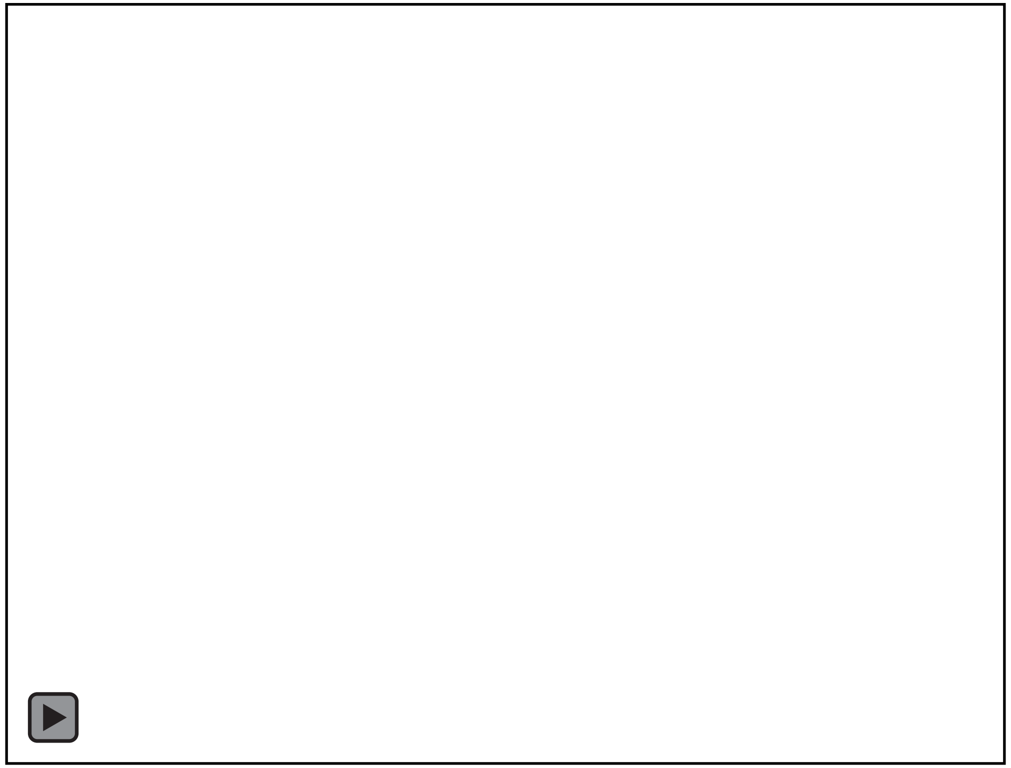
NeRF: Neural Radiance Fields



Computational Photography

Derek Hoiem, University of Illinois

This class

- NeRF: recover 3D model with view-dependent rendering from multiple registered images

- RawNeRF: operate directly on raw images to achieve HDR, denoising, and defocus

- DreamFusion: generate 3D models from text

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

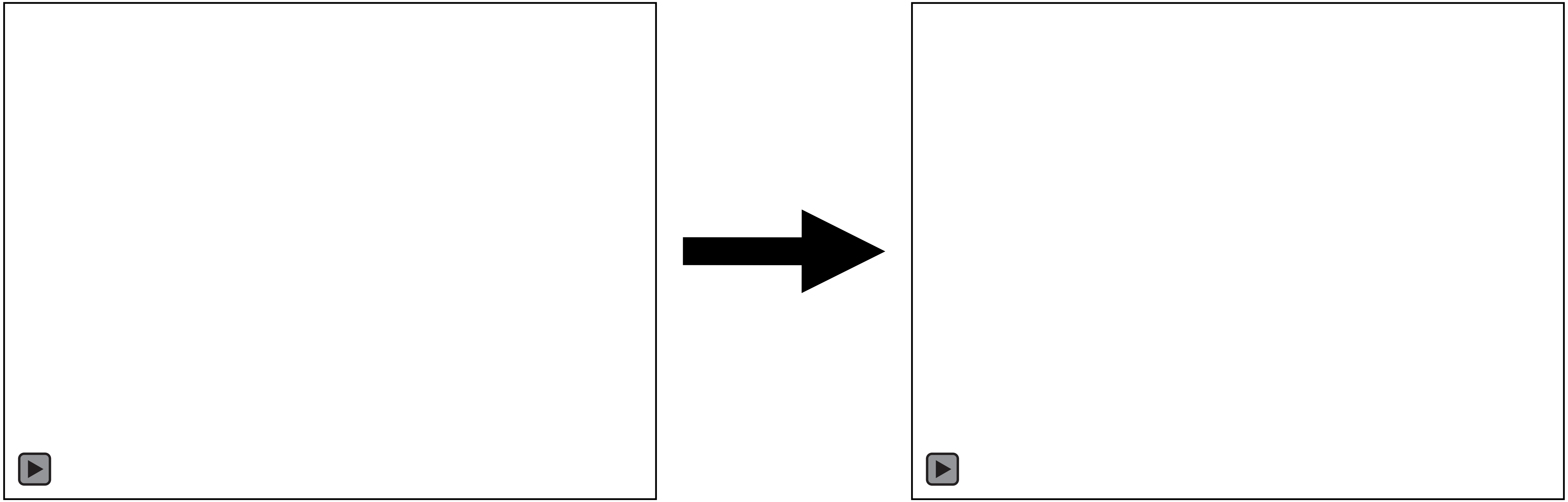
Ravi Ramamoorthi

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

NeRF (neural radiance fields)

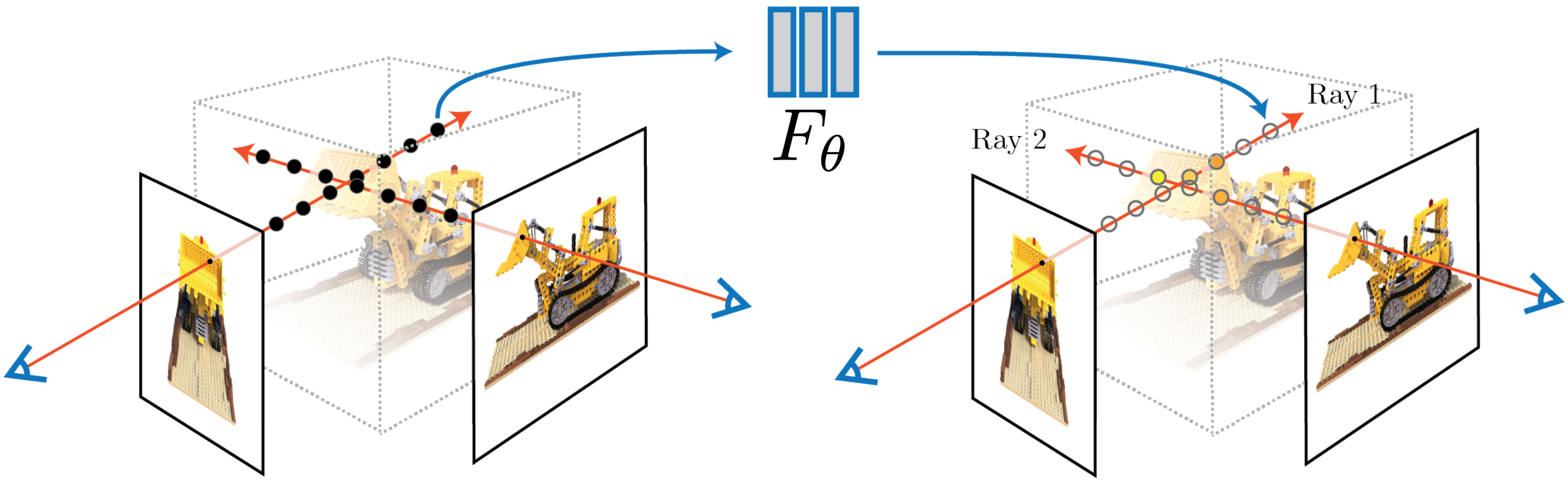
$$(\underbrace{x, y, z}_{\text{Spatial location}}, \underbrace{\theta, \phi}_{\text{Viewing direction}}) \rightarrow F_\theta \rightarrow (\underbrace{r, g, b}_{\text{Output color}}, \underbrace{\sigma}_{\text{Output density}})$$

F_θ

Fully-connected neural network
9 layers,
256 channels

Slide credit: Jon Barron

Volume rendering averages over colors along each ray



Slide credit: Jon Barron

Volume rendering is differentiable

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

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

weights

colors

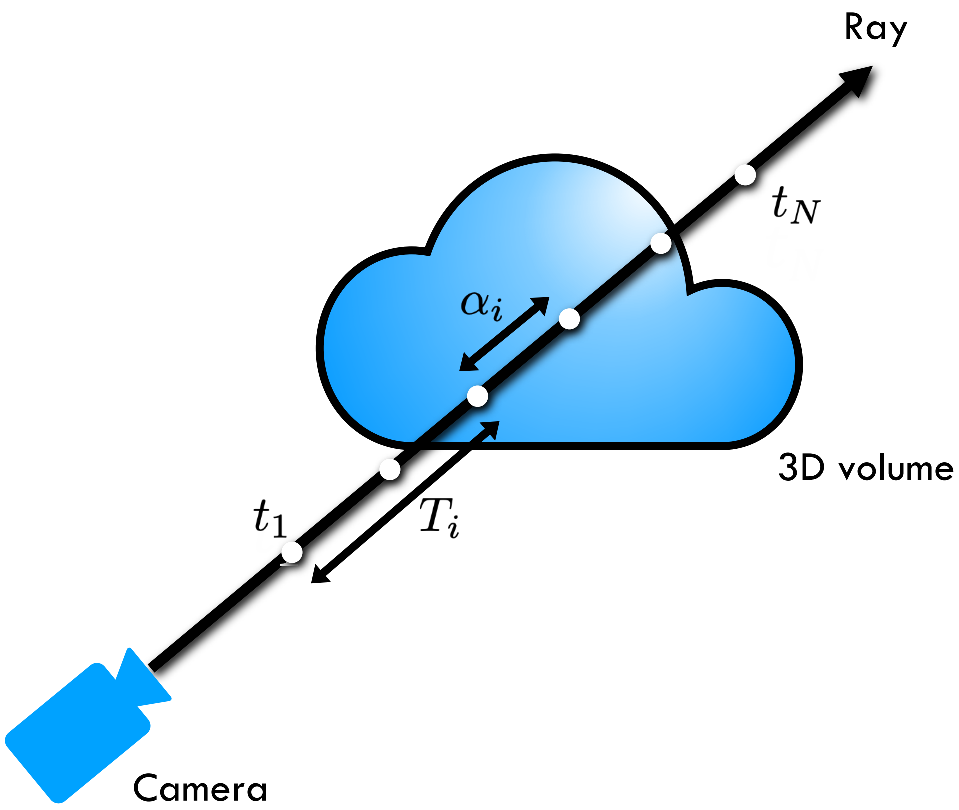
How much light is not blocked earlier along ray:

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

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

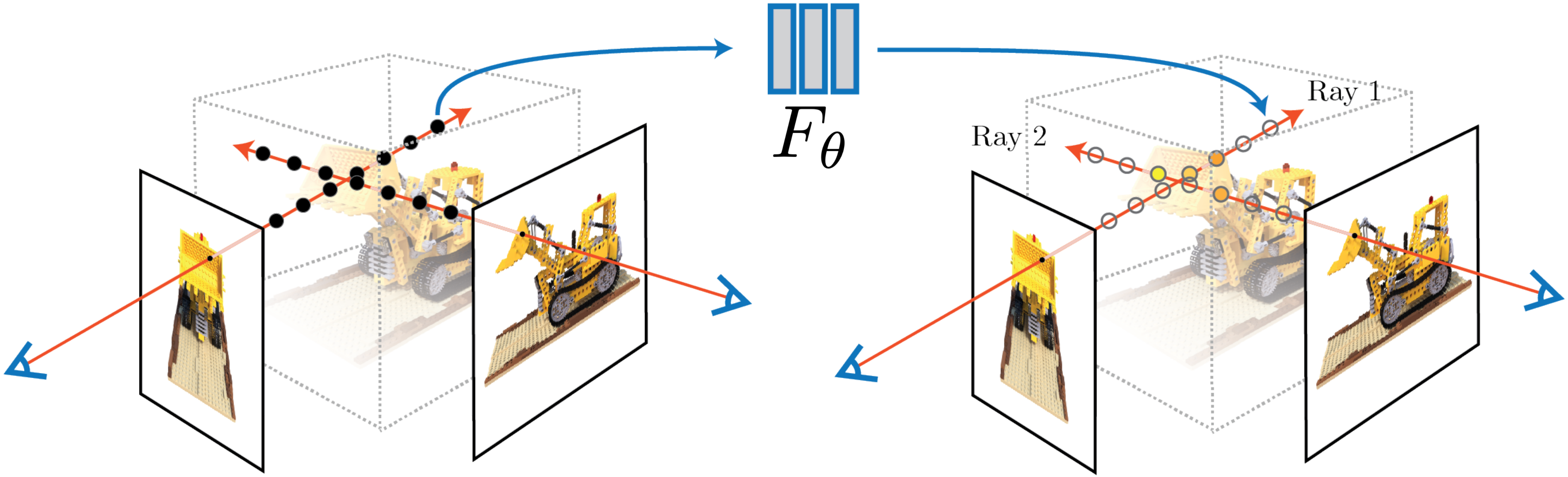
$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$

Density * Distance Between Points



Slide credit: Jon Barron

Optimize with gradient descent on rendering loss



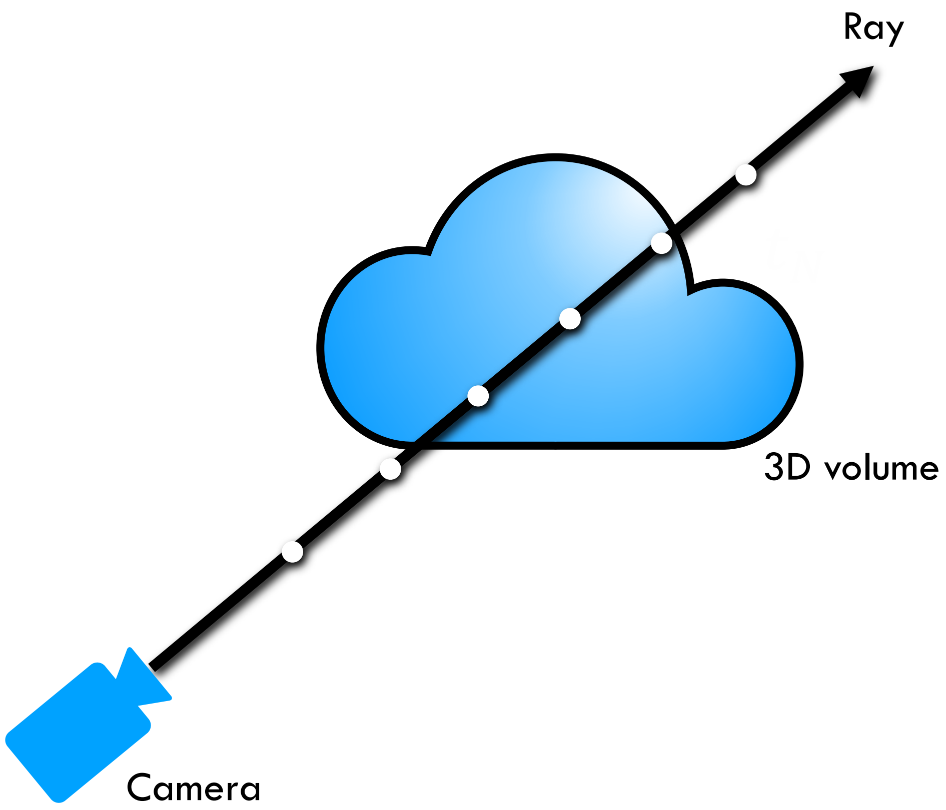
$$\min_{\theta} \sum_{i} ||\text{render}_i(F_\theta) - I_i||^2$$

Slide credit: Jon Barron

Training network to reproduce all input views of the scene

Slide credit: Jon Barron

Do two-pass rendering to focus samples near surfaces

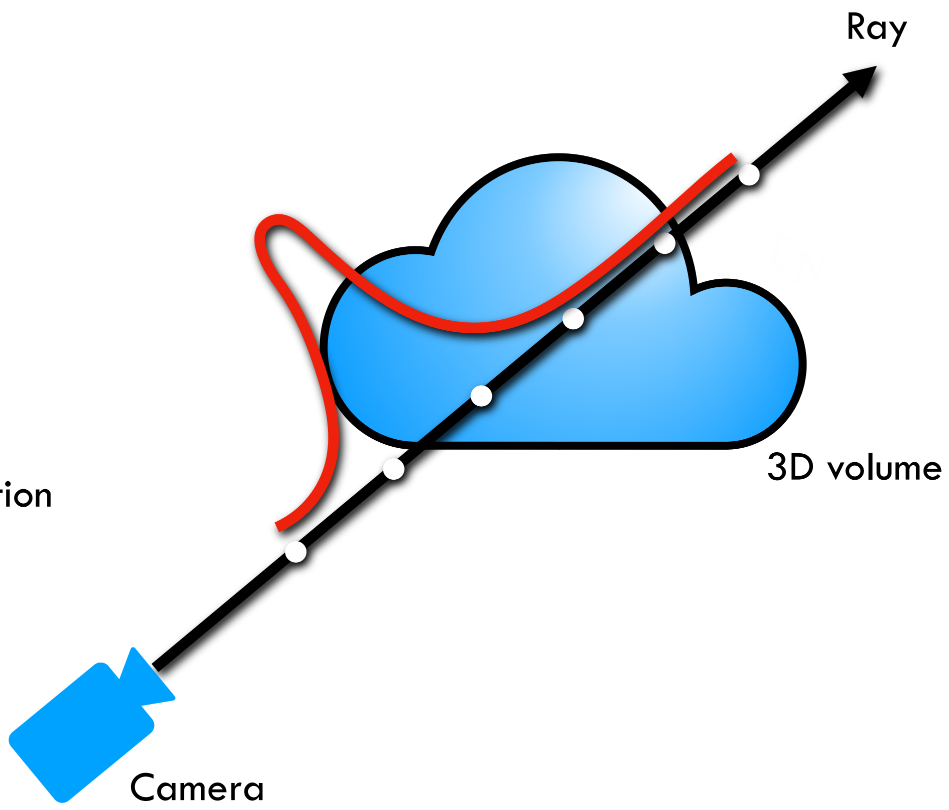


Slide credit: Jon Barron

Two pass rendering: coarse

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

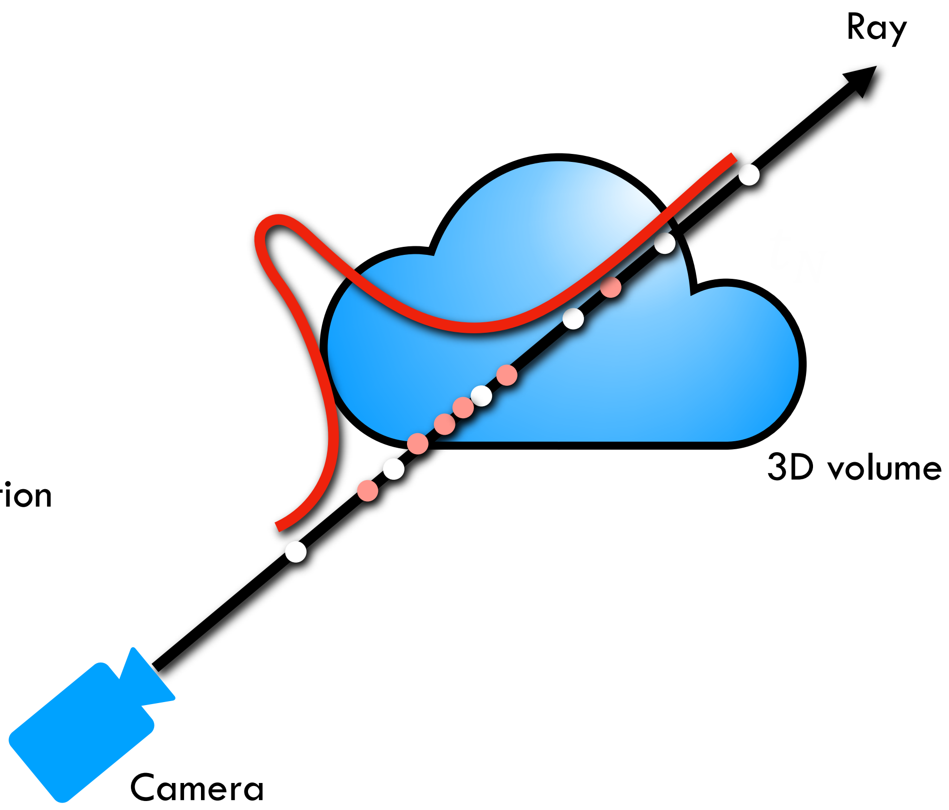
treat weights as probability distribution
for new samples



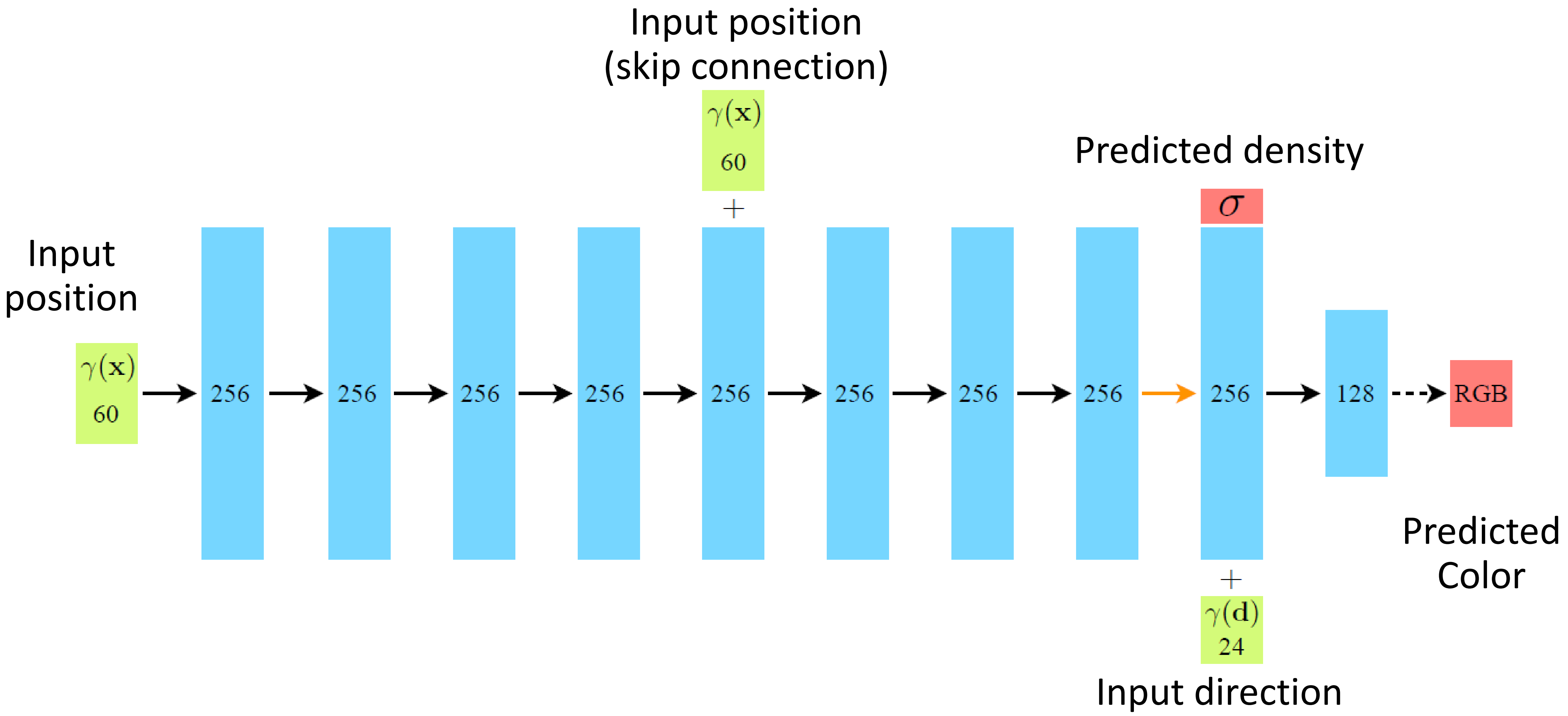
Two pass rendering: fine

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

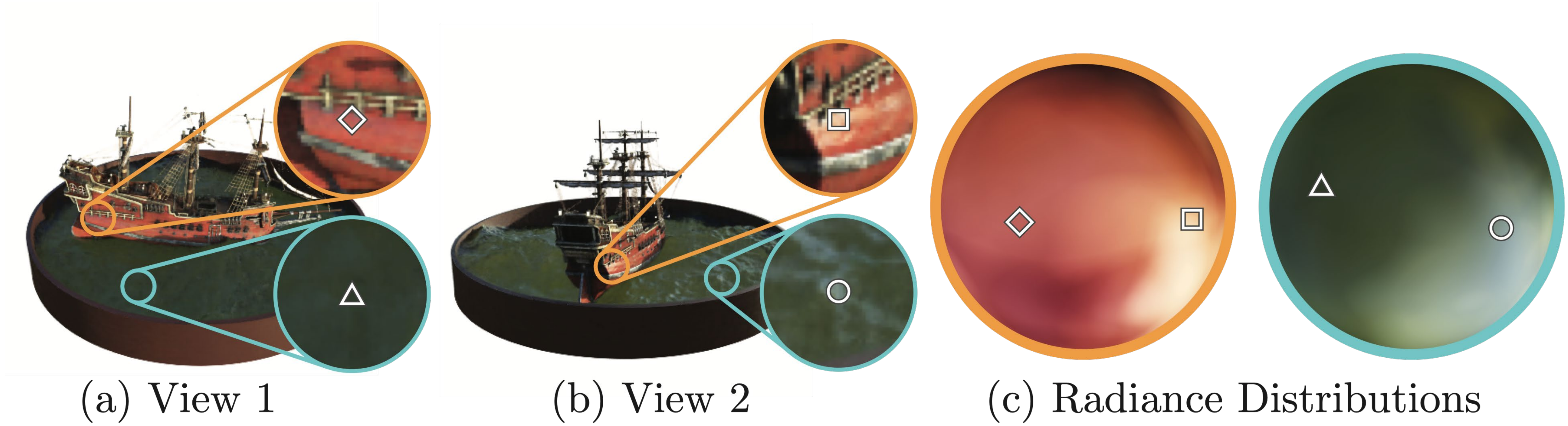
treat weights as probability distribution
for new samples



Network Structure is Dense Multilayer Perceptron

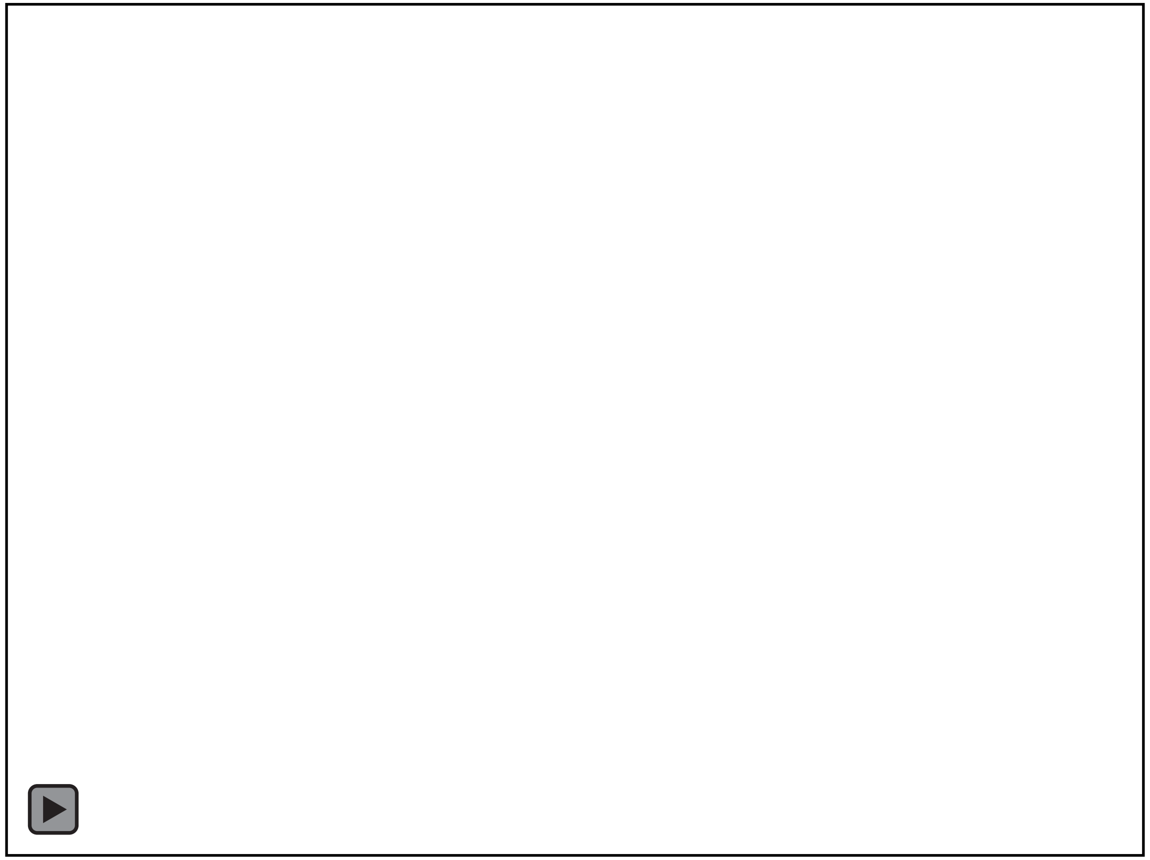


Predicted color depends on the input



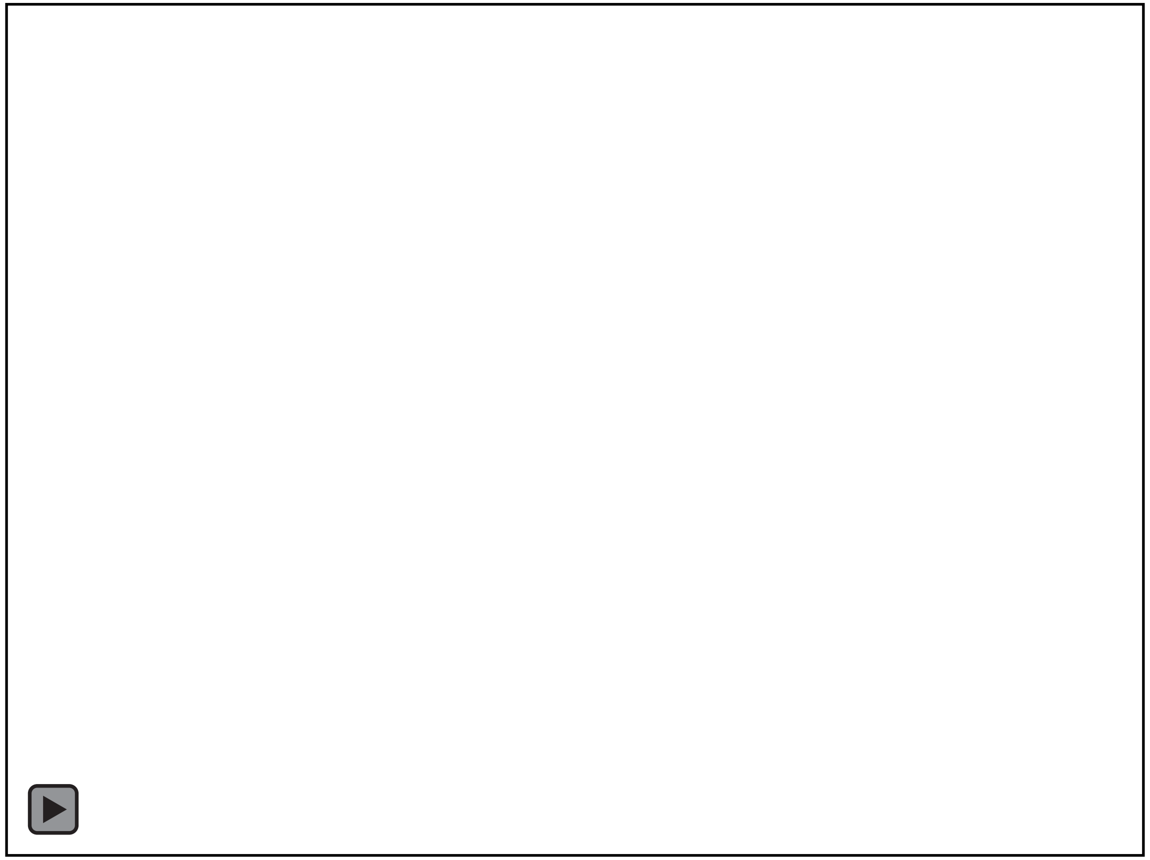
Slide credit: Jon Barron

Naive implementation produces blurry results

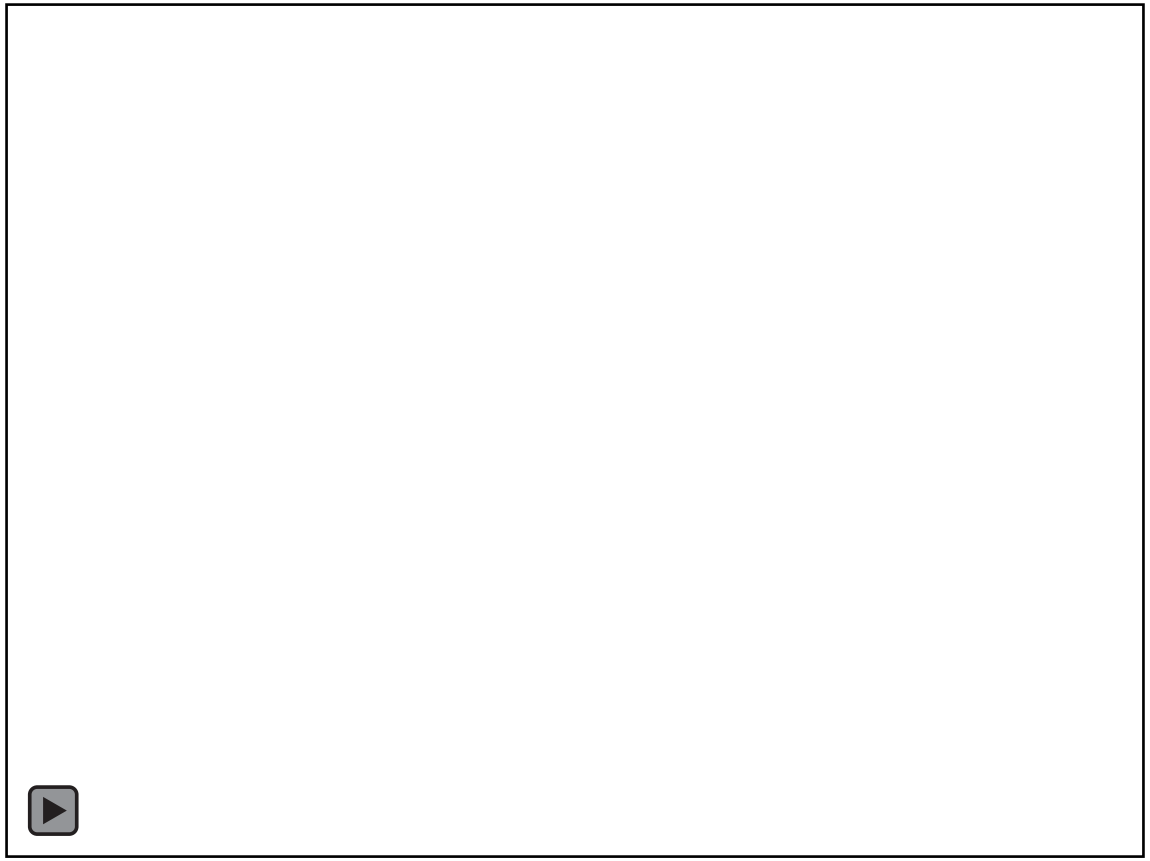


NeRF (Naive)

Naive implementation produces blurry results



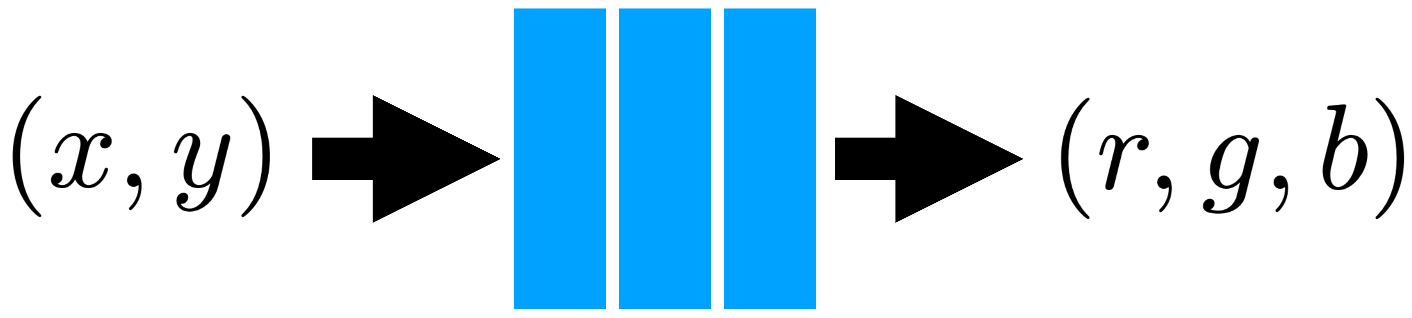
NeRF (Naive)



NeRF (with positional encoding)

Slide credit: Jon Barron

Toy problem: memorizing a 2D image



Slide credit: Jon Barron

Toy proble

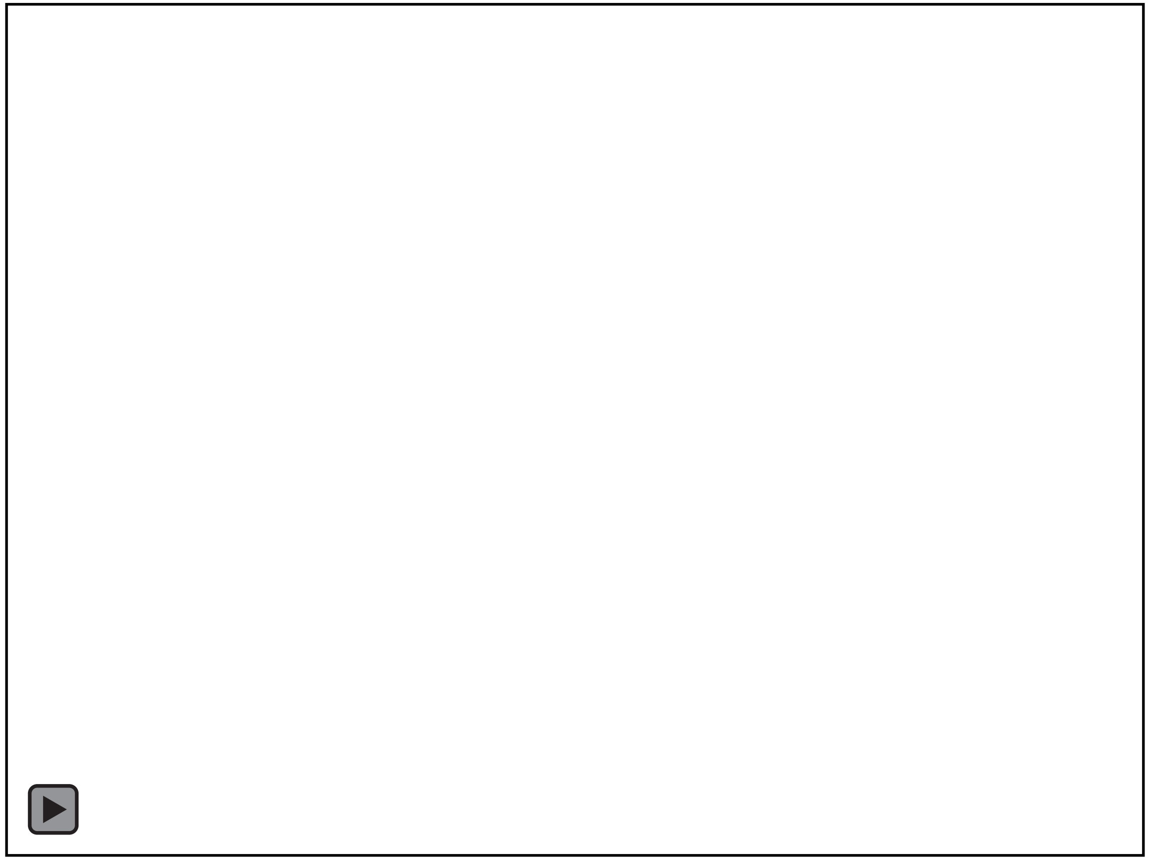
Ground truth image

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

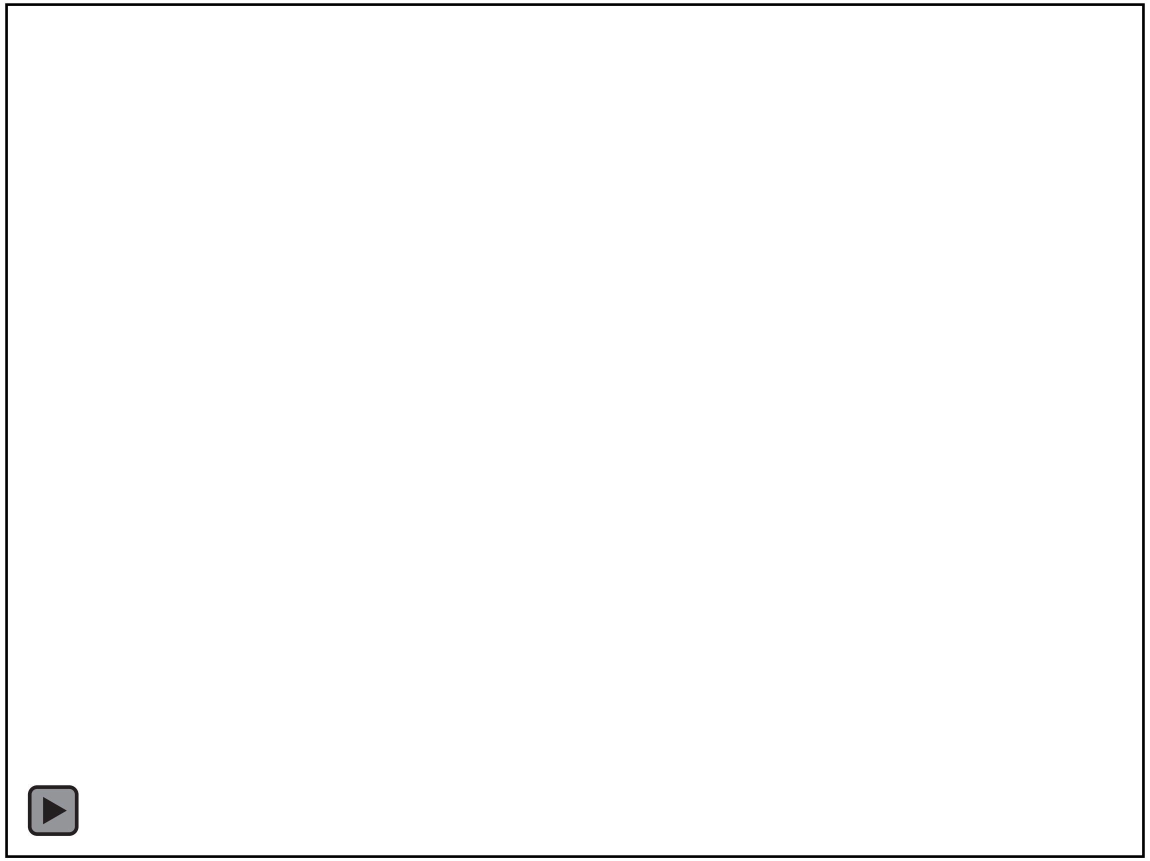
Slide credit: Jon Barron

Ground truth image

Positional encoding also directly improves the 3D scene representation



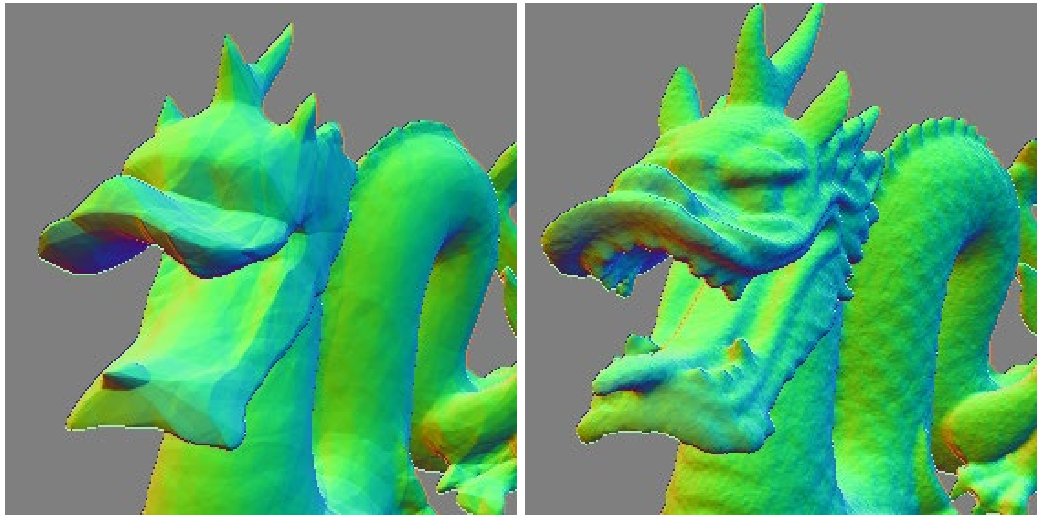
NeRF (Naive)



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

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

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

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

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

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

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

Fourier features turn a dot product of v1 and v2 into a shift-invariant function of their distance

MLP of Fourier-encoded positions interpolates values between positions

Mapping bandwidth controls underfitting / overfitting

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

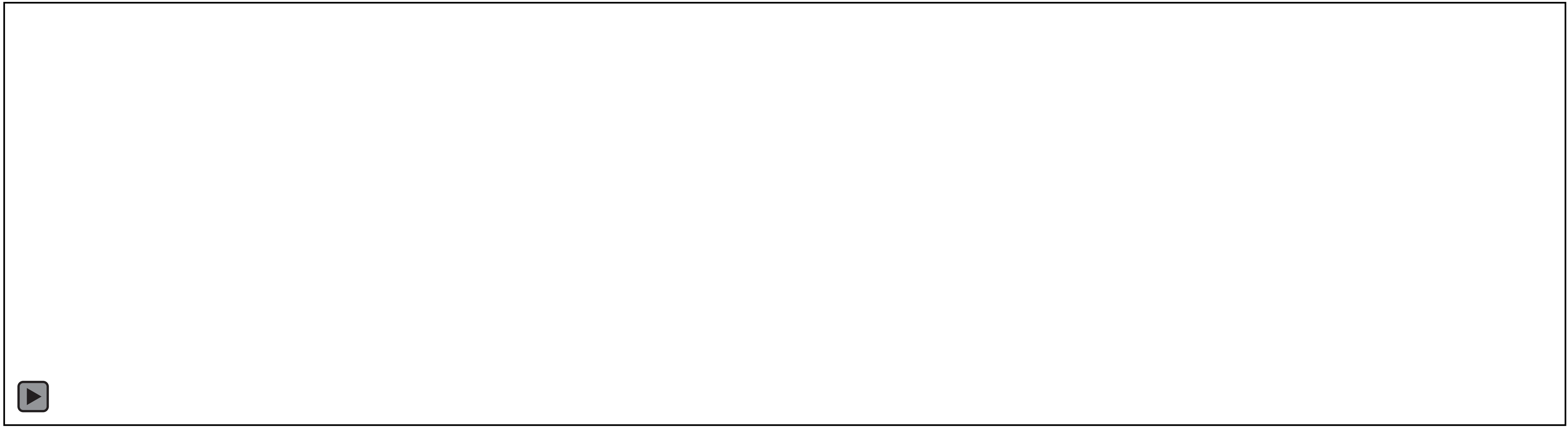
Slide credit: Jon Barron

Mapping bandwidth controls underfitting / overfitting

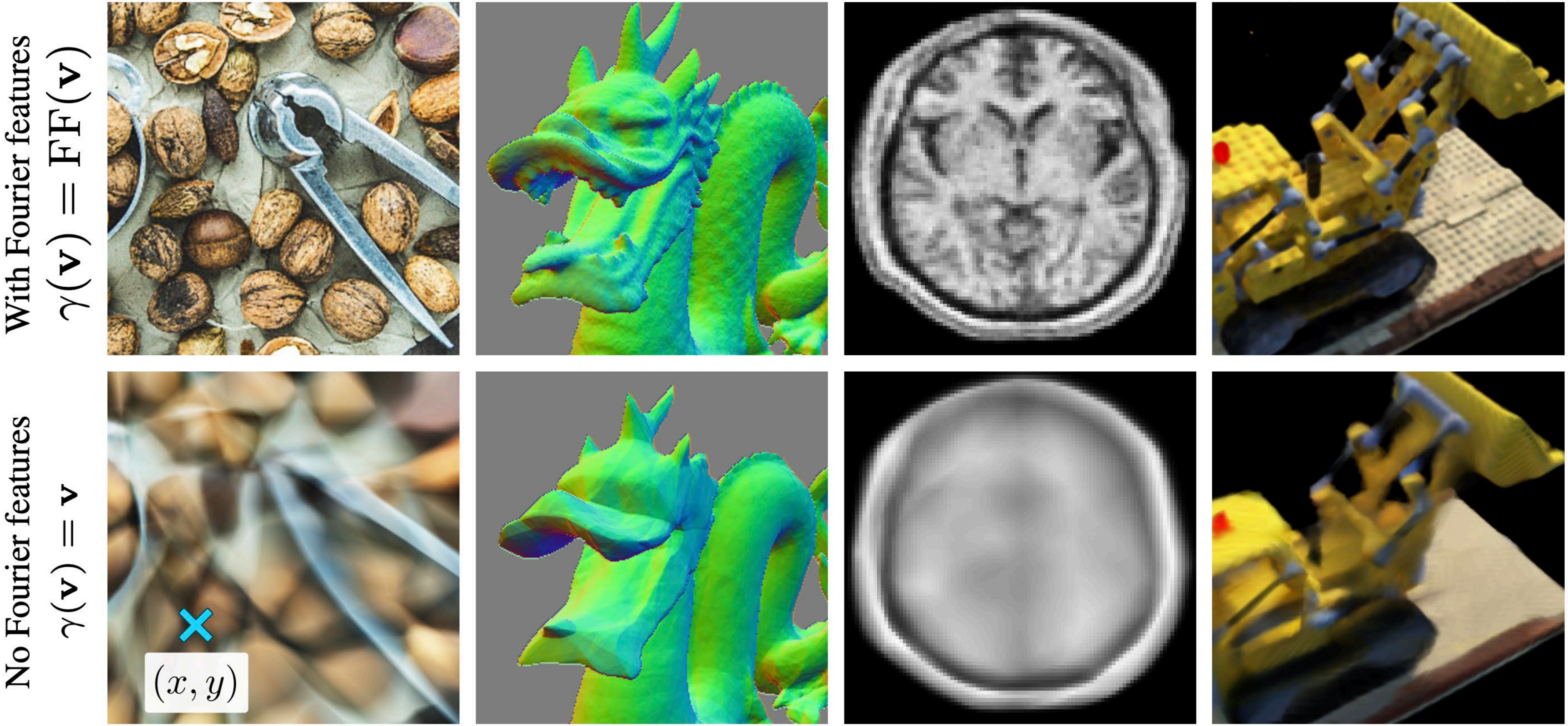
$$\gamma(\mathbf{v}) = [\cos(\mathbf{B}\mathbf{v}), \sin(\mathbf{B}\mathbf{v})] \qquad \mathbf{B} \sim \mathcal{N}(0, \boxed{\sigma}^2)$$

Slide credit: Jon Barron

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
$(x,y) \to$ RGB

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

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

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

Slide credit: Jon Barron

Very simple!

```
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

View-Dependent Effects

Detailed Geometry & Occlusion

Detailed Geometry & Occlusion

Meshable

MonoSDF (2022)

- Solve for features in multi-resolution grid and pass into MLP w/ positional encoding
- Use per-view surface normal prediction as constraint
- Solve SDF (distance to surface)

MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction

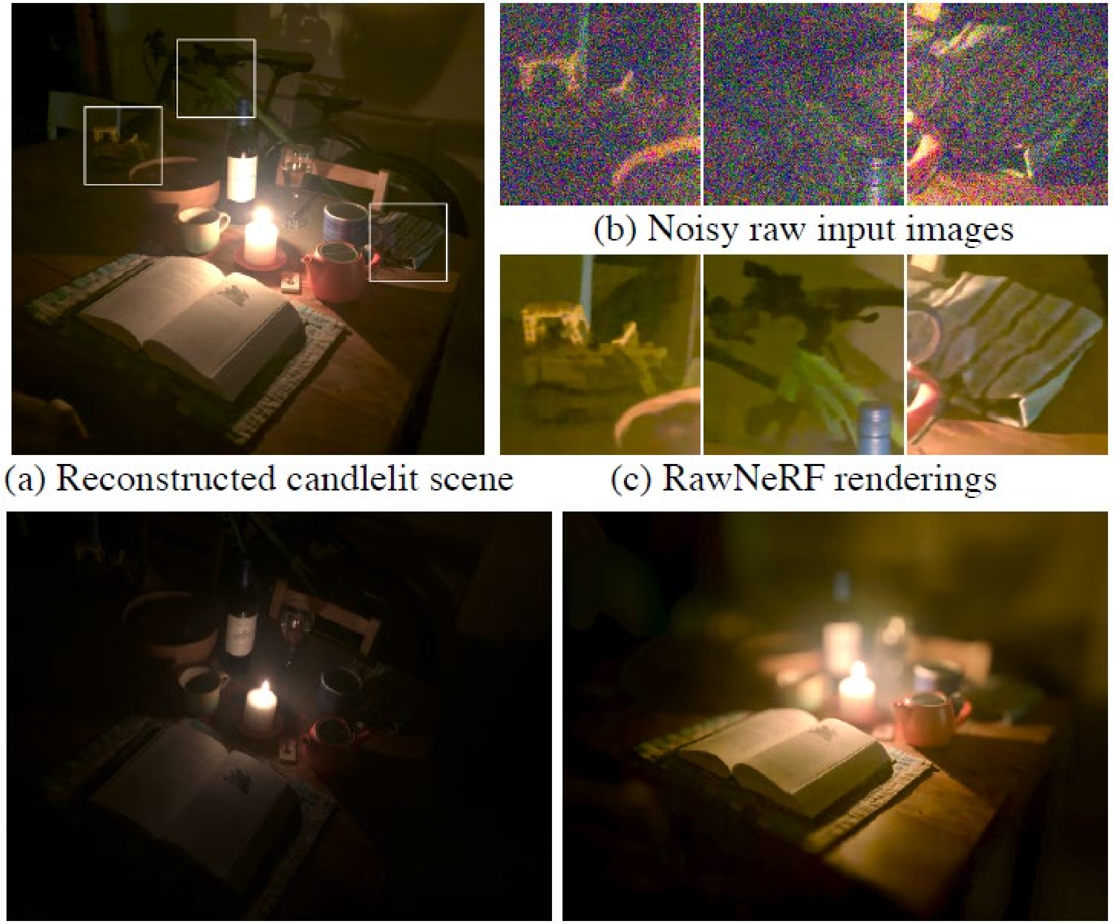
Zehao Yu[1] Songyou Peng[2,3] Michael Niemeyer[1,3] Torsten Sattler[4] Andreas Geiger[1,3]

NeRF in the Dark:

High Dynamic Range View Synthesis from Noisy Raw Images

Ben Mildenhall Peter Hedman Ricardo Martin-Brualla Pratul P. Srinivasan Jonathan T. Barron

CVPR 2022



(a) Reconstructed candlelit scene

(b) Noisy raw input images

(c) RawNeRF renderings

(d) Changing viewpoint, focus, exposure, and tonemapping

https://www.youtube.com/watch?v=JtBS4KBcKVc

Measurement noise is complicated by camera processes

1. Raw camera measurement: photons converted to electrical charge, recorded in 10-14 bits
   - Black level subtraction removes noise effects
   - Gaussian measurement noise

2. Color filter demosaicking
   - Color light captured in Bayer pattern
   - Interpolation completes RGB values

3. Color correction and white balance
   - Convert from camera color space to standard RGB
   - White balance to remove tint due to lighting

4. Gamma compression and tone mapping
   - Gamma compression to shift toward darker values with higher human sensitivity
   - Preserve contrast when mapping to 8 bits

Improves viewing quality but complicates noise model and discards information

RawNeRF Method

- COLMAP on jpg images is used for camera registration

- NeRF operates on raw images

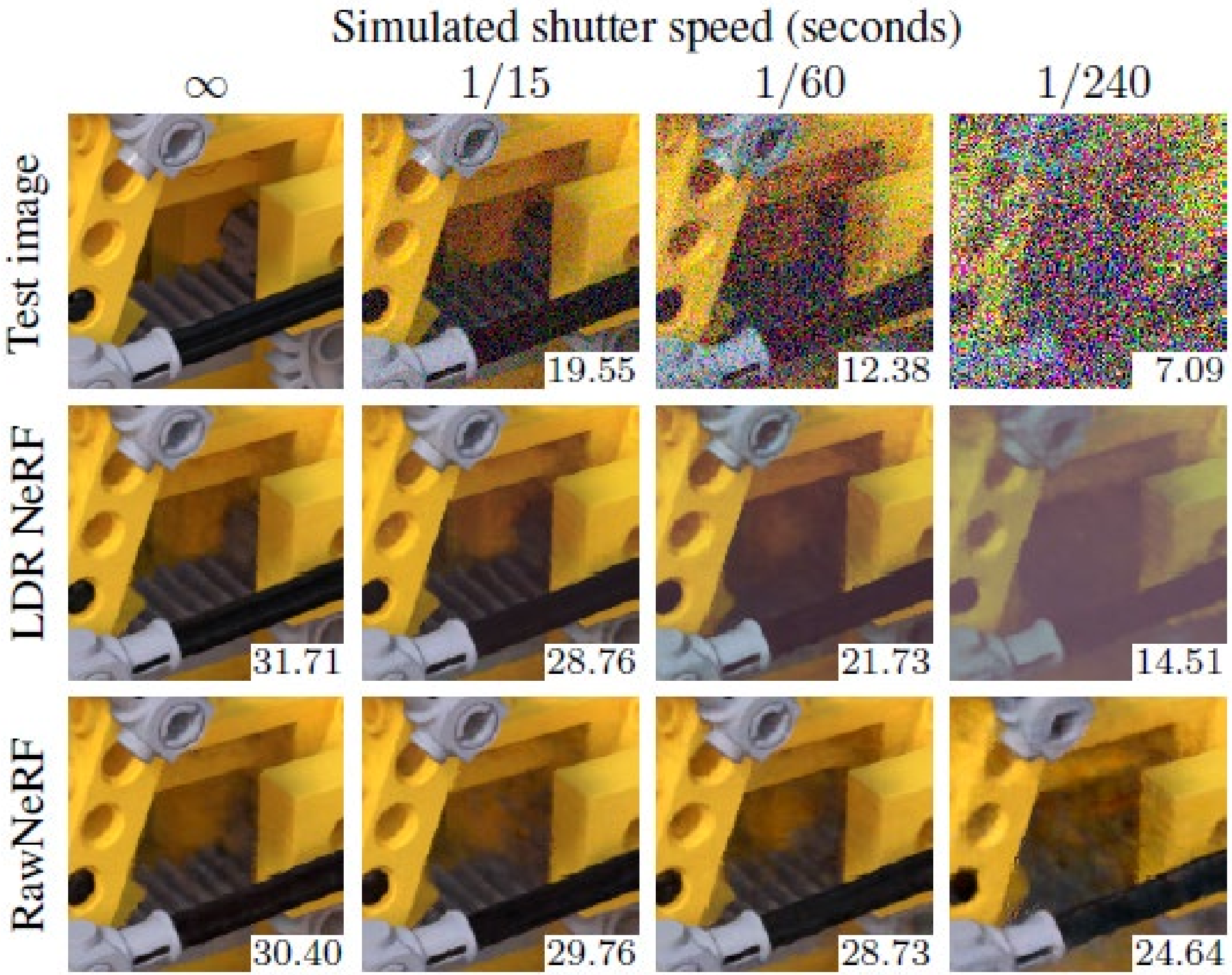
  – Loss emphasizes darker pixels, since HDR values can get very high

  $$\mathrm{L}(\hat{y}, y) = \sum_i w_i(\hat{y}_i - y_i)^2 \qquad w_i = 1/(\hat{y}_i + \epsilon)$$

  – Optimize over exposure, accounting for shutter time

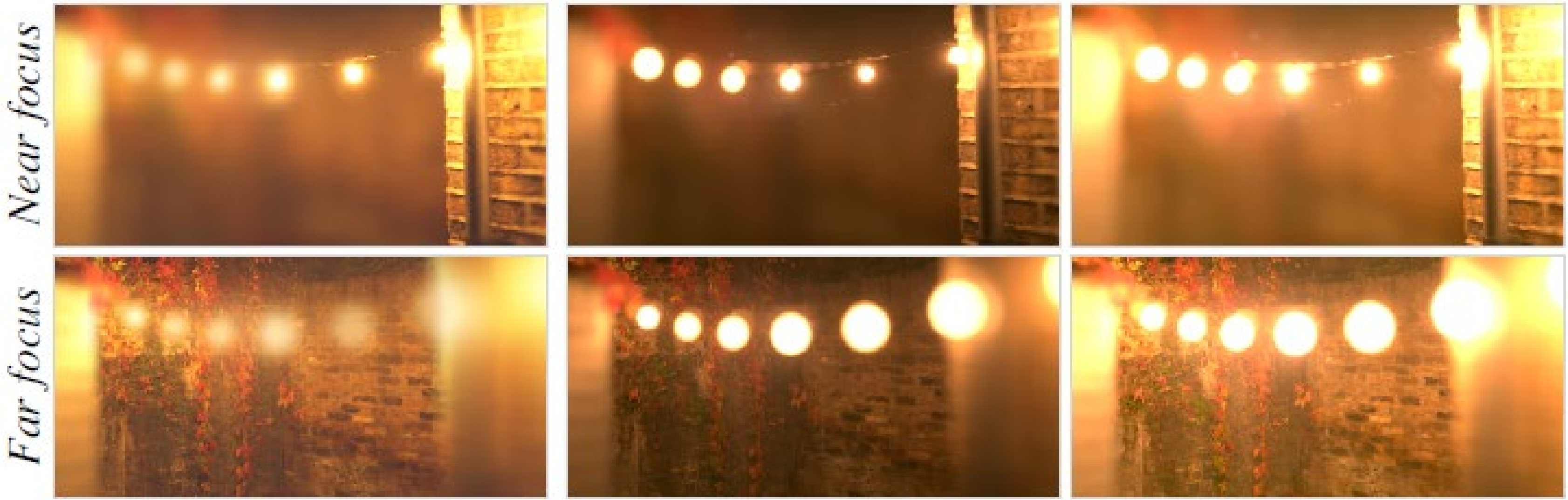
  – Use regular NeRF architecture, except final layer is exponential instead of sigmoid

RawNeRF handles noise better



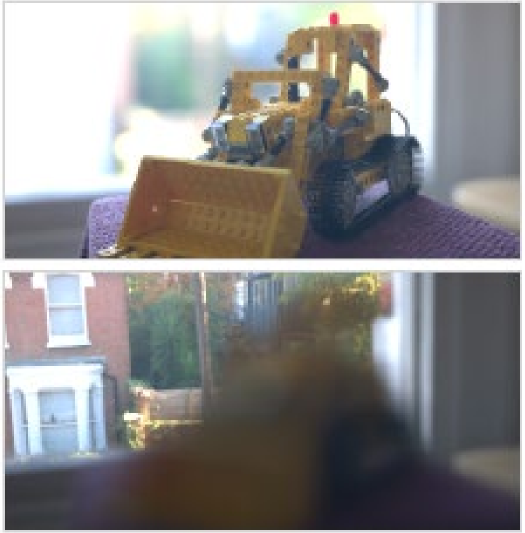
Method	Simulated shutter speed (seconds) ∞	1/7	1/15	1/30	1/60	1/120	1/240
Noisy input	-	23.33	19.65	16.03	12.51	9.40	7.18
LDR NeRF	**33.16**	31.25	29.14	26.10	22.31	18.27	14.87
RawNeRF	32.15	**32.11**	**31.94**	**31.59**	**30.94**	**29.69**	**27.73**

Synthetic re-focusing benefits from HDR and 3D model



(b) LDR NeRF defocus (c) RawNeRF defocus and exposure variation

(a) Full RawNeRF output



(d) Seeing behind objects

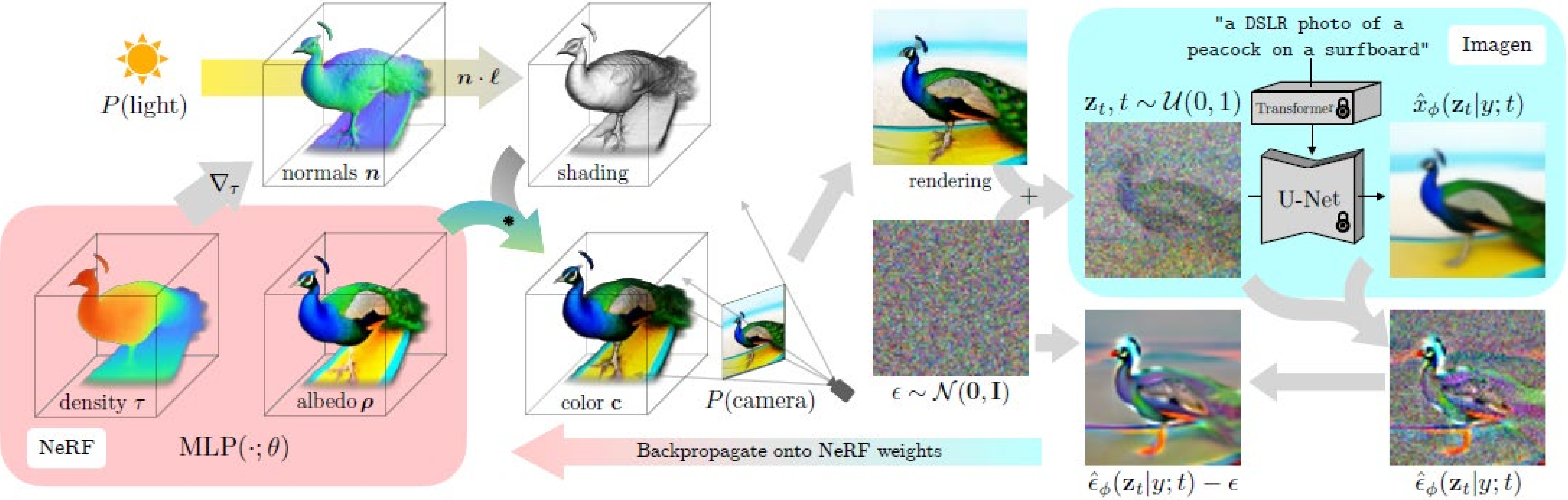
(e) Revealing reflections

Noisy test image | RawNeRF rendering | New viewpoint, HDR tonemapping

DreamFusion: Text-to-3D using 2D Diffusion

Ben Poole[1], Ajay Jain[2], Jonathan T. Barron[1], Ben Mildenhall[1]

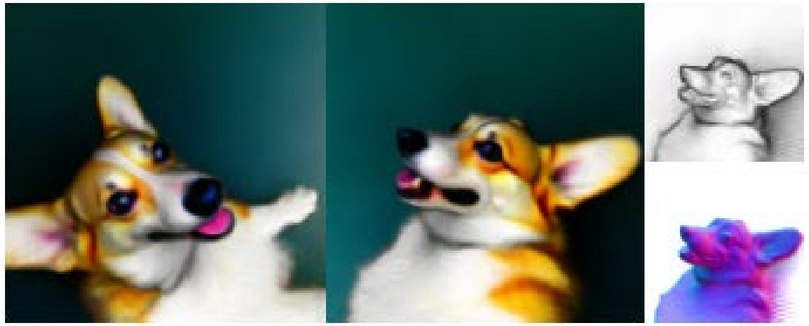
1. **Input**: “a DSLR photo of a peacock on a surfboard, randomly initialized NeRF model, Imagen generator
2. **Generate with Nerf:** albedo and surface normal image, compute shading, render final color
3. **Correct with Imagen**: the generated image (plus noise), subtract noise from delta, and provide as feedback to NeRF



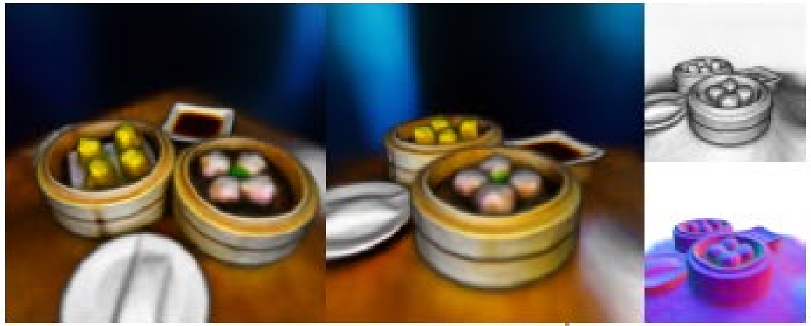
an orangutan making a clay bowl on a throwing wheel*

a raccoon astronaut holding his helmet†

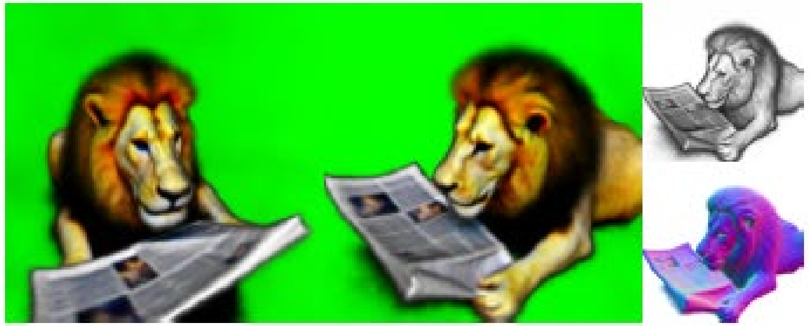
a blue jay standing on a large basket of rainbow macarons*



a corgi taking a selfie*



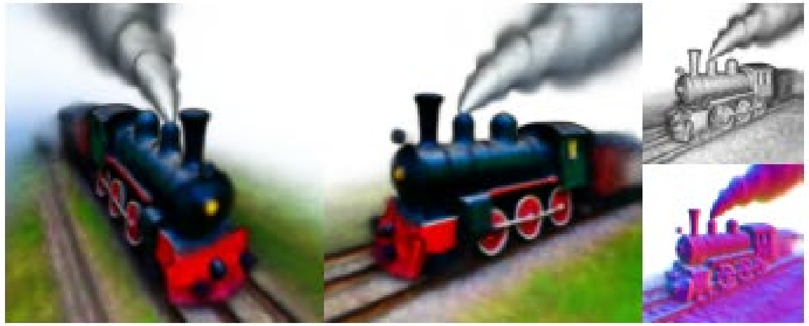
a table with dim sum on it†



a lion reading the newspaper*

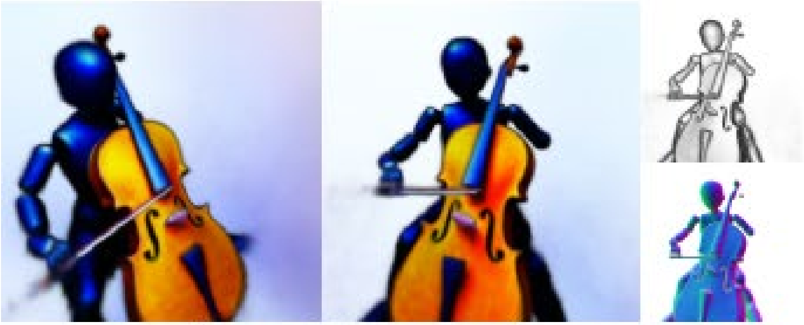
Michelangelo style statue of dog reading news on a cellphone

a tiger dressed as a doctor*



a steam engine train, high resolution*

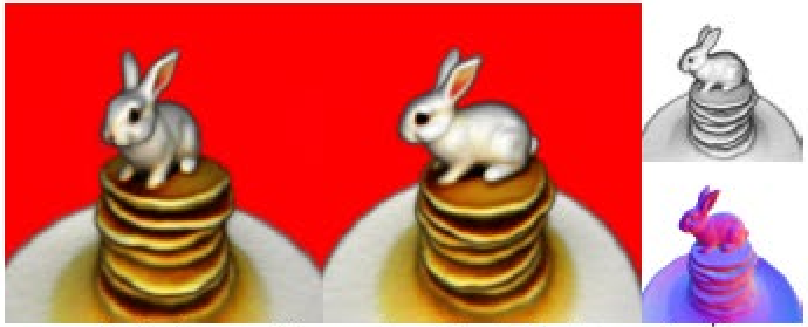

a frog wearing a sweater*


a humanoid robot playing the cello*


Sydney opera house, aerial view†

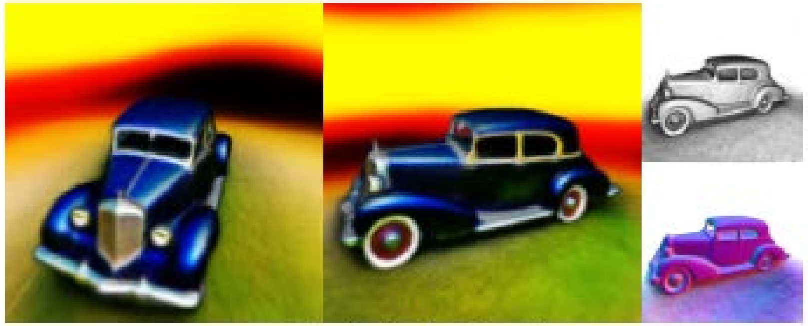

an all-utility vehicle driving across a stream†


a chimpanzee dressed like Henry VIII king of England*


a baby bunny sitting on top of a stack of pancakes†


a sliced loaf of fresh bread

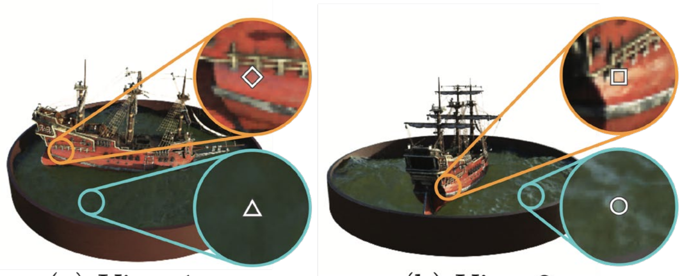

a bulldozer clearing away a pile of snow*


a classic Packard car*

https://dreamfusion3d.github.io/index.html

Summary

- NeRF encodes a surface with diffuse and non-diffuse color components by mapping (x,y,z,direction) to (density, r,g,b)



- RawNeRF operates directly in raw measurement space, achieving denoising, HDR, and refocus effects

- DreamFusion combines image generation with NeRF to create 3D models from text

a raccoon astronaut holding his helmet†

Coming up

- Enjoy Fall break next week!
- Project 5 due Monday, Nov 28