# NeRF Review

박지호

# Content

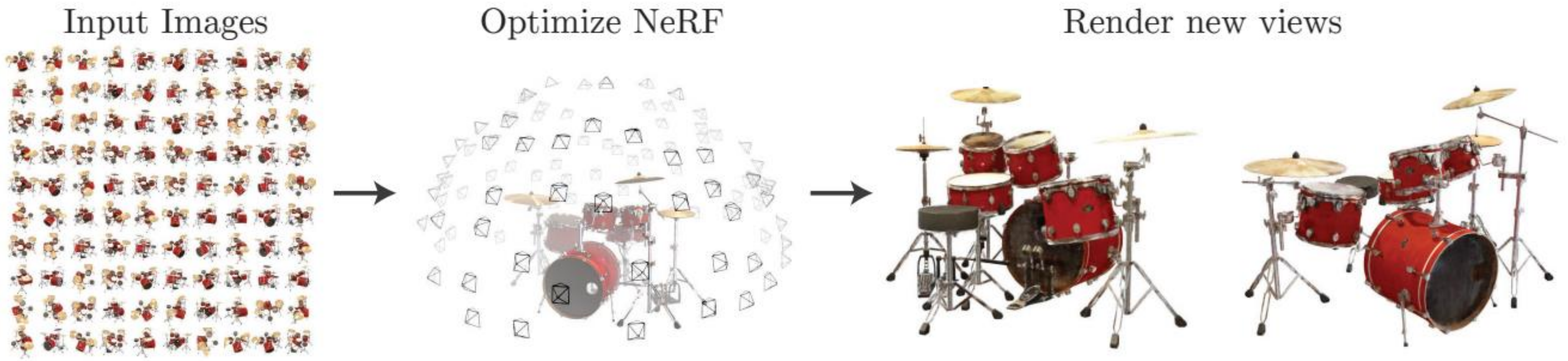
1. Concept
2. NeRF
3. Volume Rendering
4. Details
5. Results

https://miro.medium.com/v2/resize:fit:512/1*TMVO-IsZNM0kl-2xDQqW6A.gif

# 1. Concept

## View Synthesis of a Single Scene

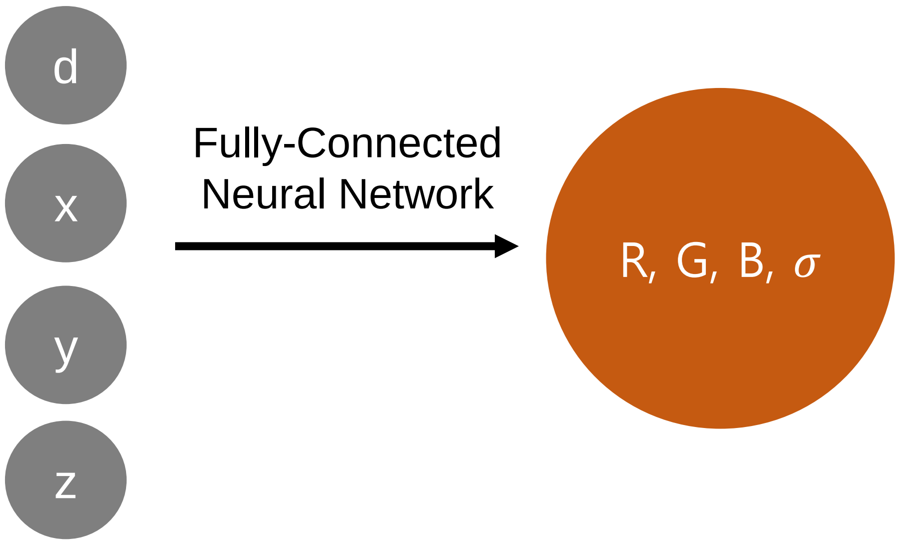
- Train Data: various views of a scene (2d image)
- Target Prediction: unseen view of a scene (2d image)



# 1. Concept

## View Synthesis of a Single Scene

### 1) ***3D Scene Representation(NeRF)***




3D Coordinate Information
(+direction)

Color Information
(+density)

### 2) ***Volume Rendering***

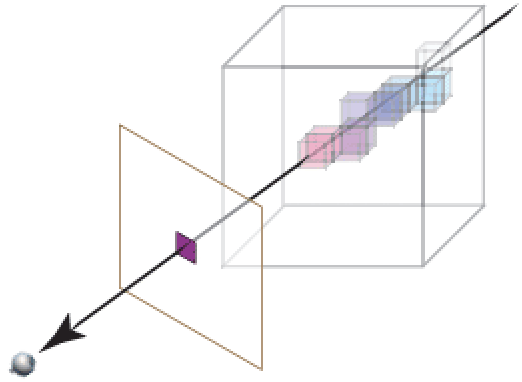


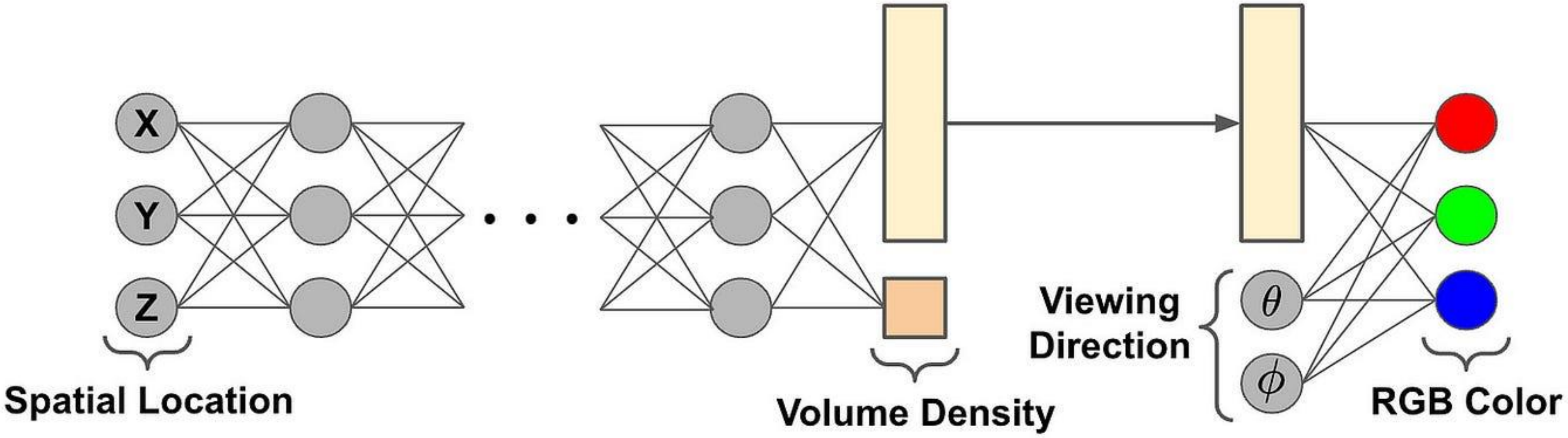
**Figure 2:** *The ray casting integral sums the color and opacity properties of each data voxel that intersects the ray.*

Generating 2D View
from
3D Scene Representation

# 2. Neural Radiance Field Scene Representation

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

- Input: $\mathrm{x}, \mathrm{y}, \mathrm{z}, \theta, \phi$
- Output: $\mathrm{R}, \mathrm{G}, \mathrm{B}, \sigma$
- 8 fully-connected layers



# 3. Volume Rendering

- $C(r)$: ***Expected Color*** of the camera ray $r(t) = o + td$
- $T(t)$: Accumulated ***Transmittance***
- Differentiable Equation → enables end-to-end training with backprop

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

$$\text{where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

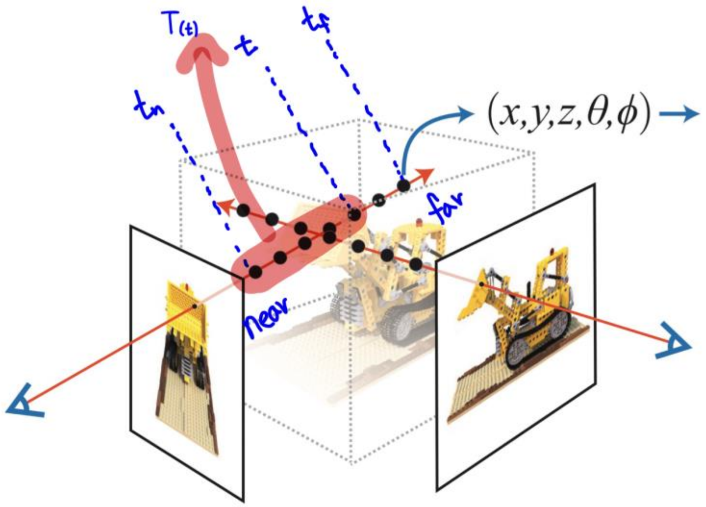



Image from https://nuggy875.tistory.com/168

# 3. Volume Rendering

Quadrature: estimation of continuous integral

1) ***Sampling*** the x value

→ avoid deterministic quadrature
(fixed discrete set of location)

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n),\ t_n + \frac{i}{N}(t_f - t_n)\right]$$

2) ***Quadrature Equation***

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

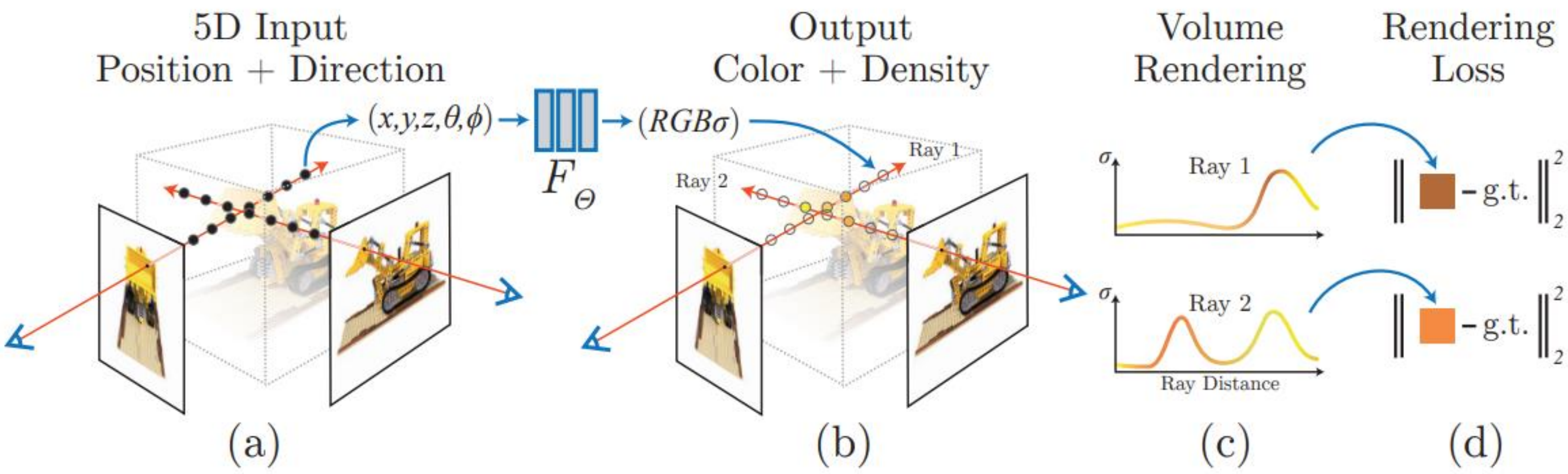
$$\delta_i = t_{i+1} - t_i$$

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i \,,$$

$$\text{where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

# Total Network

- NeRF mapping + Volume Rendering
- Training with ground truth 2D view images



# 4. Details

## ***Positional Encoding***

- Bad Result with cartesian coordinate input (x, y, z) → Expand the Dimension
- Instead of two angle values, unit direction vector is used for direction input

  : $(\theta, \phi) \rightarrow (x_0, y_0, z_0) \rightarrow encode$

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

$$L = 10\ for\ \gamma(x),\ L = 4\ for\ \gamma(d)$$

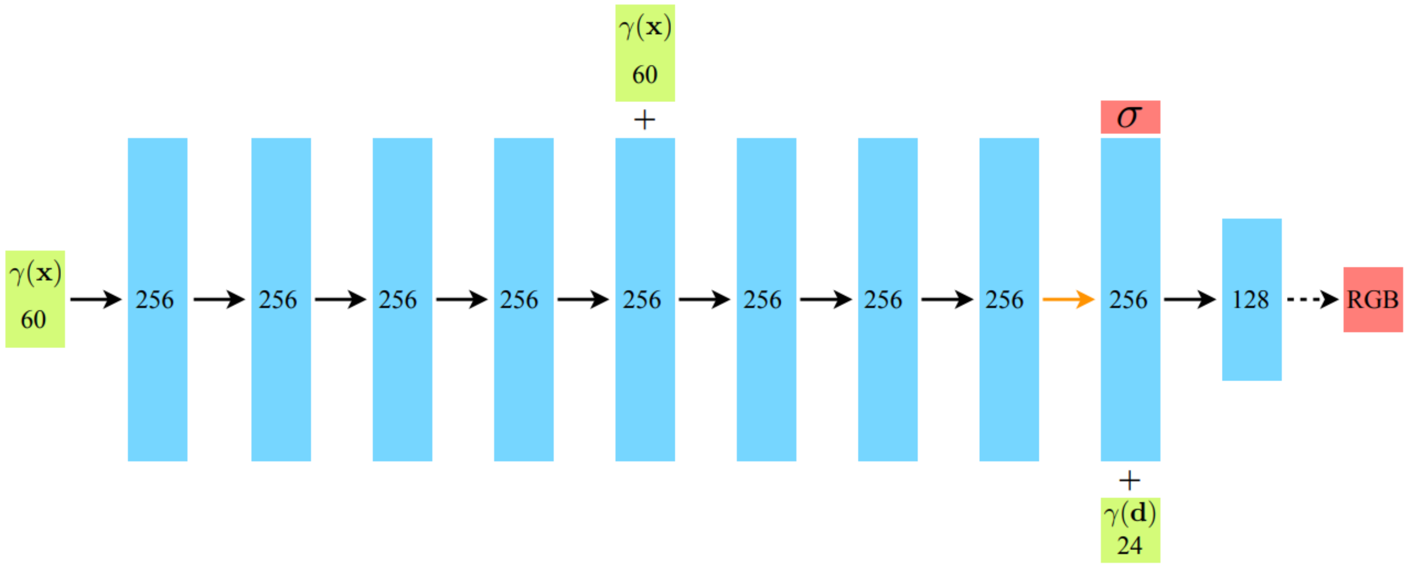
# 4. Details

## ***Positional Encoding**: Final Network Architecture*

encoded dim = input dim x 2 x L

- dim of $\gamma(x)$ = 3 x 2 x 10 = 60

- dim of $\gamma(d)$ = 3 x 2 x 4  = 24



# 4. Details

## *Hierarchical Volume Sampling*

- Let ***Course network*** and ***Fine network***
- Normalize the $w_i$, and Compute the PDF of $w_i$

→ Sample a second set from this PDF
(more likely to sample the point where more visible objects exists)

→ Use it on Fine network

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i \,, \qquad w_i = T_i(1 - \exp(-\sigma_i \delta_i)) \,.$$

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

# 4. Details

## *Hierarchical Volume Sampling*

- Total Loss:

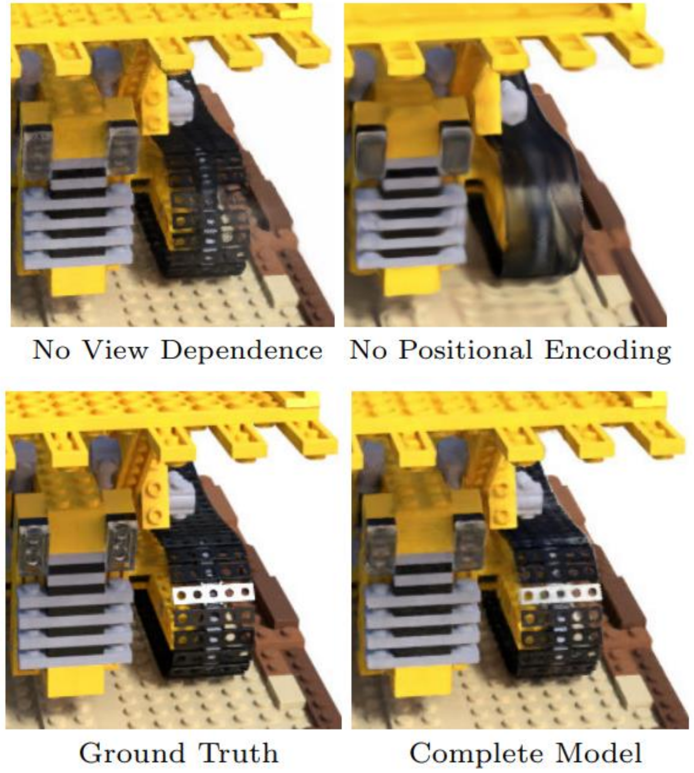
$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[ \left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

- Final Rendering: ***Fine network*** (using all samples from course/fine)
- ***Course network*** learns to allocate the important samples for ***fine network***

# 5. Results

## Model Experiments:

Using the view direction and positional encoding improves the performance

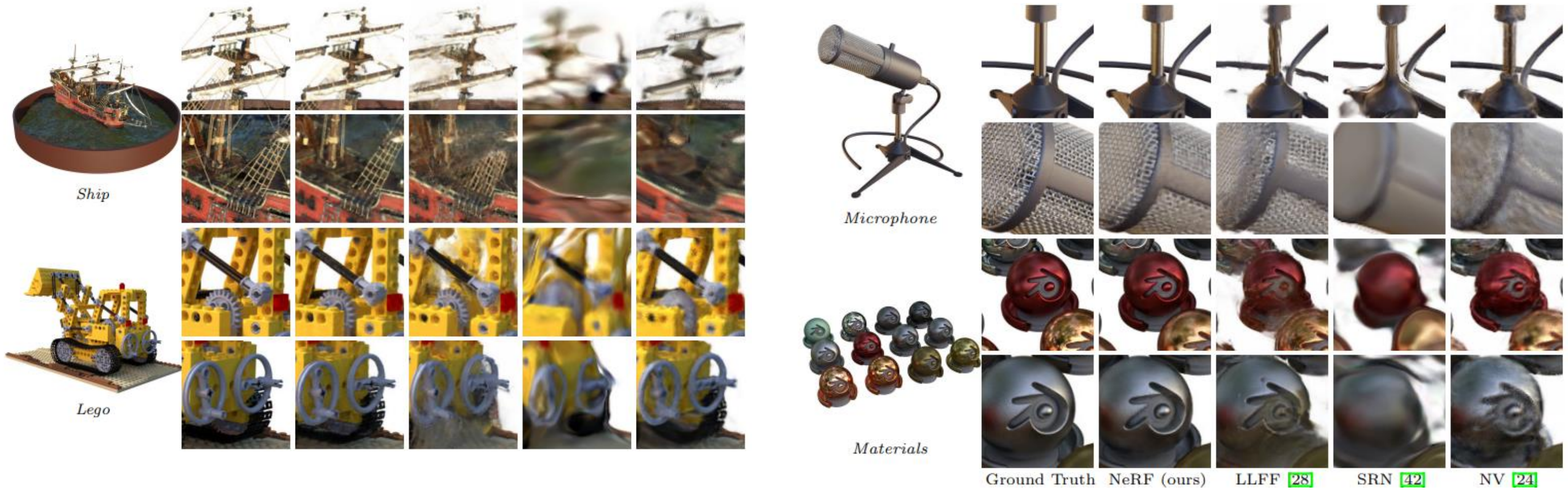


| | Input | #Im. | $L$ | $(N_c, N_f)$ | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|---|---|---|---|
| 1) No PE, VD, H | $xyz$ | 100 | - | (256, - ) | 26.67 | 0.906 | 0.136 |
| 2) No Pos. Encoding | $xyz\theta\phi$ | 100 | - | (64, 128) | 28.77 | 0.924 | 0.108 |
| 3) No View Dependence | $xyz$ | 100 | 10 | (64, 128) | 27.66 | 0.925 | 0.117 |
| 4) No Hierarchical | $xyz\theta\phi$ | 100 | 10 | (256, - ) | 30.06 | 0.938 | 0.109 |
| 5) Far Fewer Images | $xyz\theta\phi$ | 25 | 10 | (64, 128) | 27.78 | 0.925 | 0.107 |
| 6) Fewer Images | $xyz\theta\phi$ | 50 | 10 | (64, 128) | 29.79 | 0.940 | 0.096 |
| 7) Fewer Frequencies | $xyz\theta\phi$ | 100 | 5 | (64, 128) | 30.59 | 0.944 | 0.088 |
| 8) More Frequencies | $xyz\theta\phi$ | 100 | 15 | (64, 128) | 30.81 | 0.946 | 0.096 |
| 9) Complete Model | $xyz\theta\phi$ | 100 | 10 | (64, 128) | **31.01** | **0.947** | **0.081** |

# 5. Results

Comparison → state-of-the-art performance

| Method | Diffuse Synthetic 360° [41] | | | Realistic Synthetic 360° | | | Real Forward-Facing [28] | | |
|---|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ |
| SRN [42] | 33.20 | 0.963 | 0.073 | 22.26 | 0.846 | 0.170 | 22.84 | 0.668 | 0.378 |
| NV [24] | 29.62 | 0.929 | 0.099 | 26.05 | 0.893 | 0.160 | - | - | - |
| LLFF [28] | 34.38 | 0.985 | 0.048 | 24.88 | 0.911 | 0.114 | 24.13 | 0.798 | **0.212** |
| Ours | **40.15** | **0.991** | **0.023** | **31.01** | **0.947** | **0.081** | **26.50** | **0.811** | 0.250 |



# Reference

- NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis
  https://arxiv.org/abs/2003.08934
- https://nuggy875.tistory.com/168