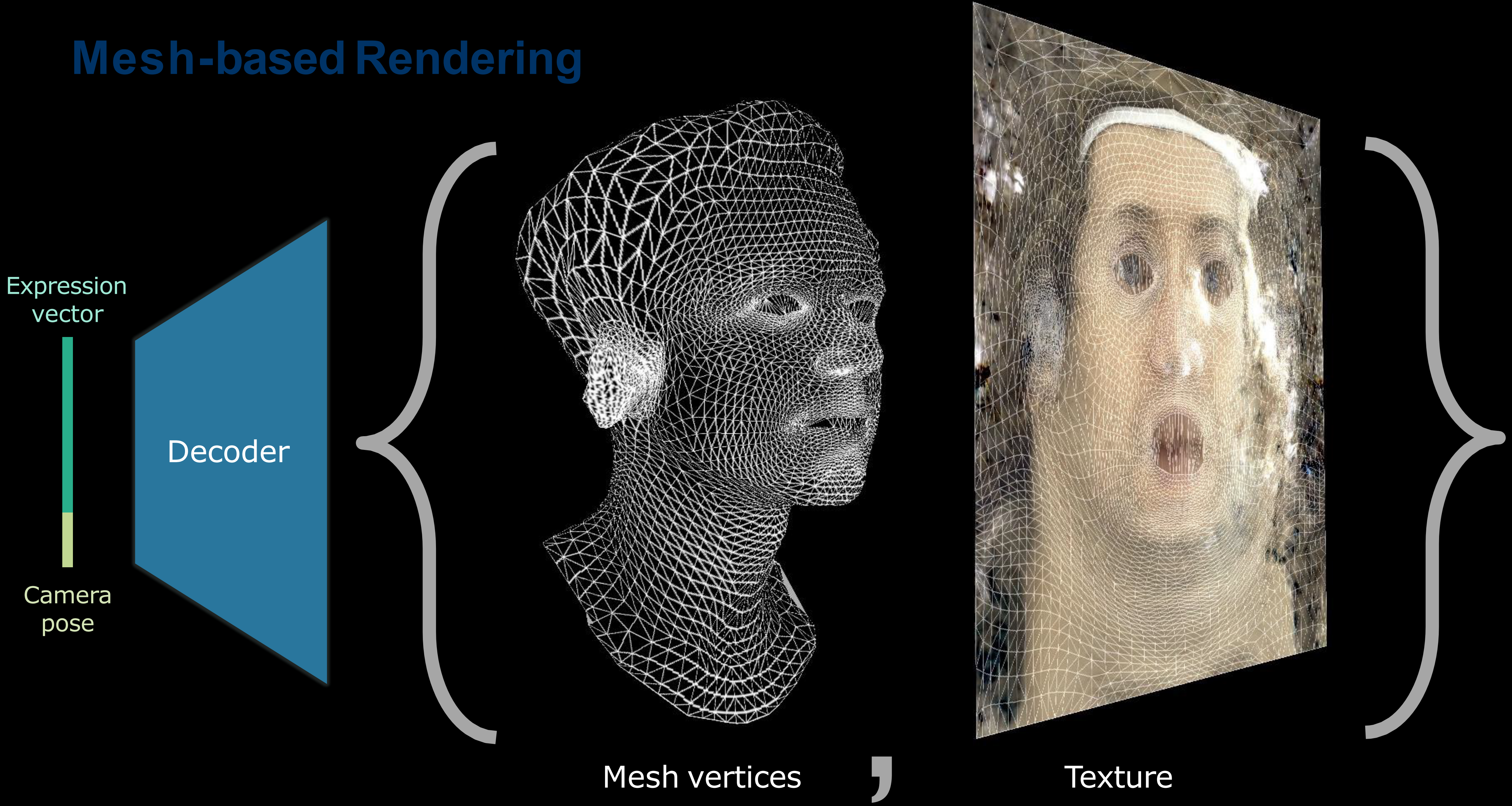
Image from [Rebain et al., 2021]

Mixture of Volumetric Primitives

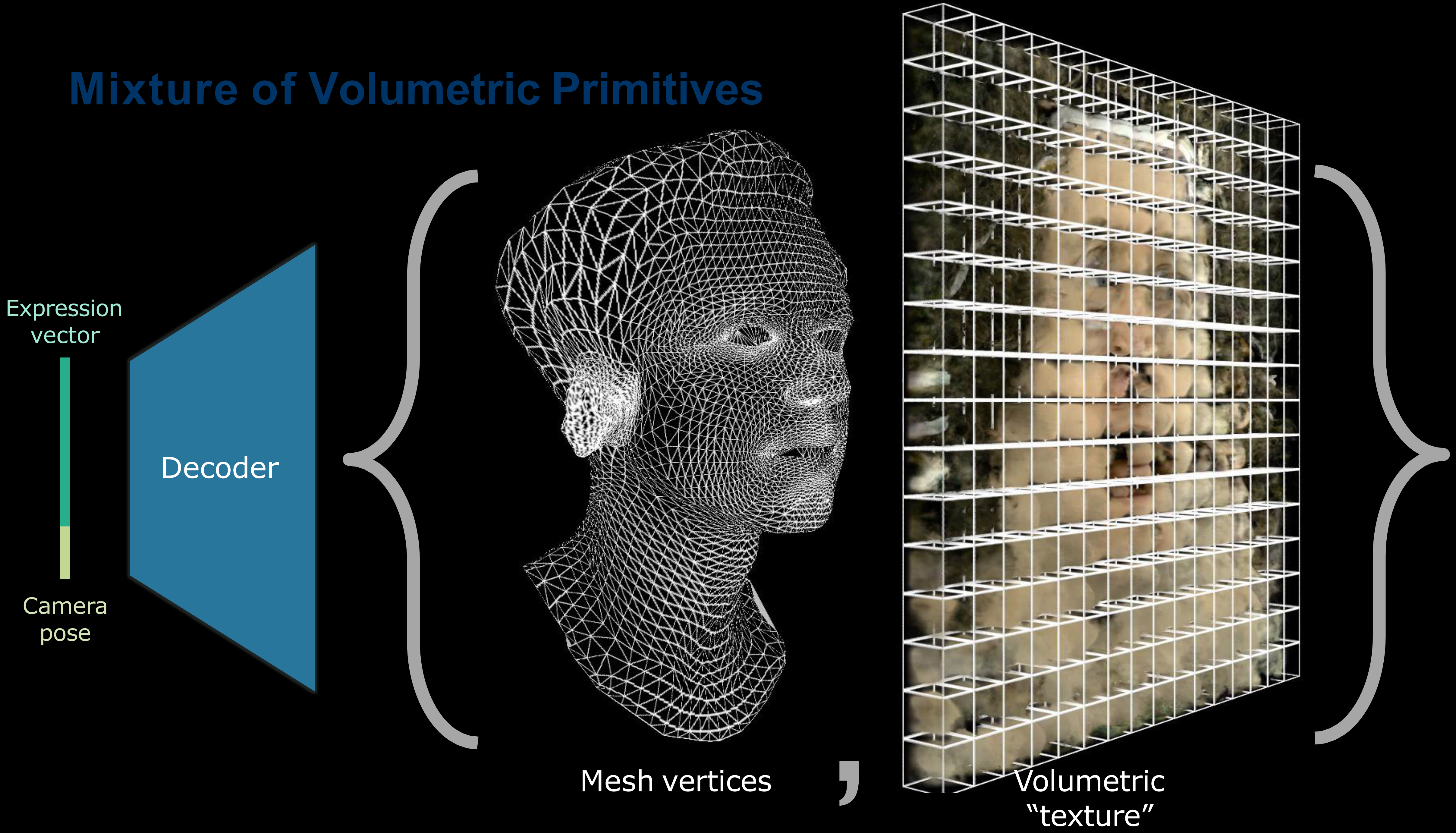
Lombardi, Simon, Schwartz, Zollhoefer, Sheikh, Saragih

SIGGRAPH 2021

Mesh-based Rendering

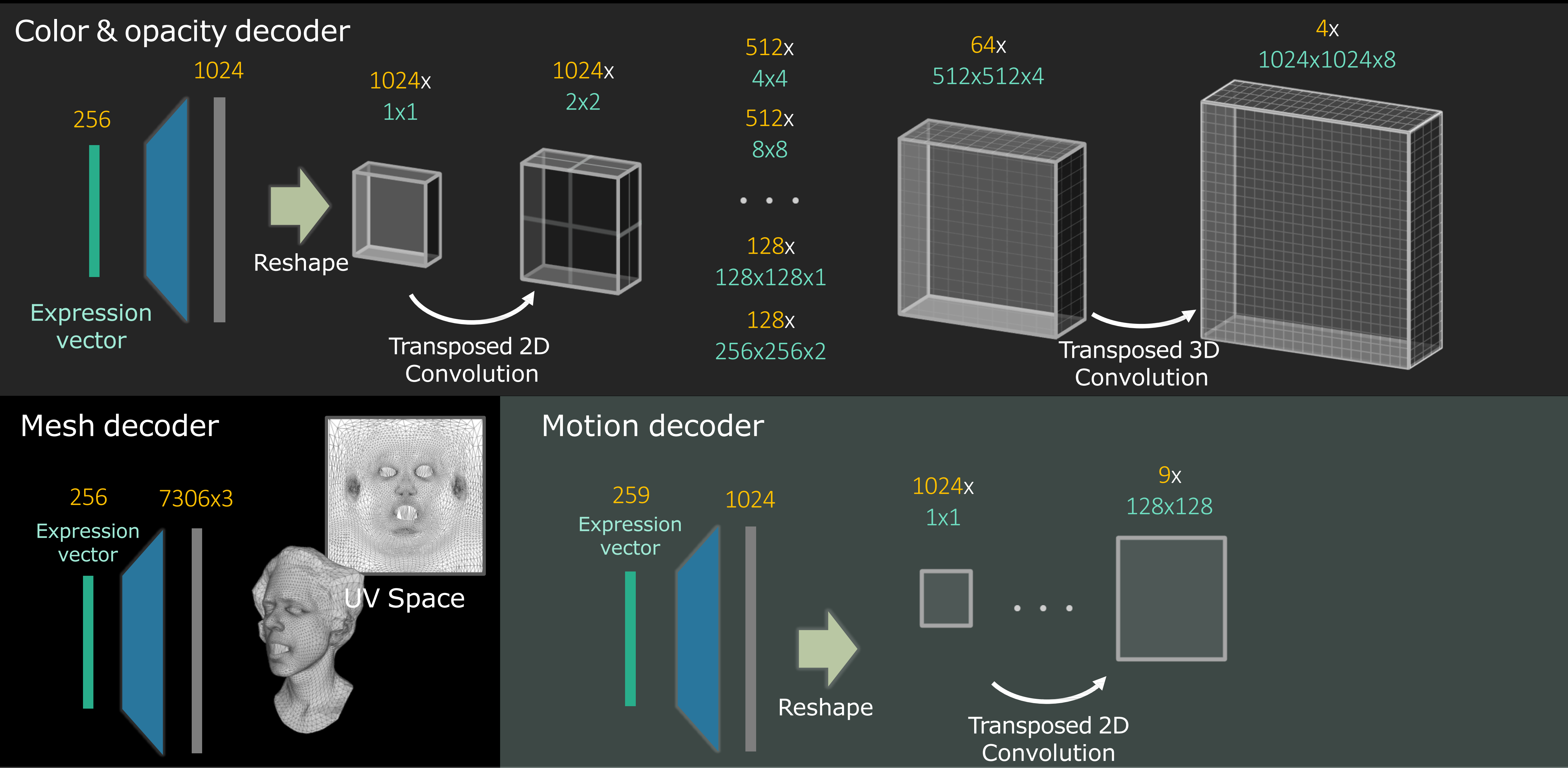


Mixture of Volumetric Primitives

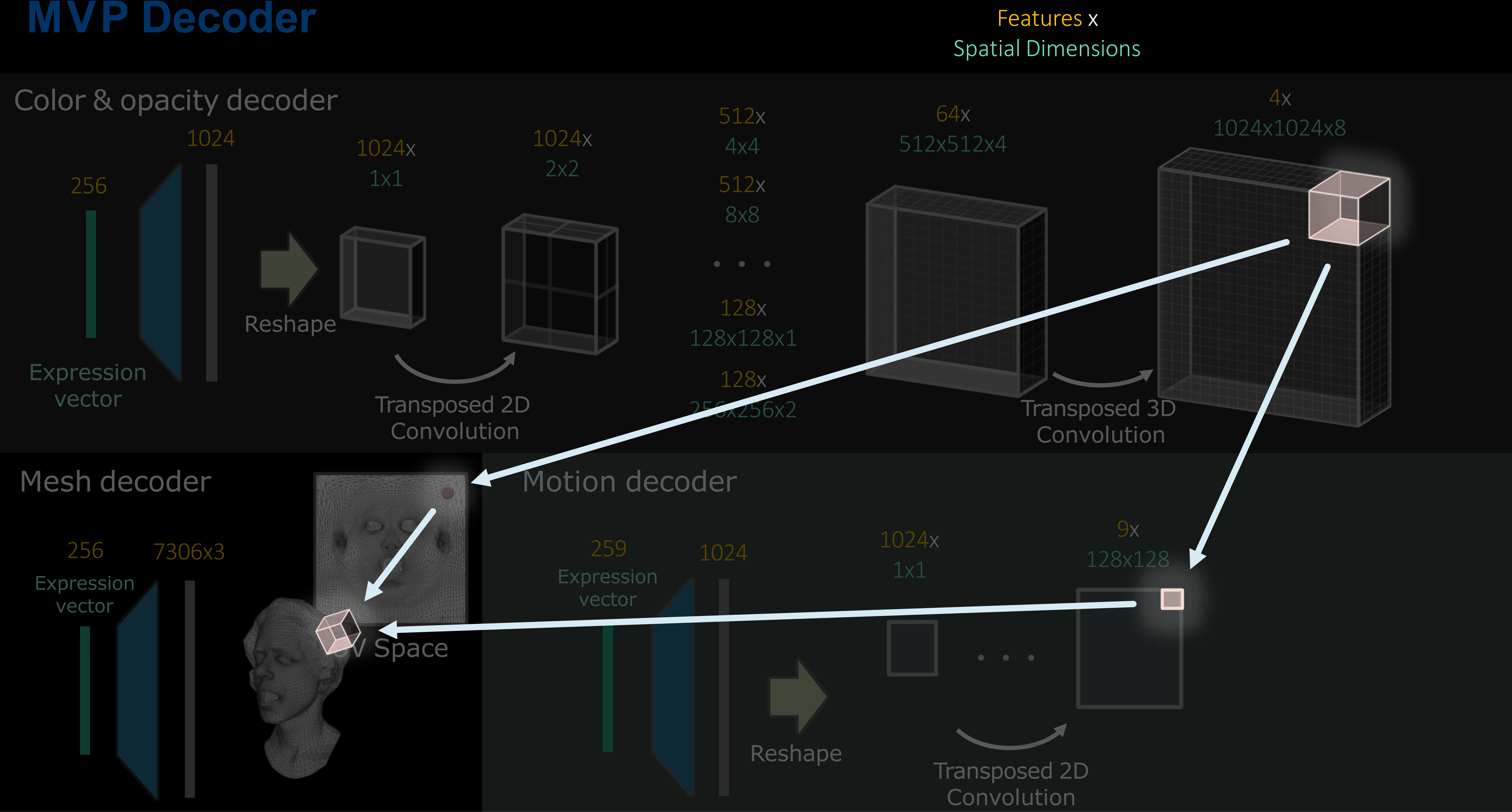


MVP Decoder

Features x
Spatial Dimensions



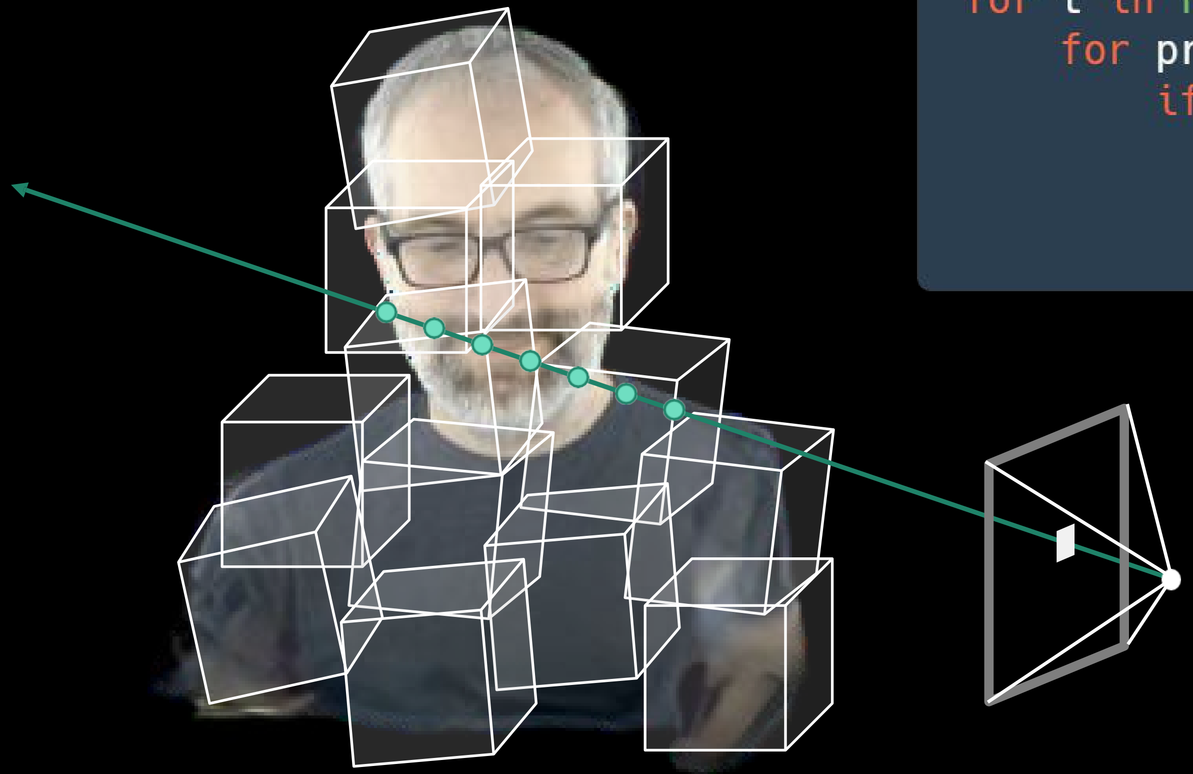
MVP Decoder



Features x
Spatial Dimensions
Color & opacity decoder
256
Expression vector
1024
Reshape
1024x
1x1
1024x
2x2
Transposed 2D Convolution
512x
4x4
512x
8x8
. . .
128x
128x128x1
128x
256x256x2
64x
512x512x4
Transposed 3D Convolution
4x
1024x1024x8
Mesh decoder
256
Expression vector
7306x3
UV Space
Motion decoder
259
Expression vector
1024
Reshape
1024x
1x1
. . .
9x
128x128
Transposed 2D Convolution

Raymarching MVP

```
for t in range(t_min, t_max, step_size):
    for primitive in primitive_list:
        if raystart + t * raydir in primitive:
            rgb, alpha = sample_rgba(primitive, t)
            aggregate_radiance(rgb, alpha)
```

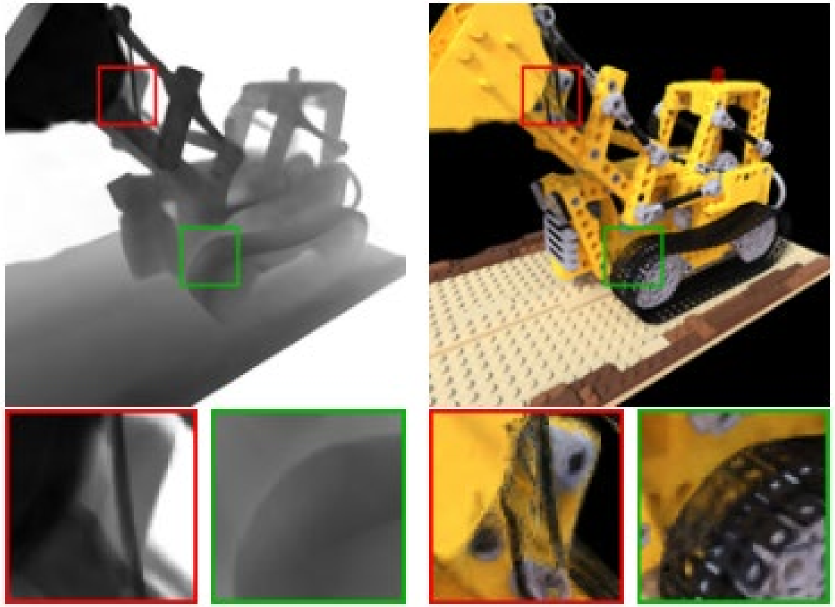


Results

Depth-guided Sampling

- Predicting depths for more efficient sampling:

  DONeRF: Towards Real-Time Rendering of Neural Radiance Fields using Depth Oracle Networks, Neff et al., Arxiv 2021 ~15 FPS



(a) Depth Prediction (b) 3 Local Samples

Image from [Neff et al. 2021]

Learning Integral by a Neural Network

- A general framework to integrate signals with implicit neural representation, which can be used in volume rendering.

AutoInt: Automatic Integration for Fast Neural Volume Rendering, Lindell et al., CVPR 2021. ~0.4 FPS

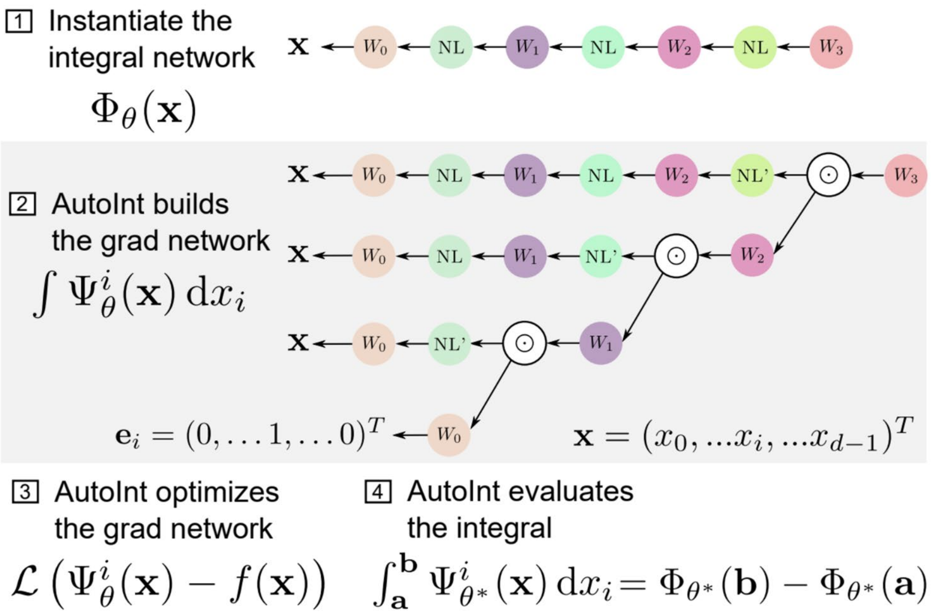


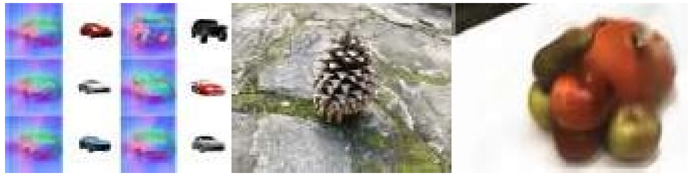
Image from [Lindell et al. 2021]

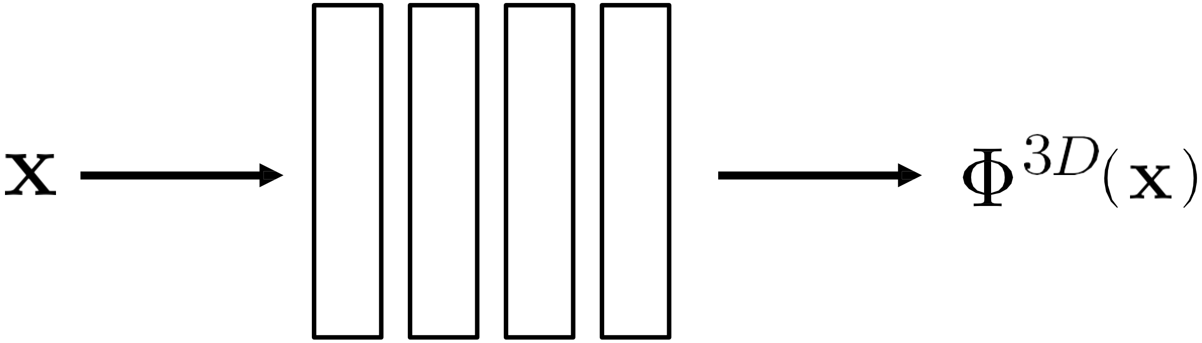
Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering

Vincent Sitzmann* Semon Rezchikov*

William T. Freeman Joshua B. Tenenbaum Frédo Durand

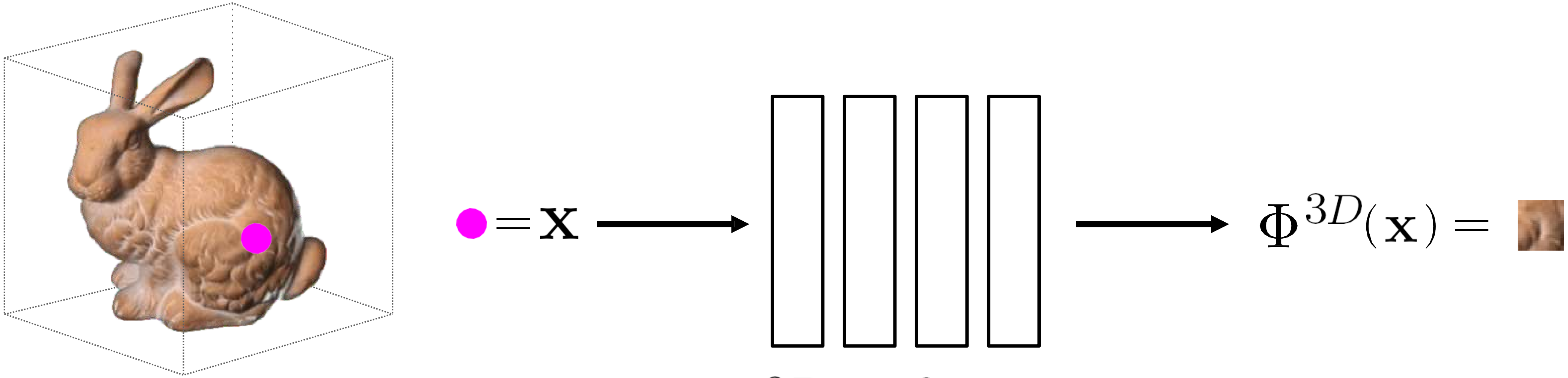
3D-structured Neural Scene Representations





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

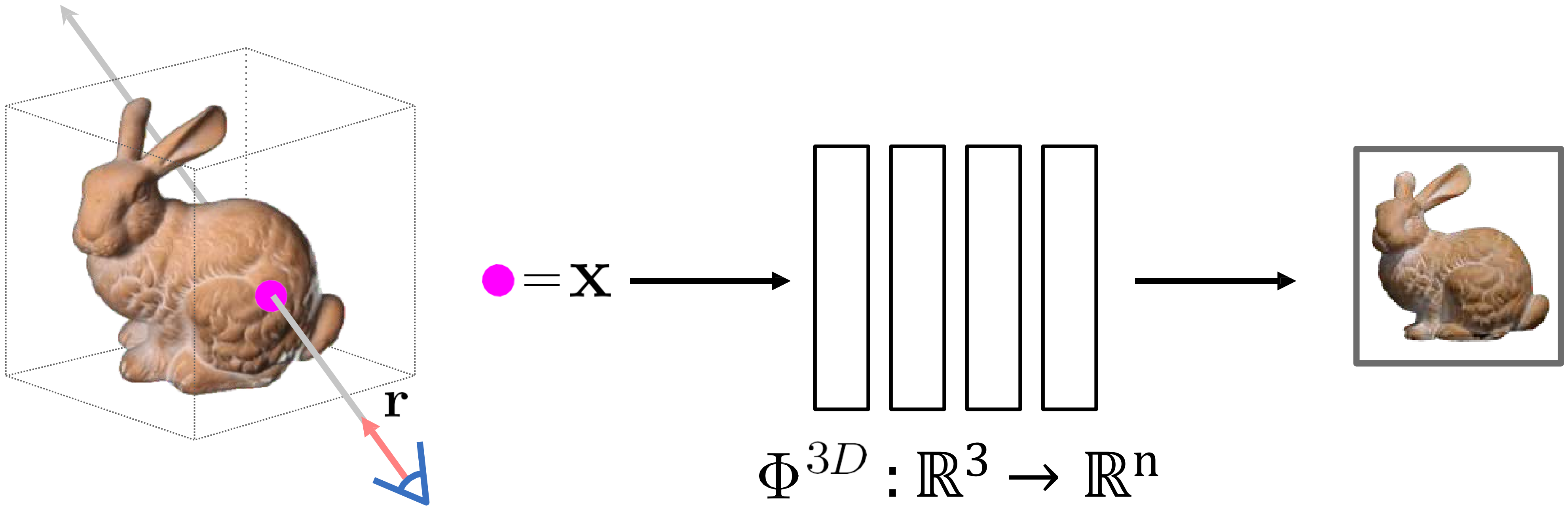
3D-structured Neural Scene Representations



$$= \mathbf{x} \longrightarrow \qquad \longrightarrow \Phi^{3D}(\mathbf{x}) =$$

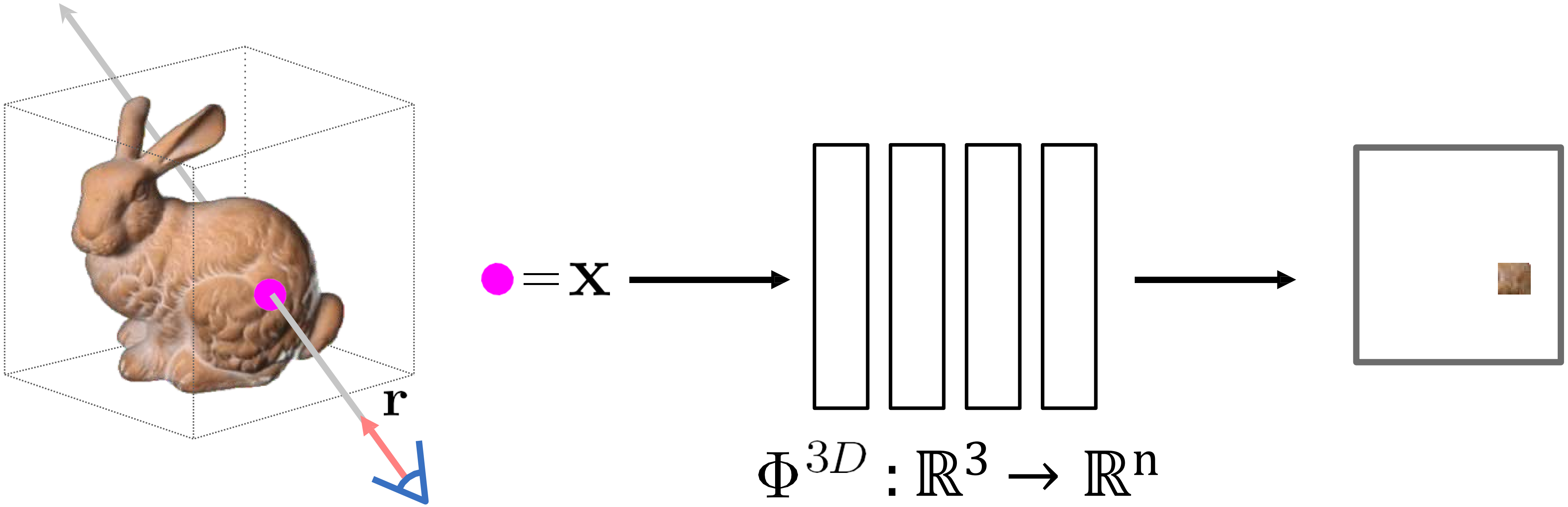
$$\Phi^{3D} : \mathbb{R}^3 \rightarrow \mathbb{R}^{\mathrm{n}}$$

3D-structured Neural Scene Representations

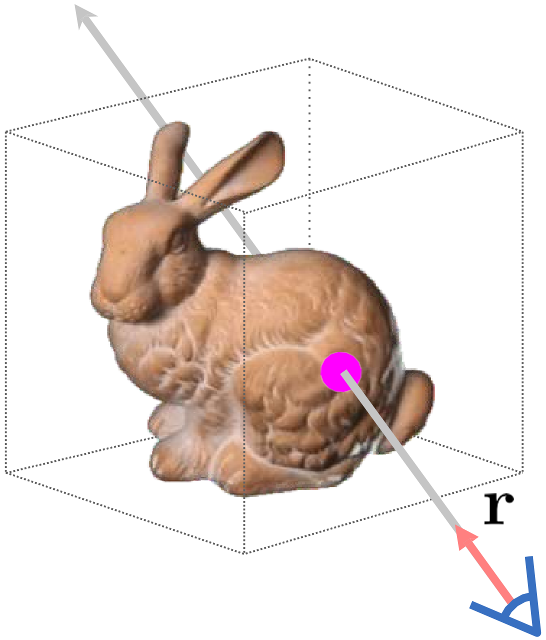


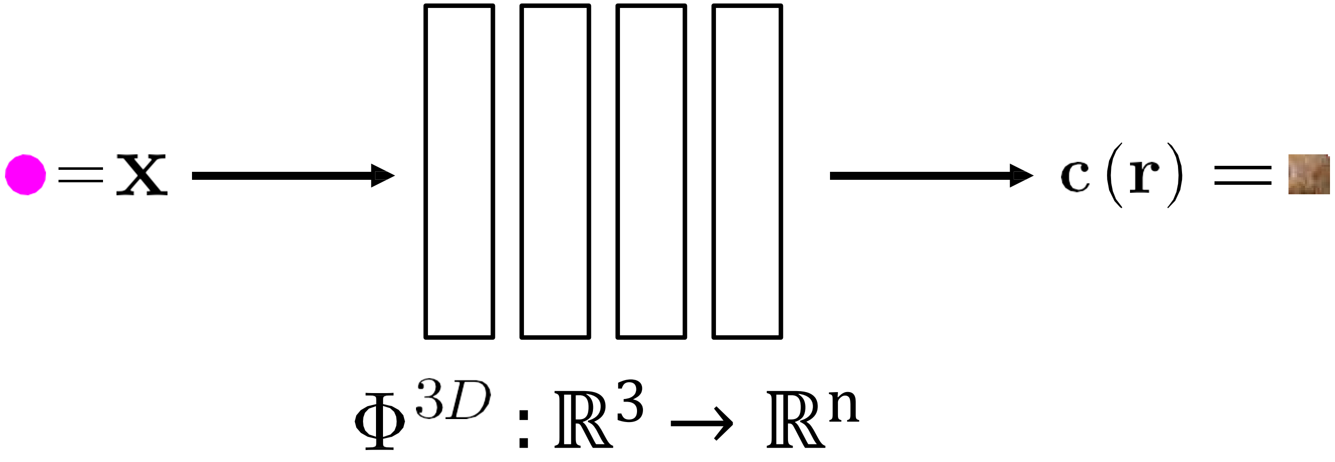
$$\Phi^{3D} : \mathbb{R}^3 \rightarrow \mathbb{R}^n$$

3D-structured Neural Scene Representations



3D-structured Neural Scene Representations

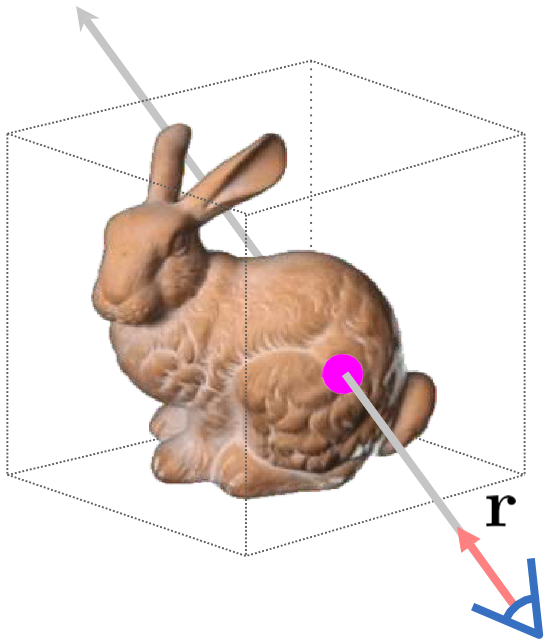


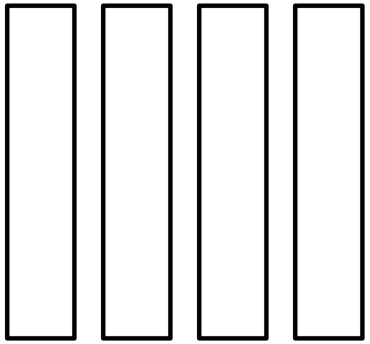


Fast Rendering of Neural Radiance Fields, Lingjie Liu

Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

3D-structured Neural Scene Representations

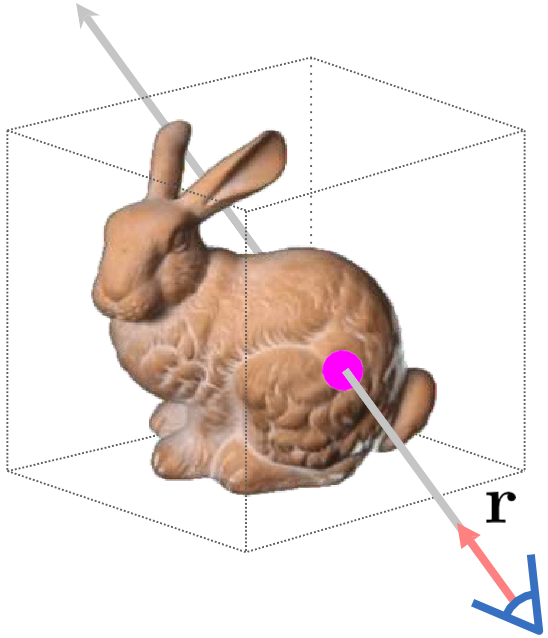


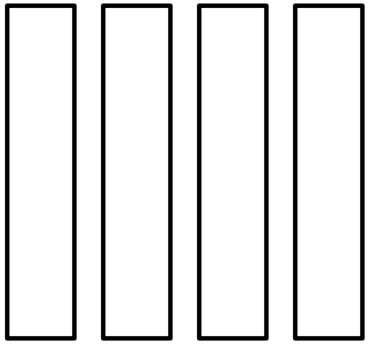


$$\Phi^{3D} : \mathbb{R}^3 \rightarrow \mathbb{R}^n$$

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

3D-structured Neural Scene Representations

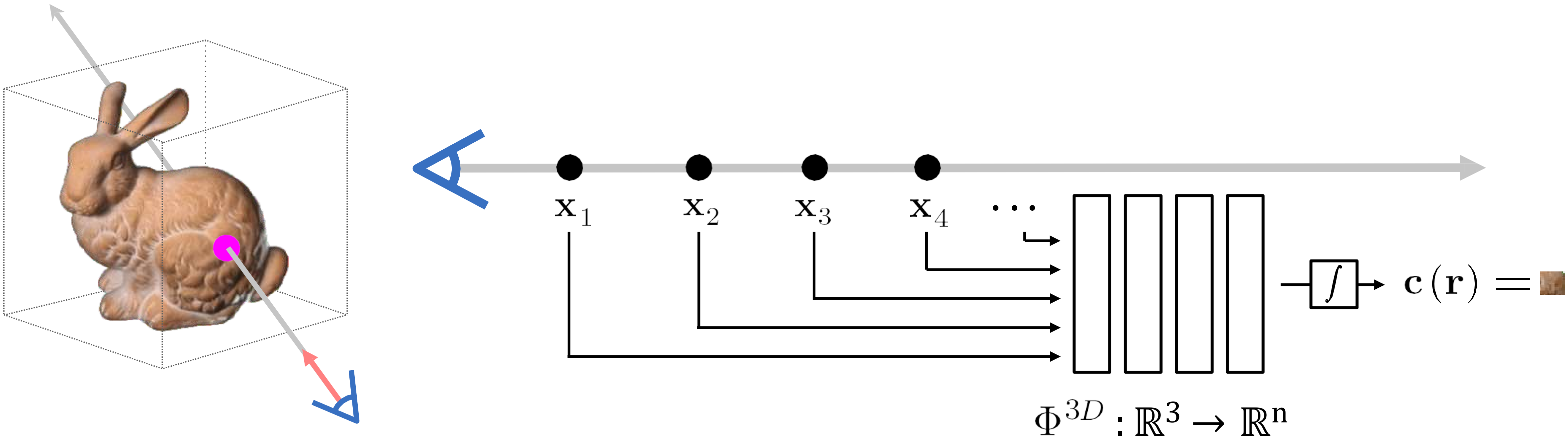




$$\Phi^{3D} : \mathbb{R}^3 \rightarrow \mathbb{R}^{\mathrm{n}}$$

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

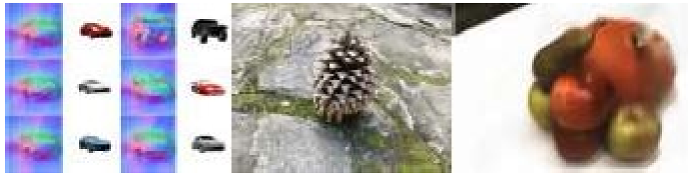
3D-structured Neural Scene Representations



Hundreds of samples per ray.

256x256 image takes >30 seconds (volumetric).

Time- and memory-intensive training.

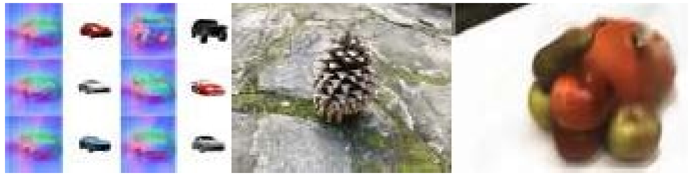


Light Field

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

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

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



Light Field Networks

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

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

Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

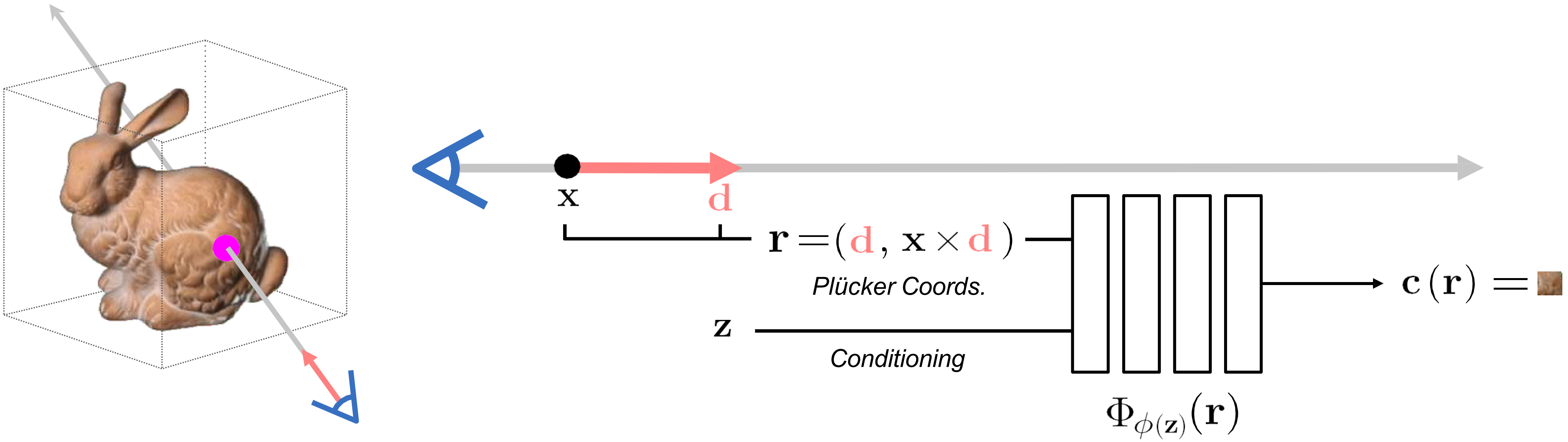
Light Field Networks

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

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

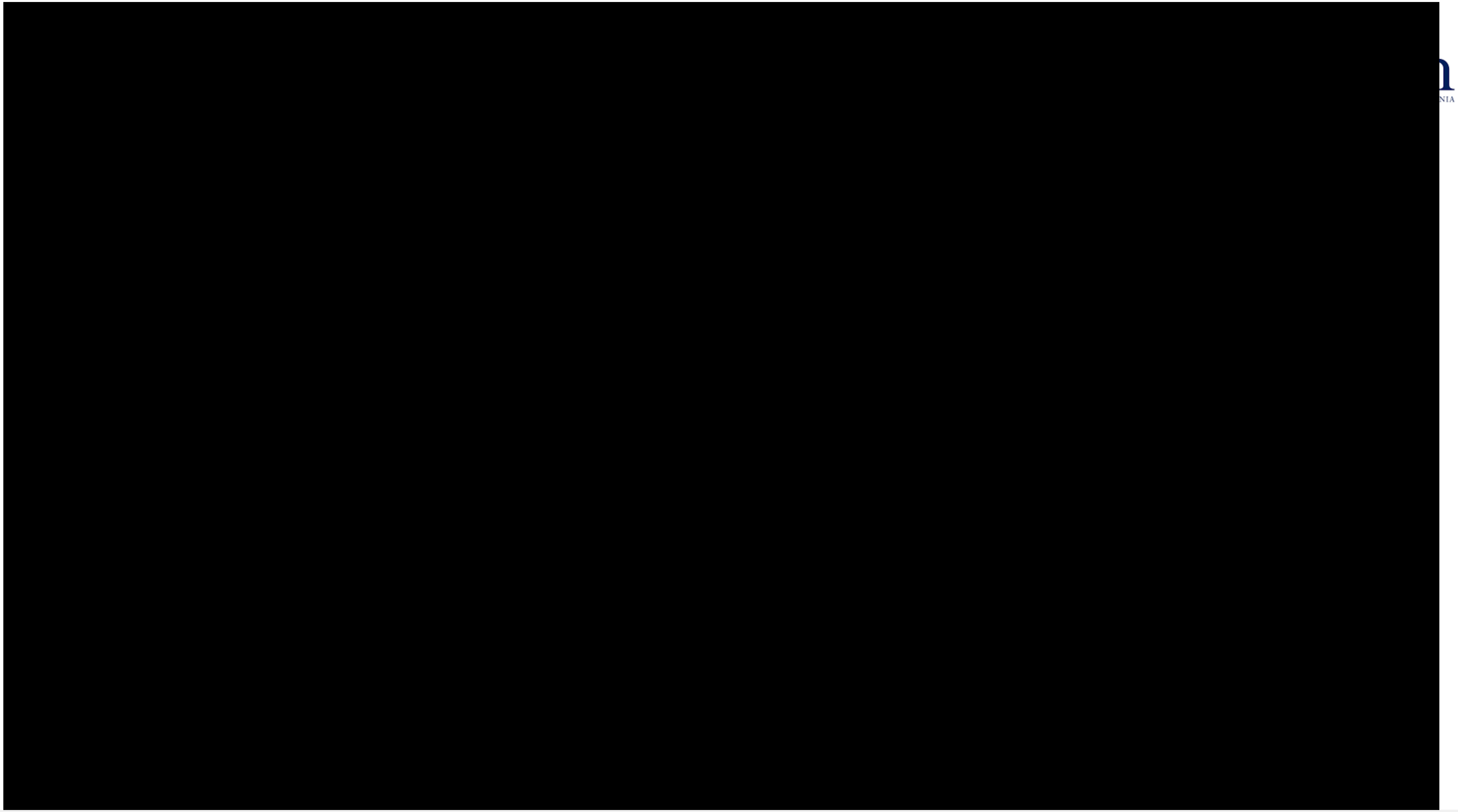
Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

Light Field Networks



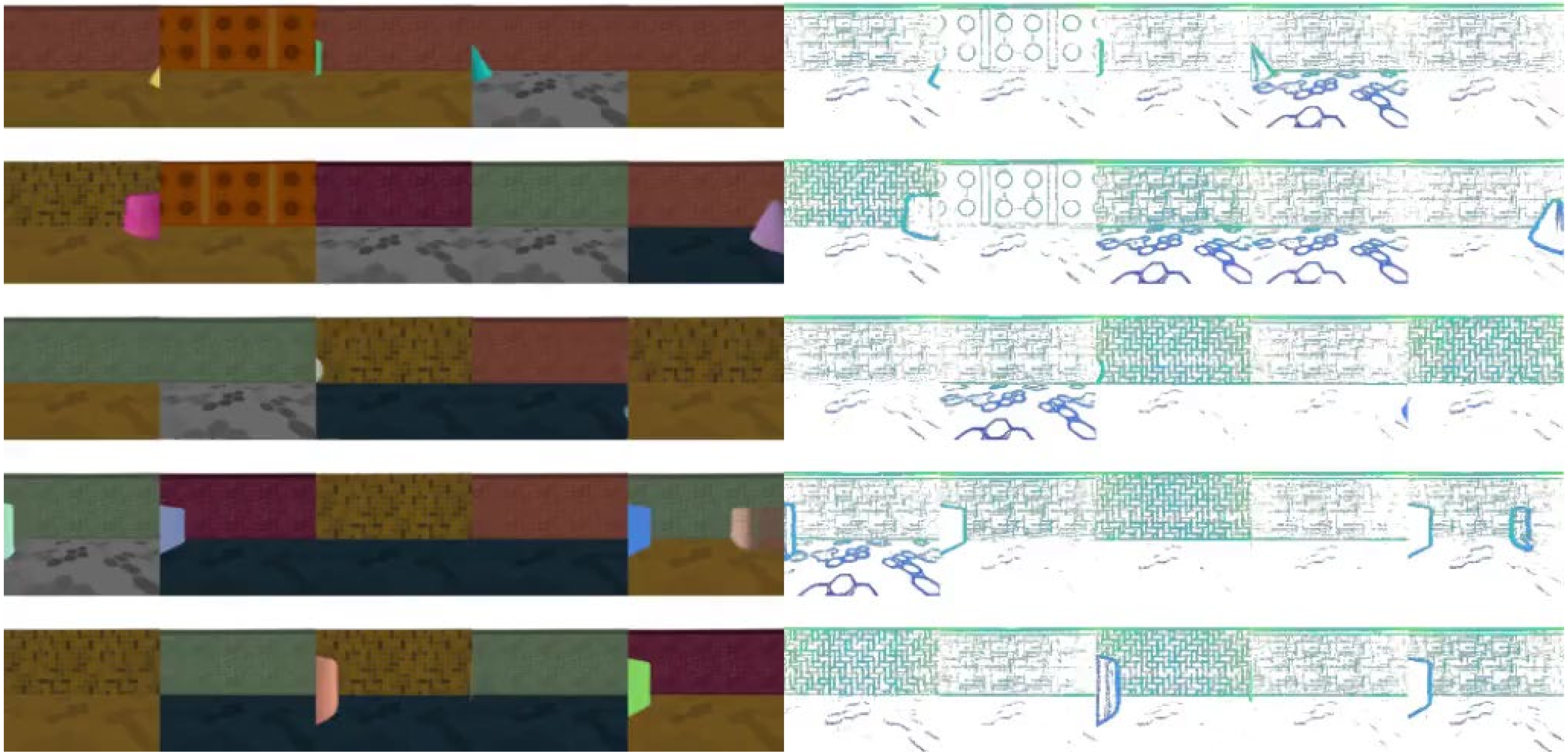
An Alternative Scene Representation

Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021



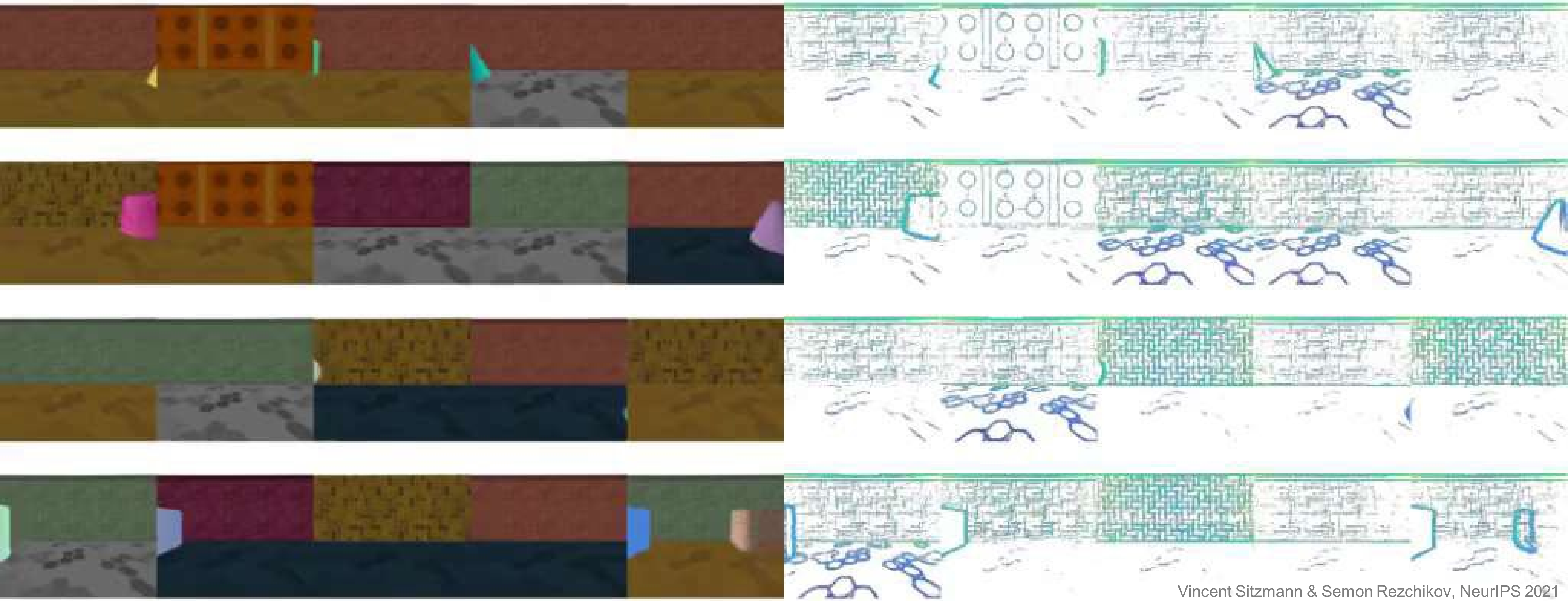
Real-time. No post-processing, no discrete data structures (octrees, voxelgrids, …).
>100x reduction in memory: Can be trained on small GPUs!

Light Field Networks
500 FPS
1 evaluation per ray



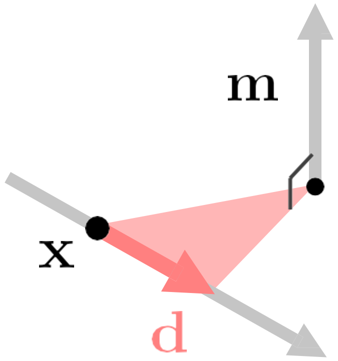
Fast Rendering of Neural Radiance Fields, Lingjie Liu

Also Encode Depth in their 4D derivatives: can be extracted via single evaluation of neural network and its gradient!

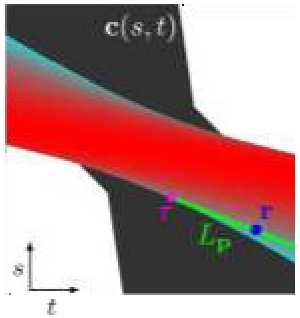


Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

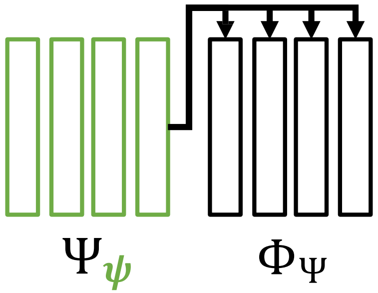
Parameterization



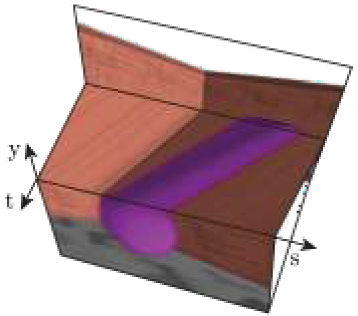
LFN Geometry



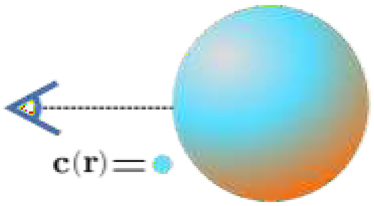
Meta-Learning



Results



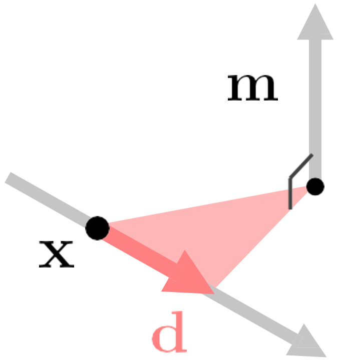
Limitations



Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

Ray Parameterizations for LFNs

Parameterization



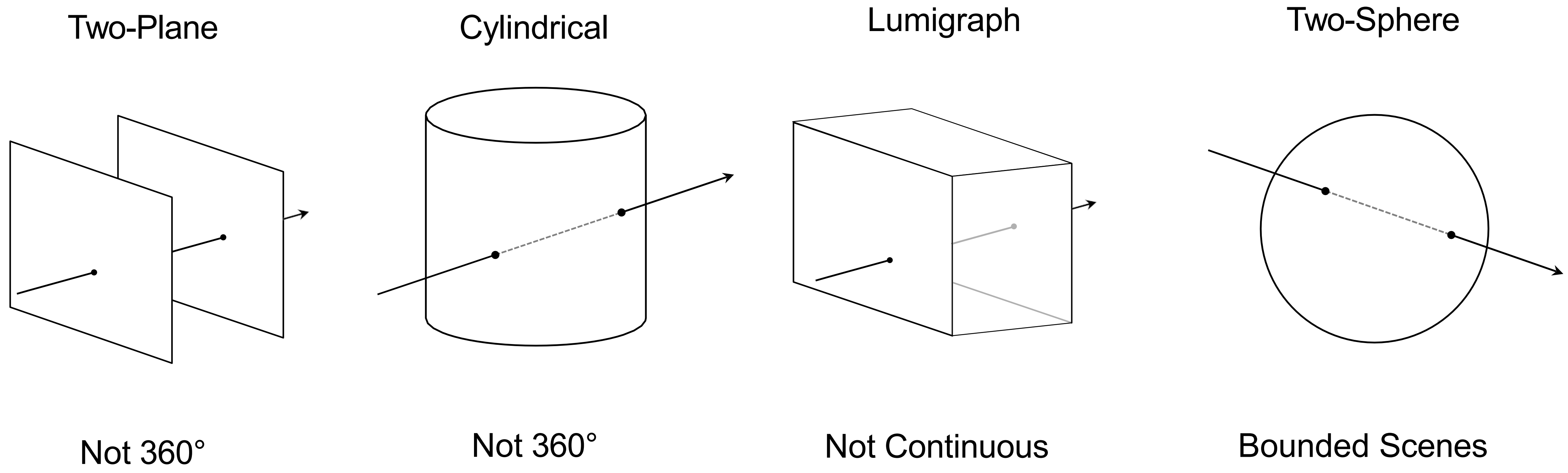
LFN Geometry

Meta-Learning

Results

Limitations

Conventional Light Field Parameterizations

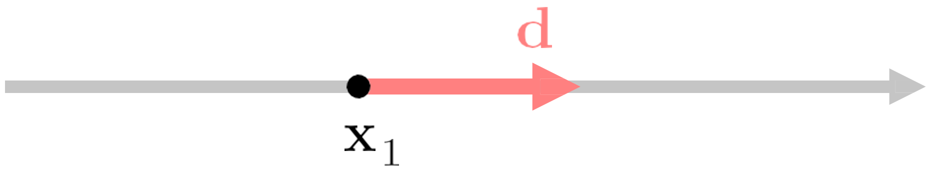


Difficult to use as a complete scene representation

Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

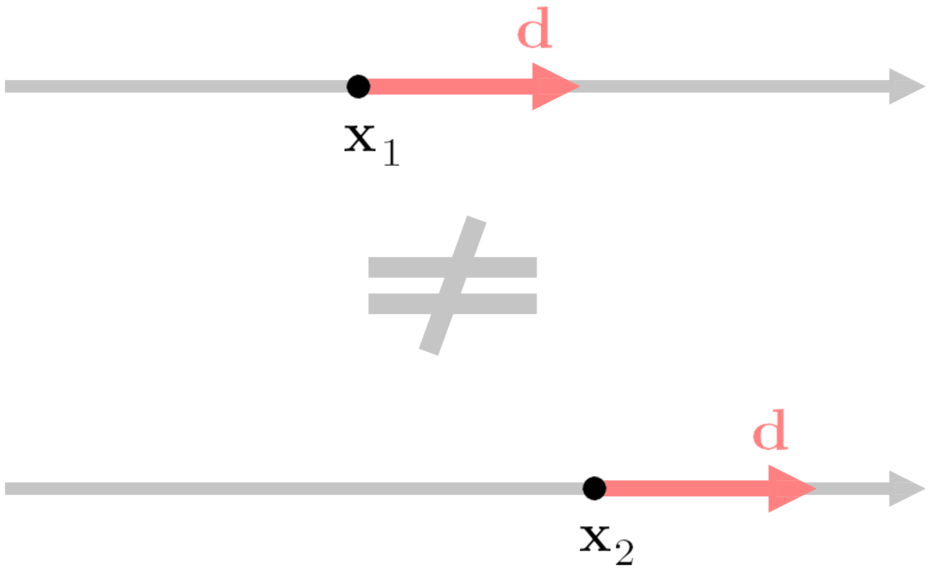
“Point-direction” coordinates

$$\mathbf{r} = (\mathbf{x}_1, \mathbf{d})$$



"Point-direction" coordinates

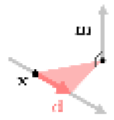
$$\mathbf{r} = (\mathbf{x}_1, \mathbf{d})$$

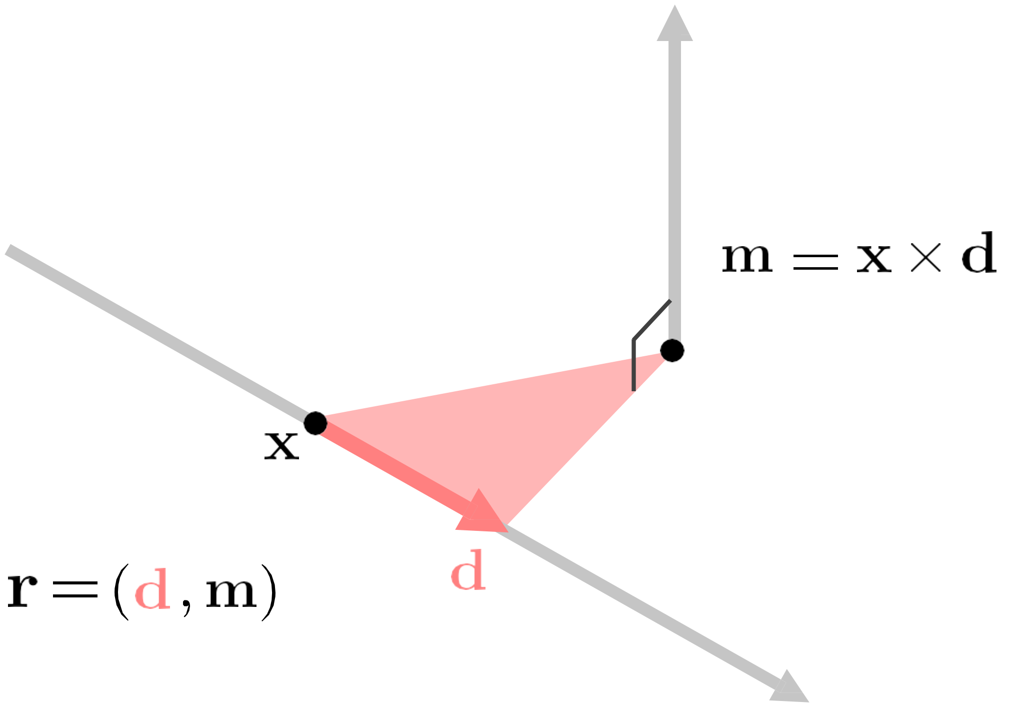


$$\mathbf{r} = (\mathbf{x}_2, \mathbf{d})$$

Not unique: Same ray, two different coordinates.

Plücker coordinates



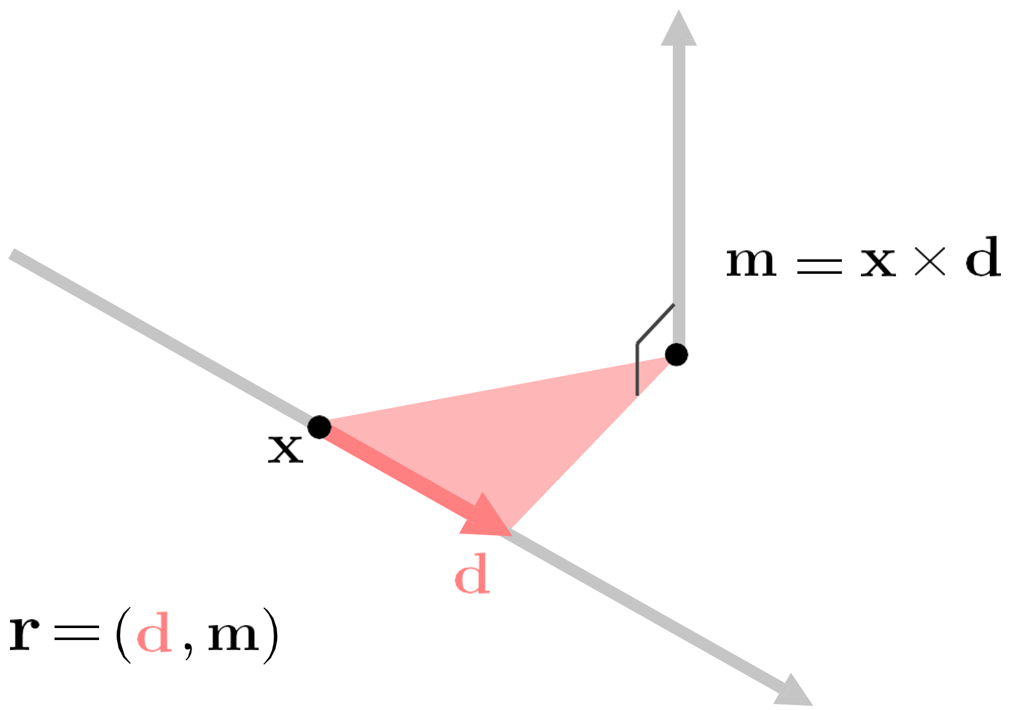


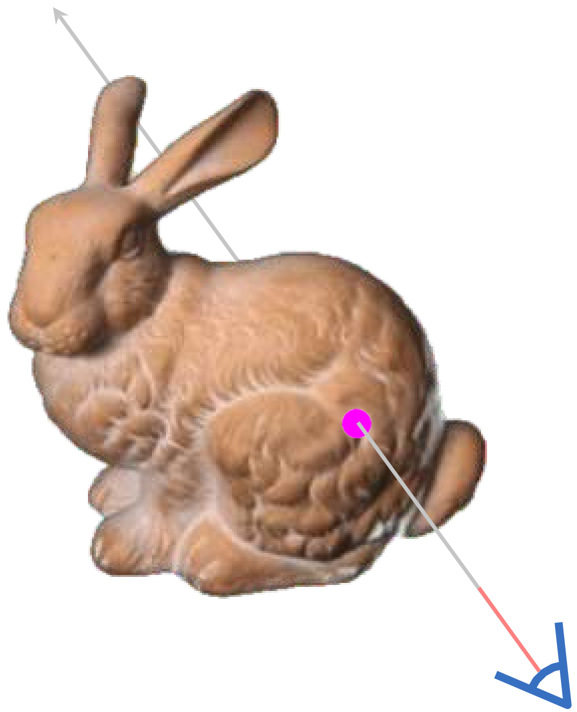
Unique: invariant to choice of $\mathbf{x}$.

Parameterize all rays without special cases.

Impractical for discrete representations, since $\mathbf{r} \in \mathbb{R}^6$.

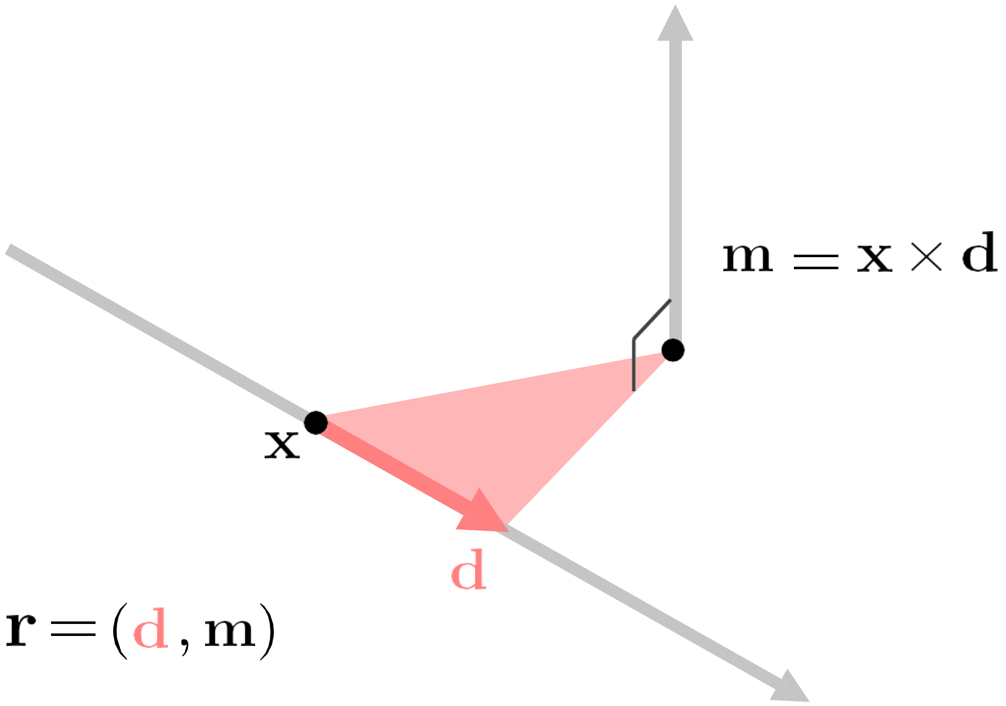
Plücker coordinates



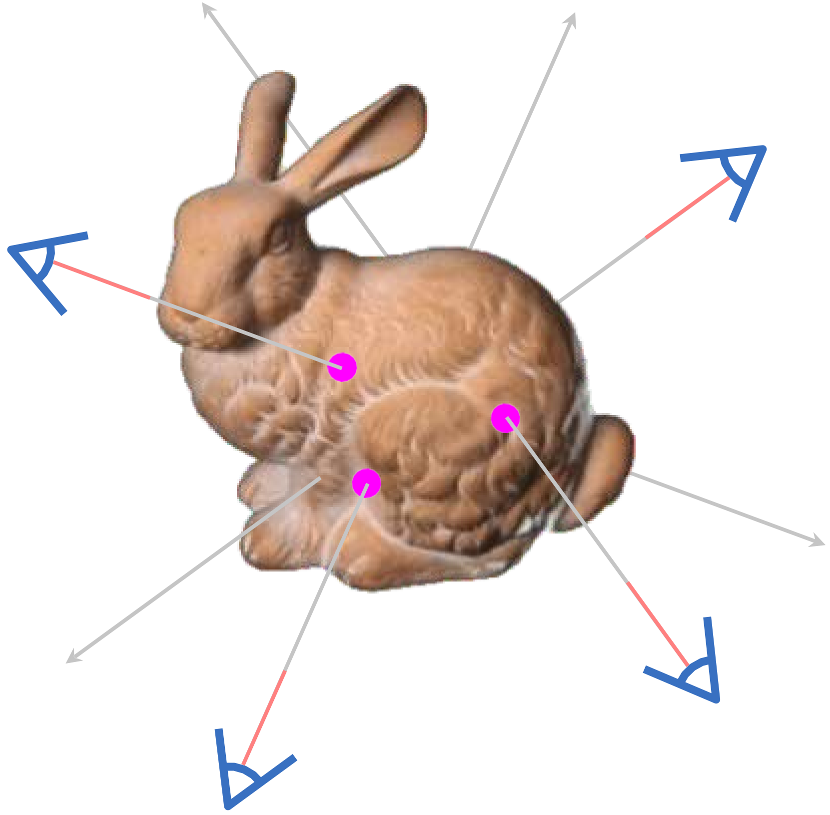


Fast Rendering of Neural Radiance Fields, Lingjie Liu

Plücker coordinates



$\mathbf{r} = (\mathbf{d}, \mathbf{m})$



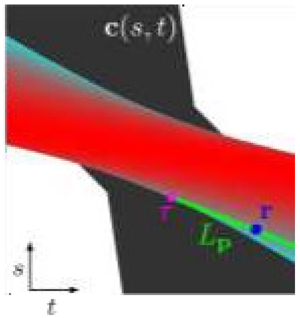
Parameterize 360 degree light fields of unbounded scenes.

Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

Extracting Scene Geometry from LFNs

Parameterization

LFN Geometry

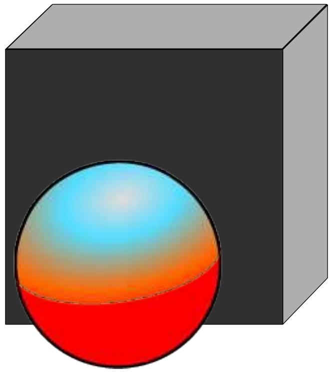


Meta-Learning

Results

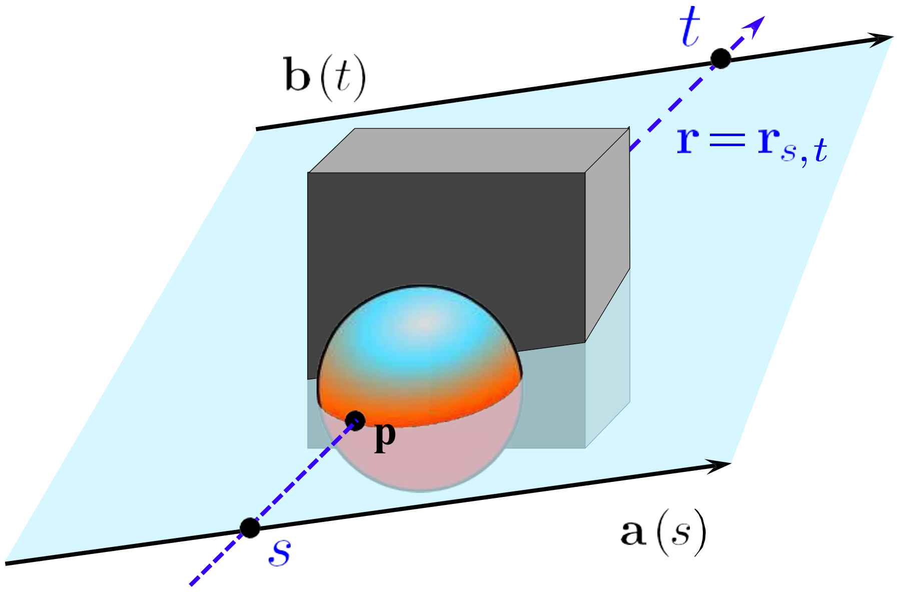
Limitations

The geometry of LFNs



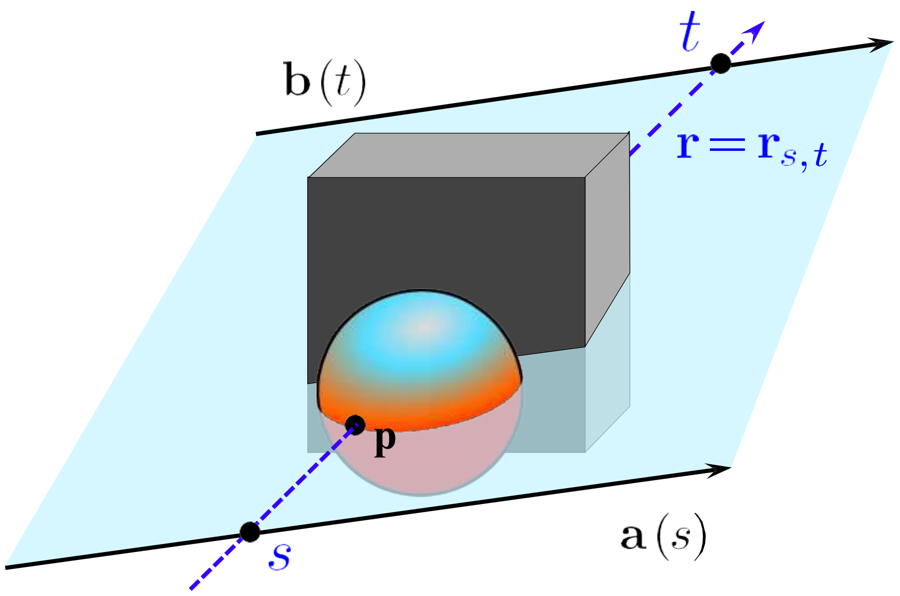
Fast Rendering of Neural Radiance Fields, Lingjie Liu

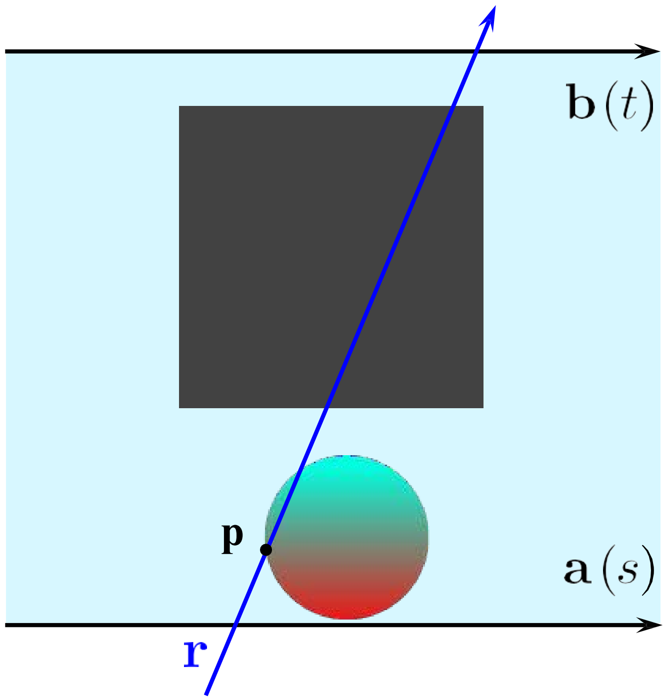
The geometry of LFNs



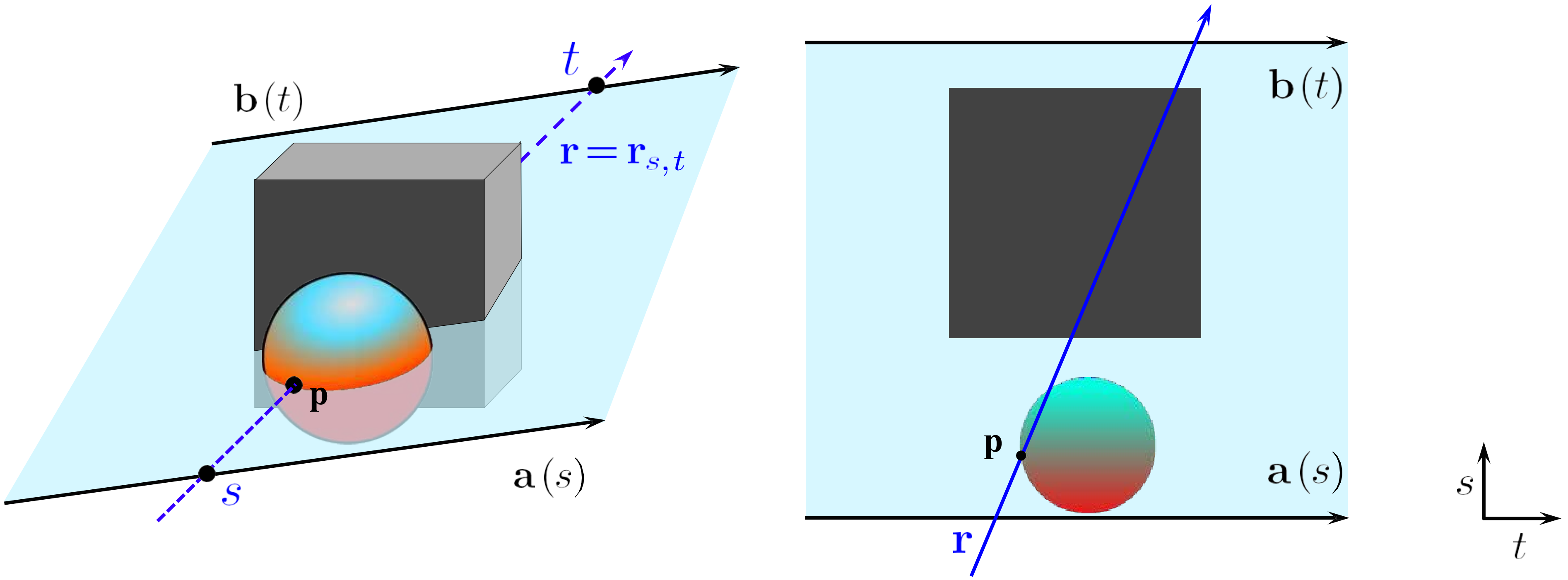
Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

The geometry of LFNs

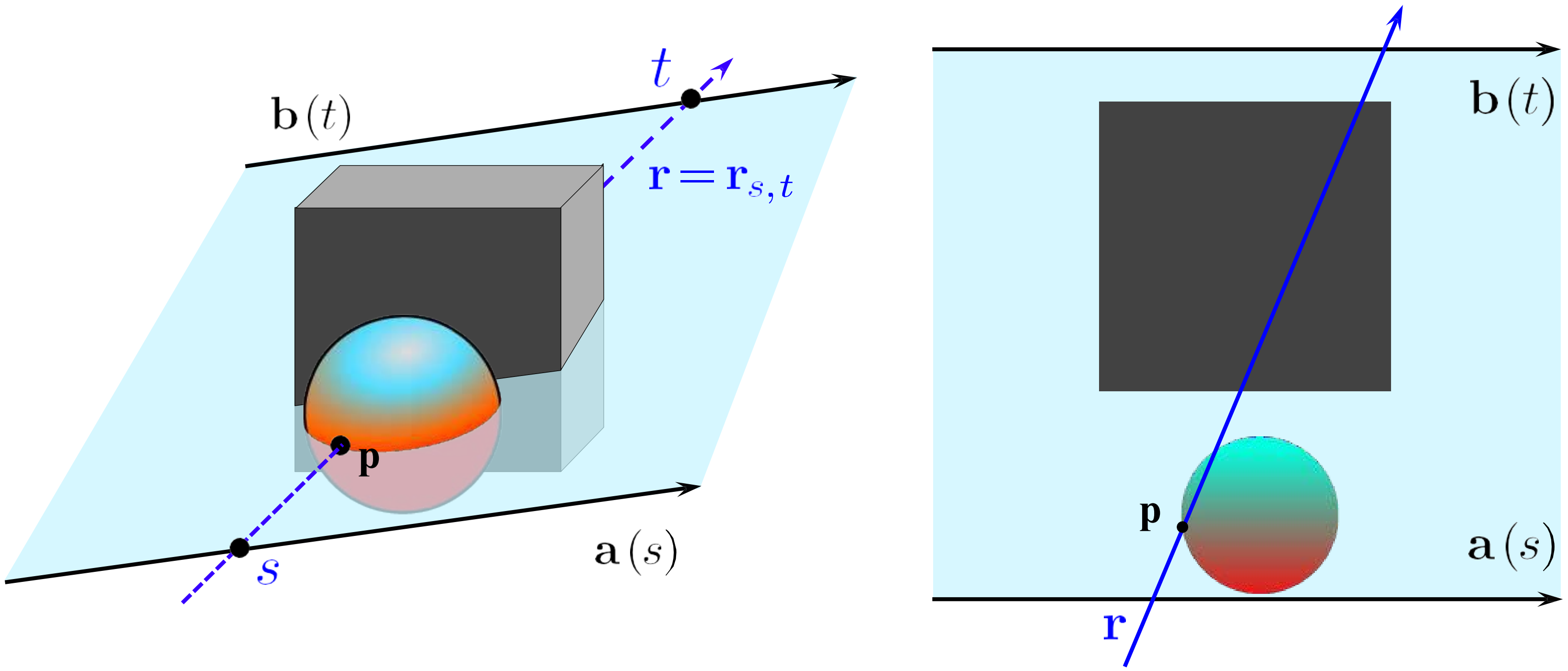




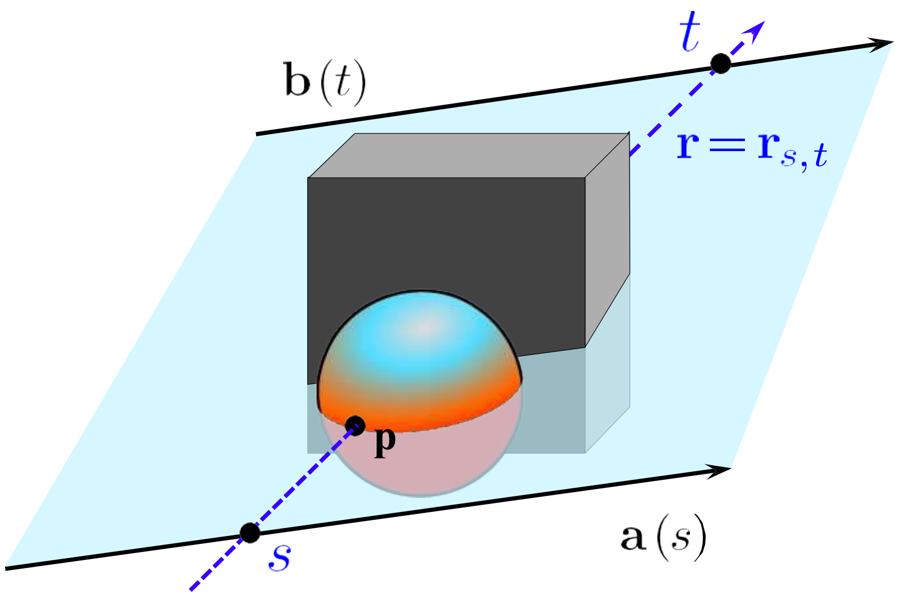
The geometry of LFNs

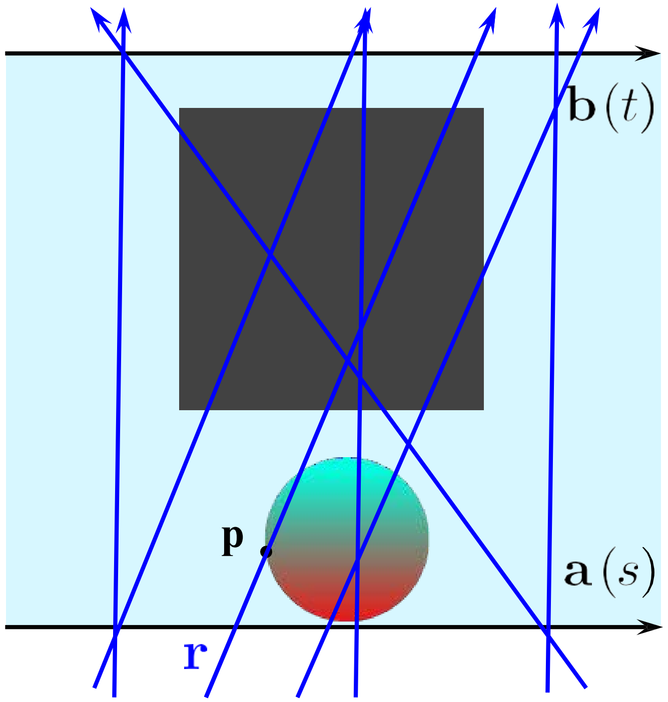


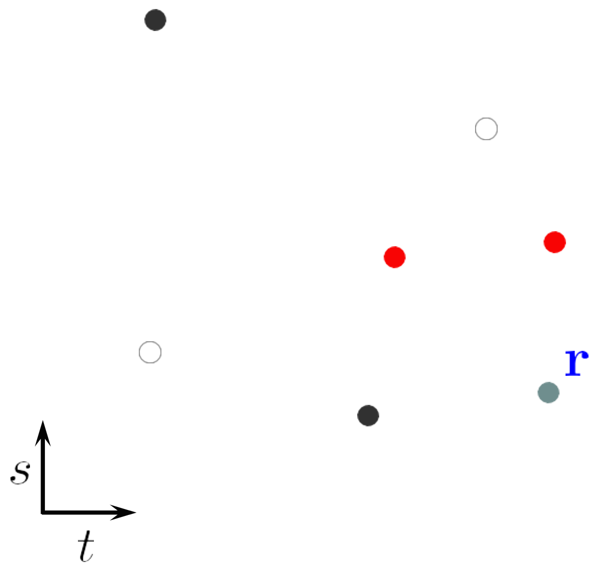
The geometry of LFNs



The geometry of LFNs



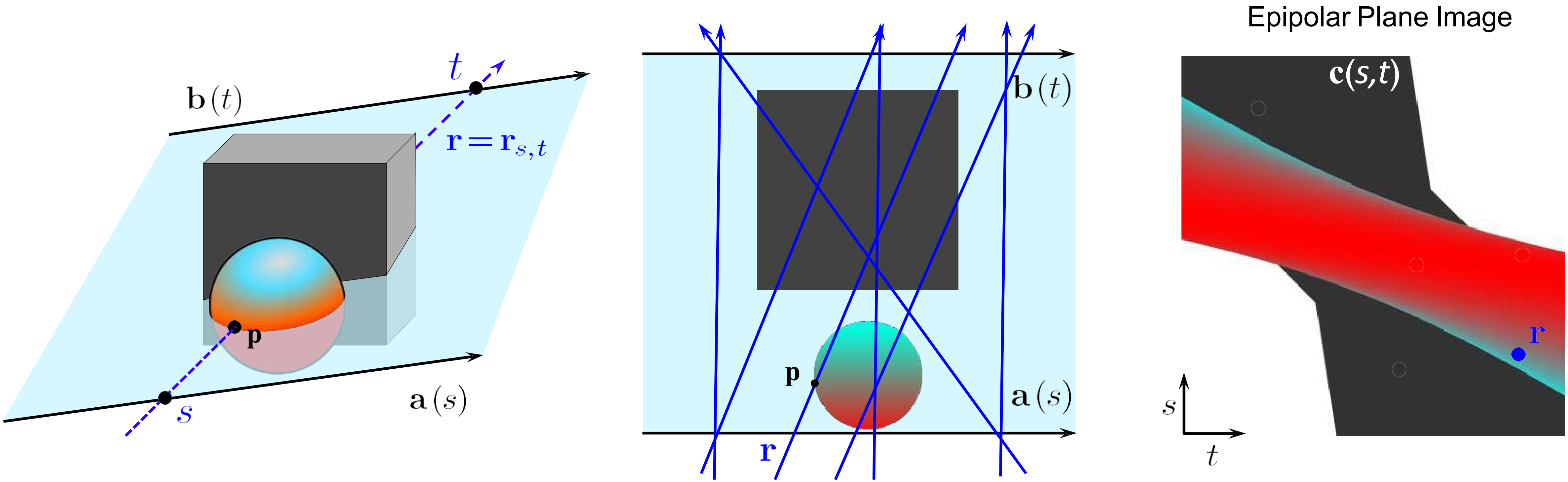




Fast Rendering of Neural Radiance Fields, Lingjie Liu

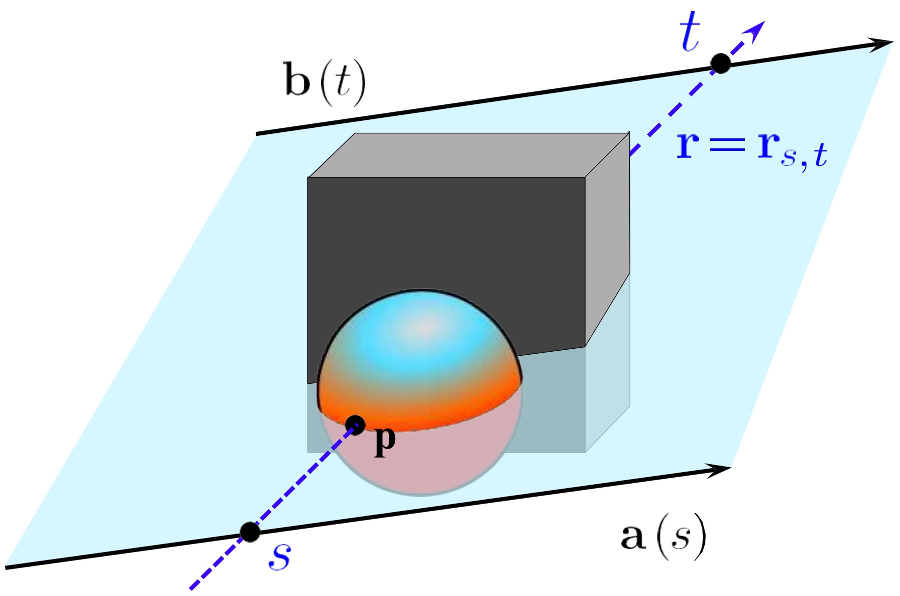
Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

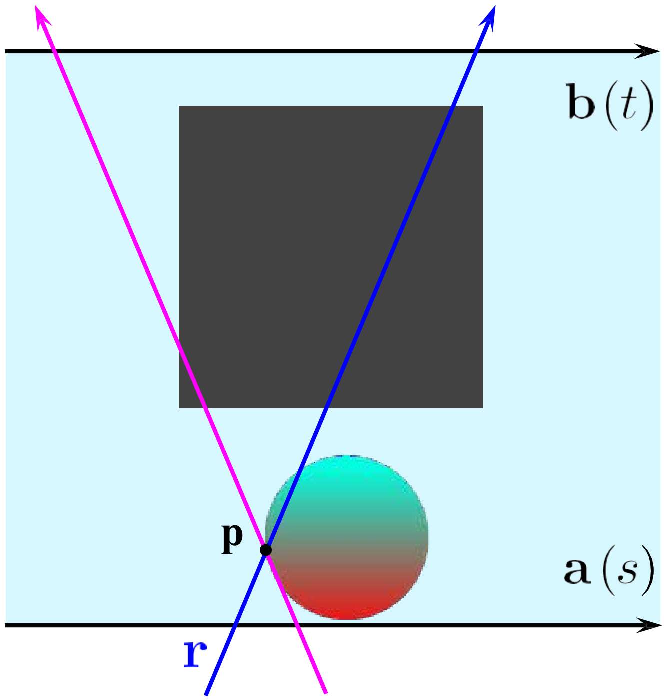
The geometry of LFNs



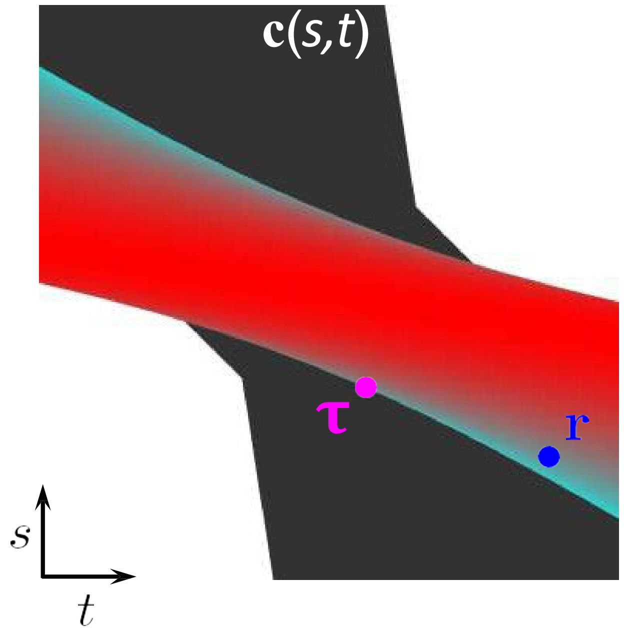
Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

The geometry of LFNs



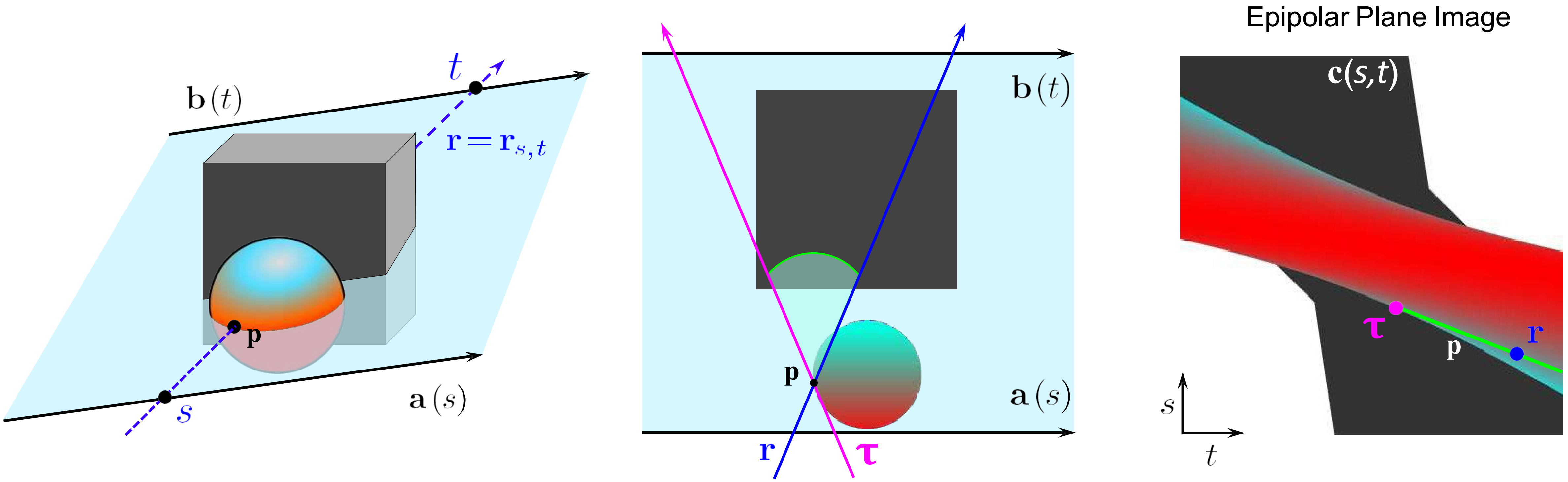


Epipolar Plane Image



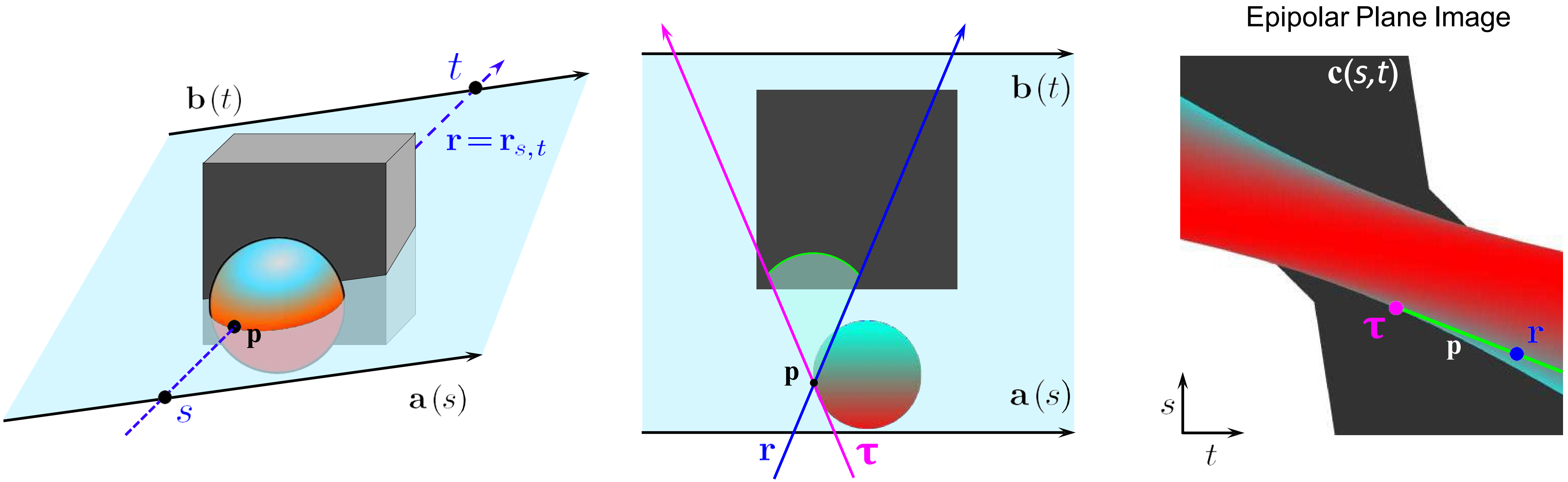
Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

The geometry of LFNs



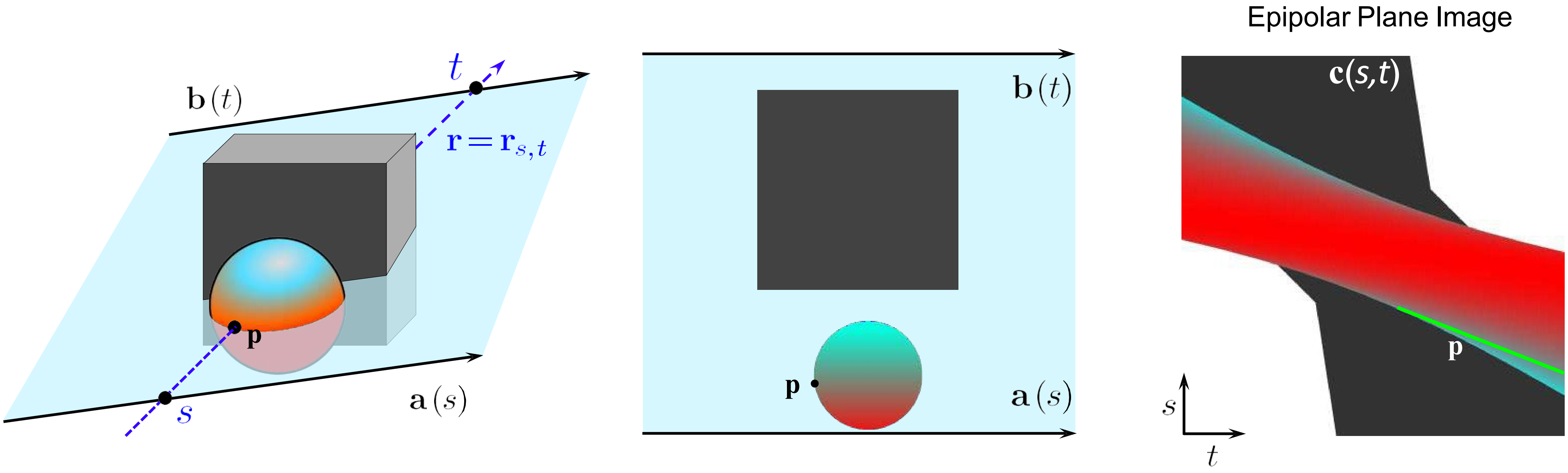
Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

The geometry of LFNs



Points give lines of constant color in EPI **c**(*s*,*t*) – line is a **levelset** of the EPI.

The geometry of LFNs



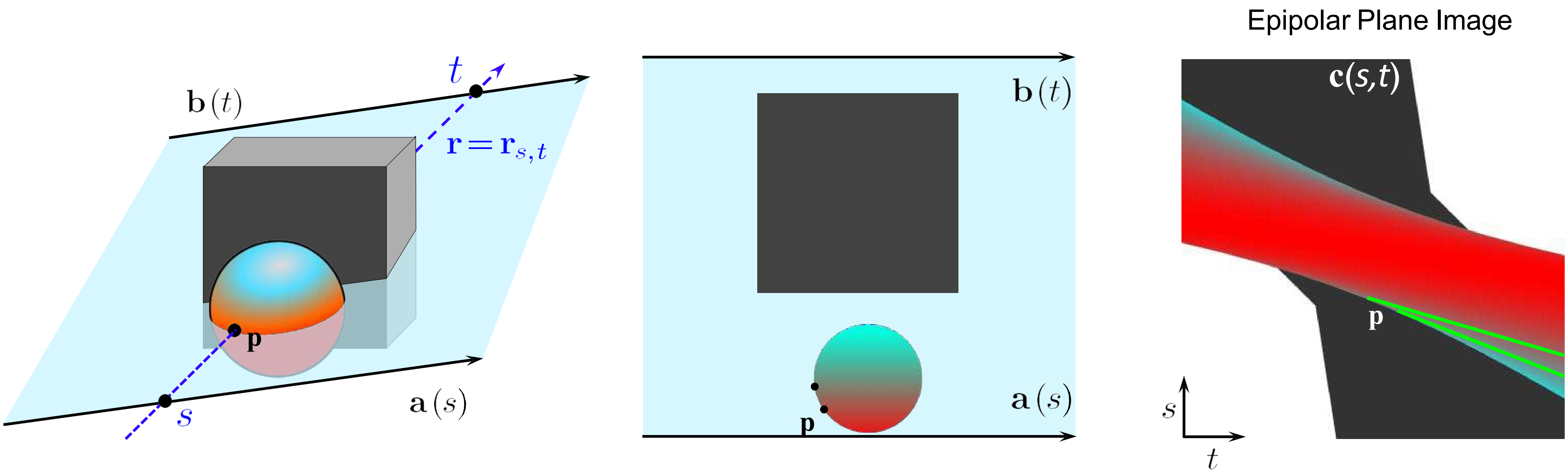
Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a **levelset** of the EPI.

Slope of line decreases as point moves closer.

Fast Rendering of Neural Radiance Fields, Lingjie Liu

Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

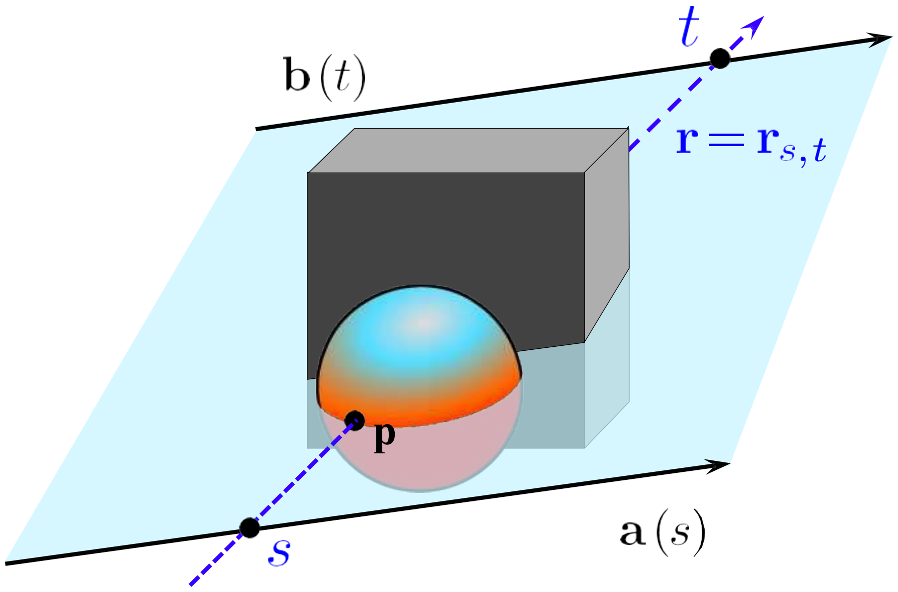
The geometry of LFNs

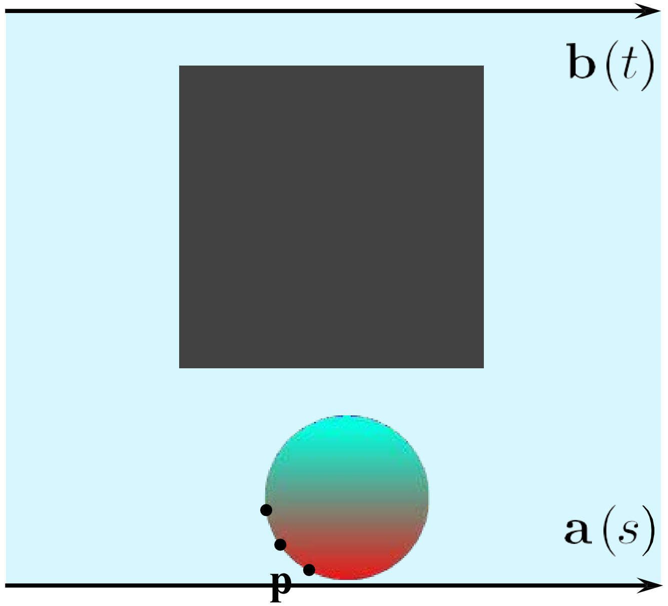


Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a **levelset** of the EPI.

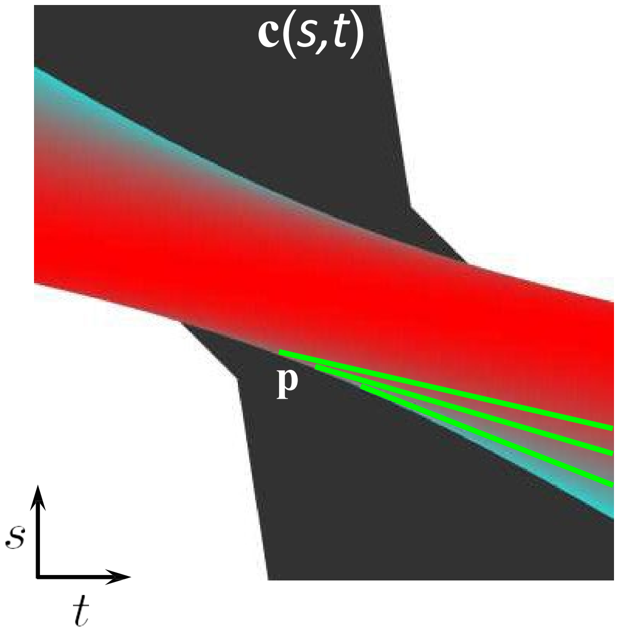
Slope of line decreases as point moves closer.

The geometry of LFNs





Epipolar Plane Image

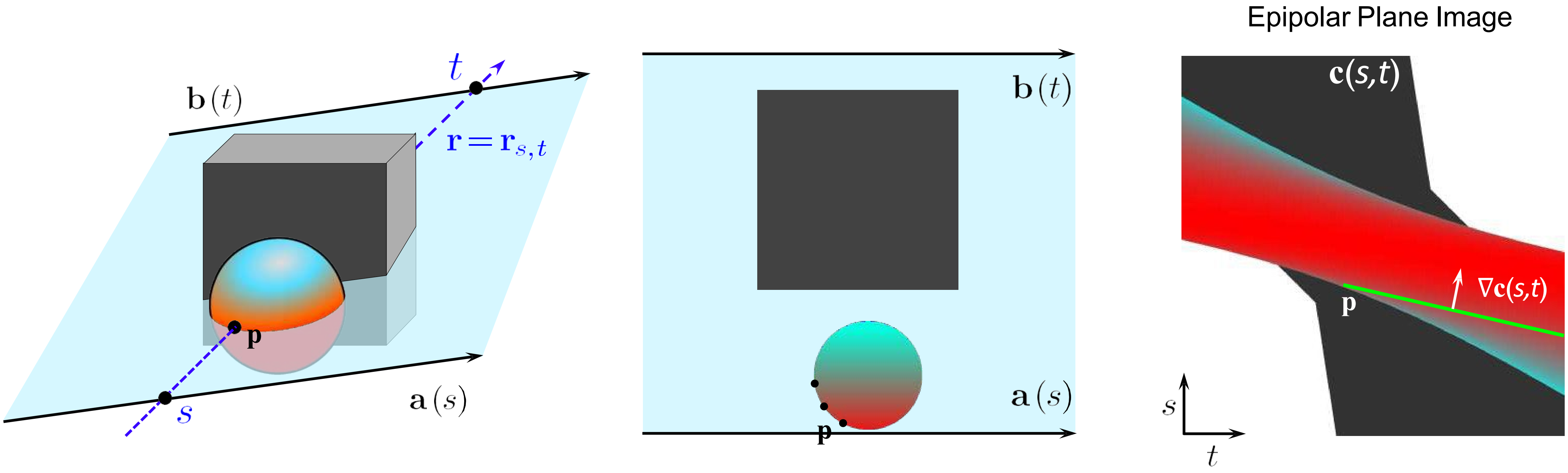


Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a **levelset** of the EPI.

Slope of line decreases as point moves closer.

Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

The geometry of LFNs



Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a **levelset** of the EPI.

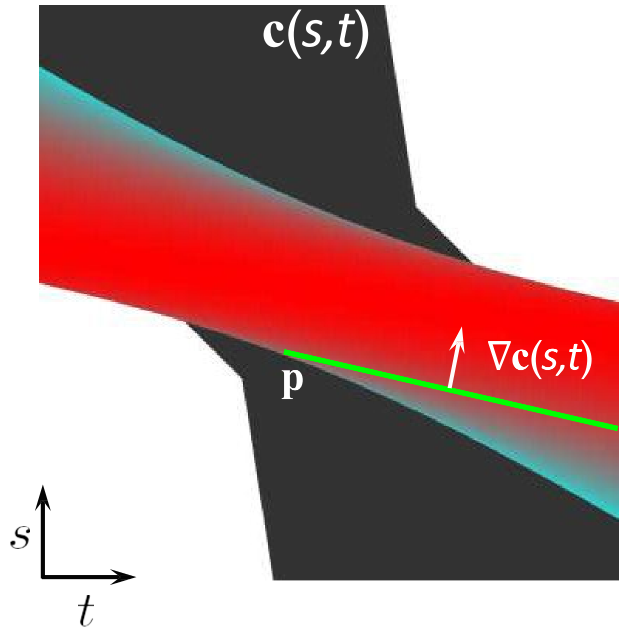
Slope of line decreases as point moves closer.

Gradient of $\mathbf{c}(s,t)$ is orthogonal to levelset -

The geometry of LFNs

$$d(s,t) = D\frac{\partial_t \mathbf{c}(s,t)}{\partial_s \mathbf{c}(s,t) + \partial_t \mathbf{c}(s,t)}$$

Epipolar Plane Image



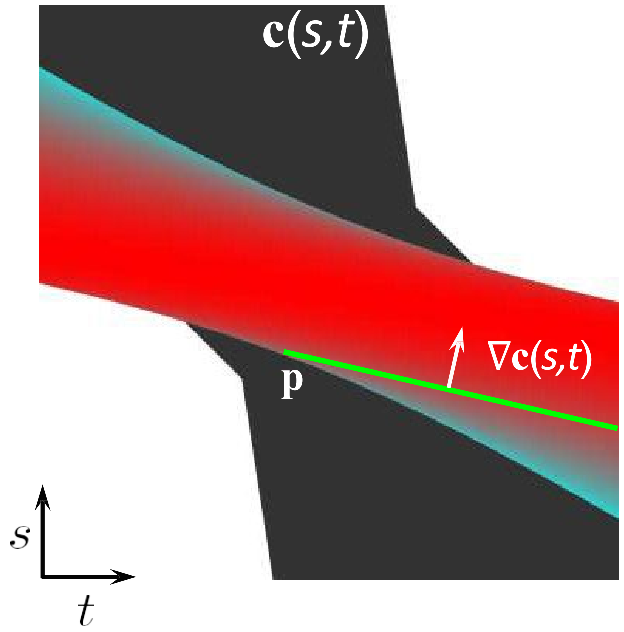
Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a **levelset** of the EPI.

Slope of line decreases as point moves closer.

Gradient of $\mathbf{c}(s,t)$ is orthogonal to levelset **-** can extract depth from **gradients of light field**.

The geometry of LFNs

Epipolar Plane Image

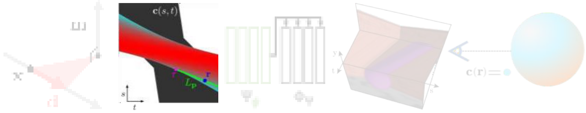


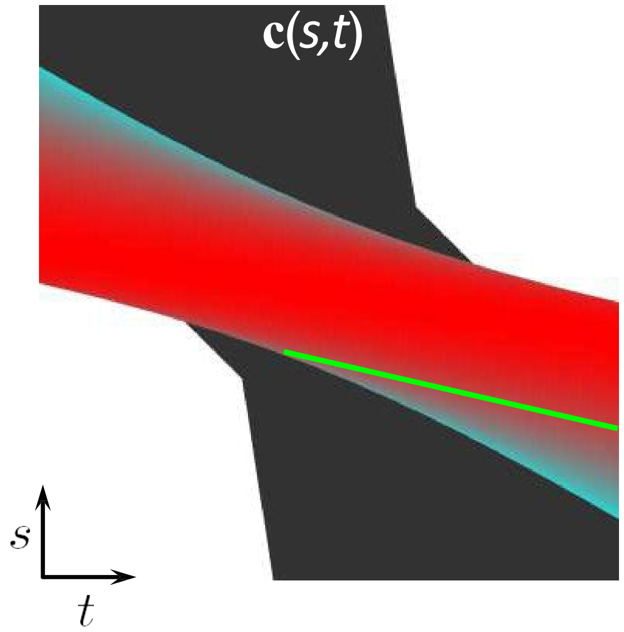
Points give lines of constant color in EPI $\mathbf{c}(s,t)$ – line is a **levelset** of the EPI.

Slope of line decreases as point moves closer.

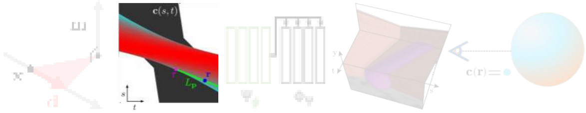
Gradient of $\mathbf{c}(s,t)$ is orthogonal to levelset **-** can extract depth from **gradients of light field**.

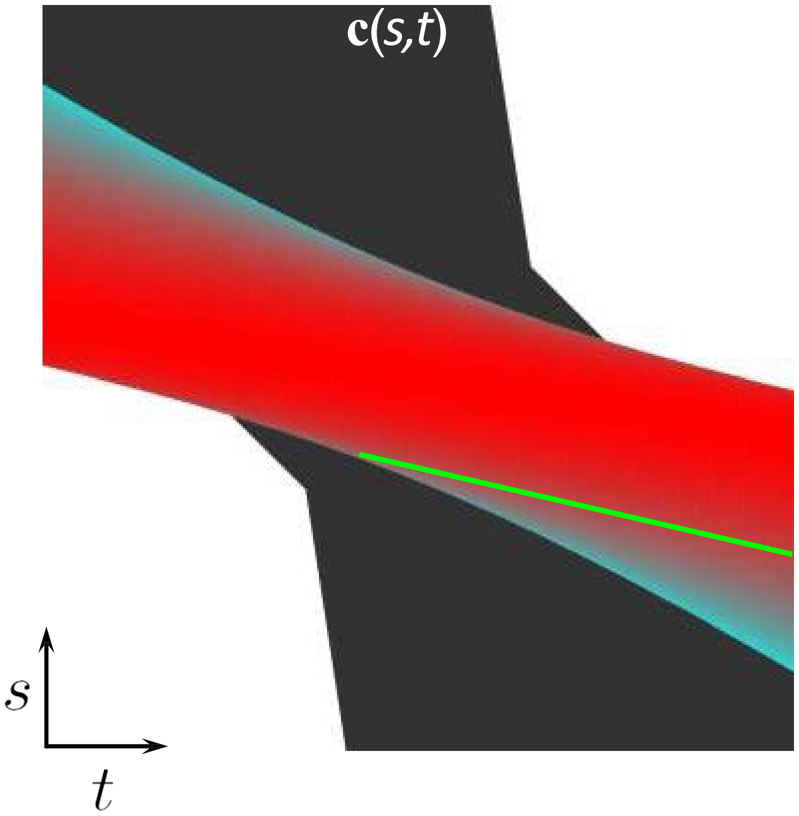
Multi-view consistency



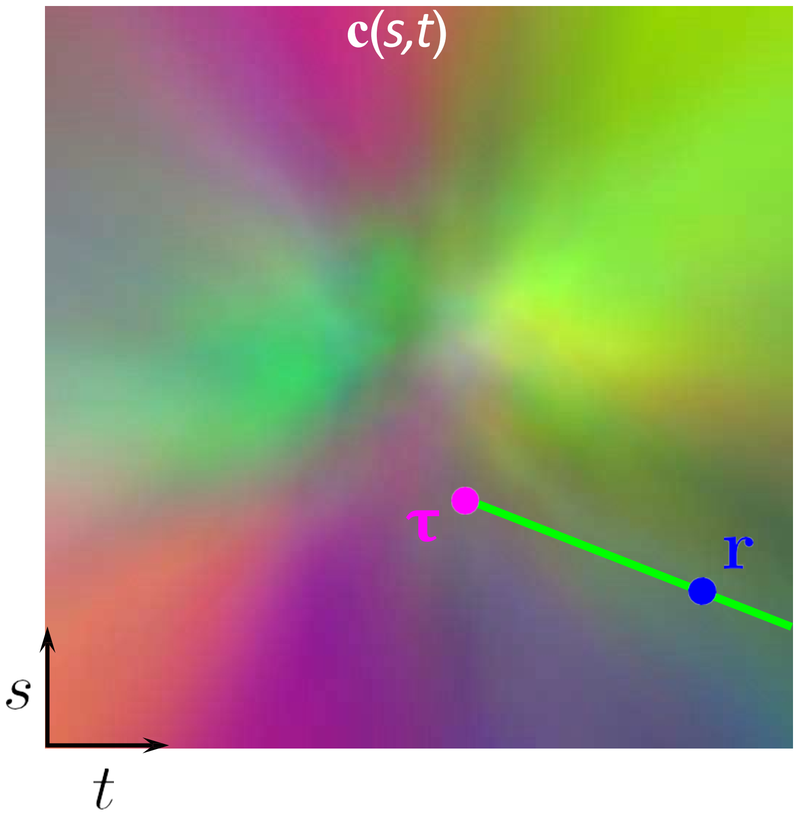


Multi-view consistency





Multi-view consistency



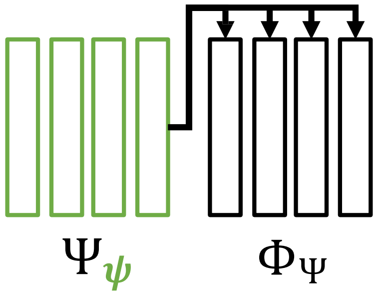
Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

Meta-Learning Multi-View Consistency

Parameterization

LFN Geometry

Meta-Learning

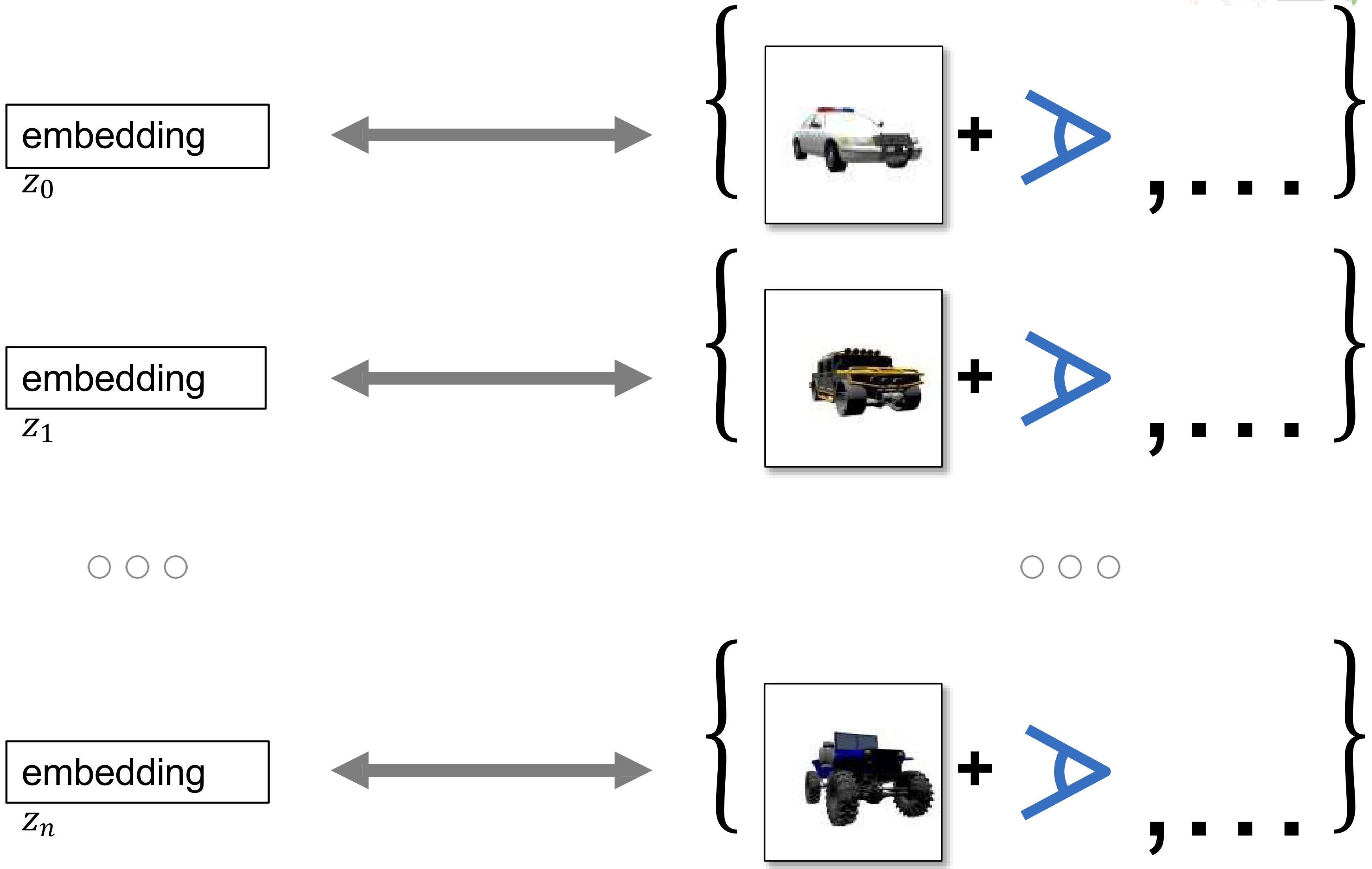


Results

Limitations

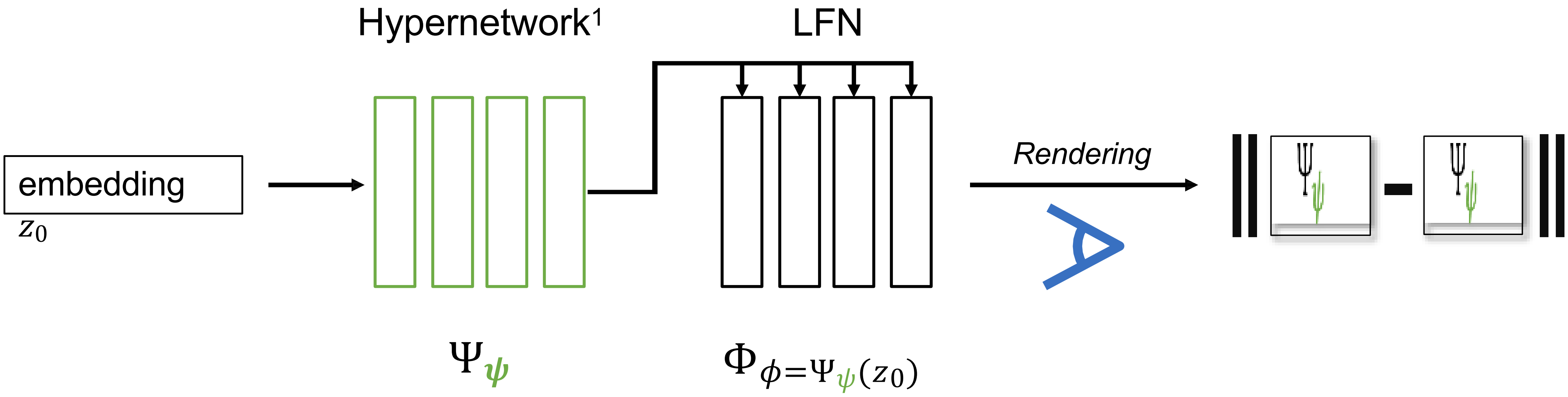
Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

Learning a space of multi-view consistent light fields



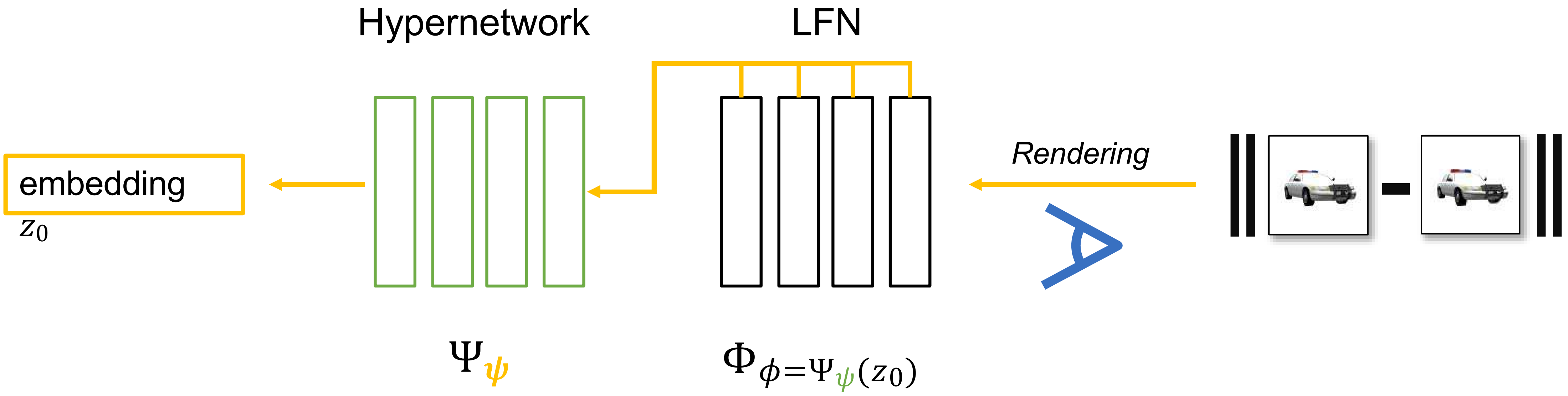
$$z_{j=0,\ldots,n} \sim \mathcal{N}(0, \sigma^2)$$

Decode embedding into scene representation



[1][Schmidhuber et al. 1992, Schmidhuber et al. 1993, Stanley et al. 2009, Ha et al., 2016]

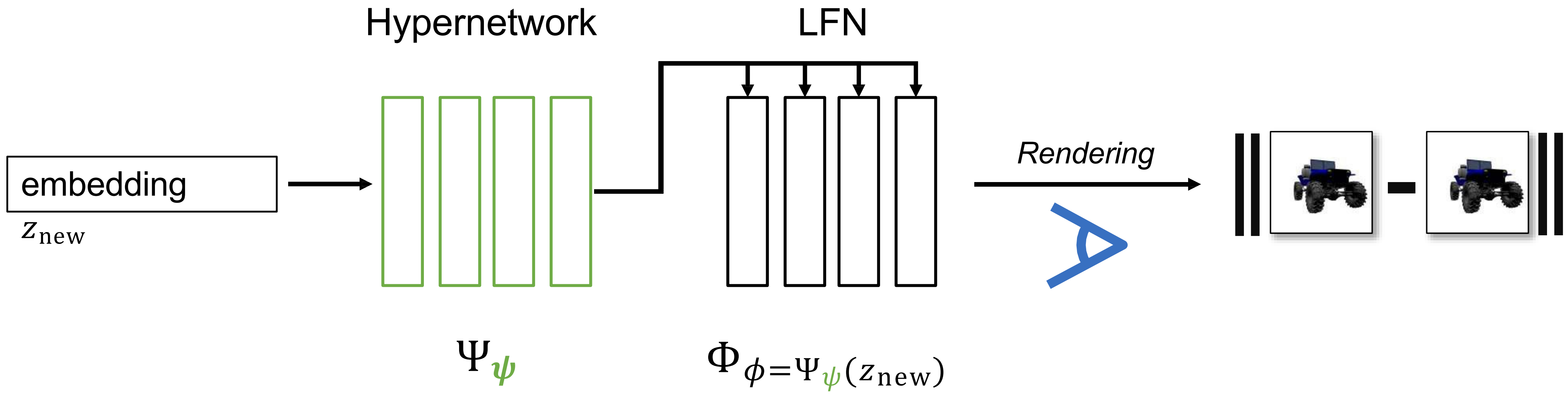
Decode embedding into scene representation



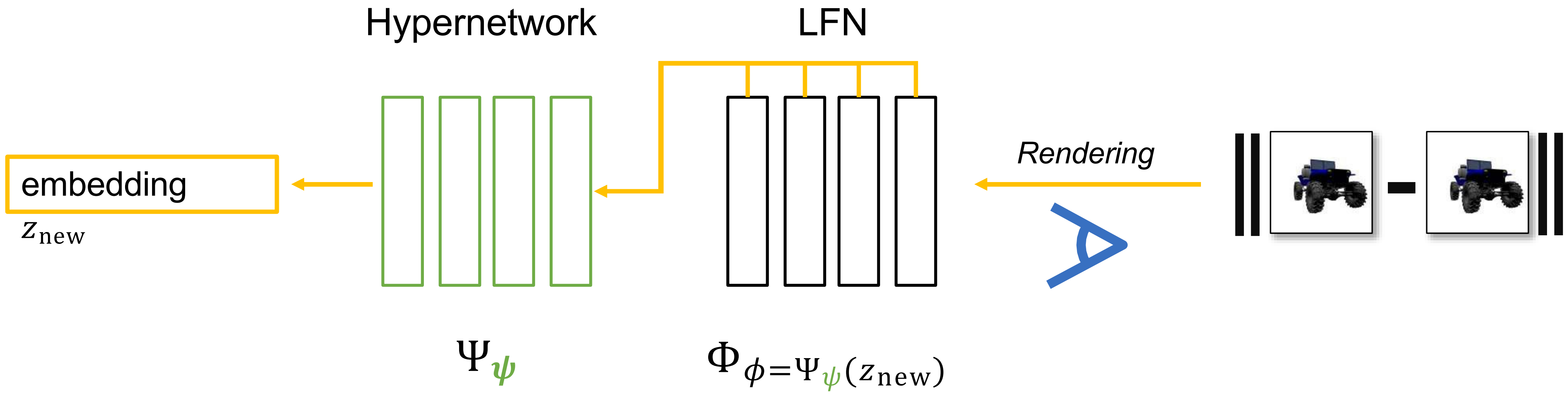
$$\arg\min_{\{z_j\}_{j=1}^M,\psi} \;{}_j \;{}_i \left\| \text{RENDER}(\Phi_{\phi=\Psi_\psi(z_j)}, \xi_i) - \mathcal{I}_i^j \right\|$$

Fast Rendering of Neural Radiance Fields, Lingjie Liu

Test time: Initialize new embedding



Freeze weights & optimize latent code only.



$$z = \arg\min_{z} \left\| \textsc{Render}(\Phi_{\phi=\Psi_{\psi}(z_0)}, \xi) - \mathcal{I} \right\|$$

Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

Vincent Sitzmann & Semon Rezchikov, NeurIPS 2021

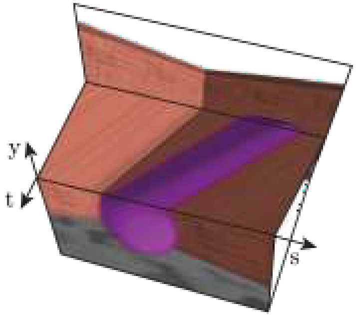
Results

Parameterization

LFN Geometry

Meta-Learning

Results

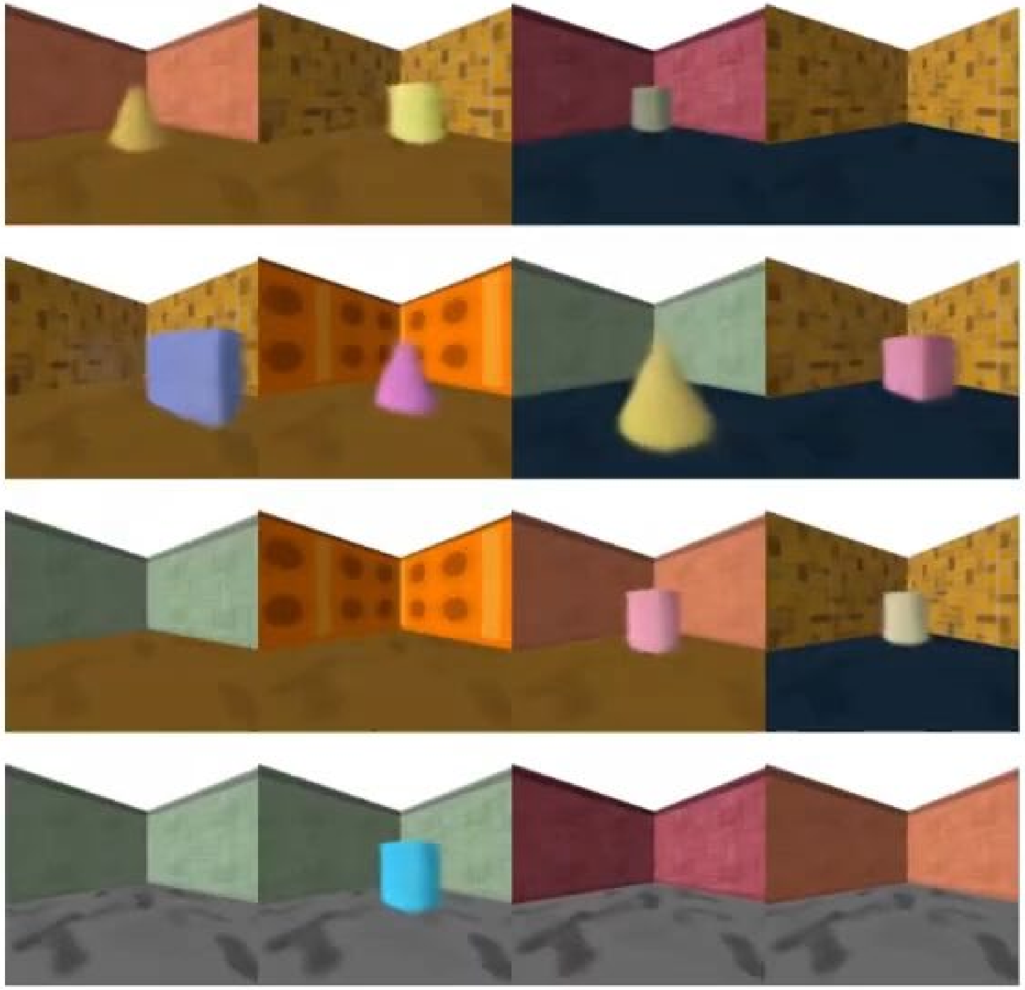


Limitations

LFNs learn multi-view consistent 360-degree light fields

500 FPS, single evaluation per ray.

GQN Rooms



Limitations

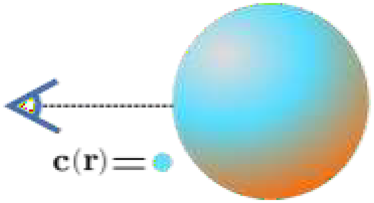
Parameterization

LFN Geometry

Meta-Learning

Results

Limitations

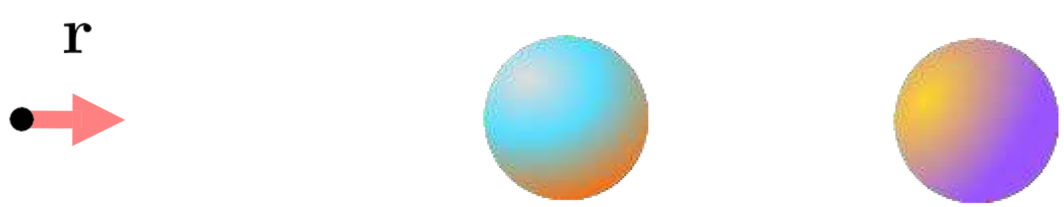


Limitations

One color per ray

Multi-view Consistency

Local conditioning

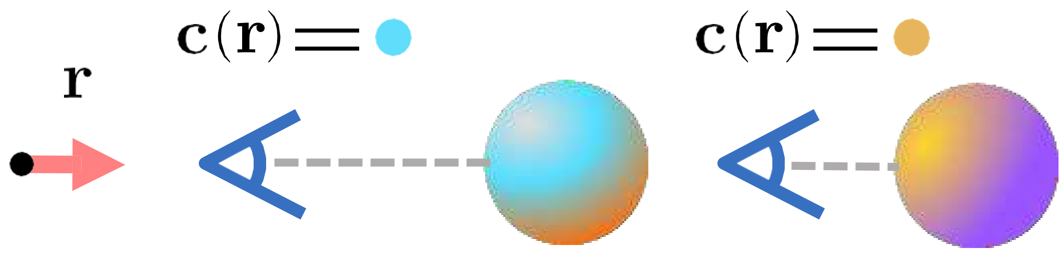


Limitations

One color per ray

Multi-view Consistency

Local conditioning

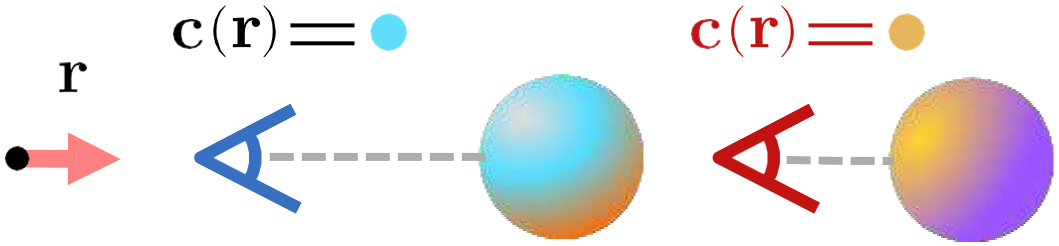


Limitations

One color per ray

Multi-view Consistency

Local conditioning

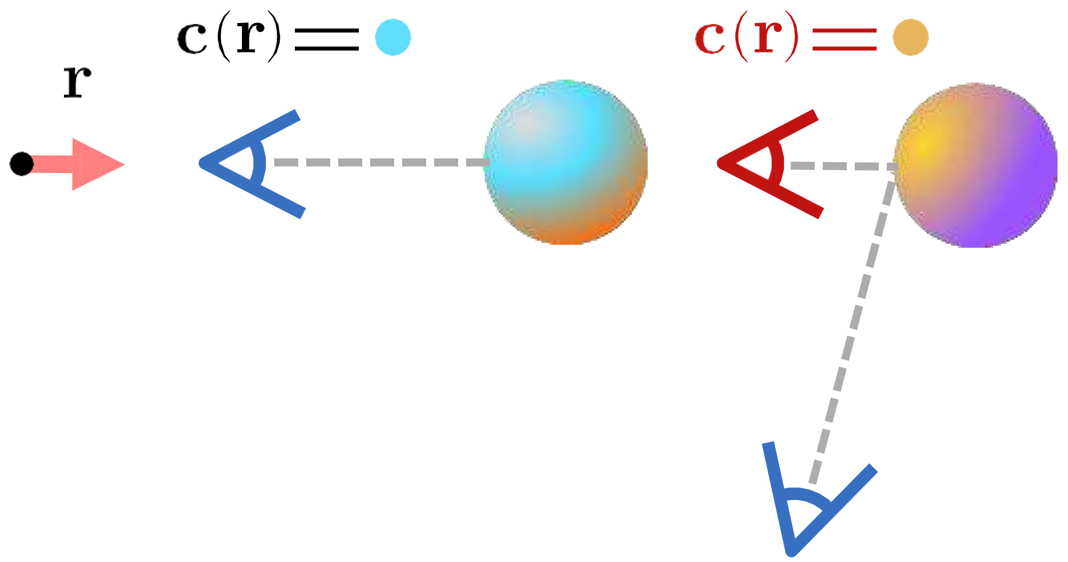


Limitations

One color per ray

Multi-view Consistency

Local conditioning



Limitations

One color per ray

Multi-view Consistency

Local conditioning

Fast Rendering of Neural Radiance Fields, Lingjie Liu

Limitations

One color per ray

Multi-view Consistency

Local conditioning

Overfitting single scene (with positional encoding)

Context Views

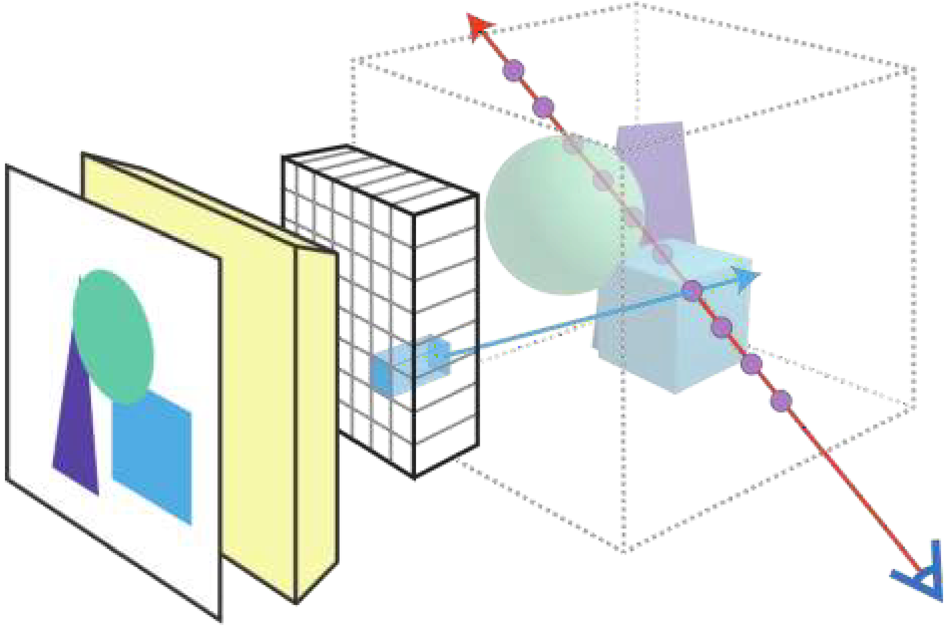
Intermediate Views

Limitations

One color per ray

Multi-view Consistency

Local conditioning



pixelNeRF Yu et al. 2020

Related Work

- NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, Mildenhall et al., ECCV 2020
- Neural Sparse Voxel Fields, Liu et al., NeurIPS 2020
- AutoInt: Automatic Integration for Fast Neural Volume Rendering, Lindell et al., CVPR 2021
- DeRF: Decomposed Radiance Fields, Rebain et al., CVPR 2021
- DONeRF: Towards Real-Time Rendering of Neural Radiance Fields using Depth Oracle Networks, Neff et al., Arxiv 2021
- FastNeRF: High-Fidelity Neural Rendering at 200FPS, Garbin et al., Arxiv 2021
- KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs, Reiser et al., Arxiv 2021
- PlenOctrees for Real-time Rendering of Neural Radiance Fields, Yu et al., Arxiv 2021
- Baking Neural Radiance Fields for Real-Time View Synthesis, Hedman et al., Arxiv 2021
- NeX: Real-time View Synthesis with Neural Basis Expansion. Wizadwongsa et al., CVPR 2021

Related Work

- Mixture of Volumetric Primitives, Lombodi et al., SIGGRAPH 2021
- Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering, Sitzmann et al., NeurIPS 2021

Acknowledgments

- Advances in Neural Rendering
- Neural Fields in Visual Computing and Beyond
- awesome-NeRF: a curated list of awesome neural radiance fields papers
- MPII Summer Semester 2023: Computer Vision and Machine Learning for Computer Graphics
- Vincent Sitzmann from MIT

Any Questions?