CMSC740
Advanced Computer Graphics

Matthias Zwicker

Fall 2025

Scene reconstruction from images

- Inverse rendering problem: given images of static scene from different viewpoints, camera parameters; reconstruct scene parameters, such that rendered images match input
- Applications
  - Extract 3D representation from scene parameters
  - Novel view synthesis from arbitrary viewpoints (simply plug new camera parameters into f)
  - Re-lighting (required for most practical applications)
  - Material editing

Scene reconstruction from images

- Assume: have rendering function f

$$\text{image} = \boxed{f(\text{scene}, \text{camera})}$$

  - Scene parameters (scene): could be array of triangle vertices, light sources, BRDF parameters, neural network-based geometry representation, etc.

Approach

- Optimize scene parameters scene to minimize rendering loss l over all input images I

$$\arg\min_{\mathrm{scene}} \ell(\mathrm{scene})$$

$$\ell(\mathrm{scene}) = \sum_i ||\underbrace{\mathrm{image}_i}_{\text{Input image}} - \underbrace{\boxed{f(\mathrm{scene}, \mathrm{camera}_i)}}_{\text{Rendered image}}||$$

- Optimization using gradient descent

Challenges

- Suitable rendering function f
  - Need to compute gradients wrt. scene parameters

$$\frac{\partial \ell(\text{scene})}{\partial \text{scene}} = \frac{\partial \sum_i ||\text{image}_i - f(\text{scene}, \text{camera}_i)||}{\partial \text{scene}}$$

  - Automatic differentiation
- Suitable scene parameterization
  - Rendering function and scene representation need to be powerful enough to reproduce input images

NeRF

- "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis", Mildenhall et al, 2020 https://arxiv.org/abs/2003.08934
- Key ideas
  - Volumetric rendering function using volumetric emission and absorption
  - Volume density and emission (radiance) represented by neural networks
- Notes
  - Rendering function not physically-based, no model of light scattering on surfaces
  - Assumes mapping of input pixels to 3D rays is known (known camera location, orientation, field of view)
  - Doesn't directly recover 3D surfaces due to volumetric rendering model
  - Enables novel view synthesis (rendering scene from different camera viewpoints), but not re-lighting; NeRF was first highly successful method for novel view synthesis that is based on inverse rendering

Volumetric rendering function

- Color $C(\mathbf{r})$ of ray $\mathbf{r}(t) = \mathbf{o}+t\mathbf{d}$, origin $\mathbf{o}$, direction $\mathbf{d}$

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

- Transmittance $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$
- Density (scattering coefficient) $\sigma(\mathbf{r}(t))$
- Radiance (volumetric emission) $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$
- Neural network to represent both σ and $\mathbf{c}$

Volumetric rendering function

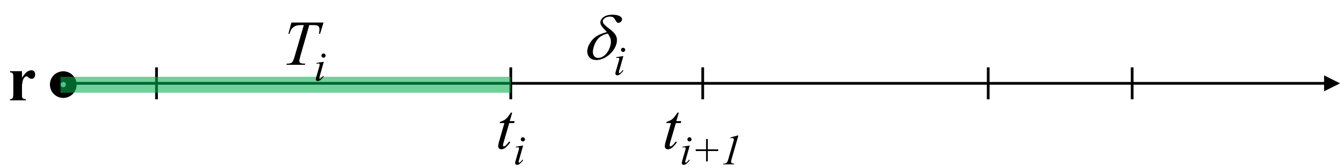
- Approximate integral using numerical quadrature (similar to ray marching)
  - N sample points along ray at locations t_i
  - Step size $\delta_i = t_{i+1} - t_i$

Probability to reach eye

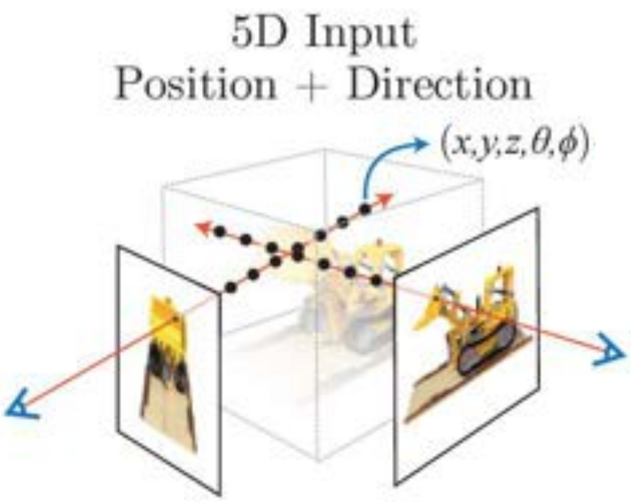
Emission in segment i

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i \,, \quad \text{where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

Probability for emission in i-th segment



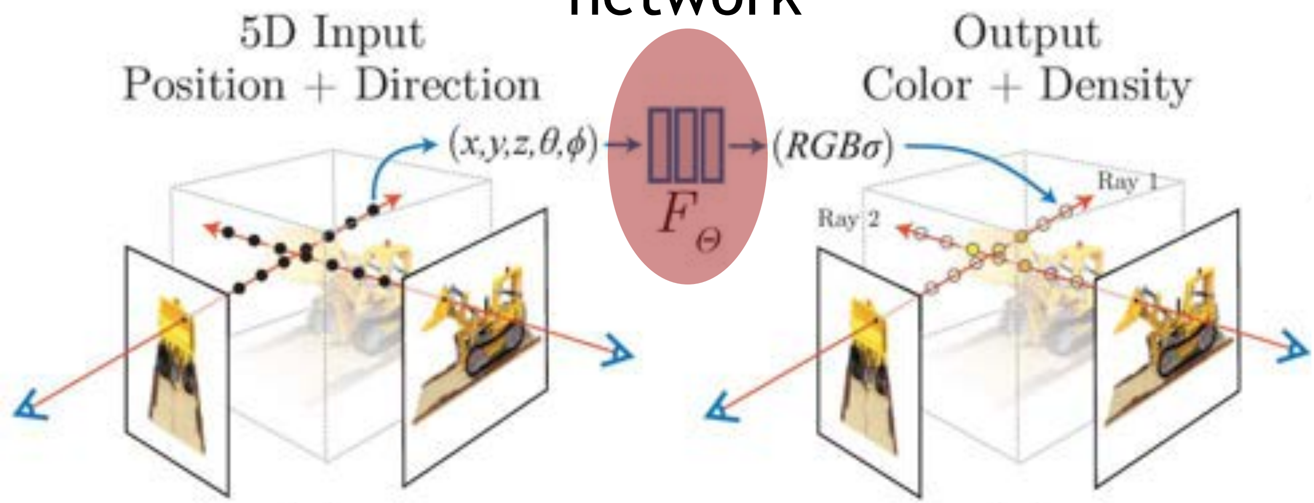
Volumetric rendering function



Ray marching, rays defined by input images
Samples on rays with positions, directions
$(\mathbf{r}(t), \mathbf{d})$

Volumetric rendering function

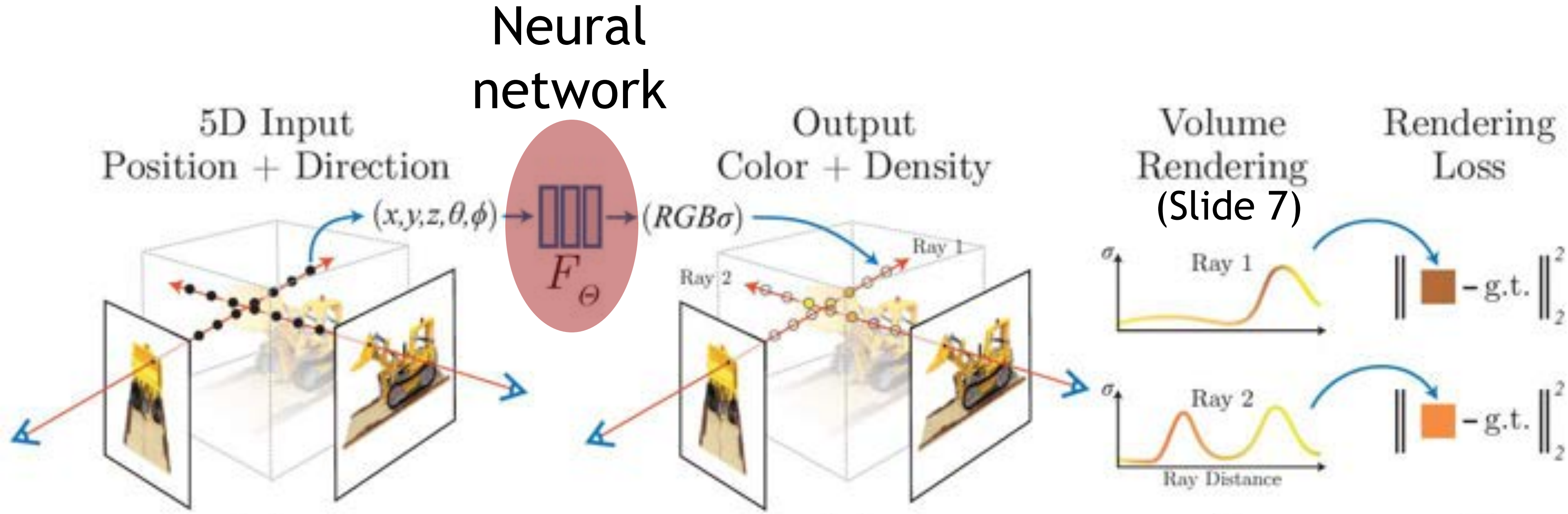
Neural network



Ray marching, rays defined by input images
Samples on rays with positions, directions
$(\mathbf{r}(t), \mathbf{d})$

Colors
$\mathbf{c}(\mathbf{r}(t), \mathbf{d})$
Densities
$\sigma\ (\mathbf{r}(t))$
for each sample

Volumetric rendering function



Ray marching, rays defined by input images
Samples on rays with positions, directions
$(\mathbf{r}(t), \mathbf{d})$

Colors
$\mathbf{c}(\mathbf{r}(t), \mathbf{d})$
Densities
$\sigma\ (\mathbf{r}(t))$
for each sample

Optimize colors, densities to minimize loss

Optimization

- Scene representation using neural network

$$F_{\Theta} : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$$

At each 3d point $\mathbf{x}$, direction $\mathbf{d}$, network outputs radiance $\mathbf{c}$, density σ

- Loss

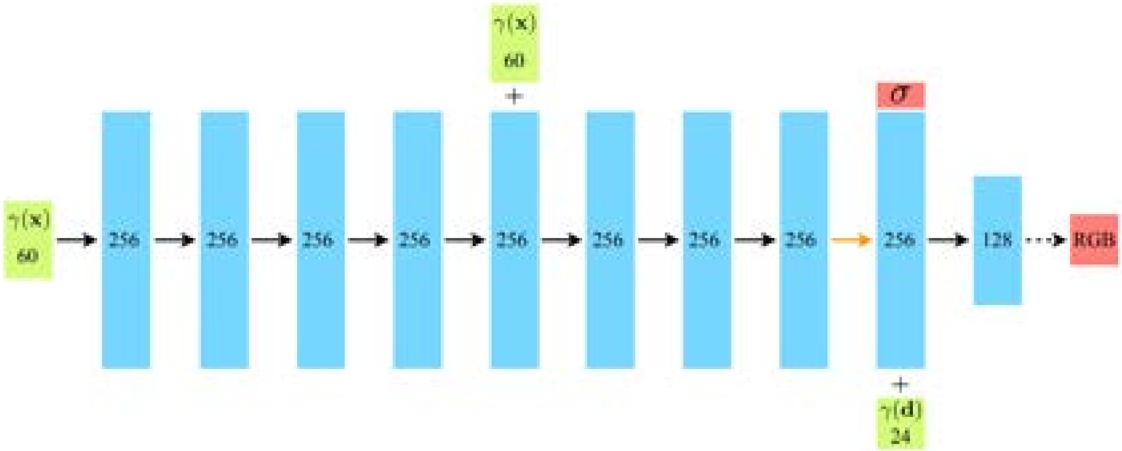
$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left\| \hat{C}(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2$$

Sum over all rays/pixels | Rendered color | Observed color in input

- Ground truth $C(\mathbf{r})$, rendered estimate $\widehat{C(\mathbf{r})}$
- Gradients of pixels $\widehat{C(\mathbf{r})}$ colors wrt. network weights/biases Θ easy to compute using backpropagation (automatic differentiation)

Network architecture

- MLP (fully connected layers), original NeRF approach



- Positional encoding: instead of position p, network input is

$$\gamma(p) = \left(\sin\left(2^0 \pi p\right), \cos\left(2^0 \pi p\right), \cdots, \sin\left(2^{L-1} \pi p\right), \cos\left(2^{L-1} \pi p\right) \right)$$

  - Applied to normalized coordinates p separately
  - Position $\mathbf{x}$: L=10, direction $\mathbf{d}$: L=4
  - Intuition: helps network to better learn high-frequency functions

Hierarchical sampling

- Train two networks simultaneously
- Coarse network: N_c=64 samples per ray
- Fine network: N_f=128 samples per ray
- Use coarse samples to define piecewise constant PDF along ray to importance sample fine samples
  - Probabilities for segments given by coarse samples

$$\hat{w}_i = w_i / \sum_{j=1}^{N_c} w_j$$

$$w_i = T_i(1 - \exp(-\sigma_i \delta_i))$$

"probability for radiance being emitted in i-th segment, and transmitted along ray all the way to eye"

Results https://www.matthewtancik.com/nerf

- View synthesis
- Depth image reconstruction
- 3D surface reconstruction using marching cubes

Limitations of NeRF

- Doesn’t attempt to represent surface directly (only volumetric density σ)
- Treats pixels as infinitesimal rays, doesn’t take into account pixel areas
- Doesn’t take into account imaging artifacts such as blur, over-/under-exposure
- Only works for static scenes
- Requires known camera parameters
- Training and rendering slow
- Requires many input images for high quality reconstruction
- Doesn’t take into account potential appearance variation in input images (different illumination, time, time of year, etc.)
- Doesn’t recover BRDF parameters and illumination
- Only uses 3D locations to predict scene (density, radiance); does not use correspondence information between 3D locations and images

Surface reconstruction

- How to reformulate problem to enable reconstruction of well-defined surfaces?

Surface reconstruction

- "NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction" https://arxiv.org/pdf/2106.10689.pdf
- Optimize neural SDF (signed distance function) instead of density to represent surface
- But still use volumetric rendering similar to NeRF
- Challenge: relationship between SDF and density, which is required for volumetric rendering?

NeuS

- Neural SDF $f(\mathbf{p})$, 3D point $\mathbf{p}$
- Neural radiance field $c(\mathbf{p}, \mathbf{v})$, direction $\mathbf{v}$
- Rendering function (almost, but not exactly same as NeRF)

$$C(\mathbf{o}, \mathbf{v}) = \int_0^{+\infty} w(t)c(\mathbf{p}(t), \mathbf{v})\mathrm{d}t$$

$$w(t) = T(t)\rho(t), \quad \text{where } T(t) = \exp\left(-\int_0^t \rho(u)\mathrm{d}u\right)$$

Weight function

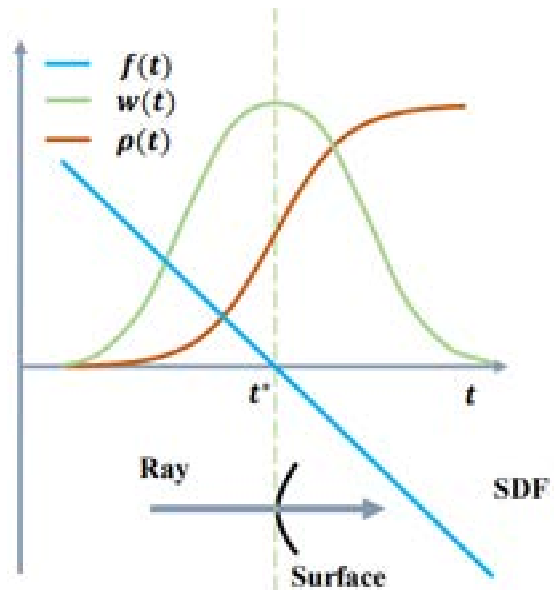
- Opaque density function ρ derived from SDF f

$$\rho(t) = \max\left(\frac{-\frac{\mathrm{d}\Phi_s}{\mathrm{d}t}(f(\mathbf{p}(t)))}{\Phi_s(f(\mathbf{p}(t)))}, 0\right) \qquad \Phi_s(x) = (1 + e^{-sx})^{-1}$$

NeuS weight function w

Designed to be

- Unbiased: $w(t)$ has local maximum for value t when $f(\mathbf{p}(t)) = 0$
  - "Color contribution from point on surface is strongest"
- Occlusion aware: given two points $t_0 < t_1$, where $f(t_0) = f(t_1)$, then $w(t_0) > w(t_1)$



$w(t)$ is unbiased, maximum where f=0

Training

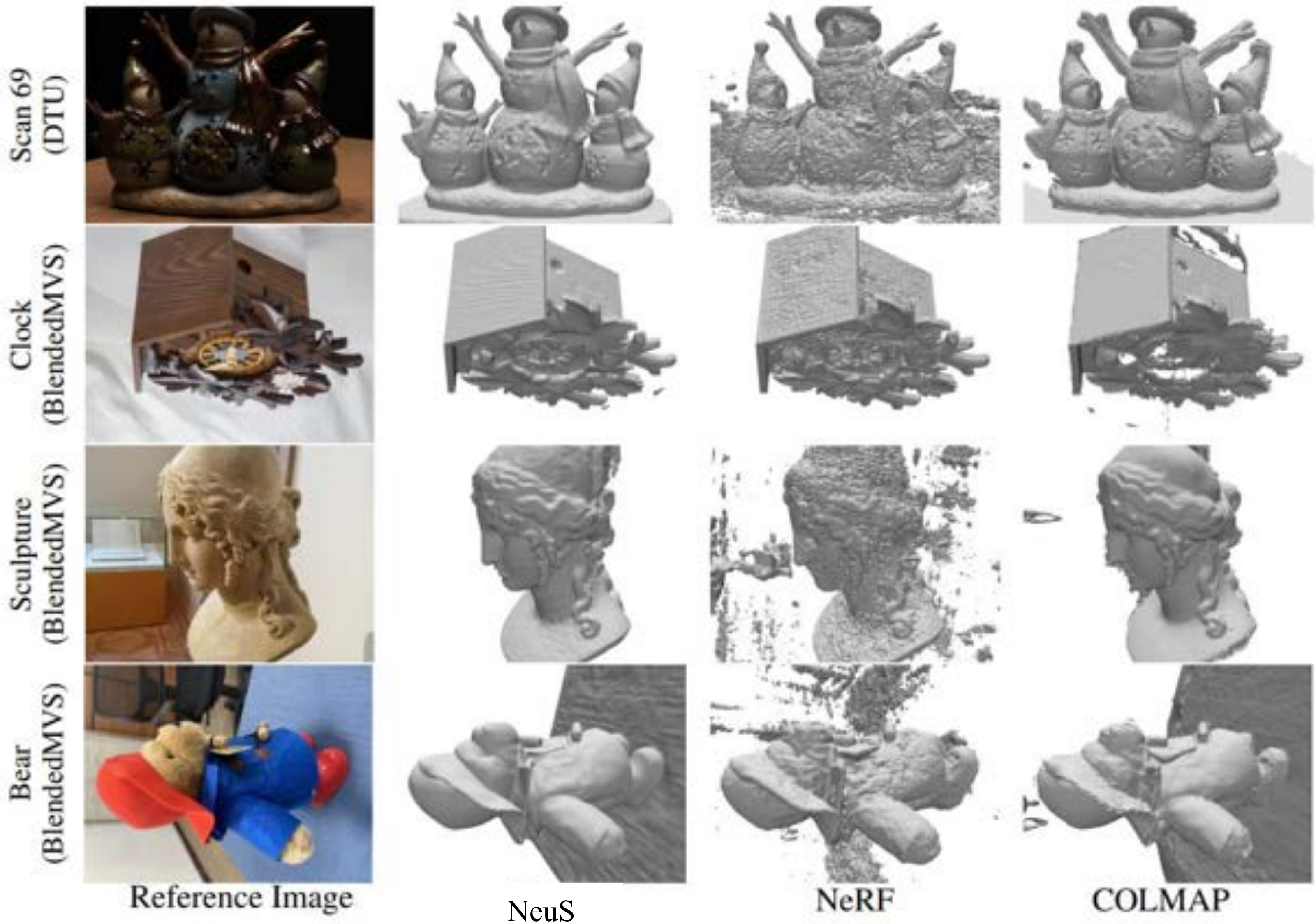
- Include Eikonal term to regularize SDF

$$\mathcal{L}_{reg} = \frac{1}{nm} \sum_{k,i} (\|\nabla f(\hat{\mathbf{p}}_{k,i})\|_2 - 1)^2$$

- Hierarchical sampling, similar to NeRF

Results

https://lingjie0206.github.io/papers/NeuS/



Results

- Chamfer distance to ground truth 3D scans

	w/ mask			w/o mask			
ScanID	IDR	NeRF	Ours	COLMAP	NeRF	UNISURF	Ours
scan24	1.63	1.83	**0.83**	**0.81**	1.90	1.32	1.00
scan37	1.87	2.39	**0.98**	2.05	1.60	**1.36**	1.37
scan40	0.63	1.79	**0.56**	**0.73**	1.85	1.72	0.93
scan55	0.48	0.66	**0.37**	1.22	0.58	0.44	**0.43**
scan63	**1.04**	1.79	1.13	1.79	2.28	1.35	**1.10**
scan65	0.79	1.44	**0.59**	1.58	1.27	0.79	**0.65**
scan69	0.77	1.50	**0.60**	1.02	1.47	0.80	**0.57**
scan83	1.33	**1.20**	1.45	3.05	1.67	1.49	**1.48**
scan97	1.16	1.96	**0.95**	1.40	2.05	1.37	**1.09**
scan105	**0.76**	1.27	0.78	2.05	1.07	0.89	**0.83**
scan106	0.67	1.44	**0.52**	1.00	0.88	0.59	**0.52**
scan110	**0.90**	2.61	1.43	1.32	2.53	1.47	**1.20**
scan114	0.42	1.04	**0.36**	0.49	1.06	0.46	**0.35**
scan118	0.51	1.13	**0.45**	0.78	1.15	0.59	**0.49**
scan122	0.53	0.99	**0.45**	1.17	0.96	0.62	**0.54**
mean	0.90	1.54	**0.77**	1.36	1.49	1.02	**0.84**

DTU dataset

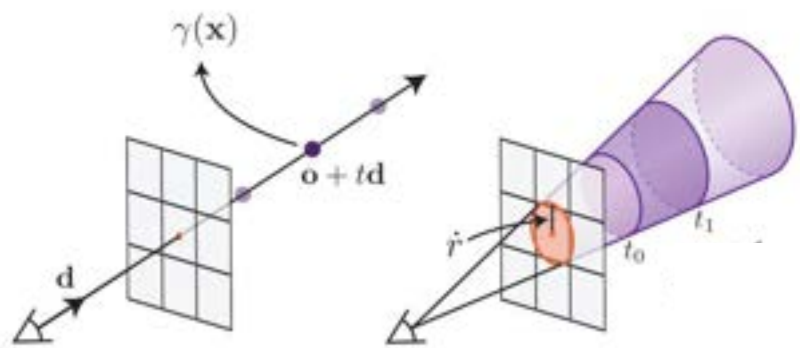
https://roboimagedata.compute.dtu.dk/?page_id=36

Pixel modeling

- How to increase reconstruction accuracy by more accurately modeling image formation in pixels of real cameras?

Mip-NeRF

- “Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields” https://github.com/google/mipnerf
- Key idea: radiance should take into account pixel size (anti-aliasing)

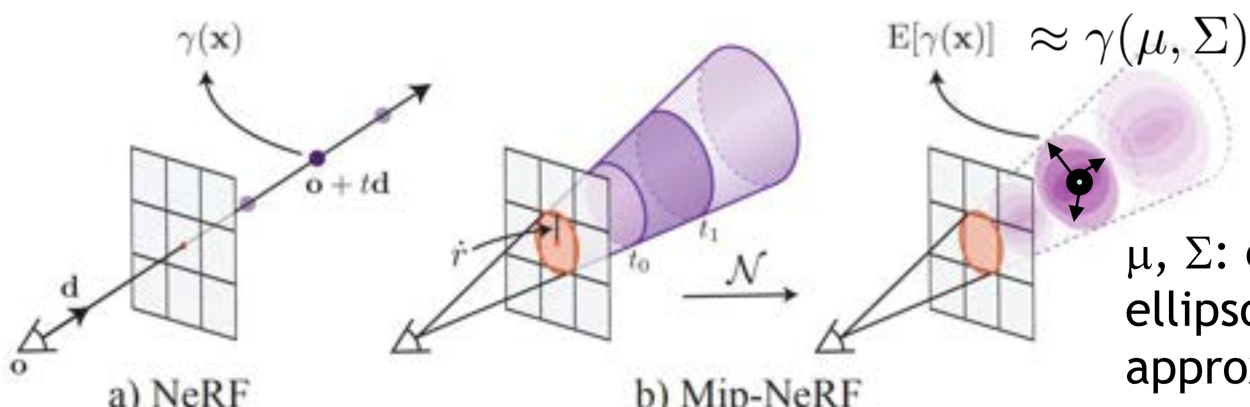


a) NeRF
Pixel modeled as single ray

b) Mip-NeRF
Pixel modeled as conical beam

Mip-NeRF

- "Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields" https://github.com/google/mipnerf
- Key idea: radiance should take into account pixel size (anti-aliasing)
- Extend spatial encoding $\gamma(\mathbf{x})$ to capture local size of conical beam (details see paper)



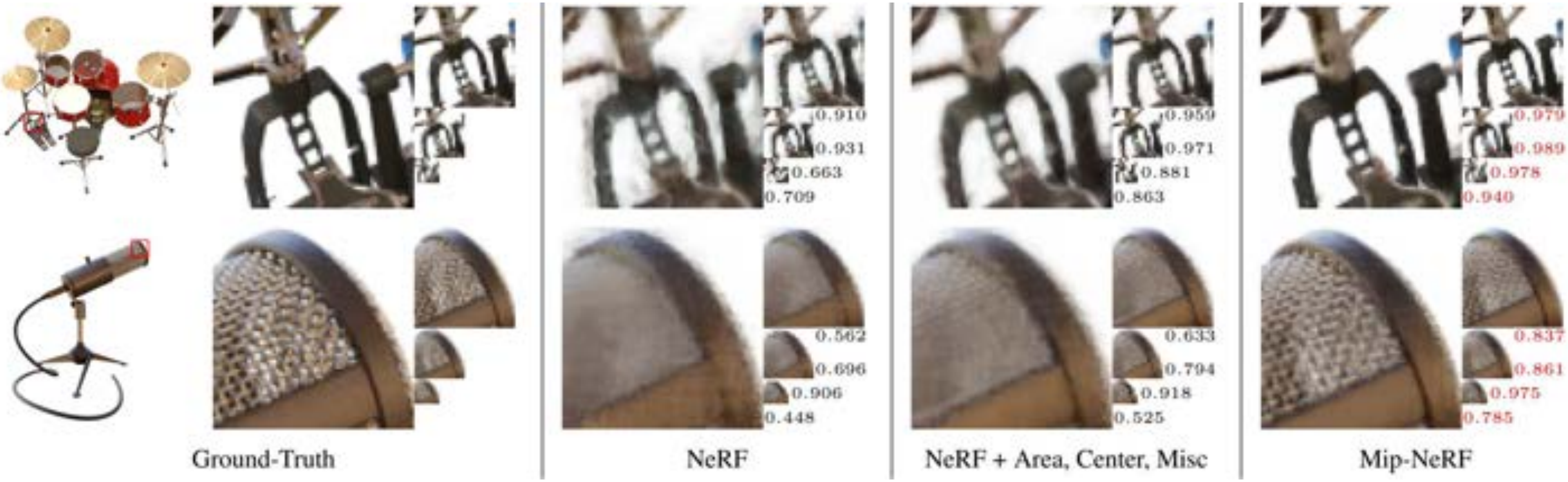
Pixel modeled as single ray

Pixel modeled as conical beam

Generalized spatial encoding $\gamma(\mu,\Sigma)$

μ, Σ: center, size of ellipsoid to locally approximate beam

Results



	PSNR ↑				SSIM ↑				LPIPS ↓						
	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res	Avg. ↓	Time (hours)	# Params
NeRF (Jax Impl.) [11, 30]	31.196	30.647	26.252	22.533	0.9498	0.9560	0.9299	0.8709	0.0546	0.0342	0.0428	0.0750	0.0288	3.05 ± 0.04	1,191K
NeRF + Area Loss	27.224	29.578	29.445	25.039	0.9113	0.9394	0.9524	0.9176	0.1041	0.0677	0.0406	0.0469	0.0305	3.03 ± 0.03	1,191K
NeRF + Area, Centered Pixels	29.893	32.118	33.399	29.463	0.9376	0.9590	0.9728	0.9620	0.0747	0.0405	0.0245	0.0398	0.0191	3.02 ± 0.05	1,191K
NeRF + Area, Center, Misc.	29.900	32.127	33.404	29.470	0.9378	0.9592	0.9730	0.9622	0.0743	0.0402	0.0243	0.0394	0.0190	2.94 ± 0.02	1,191K
Mip-NeRF	32.629	34.336	35.471	35.602	0.9579	0.9703	0.9786	0.9833	0.0469	0.0260	0.0168	0.0120	0.0114	2.84 ± 0.01	612K
Mip-NeRF w/o Misc.	32.610	34.333	35.497	35.638	0.9577	0.9703	0.9787	0.9834	0.0470	0.0259	0.0167	0.0120	0.0114	2.82 ± 0.03	612K
Mip-NeRF w/o Single MLP	32.401	34.131	35.462	35.967	0.9566	0.9693	0.9780	0.9834	0.0479	0.0268	0.0169	0.0116	0.0115	3.40 ± 0.01	1,191K
Mip-NeRF w/o Area Loss	33.059	34.280	33.866	30.714	0.9605	0.9704	0.9747	0.9679	0.0427	0.0256	0.0213	0.0308	0.0139	2.82 ± 0.01	612K
Mip-NeRF w/o IPE	29.876	32.160	33.679	29.647	0.9384	0.9602	0.9742	0.9633	0.0742	0.0393	0.0226	0.0378	0.0186	2.79 ± 0.01	612K

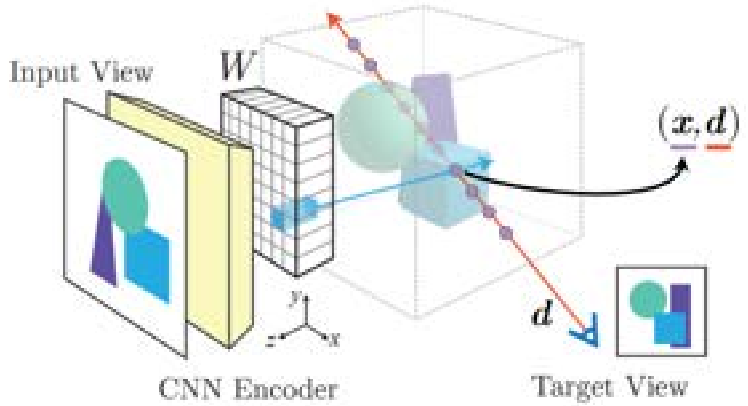
Leveraging correspondence for sparse input data

- Observation
  - To perform 3D reconstruction using correspondence and triangulation, we only need correspondence between two views
  - In contrast, inverse rendering typically requires observation of same scene point from many viewpoints
- How to leverage correspondences between 3D locations and 2D image regions to improve performance of inverse rendering under sparse input views?

Sparse input views

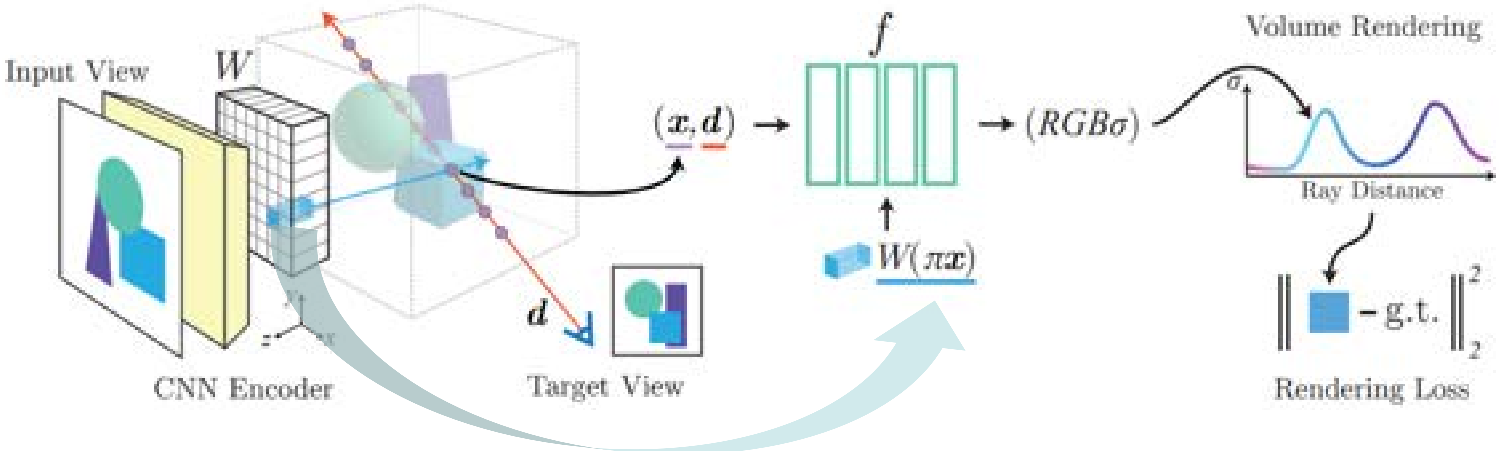
- "pixelNeRF: Neural Radiance Fields from One or Few Images" https://arxiv.org/pdf/2012.02190.pdf
- Key idea:
  - Project 3D sample points on rays to retrieve correspondence information between 3D sample points and images, encoded as neural network features
  - Use local image features to help NeRF network predict density, radiance

Architecture



- 2D feature maps $\mathbf{W}$ using CNN encoder

Architecture



- 2D feature maps $\mathbf{W}$ using CNN encoder
- NeRF f

$$f(\gamma(\mathbf{x}), \mathbf{d}; \mathbf{W}(\pi(\mathbf{x}))) = (\sigma, \mathbf{c})$$

  - Projection of 3D sample location $\mathbf{x}$ onto image plane $\pi(\mathbf{x})$
  - Positional encoding γ
  - Direction $\mathbf{d}$, density σ, radiance $\mathbf{c}$

Multiple input views

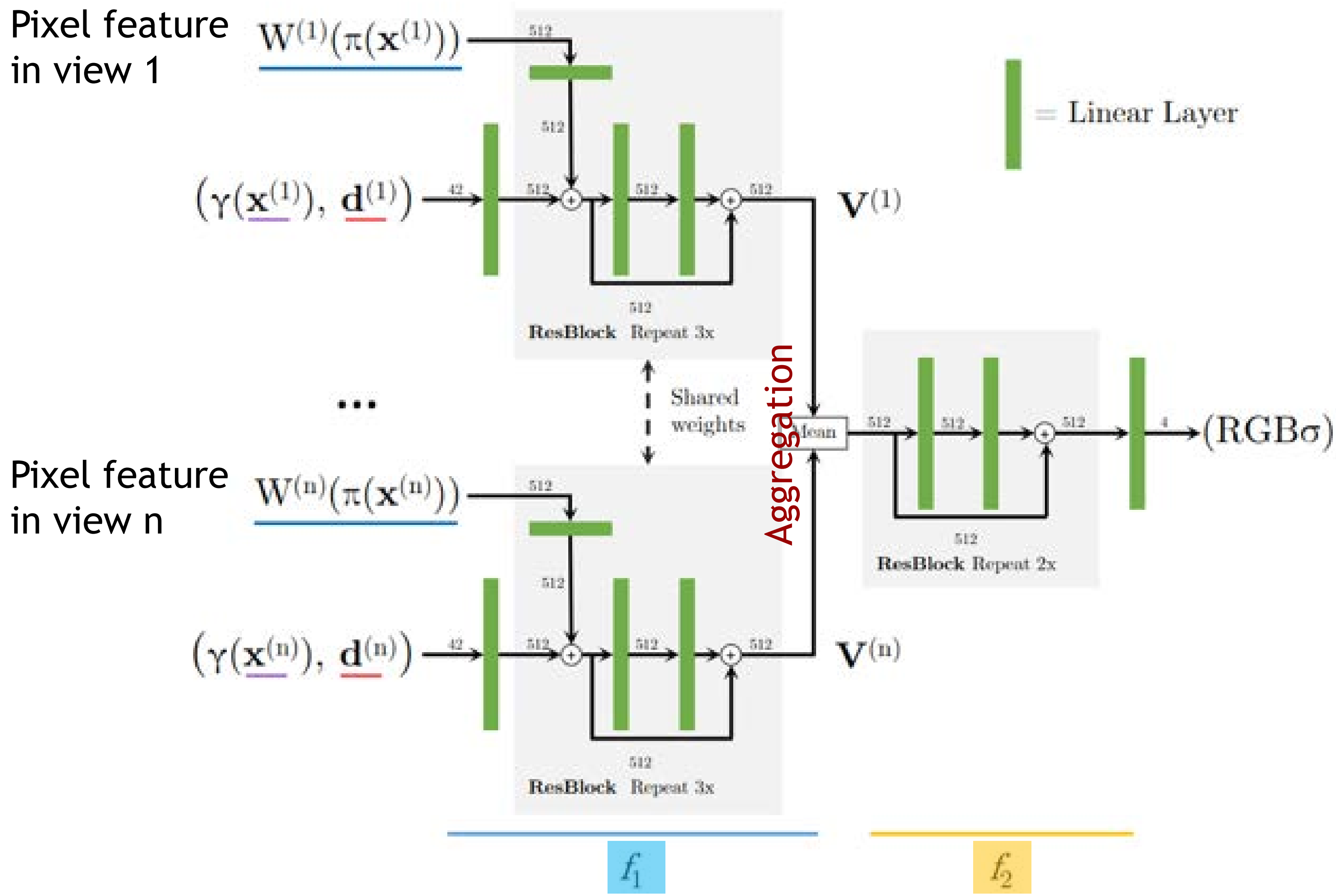
- Intermediate feature for each view i using network f_1

$$\mathbf{V}^{(i)} = f_1\left(\gamma(\mathbf{x}^{(i)}), \mathbf{d}^{(i)}; \mathbf{W}^{(i)}(\pi(\mathbf{x}^{(i)}))\right)$$

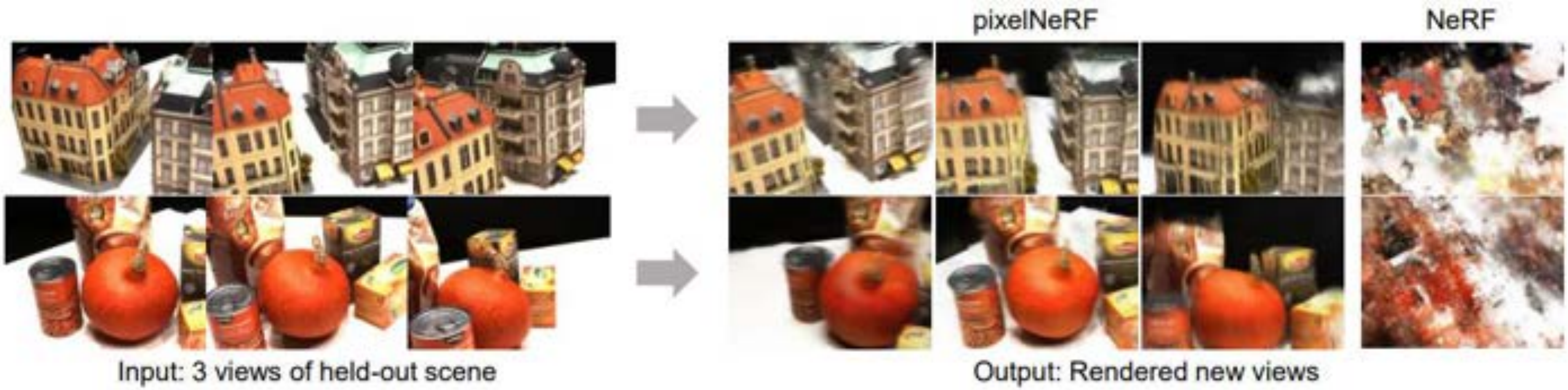
- Aggregation (averaging ψ) over multiple views and color/density prediction using network f_2

$$(\sigma, \mathbf{c}) = f_2\left(\psi\left(\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(n)}\right)\right)$$

Multiple input views



Results



- Trained on data set, applied to test scene without scene-specific training