Novel View Synthesis

3D Vision

University of Illinois

Derek Hoiem

Many slides adapted from Lana Lazebnik, Steve Seitz, Yasu Furukawa, Noah Snavely

This class: Novel View Synthesis

- Applications and problem space
- NeRF
- Mesh-based

Applications of Novel View Synthesis

- Walk-throughs and photo tours
- Merchandise inspection
- Virtual tourism / Entertainment / VR

Novel view synthesis

- View interpolation
  - Render views that are similar or between photo views
- View extrapolation
  - Render views from arbitrary positions and orientations
- View manipulation
  - Change materials, lighting, or content

Matterport example: https://matterport.com/gallery

How Matterport viewing works

- Mesh viewing
  - Solve for mesh, texture map, and render from arbitrary viewpoint
  - Enables extrapolation and free view synthesis

- Photo viewing and transitions
  - Transition by texture mapping start/destination photos onto simple mesh and cross-fading during movement
  - Enables restricted photo tour

- What is good and bad about these approaches?

Mesh:
+ simple, complete freedom of movement, can also support measurement/pins/annotations
- Cannot render view-dependent effects, artifacts due to geometry/texture errors

Photo tour w/ mesh-based cross-fade
+ simple, looks perfect at photo locations
- Very limited freedom of movement

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Most of following slides from Jon Barron

Ben Mildenhall*

UC Berkeley

Pratul Srinivasan*

UC Berkeley

Matt Tancik*

UC Berkeley

Jon Barron

Google Research

Ravi Ramamoorthi

UC San Diego

Ren Ng

UC Berkeley

Problem: View Interpolation

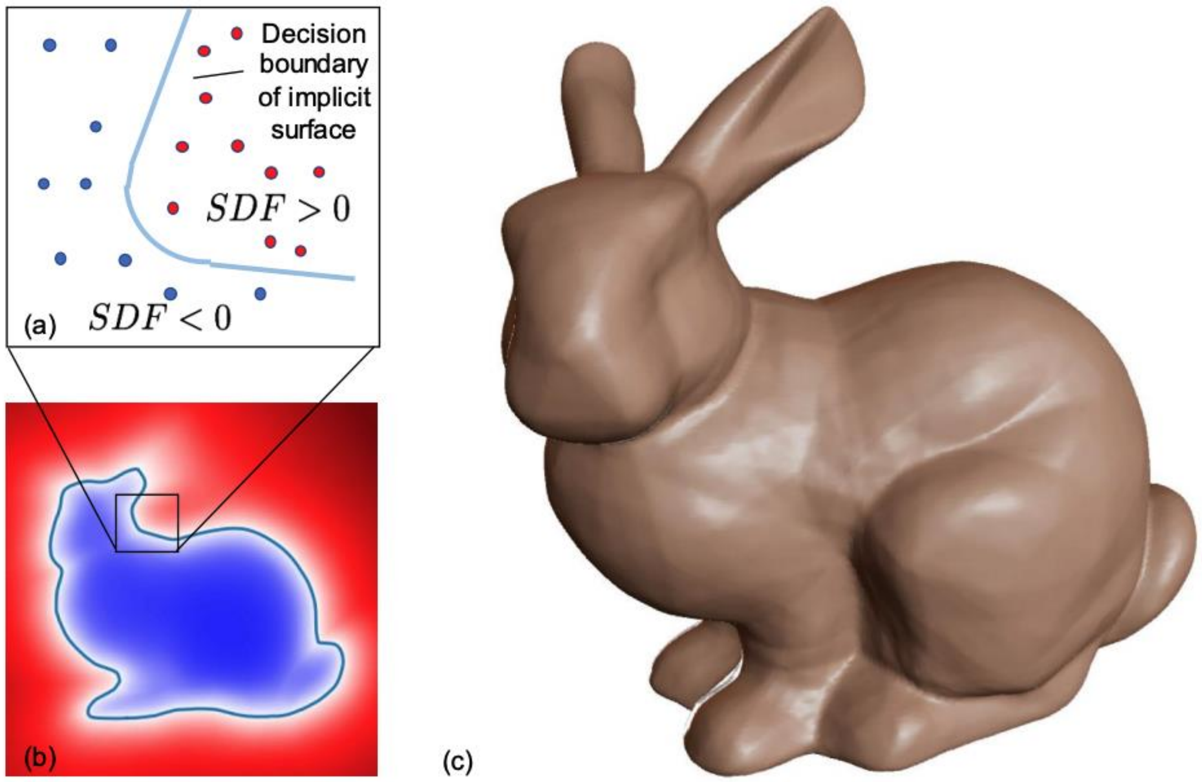
Inputs: sparsely sampled images of scene

Outputs: *new* views of same scene

tancik.com/nerf

Slide credit: Jon Barron

Neural Networks as a Continuous Shape Representation



$(x, y, z) \rightarrow occupancy$

$(x, y, z) \rightarrow distance$

$(x, y, z) \rightarrow (color, occupancy)$

$(x, y, z) \rightarrow latent\ vector$

+ Compact and expressive parameterization

− Limited rendering, difficult to optimize

Mescheder et al. Occupancy Networks, CVPR 2019, Park et al., DeepSDF, CVPR 2019,
Sitzmann et al., Scene Representation Networks, NeurIPS 2019, Niemeyer et al. Differentiable Volumetric Rendering, CVPR 2020

Slide credit: Jon Barron

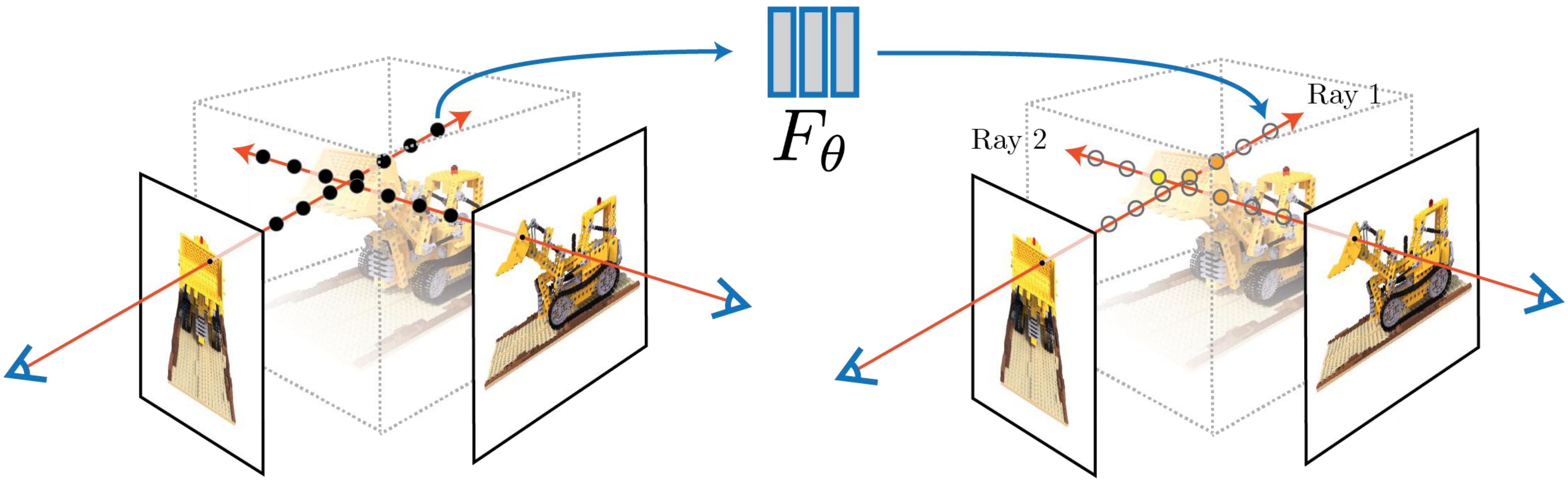
NeRF (neural radiance fields)

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

F_θ

Fully-connected neural network
9 layers,
256 channels

Generate views with traditional volume rendering



Slide credit: Jon Barron

Volume rendering is trivially differentiable

Rendering model for ray r(t) = o + td:

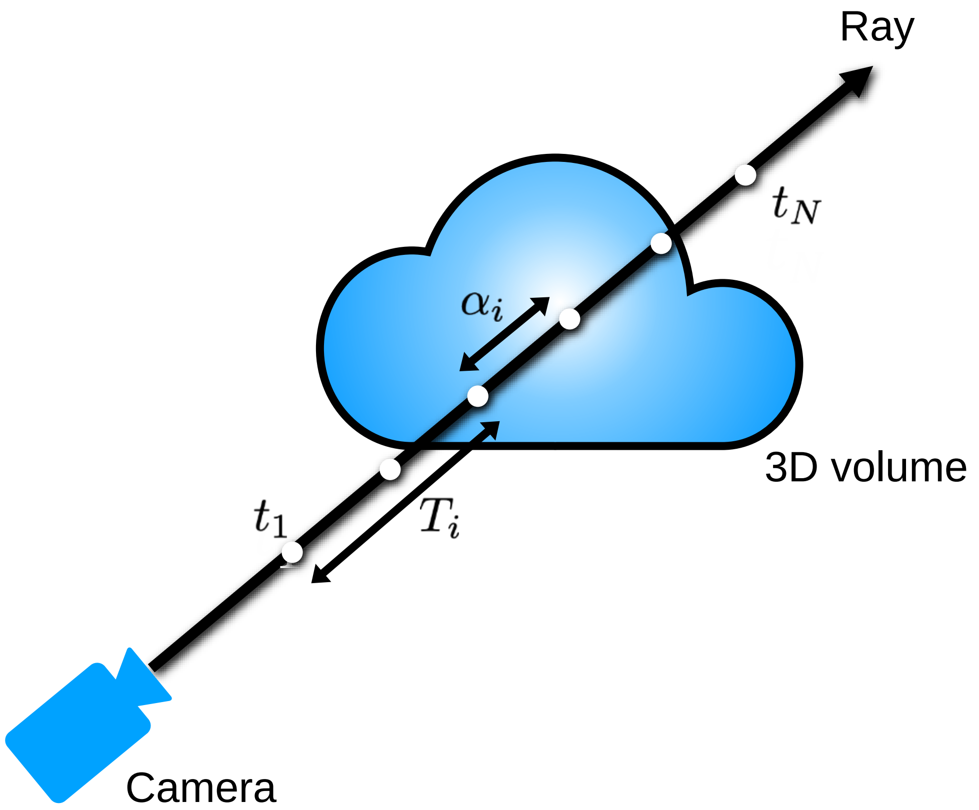
$$C \approx \sum_{i=1}^{N} \underbrace{T_i \alpha_i}_{\text{weights}} \underbrace{c_i}_{\text{colors}}$$

How much light is blocked earlier along ray:

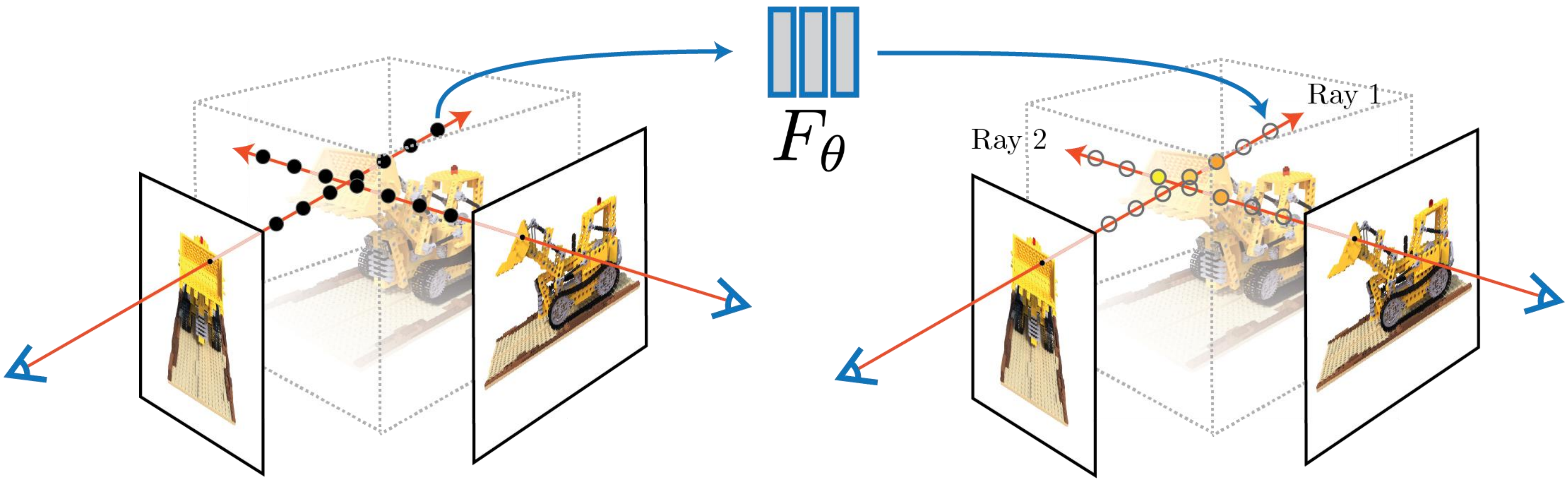
$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

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

$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$ ← Density * Distance Between Points



Optimize with gradient descent on rendering loss

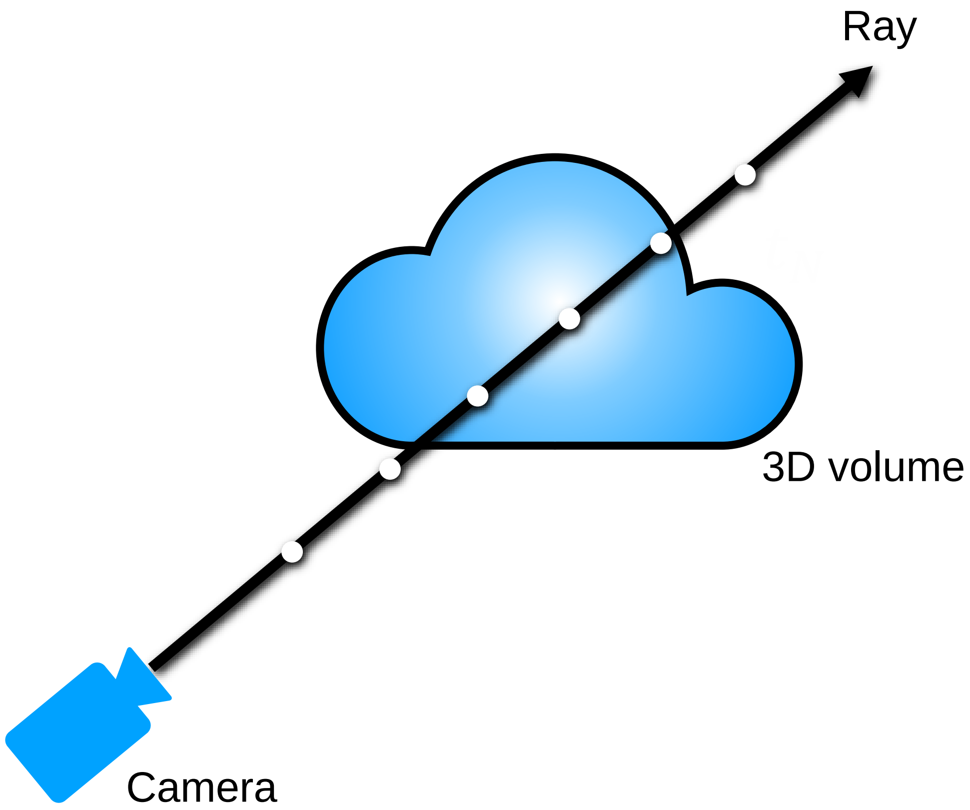


$$\min_\theta \sum_i ||\text{render}_i(F_\theta) - I_i||^2$$

Training network to reproduce all input views of the scene

Slide credit: Jon Barron

Can we allocate samples more efficiently?
Two pass rendering

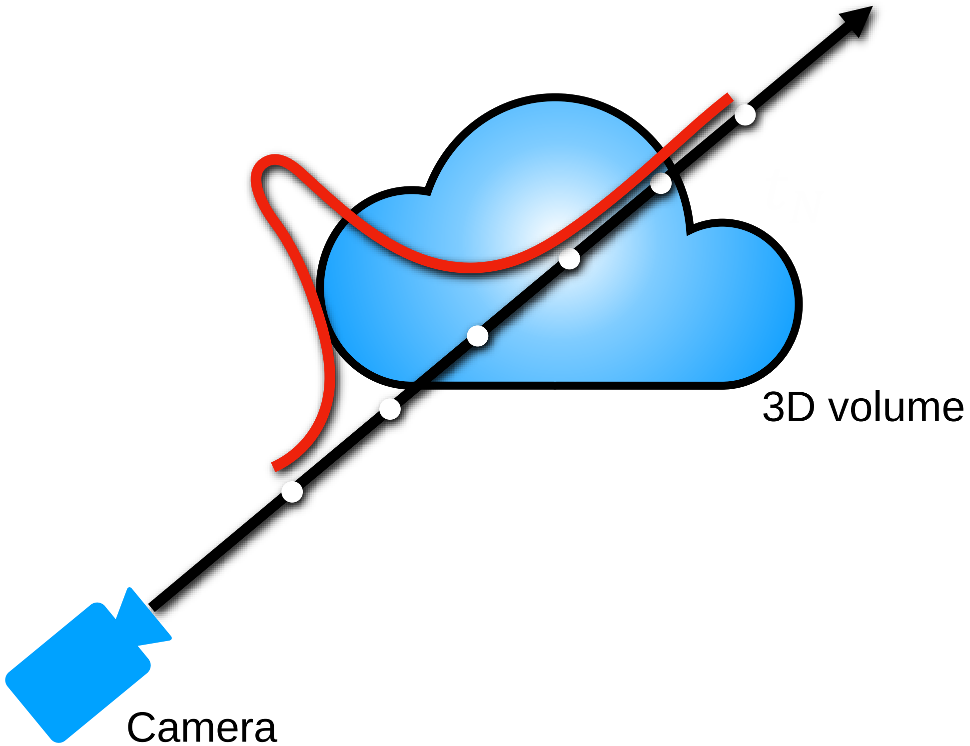


Slide credit: Jon Barron

Two pass rendering: coarse

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

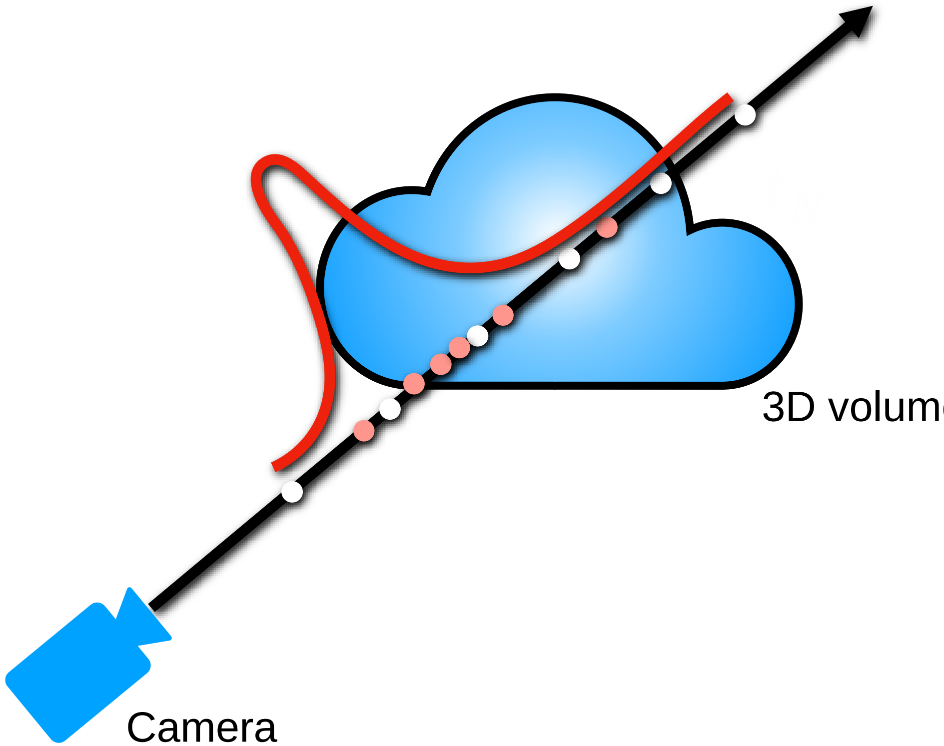
treat weights as probability distribution for new samples



Two pass rendering: fine

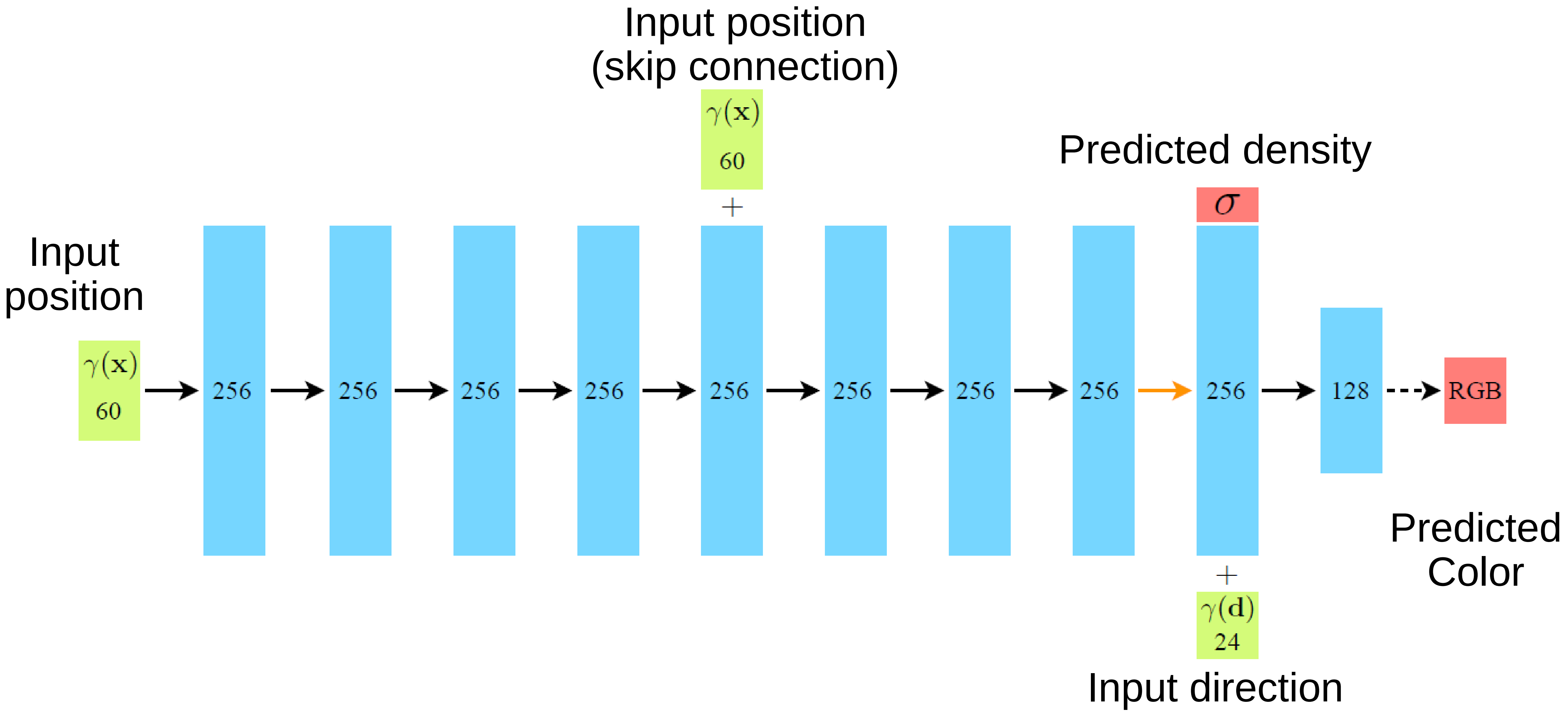
$$C \approx \sum_{i=1}^{N} \boxed{T_i \alpha_i} c_i$$

treat weights as probability
distribution for new samples

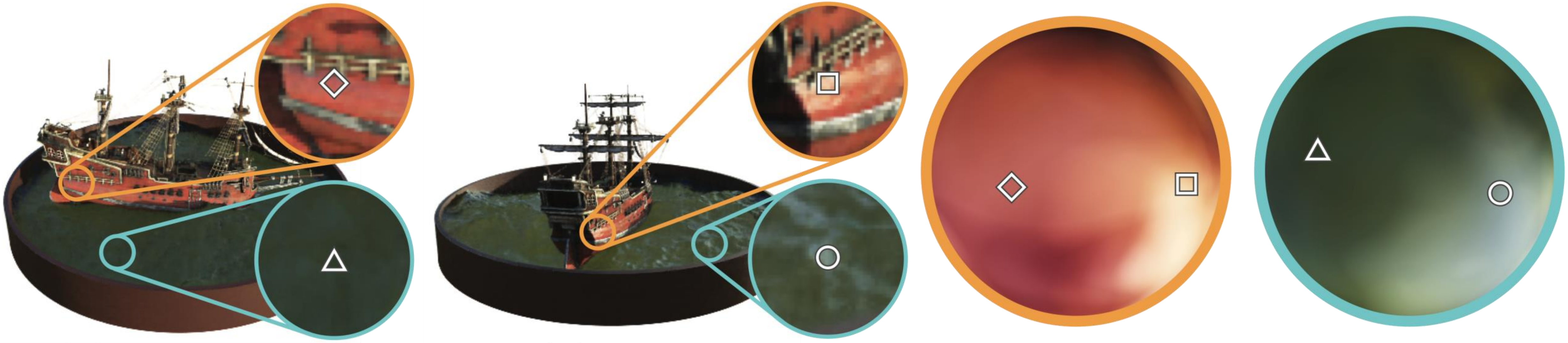


Slide credit: Jon Barron

Network Structure



Viewing directions as input



(a) View 1 (b) View 2 (c) Radiance Distributions

Slide credit: Jon Barron

Naive implementation produces blurry results

NeRF (Naive)

Naive implementation produces blurry results

NeRF (Naive)

NeRF (with positional encoding)

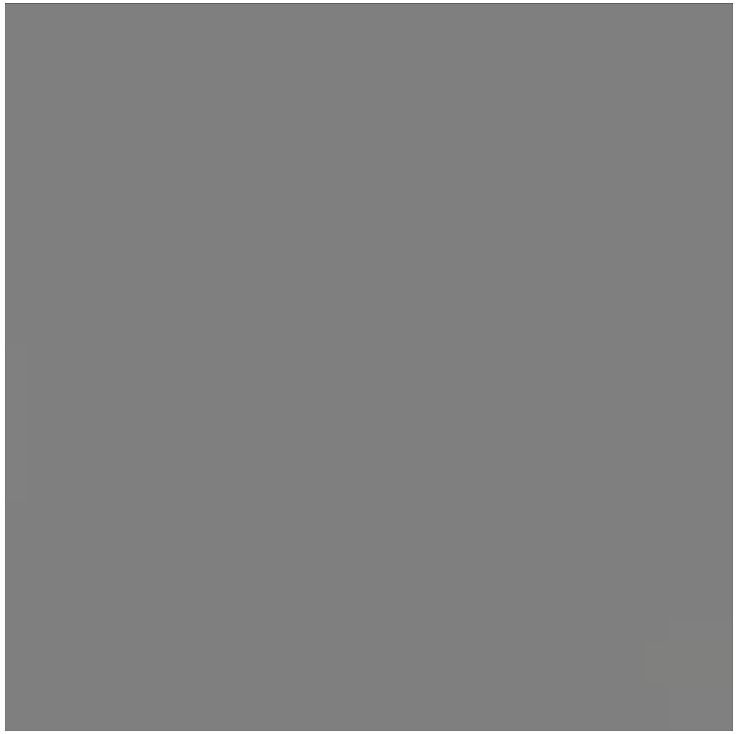
Slide credit: Jon Barron

Toy problem: memorizing a 2D image

Toy problem: memorizing a 2D image

Ground truth image

Standard fully-connected net



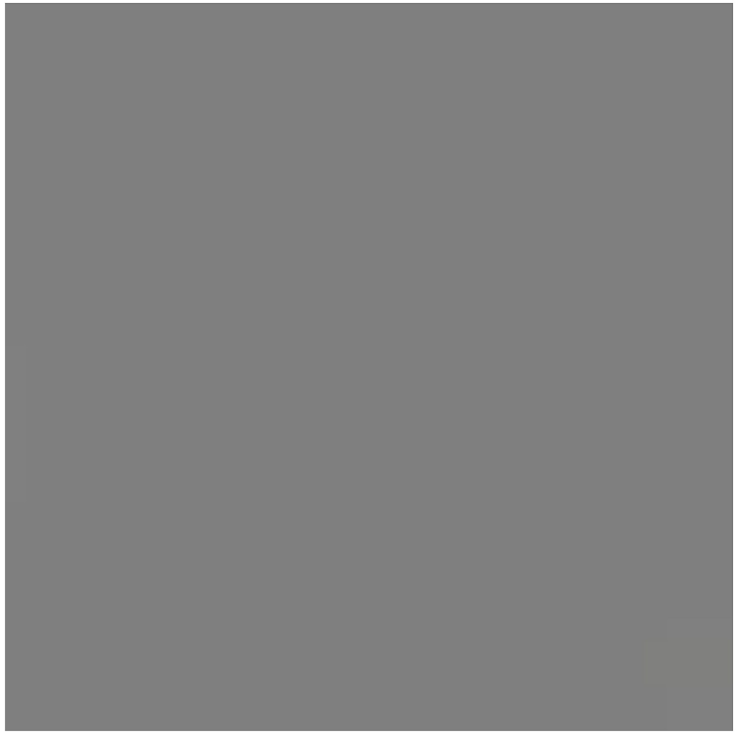
Slide credit: Jon Barron

$$\begin{pmatrix} \sin(\mathbf{v}), \cos(\mathbf{v}) \\ \sin(2\mathbf{v}), \cos(2\mathbf{v}) \\ \sin(4\mathbf{v}), \cos(4\mathbf{v}) \\ \dots \\ \sin(2^{L-1}\mathbf{v}), \cos(2^{L-1}\mathbf{v}) \end{pmatrix}$$

Slide credit: Jon Barron

Ground truth image

Standard fully-connected net



With Positional Encoding

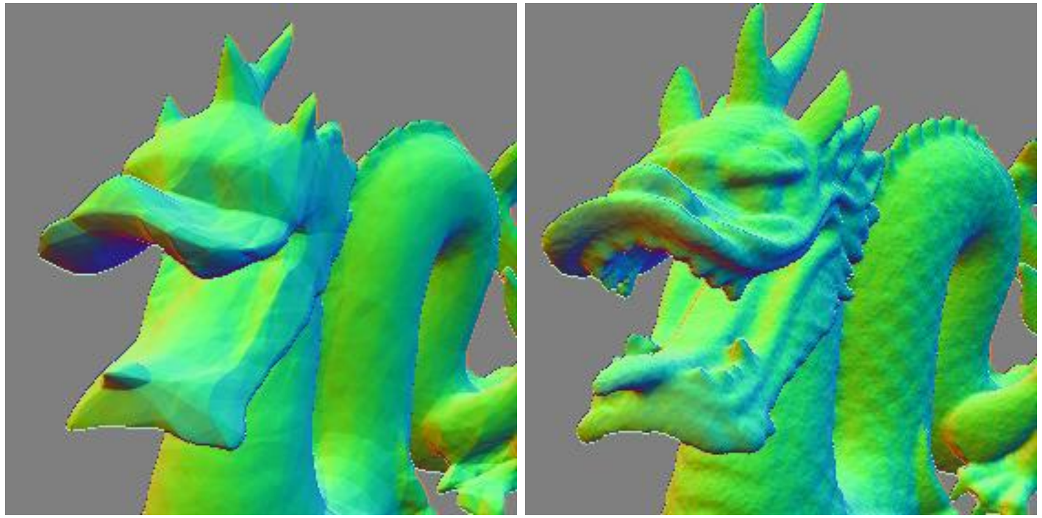
Positional encoding also directly improves our scene representation!

NeRF (Naive)

NeRF (with positional encoding)

Slide credit: Jon Barron

Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains



Matthew Tancik*, Pratul Srinivasan*, Ben Mildenhall*,
Sara Fridovich-Keil, Nithin Ragahavan, Utkarsh Singhal,
Ravi Ramamoorthi, Jonathan T. Barron, Ren Ng

Slide credit: Jon Barron

$\mathbf{v}$ Input Coordinate (position, view direction) → Input Mapping $\gamma(\cdot)$ → Relu MLP $f(\cdot\,;\theta)$ → Output Signal $f(\gamma(\mathbf{v});\theta)$ (color, density)

Positional Encoding [1]: $\gamma(\mathbf{v}) = \left[\cos(2^0\mathbf{v}), \sin(2^0\mathbf{v}), \ldots, \cos(2^{L-1}\mathbf{v}), \sin(2^{\boxed{L}-1}\mathbf{v})\right]$

Random Fourier Features [2]: $\gamma(\mathbf{v}) = \left[\cos(\mathbf{B}\mathbf{v}), \sin(\mathbf{B}\mathbf{v})\right]$ $\mathbf{B} \sim \mathcal{N}(0, \boxed{\sigma}^2)$

[1] Vaswani et al.. NeurIPS, 2017
[2] Rahimi & Recht. NeurIPS, 2007

Slide credit: Jon Barron

Neural Tangent Kernel

$$f(\mathbf{x};\theta) \approx \sum_i (\mathbf{K}^{-1}\mathbf{y})_i k(\mathbf{x}_i, \mathbf{x})$$

Under certain conditions,
neural networks are kernel regression(!)

$$k(\mathbf{x}_i, \mathbf{x}_j) = h_{\text{NTK}}(\langle \mathbf{x}_i, \mathbf{x}_j \rangle)$$

$$h_{\text{NTK}} : \mathbb{R} \to \mathbb{R}$$

ReLU MLPs correspond to a “dot product” kernel

Jacot et al., NeurIPS, 2018, Arora, et al., ICML, 2019, Basri et al., 2020., Du et al., ICLR, 2019., Lee et al., NeurIPS, 2019

Slide credit: Jon Barron

Dot Product of Fourier Features

$$
\begin{aligned}
\langle \gamma(\mathbf{v}_1), \gamma(\mathbf{v}_2) \rangle &= \sum_j \left(\cos(\mathbf{b}_j^{\mathrm{T}} \mathbf{v}_1) \cos(\mathbf{b}_j^{\mathrm{T}} \mathbf{v}_2) + \sin(\mathbf{b}_j^{\mathrm{T}} \mathbf{v}_1) \sin(\mathbf{b}_j^{\mathrm{T}} \mathbf{v}_2) \right) \\
&= \sum_j \cos\left(\mathbf{b}_j^{\mathrm{T}} (\mathbf{v}_1 - \mathbf{v}_2) \right) \qquad \text{(cosine difference trig identity)} \\
&\triangleq h_\gamma(\mathbf{v}_1 - \mathbf{v}_2)
\end{aligned}
$$

Fourier Features → stationary kernel

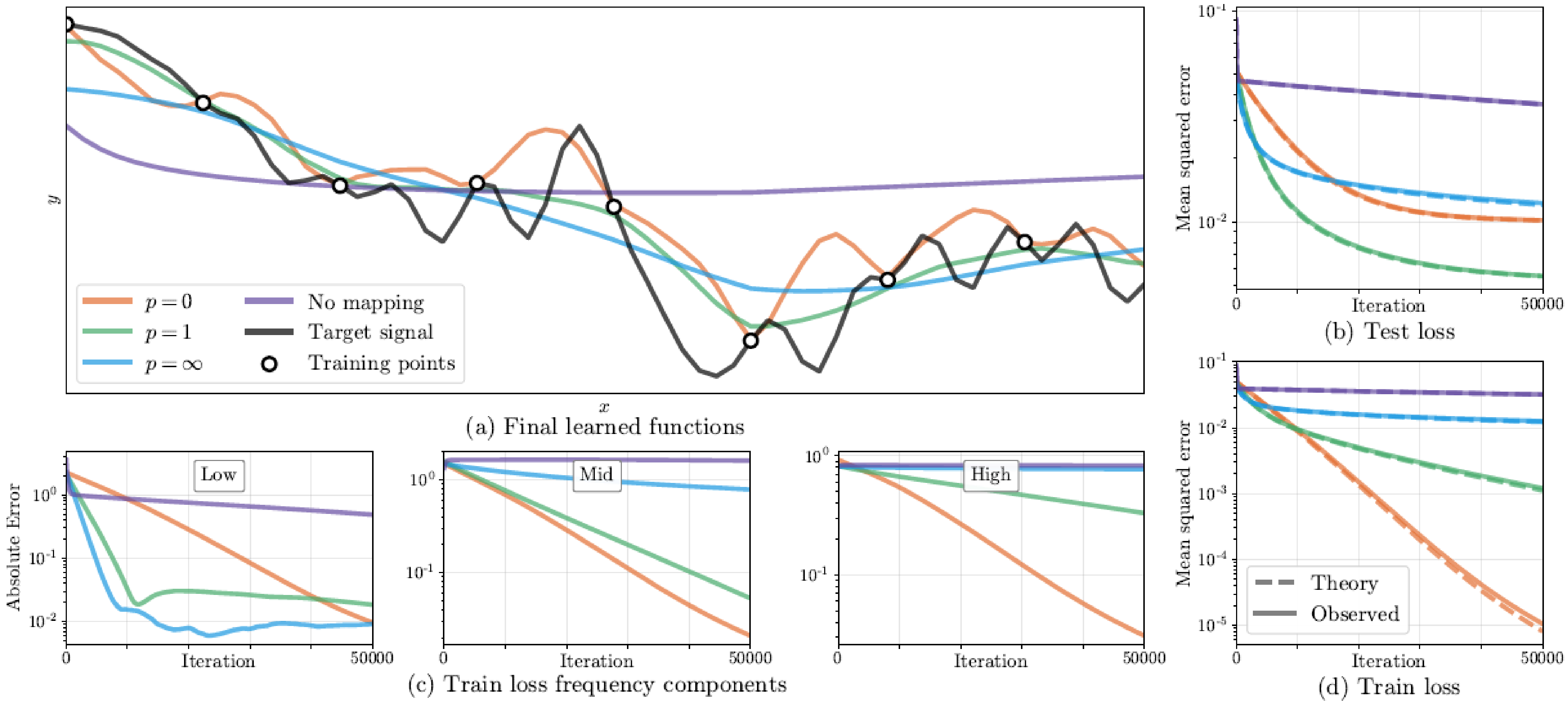
Resulting *composed* NTK is stationary

$$h_{\text{NTK}}\Big(\,\langle\gamma(\mathbf{v})_i, \gamma(\mathbf{v})_j\rangle\,\Big) = h_{\text{NTK}}(h_\gamma(\mathbf{v}_i - \mathbf{v}_j))$$

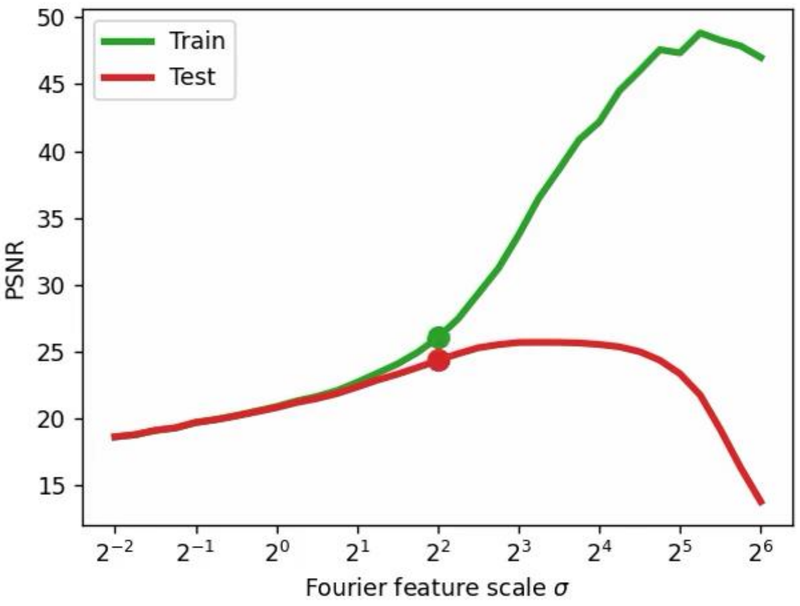
Resulting network regression function is a *convolution*

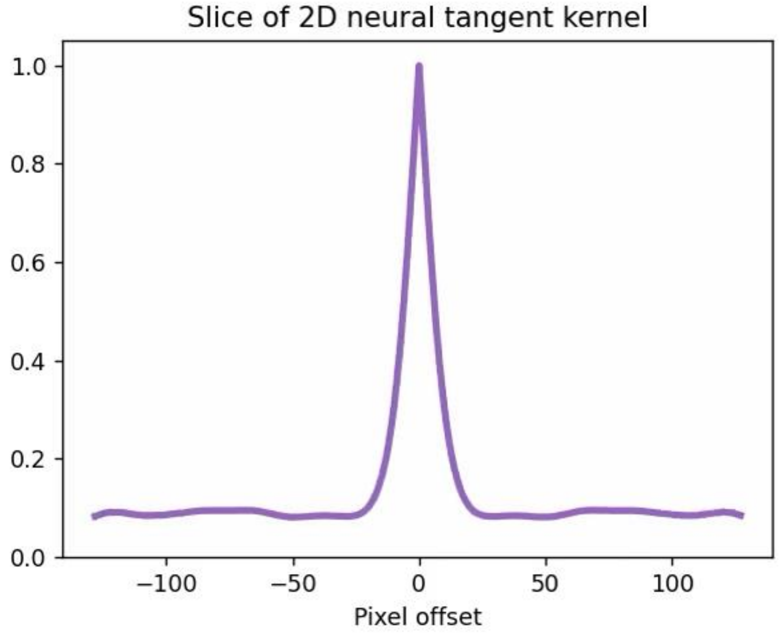
$$\hat{f} = (h_{\text{NTK}} \circ h_\gamma) * \sum_{i=1}^{n} w_i \delta_{\mathbf{v}_i}$$

Fit to 1D function with varying Fourier features (low p = high frequency FF)



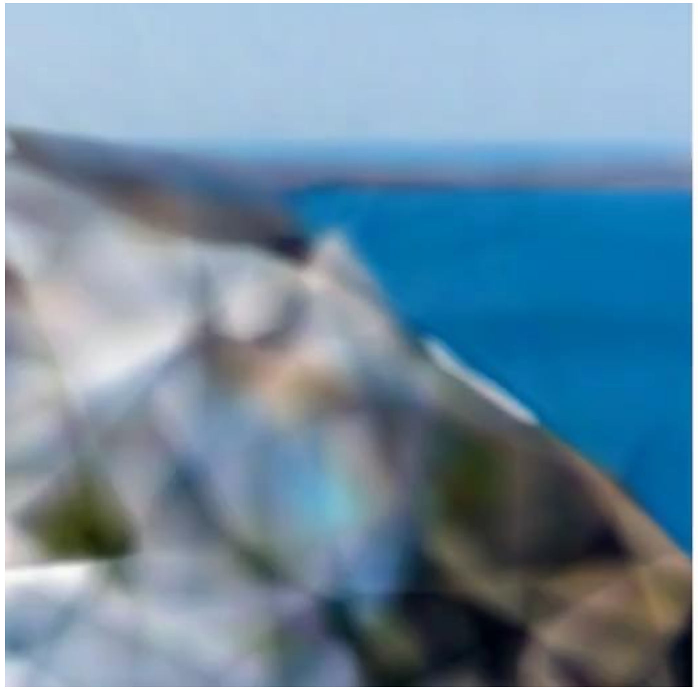
Mapping bandwidth controls underfitting / overfitting

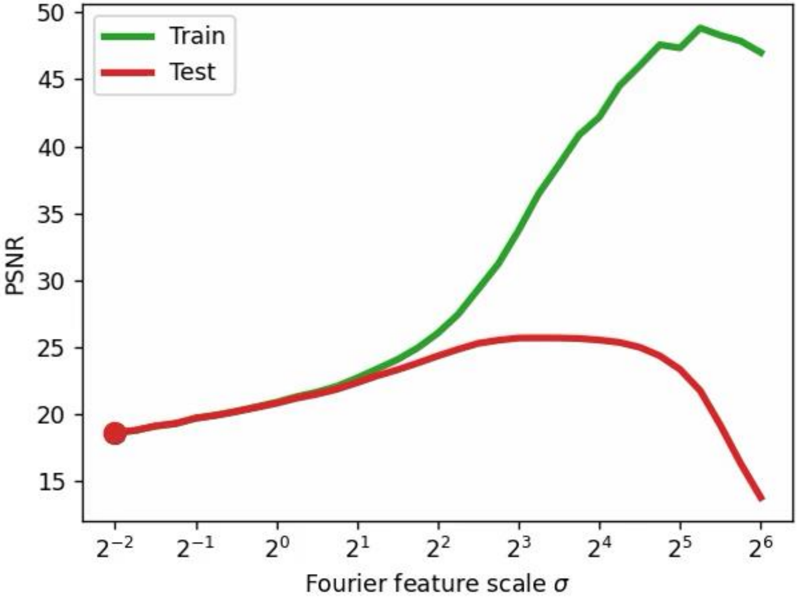


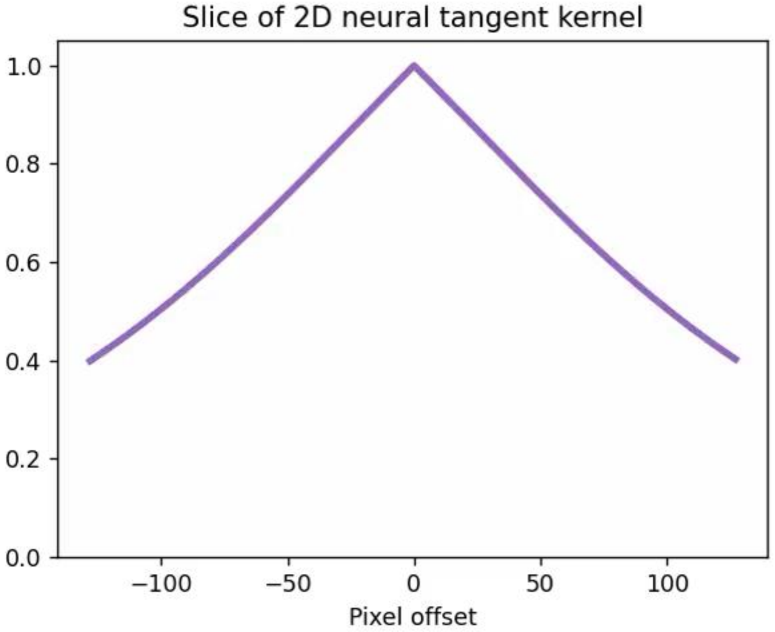


$$\gamma(\mathbf{v}) = [\cos(\mathbf{B}\mathbf{v}), \sin(\mathbf{B}\mathbf{v})] \qquad \mathbf{B} \sim \mathcal{N}(0, \boxed{\sigma}^2)$$

Mapping bandwidth controls underfitting / overfitting

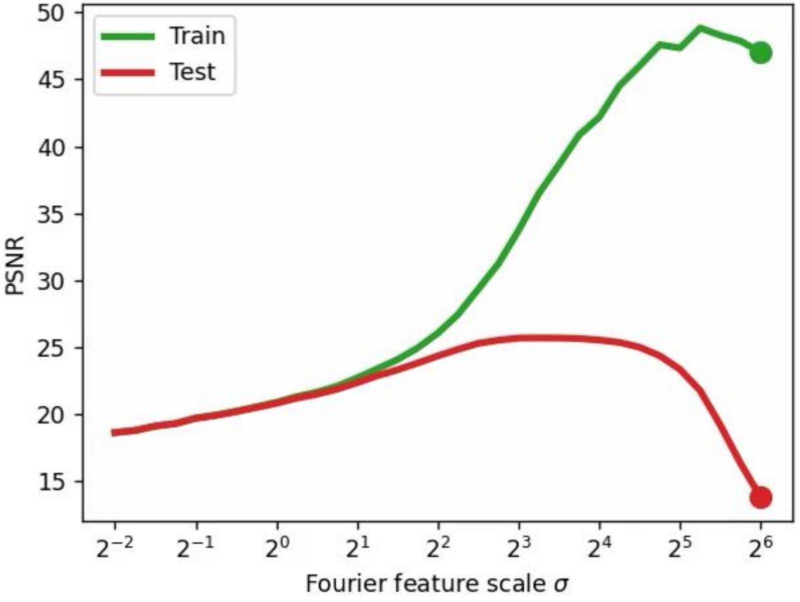


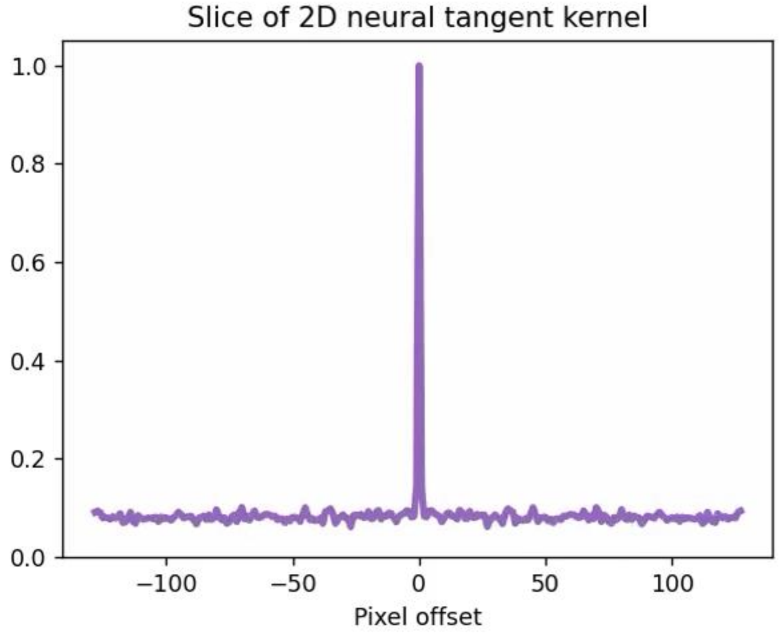




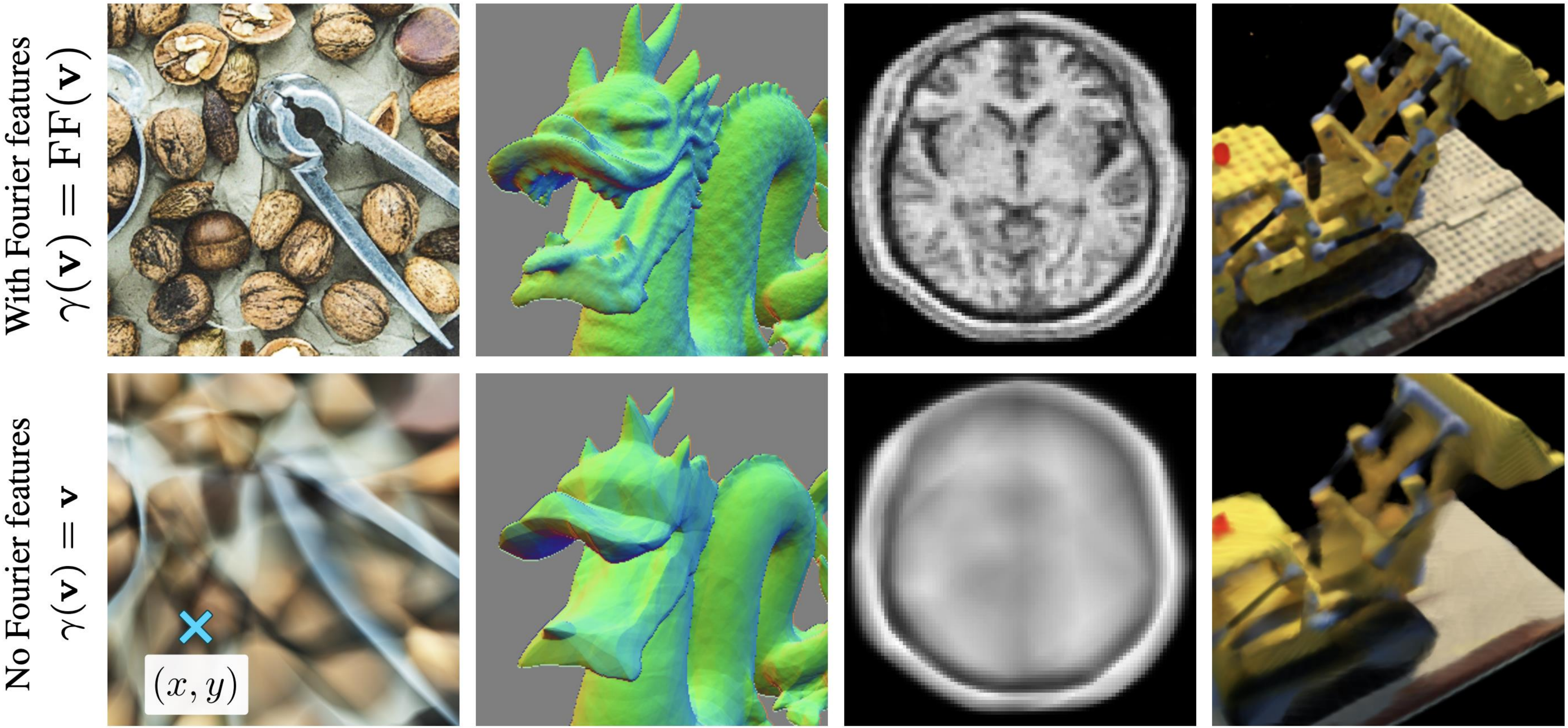
$$\gamma(\mathbf{v}) = [\cos(\mathbf{B}\mathbf{v}), \sin(\mathbf{B}\mathbf{v})] \qquad \mathbf{B} \sim \mathcal{N}(0, \boxed{\sigma}^2)$$

Mapping bandwidth controls underfitting / overfitting





$$\gamma(\mathbf{v}) = [\cos(\mathbf{B}\mathbf{v}), \sin(\mathbf{B}\mathbf{v})] \qquad \mathbf{B} \sim \mathcal{N}(0, \boxed{\sigma}^2)$$



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

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

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

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

Slide credit: Jon Barron

Try It!

```
B = SCALE * np.random.normal(shape=(input_dims, NUM_FEATURES))
x = np.concatenate([np.sin(x @ B), np.cos(x @ B)], axis=-1)

x = nn.Dense(x, features=256)
```

Slide credit: Jon Barron

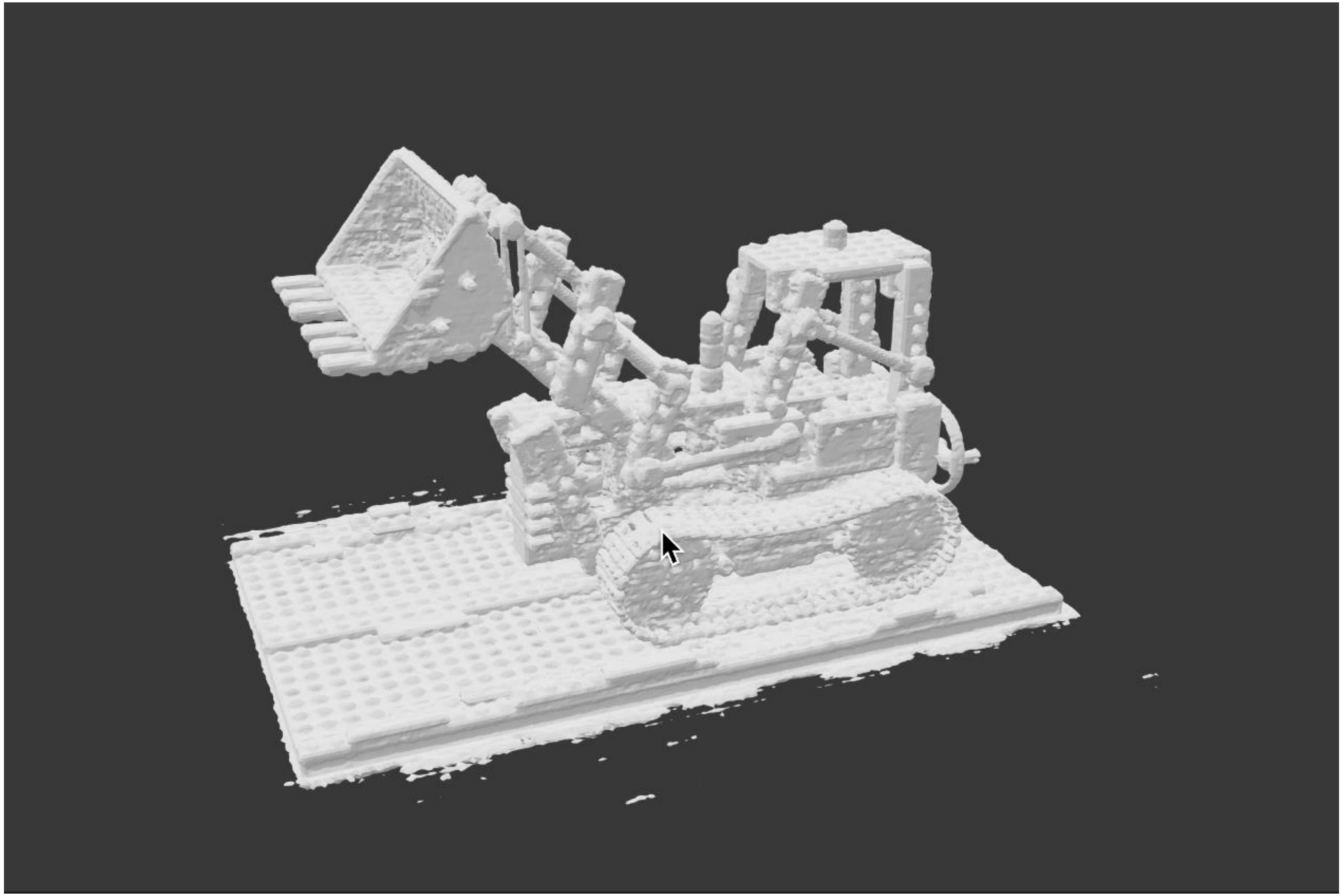
Results


View-Dependent Effects

Detailed Geometry & Occlusion

Detailed Geometry & Occlusion

Meshable



Baking Neural Radiance Fields for Real-Time View Synthesis

arXiv 2021

Peter Hedman Pratul P. Srinivasan Ben Mildenhall Jonathan T. Barron Paul Debevec

Google Research

Paper

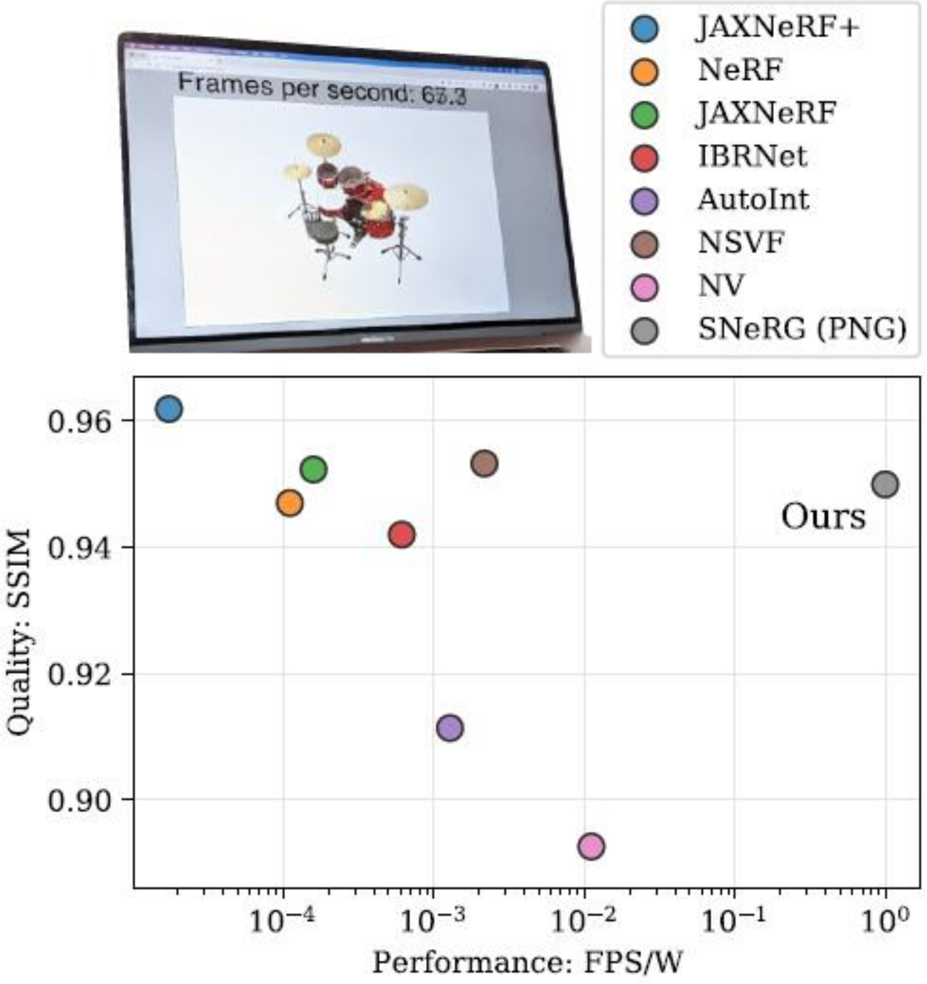
Video

Demos

http://nerf.live/

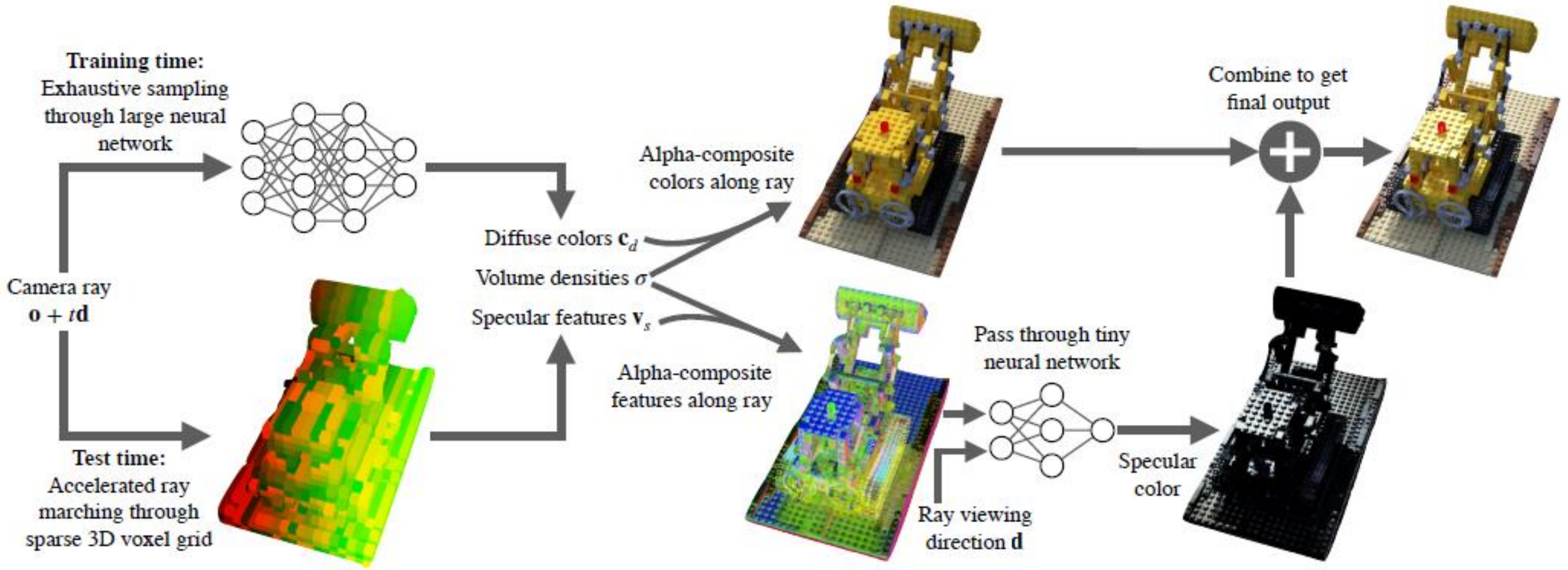
Has a demo too! →

Concurrent works:
Yu et al., PlenOctrees
Garbin et al., FastNeRF
Reiser et al., KiloNeRF



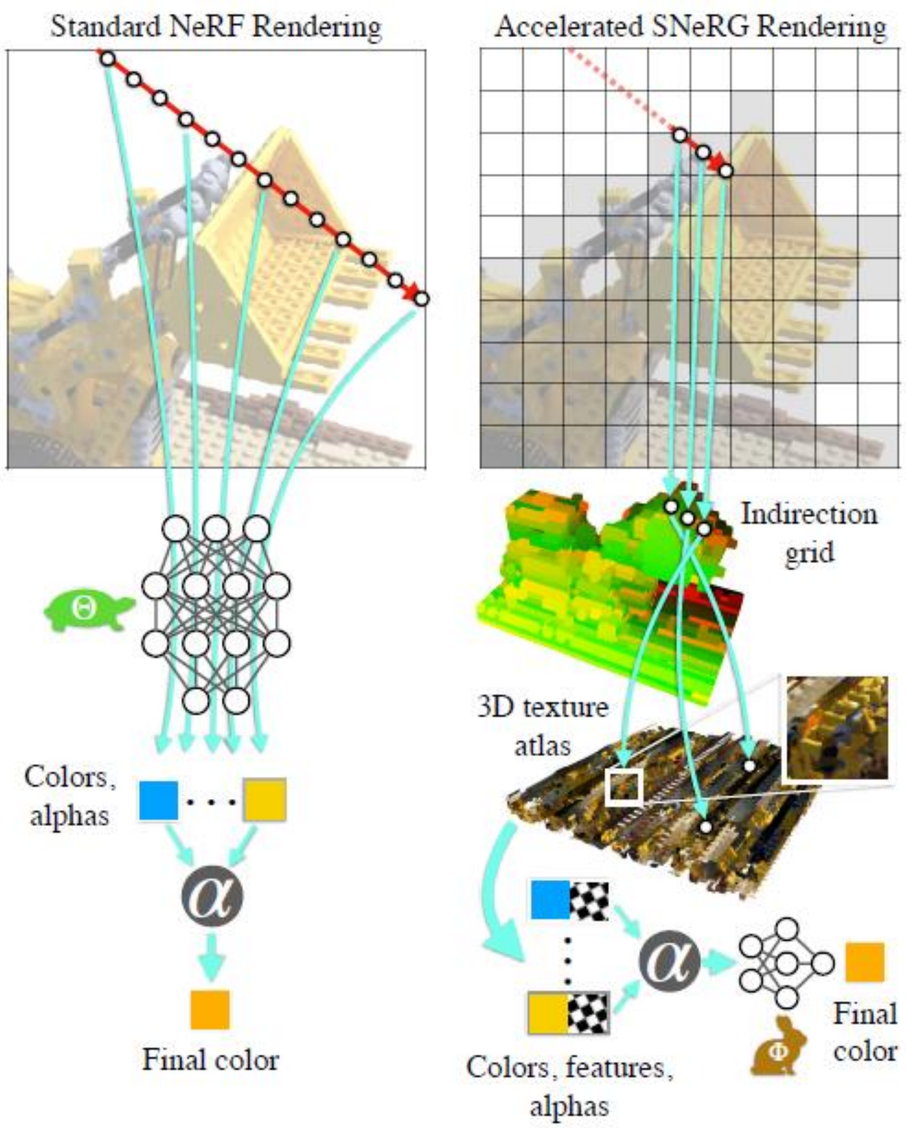
Frames per second: 63.3
JAXNeRF+
NeRF
JAXNeRF
IBRNet
AutoInt
NSVF
NV
SNeRG (PNG)
Ours
Quality: SSIM
0.96
0.94
0.92
0.90
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
Performance: FPS/W

- NeRF modified to output diffuse color, density, and 4-d specular features
- Color and features are accumulated along ray, and a small network produces a specular residual that is added to color
- Prior encourages sparse density/opacity in coarse samples



Rendering

- Precompute anti-aliased diffuse colors/features on voxel grid (1000^3 to 1300^3)
- Voxels are stored sparsely and divided into local blocks
- In coarse grid, store whether occupied and if so pointer to higher resolution color/feature info
- Compute specular component from features (only once per pixel) and add to color
- All values are quantized and compressed
- Per-pixel shading is fine-tuned to recover losses due to above process
- Result: 30+ FPS on laptop, < 100 MB model



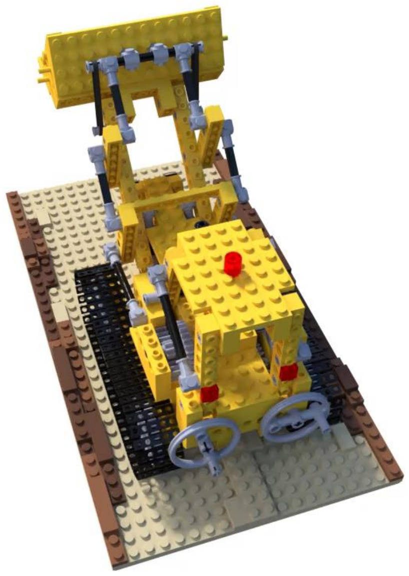
Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields

Jonathan T. Barron   Ben Mildenhall   Matthew Tancik

Peter Hedman   Ricardo Martin-Brualla   Pratul P. Srinivasan

Google

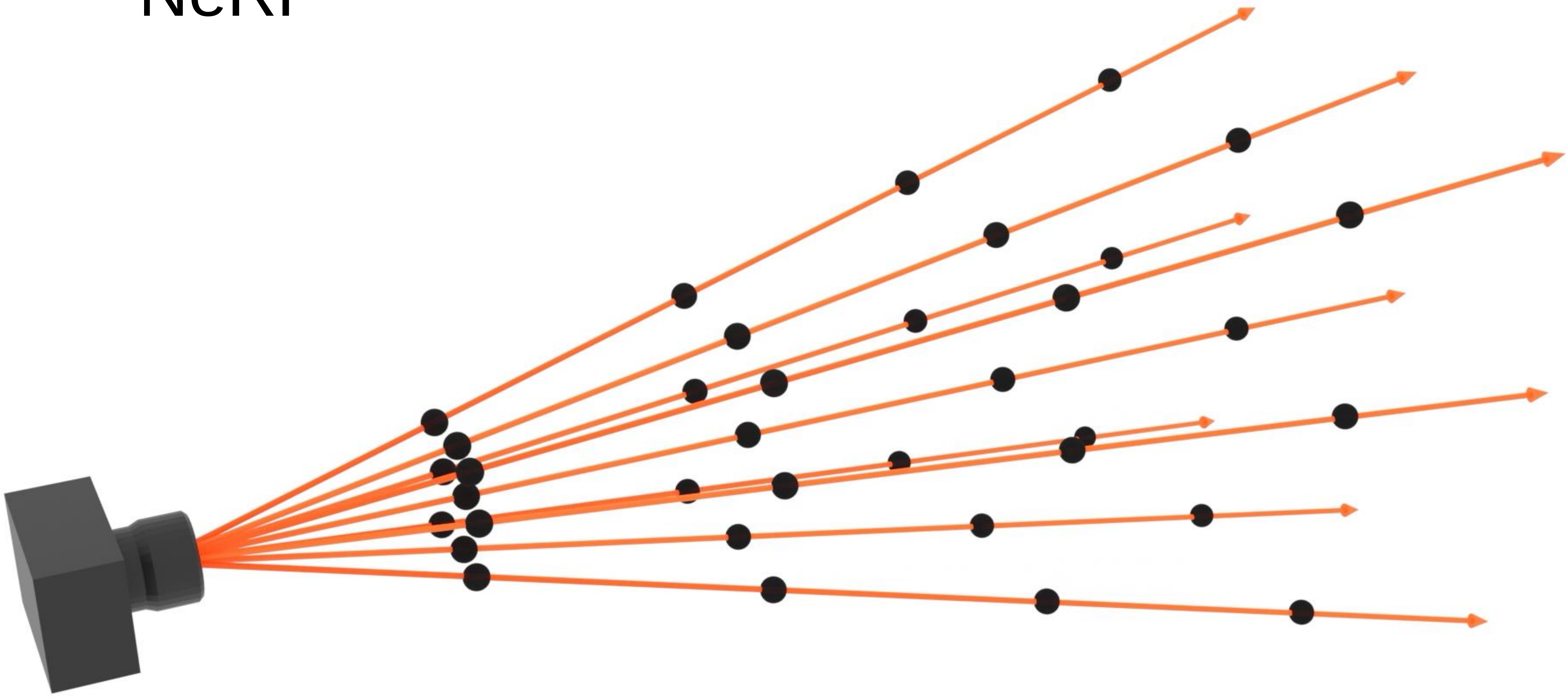
Berkeley
UNIVERSITY OF CALIFORNIA



Ground Truth

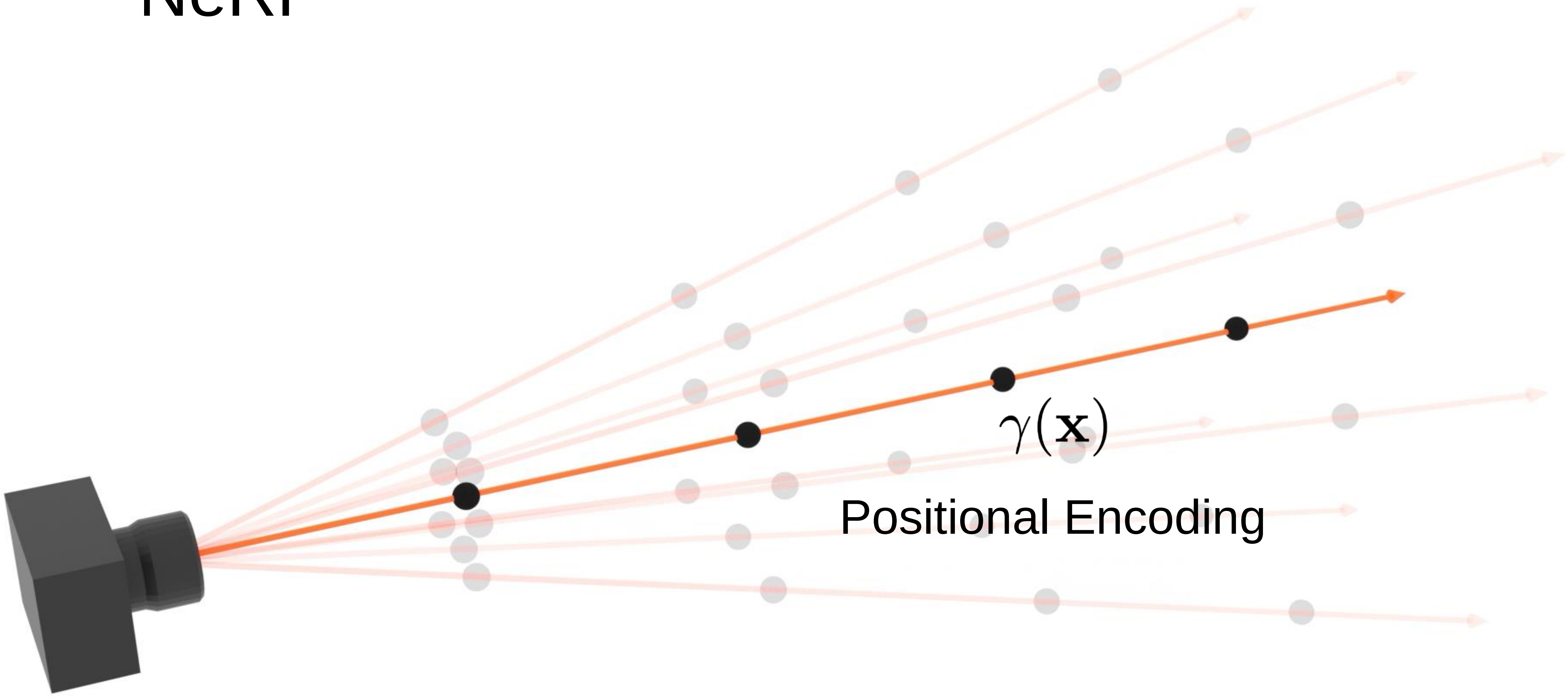
Slide credit: Jon Barron

NeRF



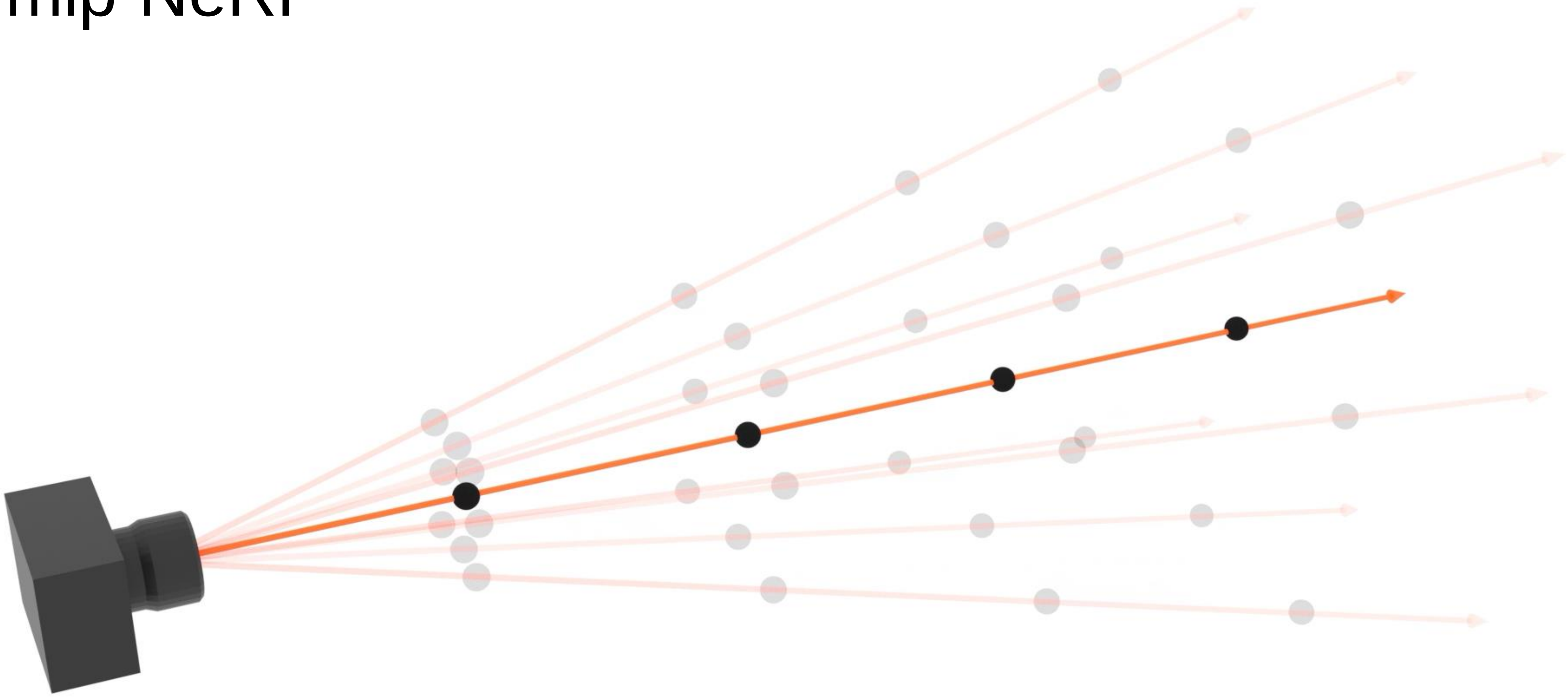
Slide credit: Jon Barron

NeRF



Slide credit: Jon Barron

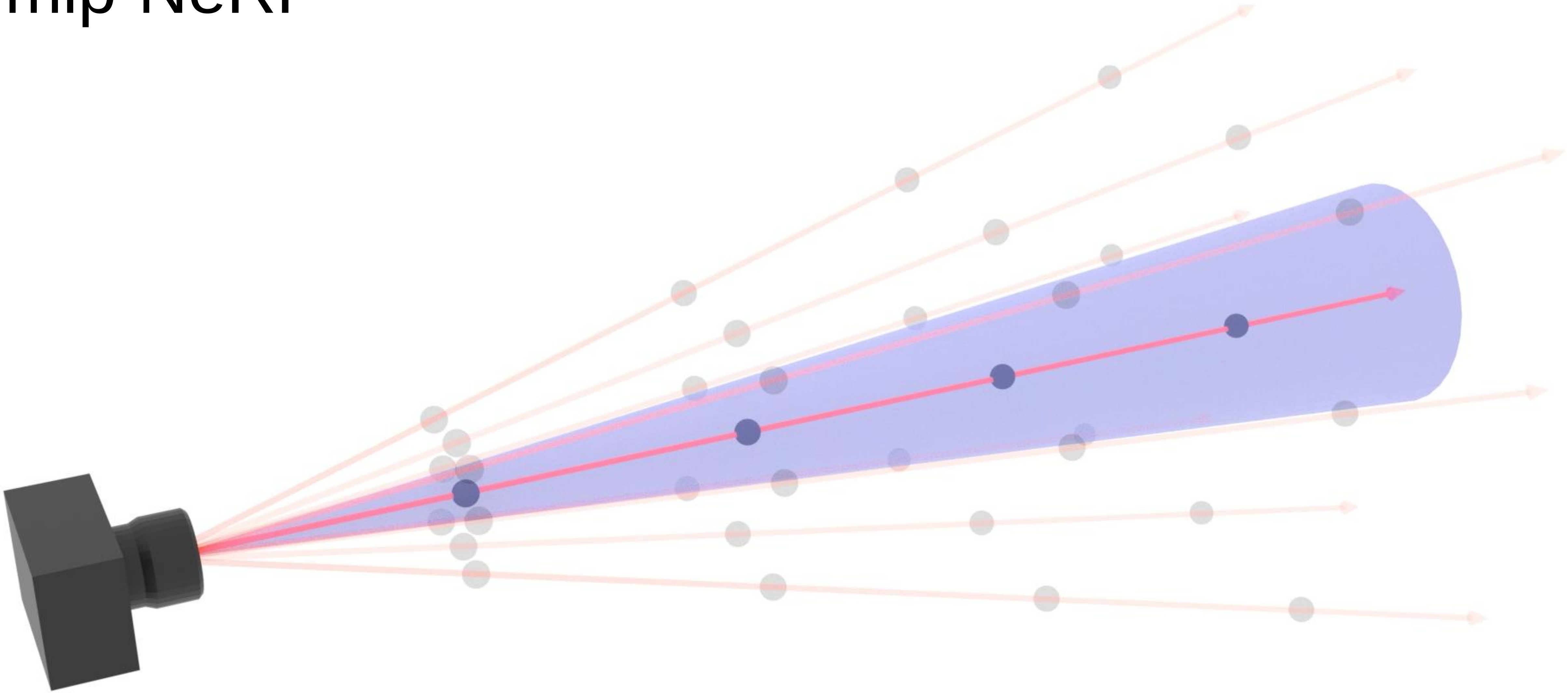
mip-NeRF



mip = "multum in parvo", Latin for "much in little"

Slide credit: Jon Barron

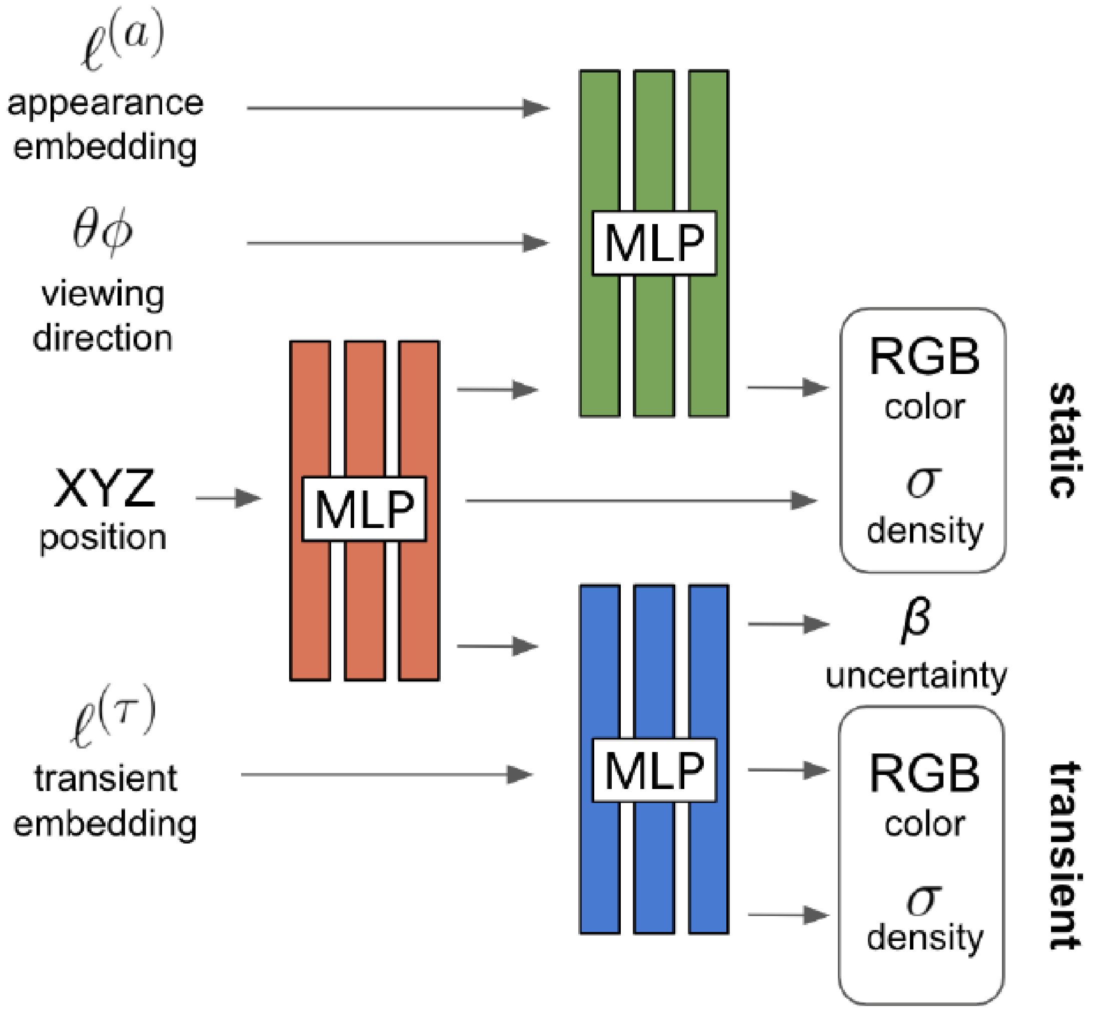
mip-NeRF



Slide credit: Jon Barron

NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections

Ricardo Martin-Brualla,* Noha Radwan,* Mehdi S. M. Sajjadi,*
Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth



NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections

Ricardo Martin-Brualla*, Noha Radwan*, Mehdi S. M. Sajjadi*,
Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth

(a) Static (b) Transient (c) Composite (d) Image (e) Uncertainty

Figure 4: NeRF-W separately renders the static (a) and transient (b) elements of the scene, and then composites them (c). Training minimizes the difference between the composite and the true image (d) weighted by uncertainty (e), which is simultaneously optimized to identify and discount anomalous image regions. Photo by Flickr user vasnic64 / CC BY.

NeRF summary

- Solves for functional mapping of position to occupancy and position/view to color

- Produces geometry/reflectance estimates that are good for interpolating views and robust to non-Lambertian surfaces

- Many follow-on works for efficient learning/storing/rendering, extending applicable settings, and manipulations

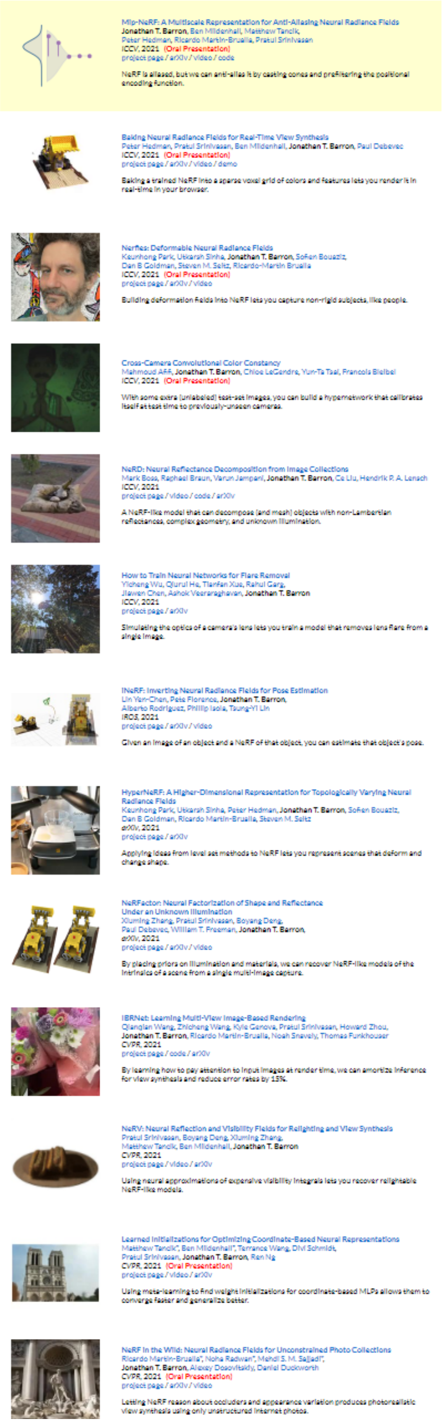
- Photometric objective and volumetric implicit surface function may not be ideal for estimating geometry in large scenes

Be wary of starting NeRF extension research

- Sooo much work on this, so fast: https://github.com/yenchenlin/awesome-NeRF

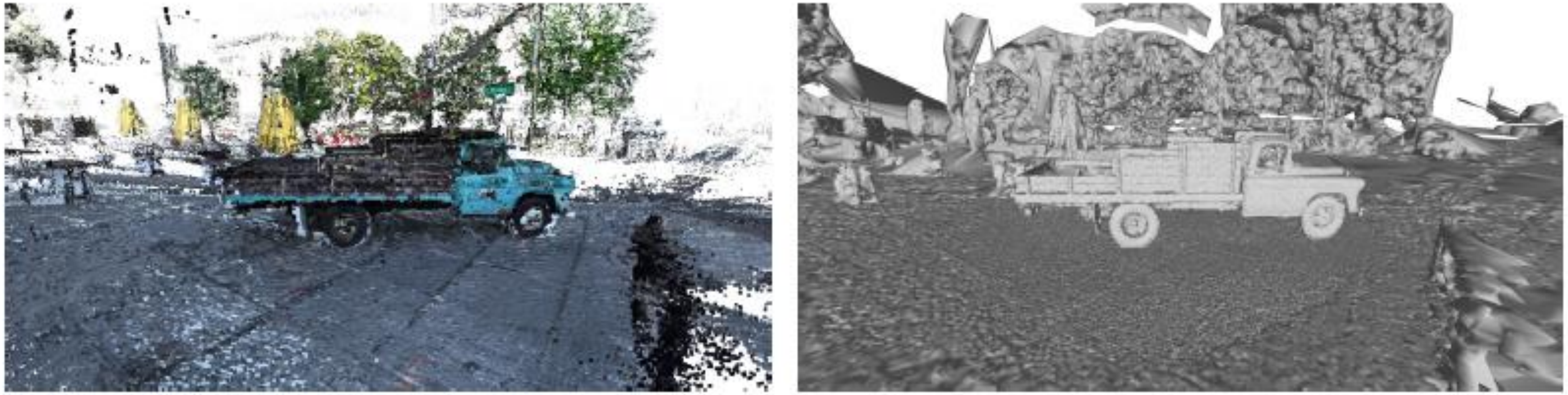
- Other people, like original authors, have a big head start

2021 NeRF papers co-authored by Jon Barron



Free view synthesis (Riegler and Koltun ECCV 2020)

- Start with mesh
  - SfM + MVS + DT/GC mesh (all in COLMAP codebase)

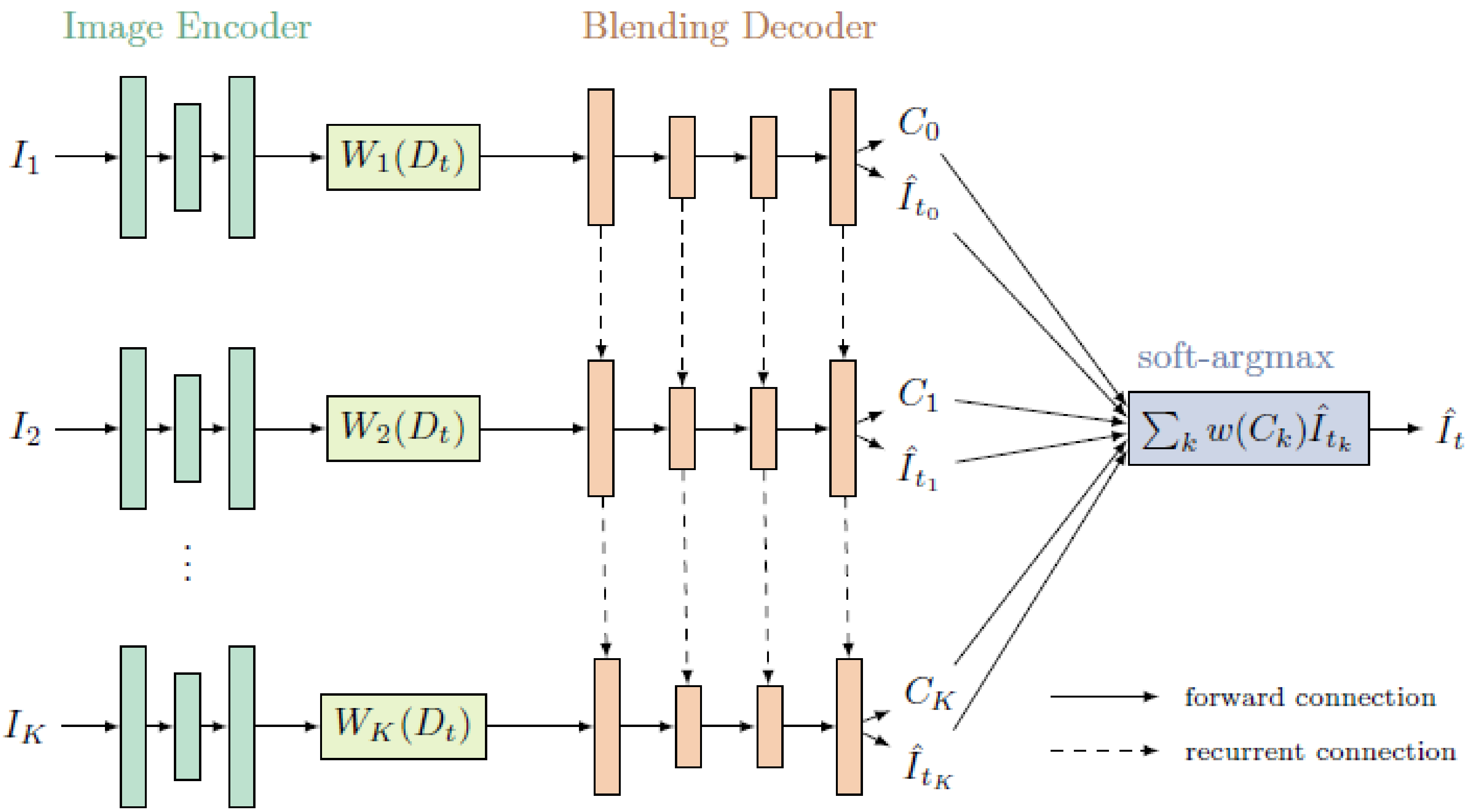


(a) Point cloud (b) Mesh

- Learn to select/blend/generate colors based on projected features from source views

Free view synthesis

1. Render mesh into target view to get its depth map D_t
2. For each source image:
   a. Extract features (using 3 stages of ImageNet pretrained VGG)
   b. Warp each pixel into each source view using D_t and get interpolated features
   c. Predict intensity $\hat{I}$ and confidence C images using blending decoder (UNet+GRU) for each source
   d. Store mask values for cases where mesh is missing or point doesn't project within source
3. Produce final intensity $\hat{I}$ and confidence C using blending decoder for each source



Training

- Minimize L1 distance to pixel intensities and VGG features of the true held out image

$$\mathcal{L}(\hat{I}_t, I_t) = ||\hat{I}_t - I_t||_1 + \sum_l \lambda_l ||\phi_l(\hat{I}_t) - \phi_l(I_t)||_1$$

- Train on 17 Tanks and Temple scenes in leave-one-image-out

Evaluation

Table 2: Results on Tanks and Temples. (Whole sequences withheld.)

	Truck			Train			M60			Playground		
	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR
EVS [8]	0.41	0.563	14.99	0.64	0.454	11.81	0.62	0.473	9.66	0.39	0.610	16.34
LLFF [26]	0.61	0.432	10.66	0.70	0.356	8.88	0.69	0.427	8.98	0.56	0.517	13.27
NeRF [27]	0.61	0.690	19.47	0.74	0.532	13.16	0.62	0.691	15.99	0.54	0.734	21.16
NPBG [2]	0.22	0.822	20.32	0.25	**0.801**	**18.08**	0.36	0.716	12.35	0.17	**0.876**	**23.03**
Our	**0.11**	**0.867**	**22.62**	**0.22**	0.758	17.90	**0.29**	**0.785**	**17.14**	**0.16**	0.837	22.03

Table 3: Quantitative results on the DTU dataset. Numbers on the left are for view interpolation, numbers on the right are for extrapolation.

	Scan 65			Scan 106			Scan 118		
	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR
EVS [8]	0.61/0.53	0.938/0.917	23.07/21.23	0.75/0.53	0.903/0.880	19.95/18.62	0.47/0.42	0.931/0.911	23.00/20.47
LLFF [26]	0.51/0.44	0.939/0.926	22.44/22.04	0.61/0.39	0.907/0.893	24.08/24.61	0.47/0.30	0.932/0.929	28.95/27.40
NeRF [27]	**0.17**/0.32	**0.987/0.963**	**34.41/27.81**	0.36/0.40	**0.973**/0.931	**34.52**/24.36	0.24/0.27	**0.985/0.952**	**37.16/28.39**
NPBG [2]	0.82/0.96	0.896/0.839	17.77/15.59	0.94/0.53	0.856/0.879	20.70/22.54	0.74/0.41	0.876/0.905	24.10/24.97
Our	0.25/**0.30**	0.972/0.950	26.96/24.08	**0.25/0.26**	0.963/**0.938**	27.24/**24.63**	**0.16/0.20**	0.975/0.951	29.21/25.75

Evaluation

Table 1: Evaluation of architectural choices on the Tanks and Temples dataset. (Leave-one-out protocol.) See the text for a detailed description of the conditions.

	Truck			Train			M60			Playground		
	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR
Fixed Identity	0.116	0.819	21.22	0.201	0.751	18.53	0.110	0.871	22.67	0.119	0.824	22.38
Fixed Encoding	0.096	0.828	21.19	0.168	0.769	19.01	0.096	0.876	22.80	0.107	0.831	22.40
Cat Global Avg.	0.089	0.842	21.49	0.175	0.773	18.73	0.093	0.887	23.41	0.098	0.845	22.92
Ours w/o Encoding	0.093	0.849	**22.13**	0.174	0.778	19.33	0.094	0.887	23.79	0.099	0.851	23.45
Ours w/o GRU	0.094	0.845	21.74	0.159	0.782	19.26	0.087	0.893	23.49	0.095	0.849	23.30
Ours w/o Masks	0.087	0.847	21.58	0.152	0.784	19.42	0.082	**0.897**	24.07	0.087	0.850	23.16
Ours w/o inf. depth	0.093	0.847	21.94	0.169	0.782	18.96	0.087	0.896	**24.08**	0.094	0.853	23.47
Ours w/o soft-argmax	0.091	0.845	21.74	0.159	0.786	19.43	0.086	0.891	23.79	0.090	0.857	23.50
Ours full	**0.082**	**0.852**	22.03	**0.147**	**0.794**	**19.54**	**0.081**	0.894	23.98	**0.084**	**0.859**	**23.51**

Free View Synthesis

Gernot Riegler and Vladlen Koltun

ECCV 2020

Open problems / research ideas

- Making NeRF faster to train (see MVSNeRF)
- NeRF on large scale scenes
- In MVS, model intensity as a mix of diffuse and specular color and make photometric cost a function of diffuse color
- Use 360 images taken from various positions within a room to enable omnidirectional and omnipositional free view synthesis

Summary

- NeRF encodes a surface with diffuse and non-diffuse color components by mapping (x,y,z,direction) to (density, r,g,b)
  - Numerous follow-on works improve the rendering time, model size, training time, ability to handle occlusions, special effects, and more

- Free view synthesis achieves results that are sometimes better than NeRF by using an MVS-derived mesh to map and blend features

- Both offer spectacular results