NeRF

Jisang Han

onground@korea.ac.kr

Artificial Intelligence in KU (AIKU)

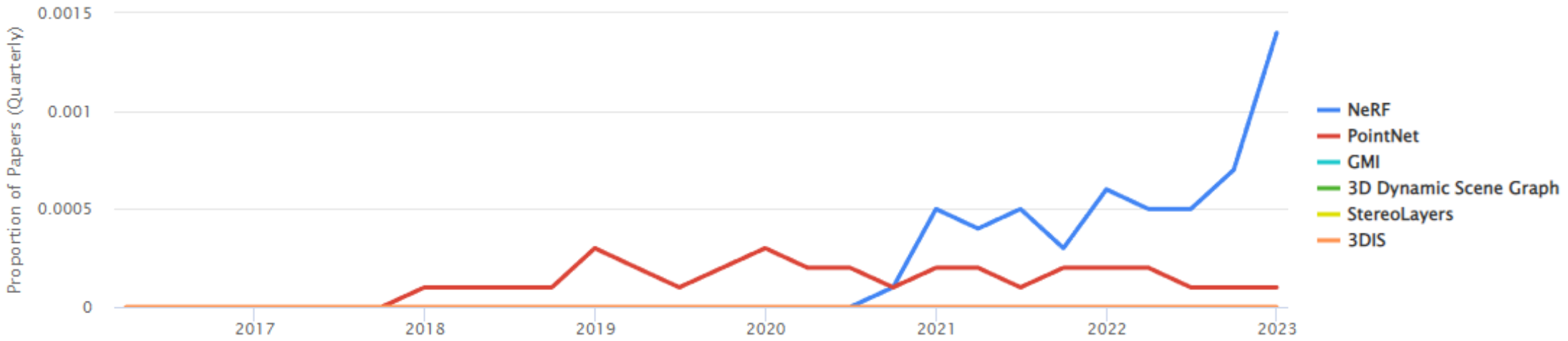
Department of Computer Science and Engineering, Korea University

AIKU

Neural **r**adiance **F**ields

Neural **r**adiance **F**ields

- 🔥🔥🔥
- Conference, Oral maker



Neural **r**adiance **F**ields

NeRF models have attracted much attention in the computer vision community in the past two years, with more than 150 papers and preprints appearing on popular code aggregation website [1], and more than 200 preprints on arXiv. Many of the preprints were eventually published in top tier computer vision conferences such as CVPR, ICCV, and ECCV, with CVPR 2021 less than 10 NeRF papers, and CVPR 2022 publishing more than 50 papers on this topic. Similar trends can be seen in the other computer vision conferences as well. In 2022, the impact of NeRF is large and ever increasing, with the original NeRF paper by Mildenhall et al. receiving more than 1300 citations, and growing interest year-over-year. Given current interest and lack of existing comprehensive survey papers, we believe it necessary to organize a one such paper to help computer vision practitioners with this new topic.

NeRF

- 2D → 3D
- (Novel) View Synthesis

View synthesis

From Wikipedia, the free encyclopedia

This article's **tone or style may not reflect the encyclopedic tone used on Wikipedia**. See Wikipedia's guide to writing better articles for suggestions. *(February 2020)* *(Learn how and when to remove this template message)*

Currently a study branch of Computer Science Research, **view synthesis** aims to create new views of a specific subject starting from a number of pictures taken from given point of views.

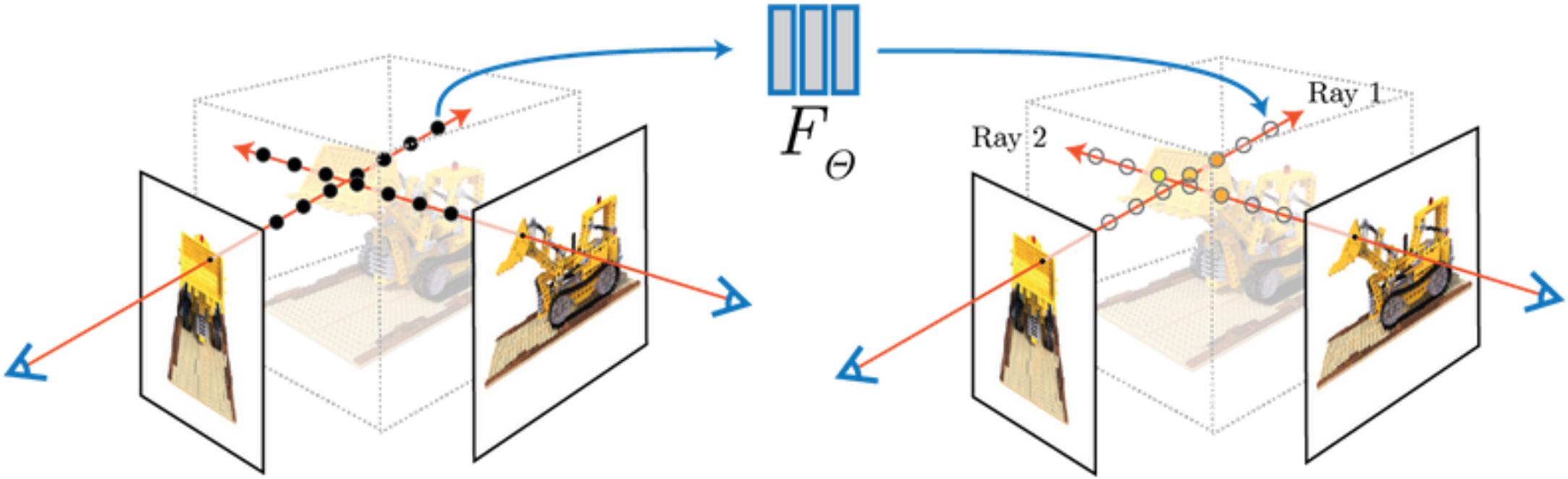
Vision Research and Artificial Intelligence fields are involved in the definition of suitable approaches to the problem. See Computer Vision

Example problem 1: Is to take a number of images of a specific subject, taken from a specific point with a specific camera orientation and setting, and then use that data to build a synthetic image that looks as if it was taken from a virtual camera that is placed at a different point and with the same settings.

Example problem 2: Two people interact through their computers, using a webcam. Try to render corrected images, as if taken from a virtual webcam positioned behind the application window. This would solve the long-standing Eye contact problem which is experienced in this environment. A double illusion is perceived by the users: each of them looks at each other's face, but neither of them get the proper feeling of it.

An example application of View Synthesis is Free view point television.

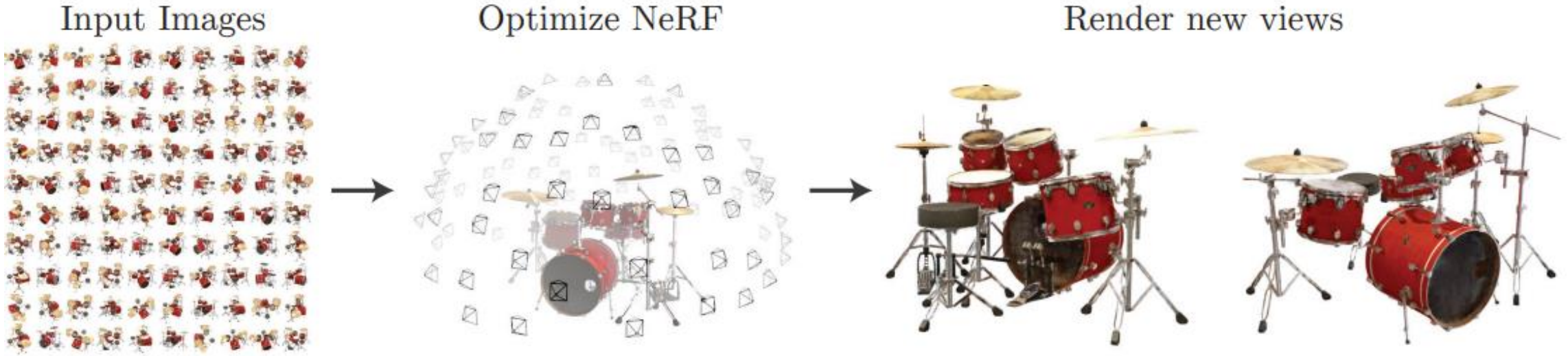
NeRF



NeRF

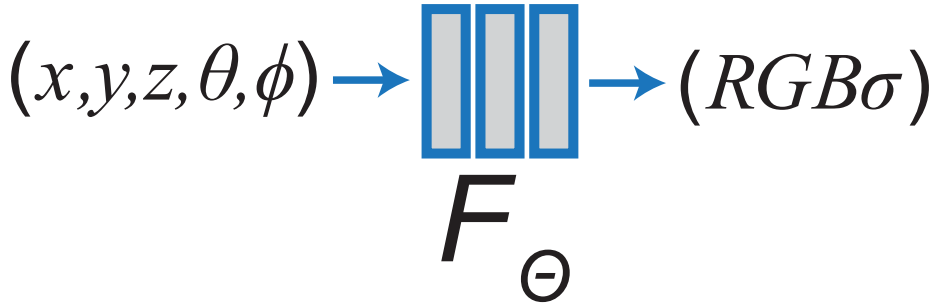
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

ECCV 2020 Oral – Best Paper Honorable Mention



It receives a discrete 2D image at n point in time to create a new image at any point in time so that the image can be constructed continuously.

NeRF: A brief explanation

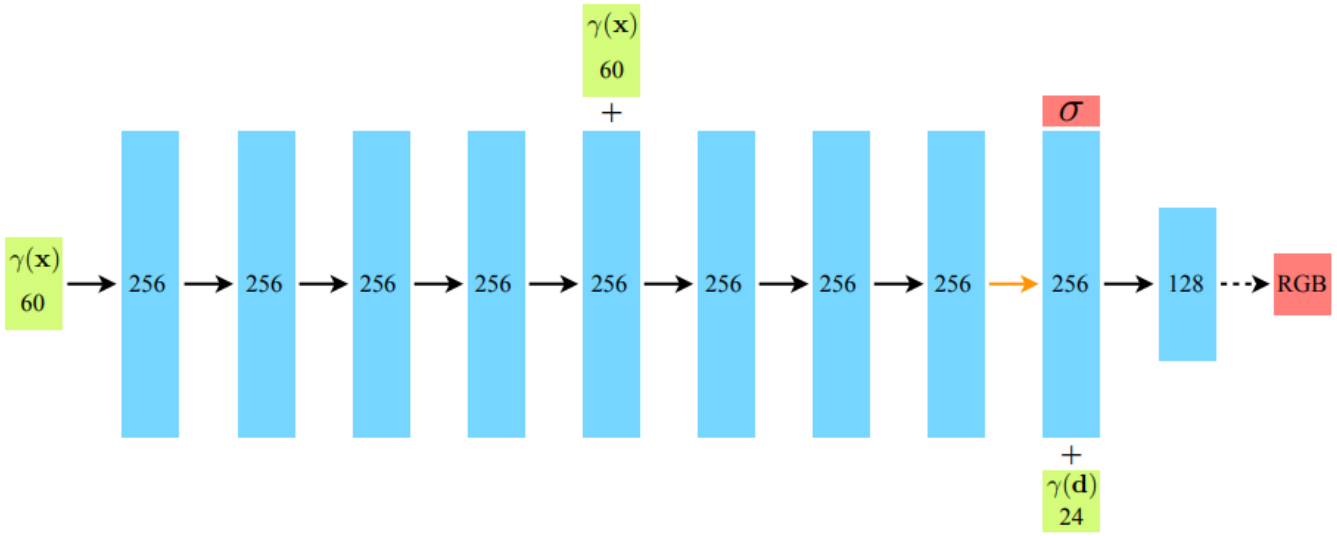


NeRF learns a **fully-connected network (FC)** that **predicts RGB values and density** of objects **by receiving five-dimensional data**.

Including **(x, y, z)**, which are the special locations of objects, and **(θ, ϕ),** which are viewing directions (points and angles of the camera).

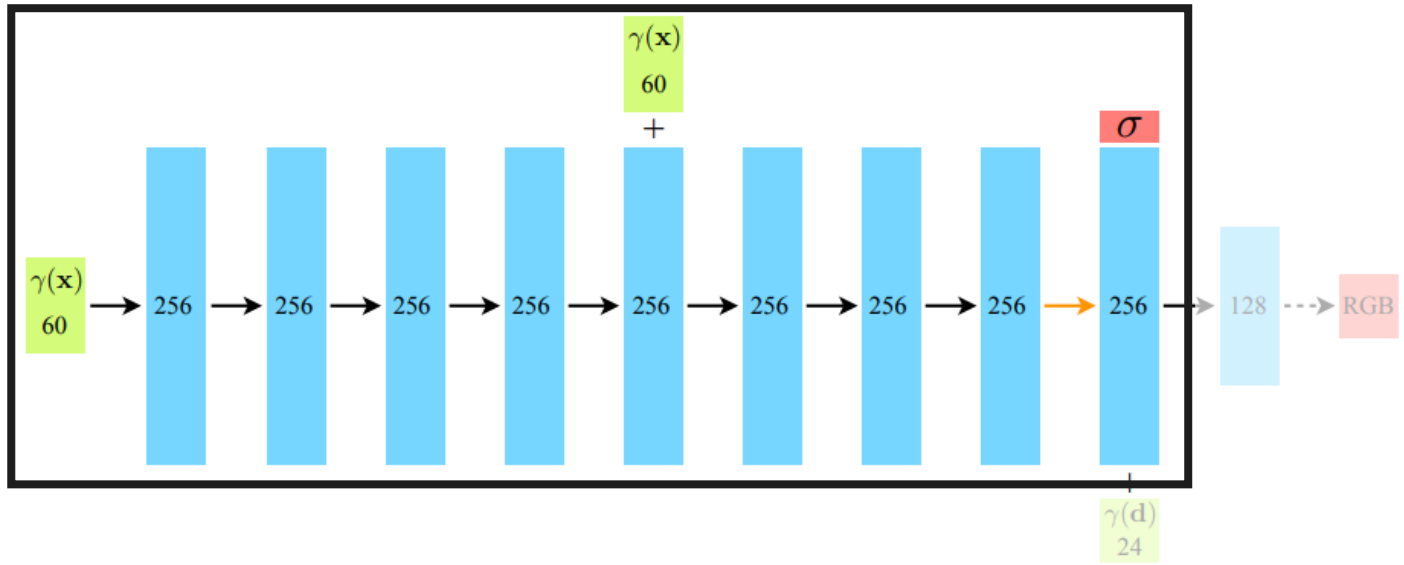
Here, the **density** of the object is the **reciprocal of transparency**, which makes the object opaque (can't see what's behind it well), and makes the object transparent when the density decreases.

NeRF: A brief explanation



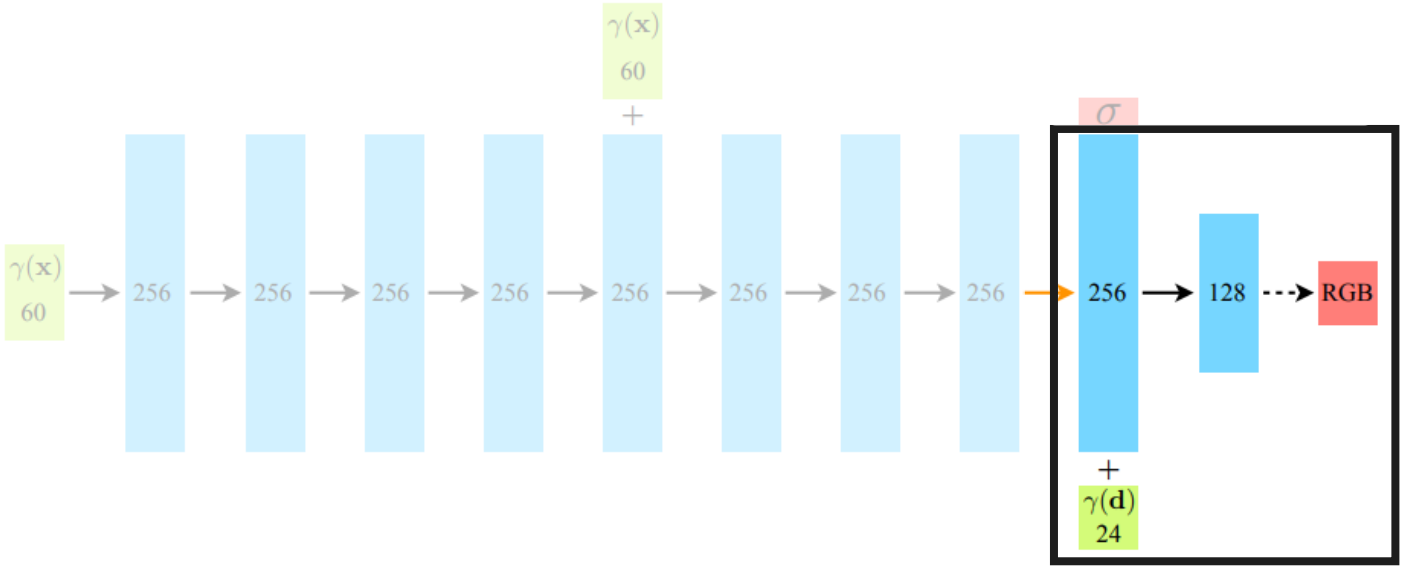
- All layers are **standard fully-connected layers**.
- Black arrows indicate layers with ReLU activations
- Orange arrows indicate layers with no activation
- Dashed black arrows indicate layers with sigmoid activation
- “+” denotes vector concatenation.

NeRF: Model Architecture



First, only (x, y, z), which are the positional information of the object, passes through 8 FCs to predict the density of the object.

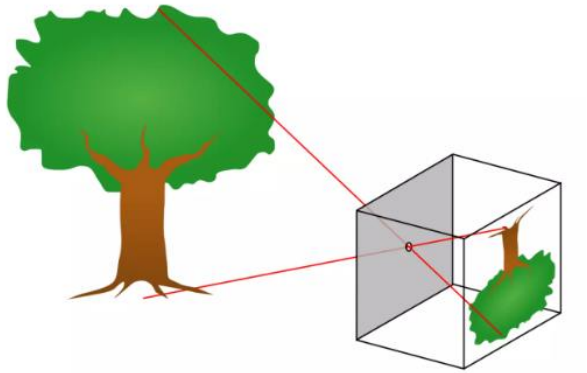
NeRF: Model Architecture

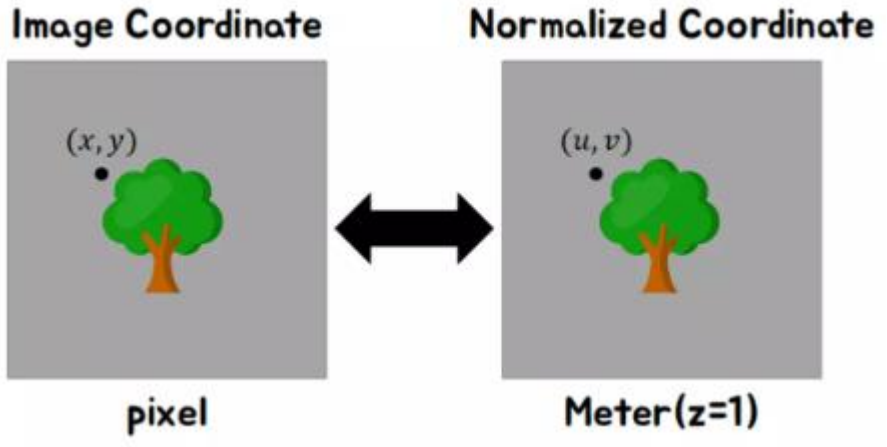


RGB is predicted through the 9th FC by combining the existing location information and the direction of the object. This is due to the Non-Lambertian Effect, which **varies in color or reflectance depending on the viewing angle.**

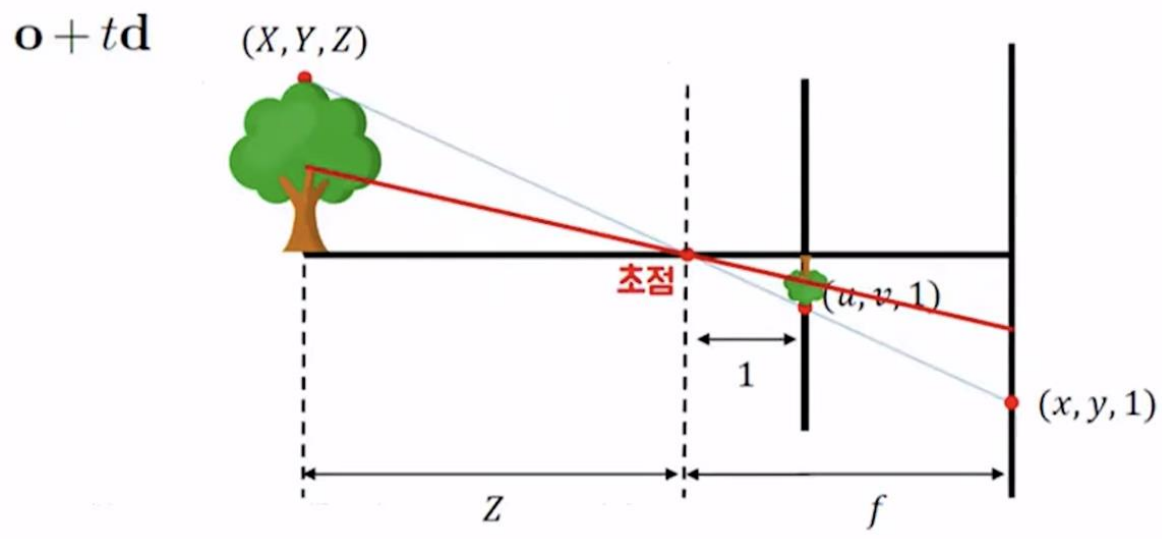
Since **the density of the object must be the same regardless of the angle**, the first eight FCs learned only from the location information, and **the color (RGB) depends on the direction in which the object is viewed**, so the object was input to the last FC.

NeRF: Volume Rendering





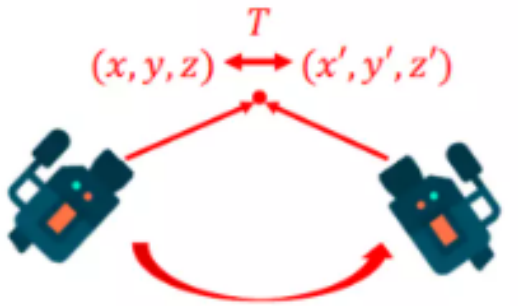
$$x = f_x u + c_x$$
$$y = f_y v + c_y$$



$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

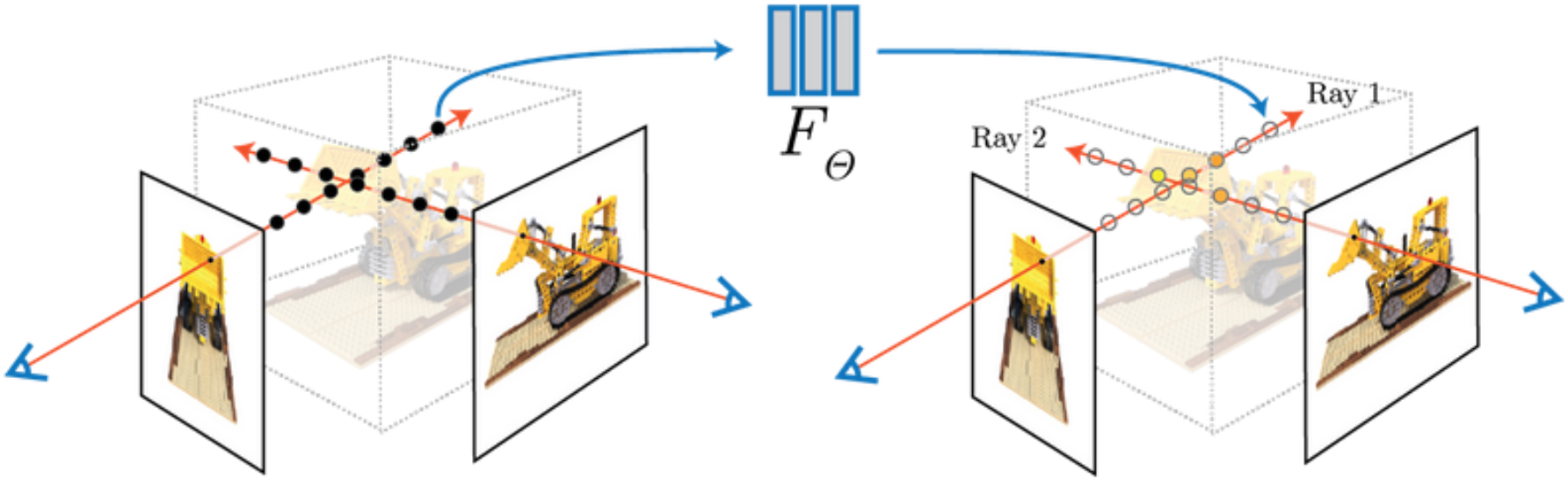
NeRF: Volume Rendering



$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \begin{bmatrix} x \\ y \\ z \end{bmatrix} + T$$

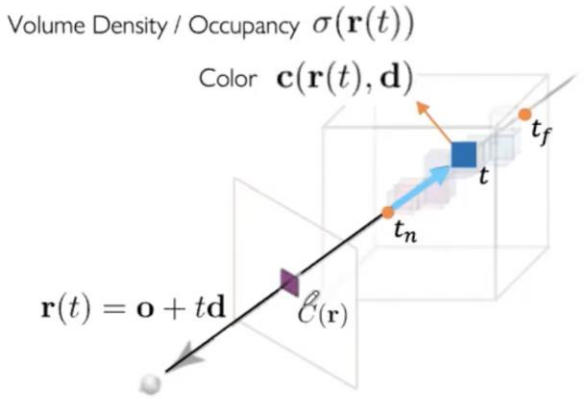
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = [R|t] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

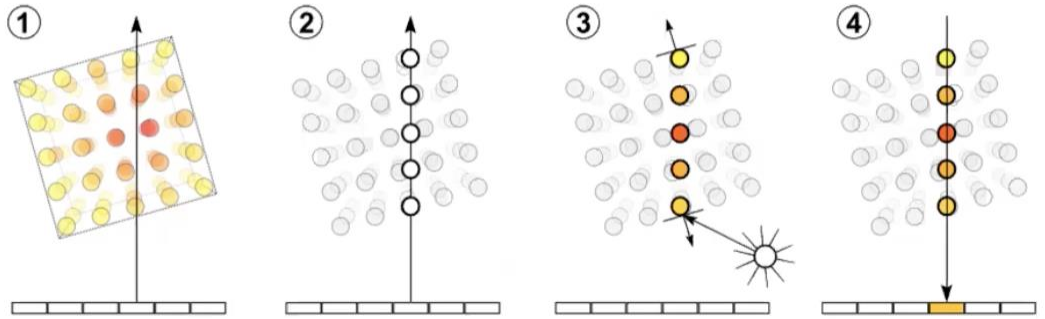
NeRF: Volume Rendering



1 pixel = Weighted sum of all points from a ray

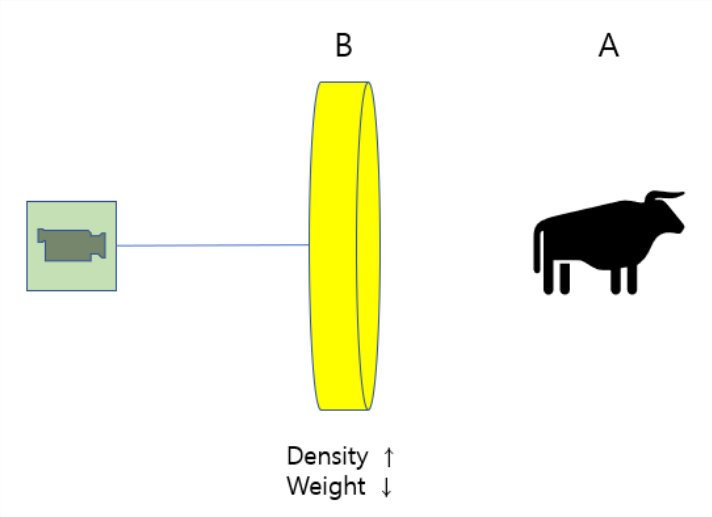
NeRF: Volume Rendering





1 pixel = Weighted sum of all points from a ray

NeRF: Volume Rendering



$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Accumulated transmittance

Color at a 3D point lying on the ray

Volume density or Occupancy at a 3D point lying on the ray

- Density ↑ = Weight ↑
- If sum of points that are in front of the point is small, density of the point is high.
- $\rightarrow T(t)$
- If $\int$ density is high, then weight should be small.

NeRF: Volume Rendering

[For all 3D points along the ray]

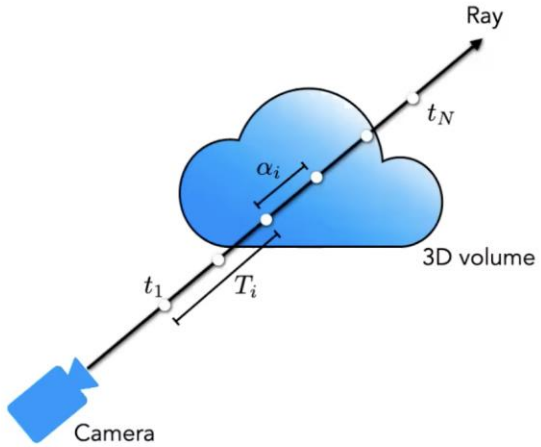
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

[Approx. for sampled points along the ray]

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

- Infinite 3D points is impossible to implement
- **Stratified sampling approach**

NeRF: Stratified sampling



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

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

We numerically estimate this continuous integral using quadrature. Deterministic quadrature, which is typically used for rendering discretized voxel grids, would effectively limit our representation's resolution because the MLP would only be queried at a fixed discrete set of locations. Instead, we use a stratified sampling approach where we partition $[t_n, t_f]$ into N evenly-spaced bins and then draw one sample uniformly at random from within each bin:

Although we use a discrete set of samples to estimate the integral, stratified sampling enables us to represent a continuous scene representation because it results in the MLP being evaluated at continuous positions over the course of optimization. We use these samples to estimate $C(\mathbf{r})$ with the quadrature rule

NeRF: Hierarchical volume sampling

- Coarse Network

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

Contribution weight, w_i

Normalize

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

Kind of a probability distribution

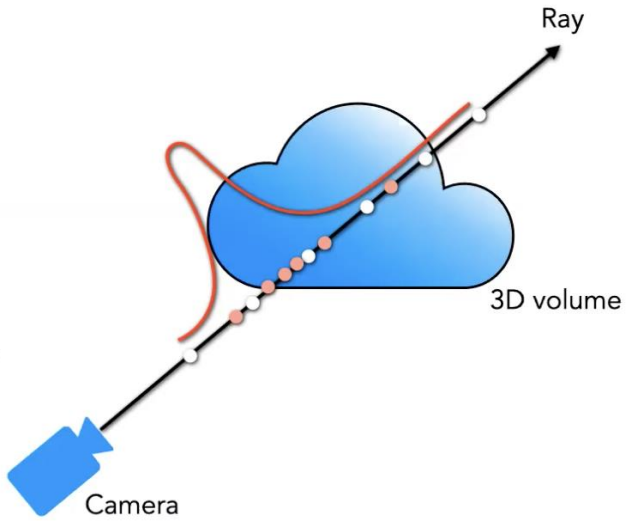
- When sampling, it is advantageous to sample the part **where the object is actually located rather than the part where the air is.**
- Divide into N_c(64) zones and draw a point from each zone.
- Use these points for inputs → density σ, color c_i
- Normalize to know the probability of which point contributes by how much

NeRF: Hierarchical volume sampling

- Fine Network

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

treat weights as probability distribution for new samples

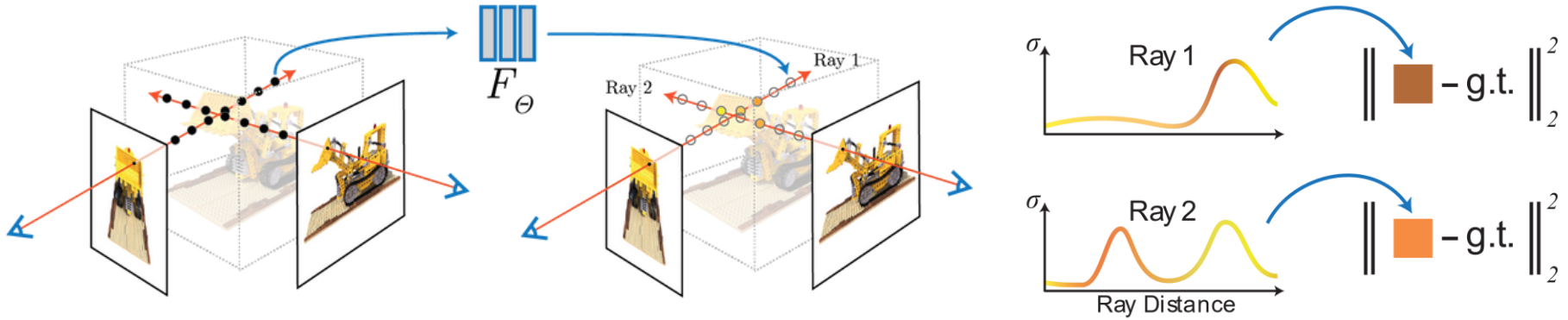


- Wants to sample more points that contribute more to color.
- A method that considers the **probability distribution obtained from the network** is needed.
- Inverse transform sampling → sample N_f(128) points

NeRF: Hierarchical volume sampling

[For total (N_c +N_f) **sampled points** along the ray]

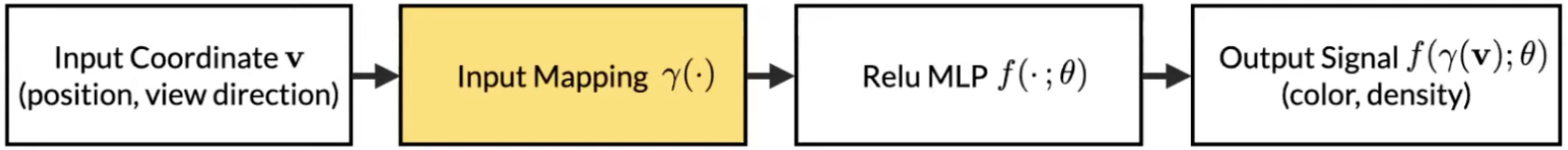
$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \underbrace{T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i}_{\text{Contribution weight}}, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right)$$



$$\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} \left[\underbrace{\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2}_{\text{Rendering loss for the "Coarse Network"}} + \underbrace{\left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2}_{\text{Rendering loss for the "Fine Network"}} \right]$$

NeRF: Positional Encoding

- How to represent high-resolution complex scene?
- **Positional Encoding** + Hierarchical sampling

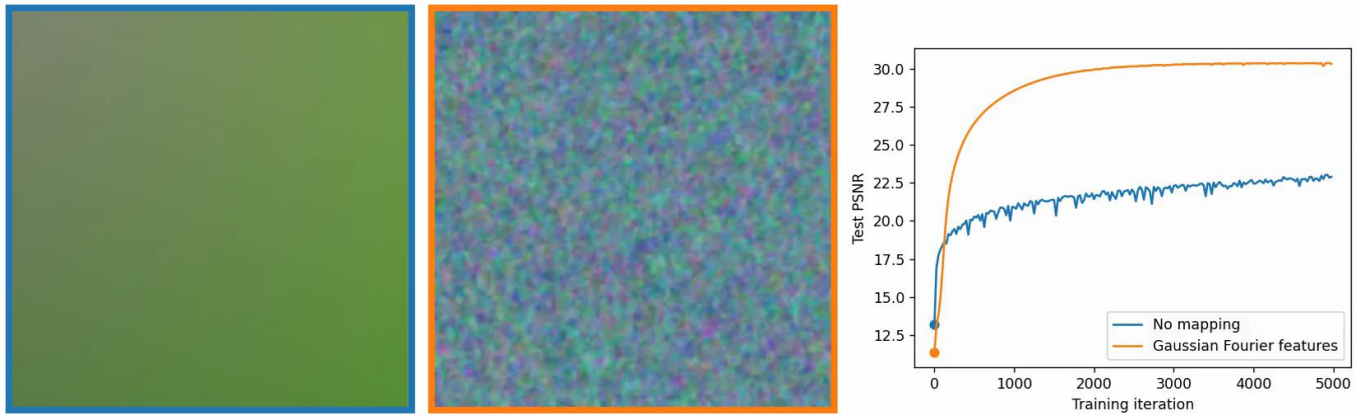


Positional Encoding [1]: $\gamma(\mathbf{v}) = [\cos(2^0\mathbf{v}), \sin(2^0\mathbf{v}), \ldots, \cos(2^{L-1}\mathbf{v}), \sin(2^{L-1}\mathbf{v})]$

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

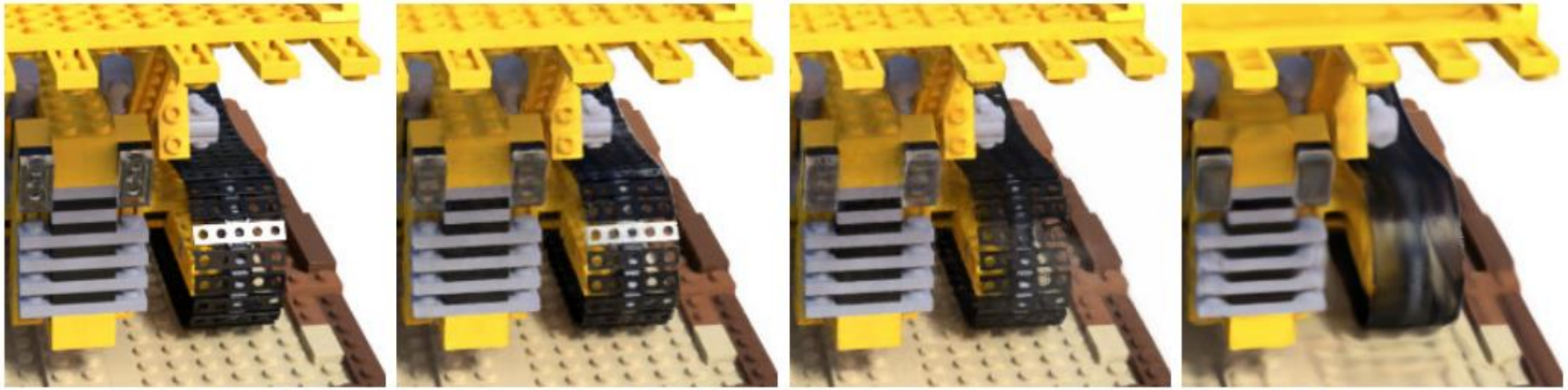
- Mapping 5D inputs to high dimension space by high frequency function
- This can represent high frequency data from real image well
- Fully-connected deep networks are biased to learn low frequencies faster.

NeRF: Positional Encoding

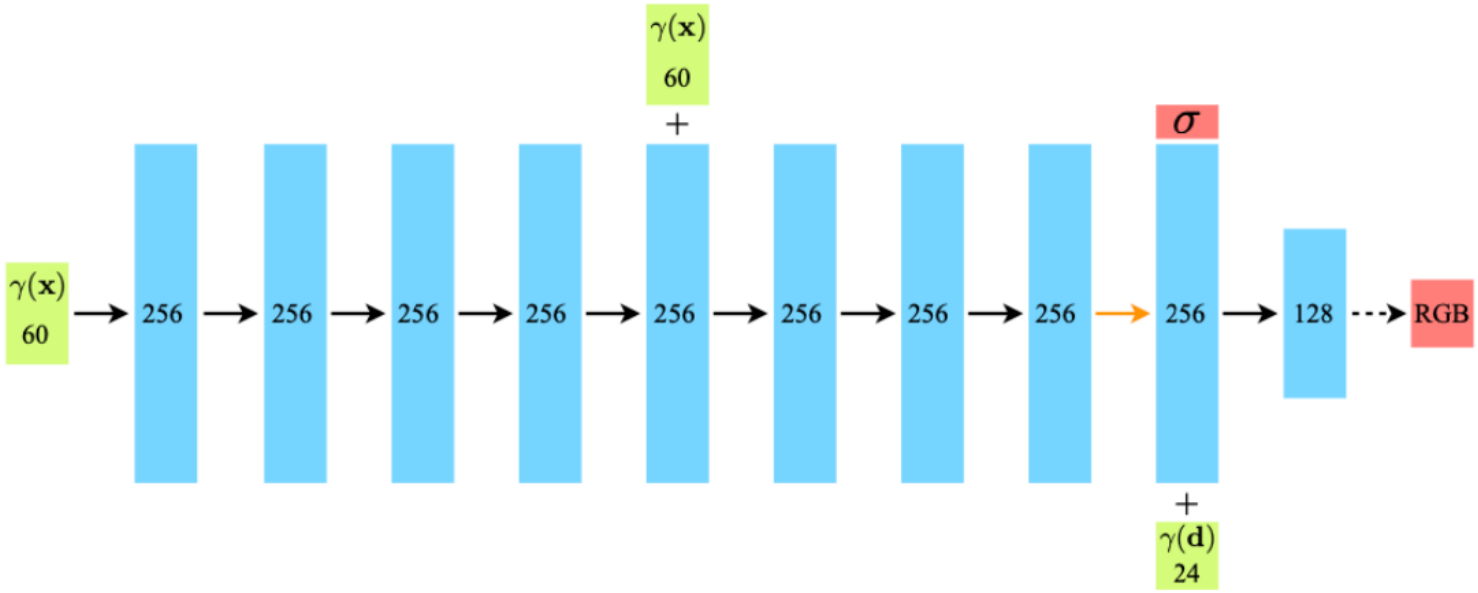


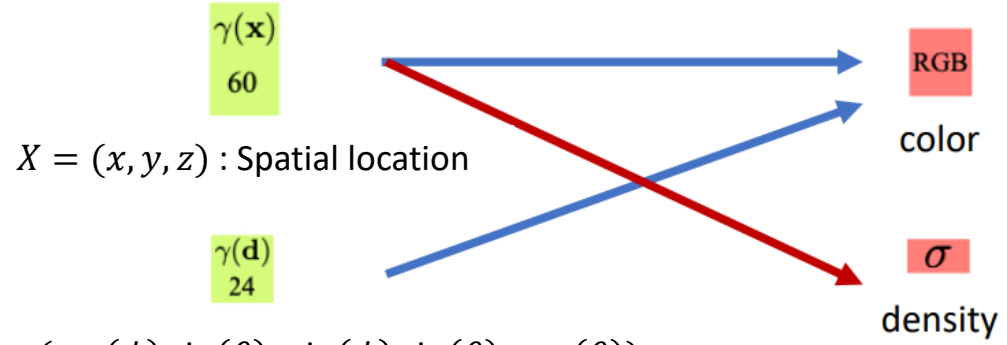
Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains NeurIPS 2020 (spotlight)

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



NeRF: Positional Encoding

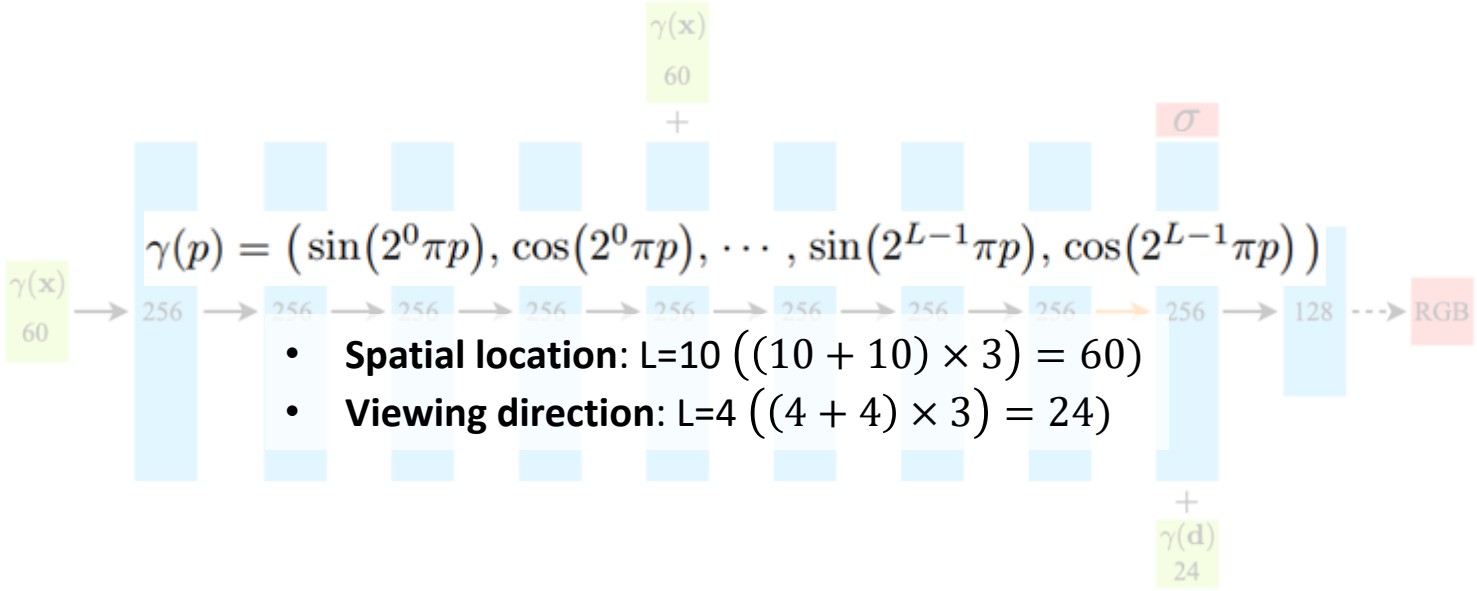




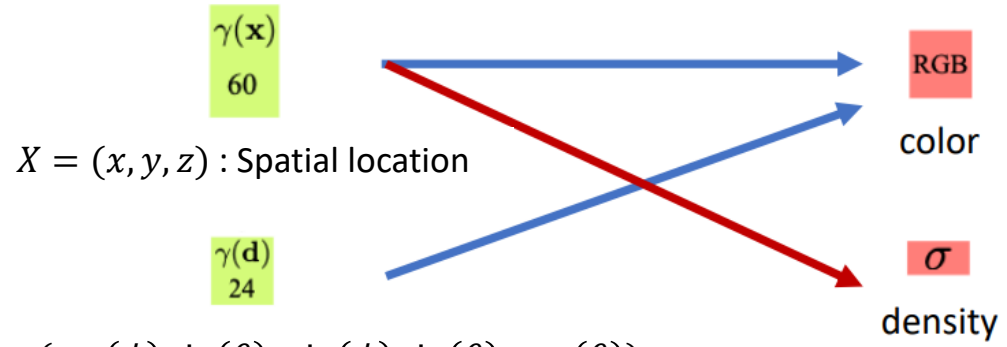
$X = (x, y, z)$: Spatial location

$\mathrm{d} = (\cos(\phi)\sin(\theta), \sin(\phi)\sin(\theta), \cos(\theta))$
: Express 2D viewing direction as 3D Cartesian unit vector

NeRF: Positional Encoding

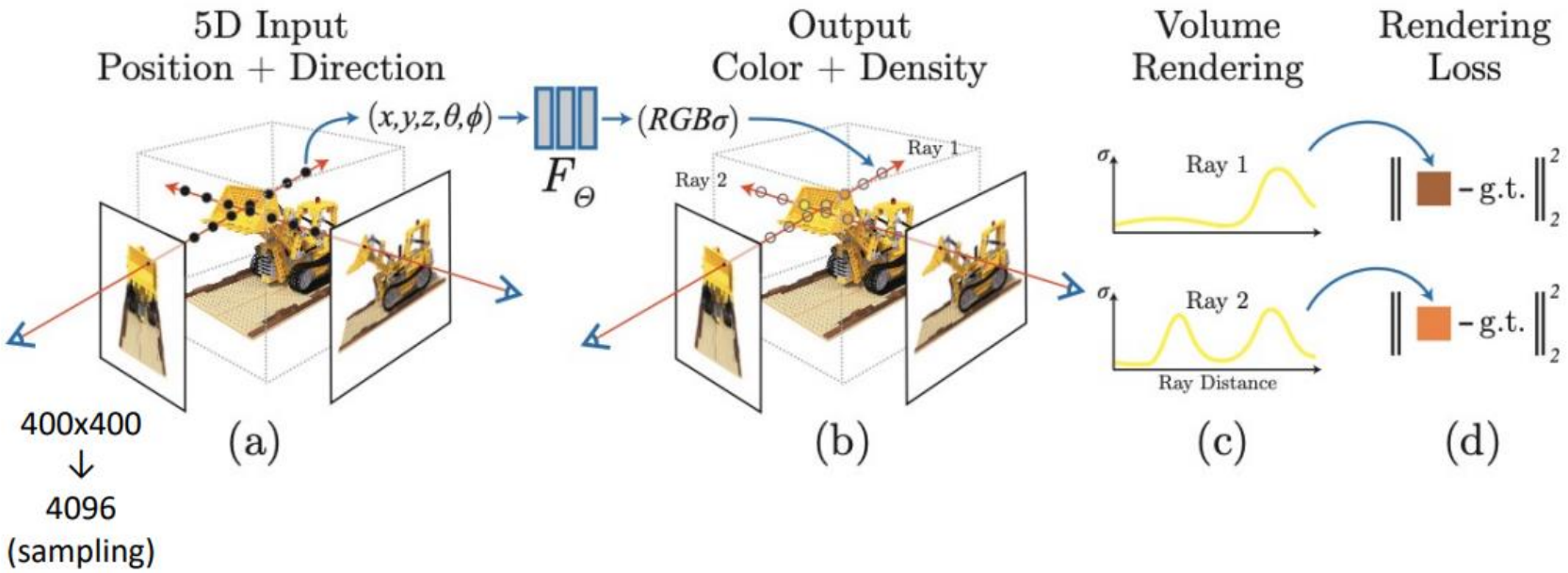


- **Spatial location**: L=10 $\left((10+10)\times 3\right)=60)$
- **Viewing direction**: L=4 $\left((4+4)\times 3\right)=24)$



$\mathrm{d}=(\cos(\phi)\sin(\theta),\sin(\phi)\sin(\theta),\cos(\theta))$
: Express 2D viewing direction as 3D Cartesian unit vector

NeRF: Implement Details



NeRF: Camera Poses

plotting.py | assignment1b.py | test_average_scores.py | scores.py | loss.py | solver.py | 7_Pytorch_num_Triplets.ipynb | colmap2nerf.py


```
{
  "camera_angle_x": 0.4653610018693349,
  "camera_angle_y": 0.8963341882297376,
  "fl_x": 1519.1629803400435,
  "fl_y": 1331.1246019650298,
  "k1": 0.038154211812483 38,
  "k2": 0.16787107186438188,
  "p1": -0.019151762239969828,
  "p2": 0.03950210566618519,
  "cx": 531.8204080667518,
  "cy": 676.7824883994541,
  "w": 720.0,
  "h": 1280.0,
  "aabb_scale": 16,
  "frames": [
    {
      "file_path": "./chair/10.jpg",
      "sharpness": 13.892344796497447,
      "transform_matrix": [
        [
          0.1651262605806869,
          0.008311703639940663,
          0.9862374124167034,
          3.9416348874494322
        ],
        [
          0.9861822566549145,
          -0.014913576905028 03,
          -0.1649913388118312,
          -0.6148413687504752
        ],
        [
          0.013336968385331369,
          0.9998542397808072,
          -0.010659477838493329,
          0.043308084030703554
        ],
        [
          0.0,
          0.0,
```


(we use ground truth camera poses, intrinsics, and bounds for synthetic data, and use the COLMAP structure-from-motion package [39] to estimate these parameters for real data). At each optimization iteration, we randomly sample

```
# For small synthetic scenes such as the original NeRF dataset, the default aabb_scale of 1 is ideal
# For natural scenes, start with an aabb_scale of 16, and subsequently reduce it
# The value can be directly edited in the transforms.json
# colmap2nerf.py에서 continue?묻는거 주석처리함.(오류나면 여기임)

# 원하는 class_[]의 폴더를 class로 변경해준다.
%cd /content/class
!python /content/instant-ngp/scripts/colmap2nerf.py --colmap_matcher exhaustive --run_colmap --aabb_scale 4
```

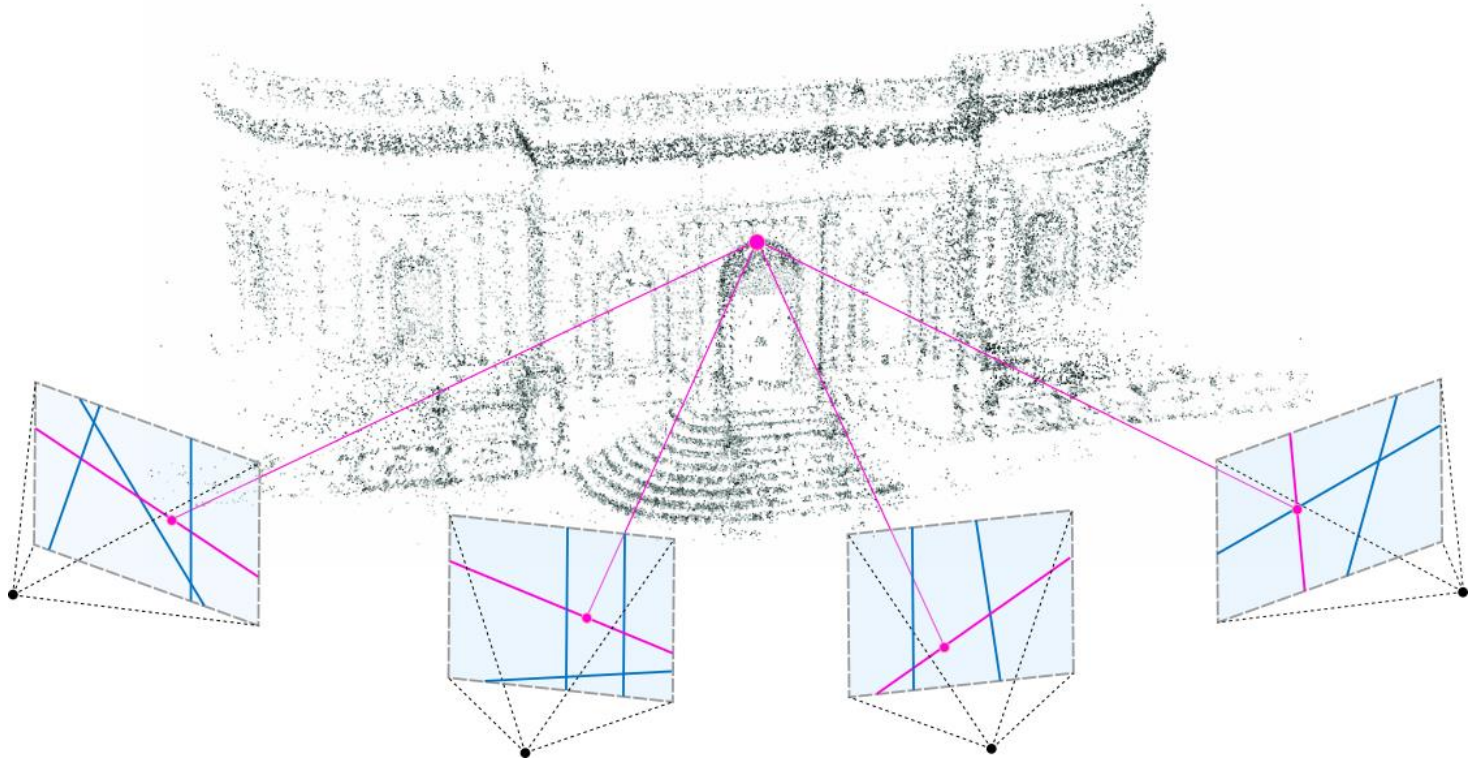
[18]

스트리밍 출력 내용이 길어서 마지막 5000줄이 삭제되었습니다.

74	1.930669e+02	1.05e-04	6.46e+00	2.79e+00	5.78e-01	3.43e+06	1	6.89e-04	5.10e-02
75	1.930668e+02	1.05e-04	6.46e+00	2.79e+00	5.78e-01	3.44e+06	1	7.45e-04	5.18e-02
76	1.930667e+02	1.04e-04	6.46e+00	2.79e+00	5.78e-01	3.45e+06	1	7.83e-04	5.26e-02
77	1.930665e+02	1.04e-04	6.46e+00	2.78e+00	5.78e-01	3.47e+06	1	7.43e-04	5.34e-02
78	1.930664e+02	1.03e-04	6.45e+00	2.78e+00	5.77e-01	3.48e+06	1	6.91e-04	5.41e-02
79	1.930663e+02	1.03e-04	6.45e+00	2.78e+00	5.77e-01	3.49e+06	1	6.72e-04	5.47e-02
80	1.930662e+02	1.03e-04	6.45e+00	2.78e+00	5.77e-01	3.51e+06	1	7.11e-04	5.55e-02
81	1.930661e+02	1.02e-04	6.45e+00	2.78e+00	5.77e-01	3.52e+06	1	1.01e-03	5.65e-02
82	1.930660e+02	1.02e-04	6.45e+00	2.78e+00	5.77e-01	3.53e+06	1	1.01e-03	5.75e-02
83	1.930659e+02	1.01e-04	6.45e+00	2.78e+00	5.77e-01	3.55e+06	1	7.97e-04	5.83e-02
84	1.930658e+02	1.01e-04	6.45e+00	2.77e+00	5.77e-01	3.56e+06	1	7.23e-04	5.91e-02
85	1.930657e+02	1.01e-04	6.45e+00	2.77e+00	5.77e-01	3.57e+06	1	7.04e-04	5.98e-02
86	1.930656e+02	1.00e-04	6.45e+00	2.77e+00	5.77e-01	3.58e+06	1	6.92e-04	6.05e-02
87	1.930655e+02	9.99e-05	6.45e+00	2.77e+00	5.77e-01	3.60e+06	1	6.47e-04	6.12e-02
88	1.930654e+02	9.95e-05	6.45e+00	2.77e+00	5.77e-01	3.61e+06	1	6.86e-04	6.19e-02
89	1.930653e+02	9.92e-05	6.45e+00	2.77e+00	5.77e-01	3.62e+06	1	7.16e-04	6.26e-02
90	1.930652e+02	9.88e-05	6.44e+00	2.77e+00	5.77e-01	3.64e+06	1	6.37e-04	6.32e-02
91	1.930651e+02	9.84e-05	6.44e+00	2.76e+00	5.77e-01	3.65e+06	1	6.38e-04	6.39e-02
92	1.930650e+02	9.80e-05	6.44e+00	2.76e+00	5.77e-01	3.66e+06	1	6.64e-04	6.45e-02
93	1.930649e+02	9.77e-05	6.44e+00	2.76e+00	5.77e-01	3.68e+06	1	7.31e-04	6.53e-02
94	1.930648e+02	9.73e-05	6.44e+00	2.76e+00	5.77e-01	3.69e+06	1	6.99e-04	6.60e-02
95	1.930647e+02	9.69e-05	6.44e+00	2.76e+00	5.77e-01	3.70e+06	1	7.31e-04	6.67e-02
96	1.930646e+02	9.66e-05	6.44e+00	2.76e+00	5.77e-01	3.72e+06	1	6.76e-04	6.74e-02
97	1.930646e+02	9.62e-05	6.44e+00	2.76e+00	5.77e-01	3.73e+06	1	6.38e-04	6.81e-02
98	1.930645e+02	9.58e-05	6.44e+00	2.75e+00	5.77e-01	3.75e+06	1	6.37e-04	6.87e-02
99	1.930644e+02	9.55e-05	6.44e+00	2.75e+00	5.77e-01	3.76e+06	1	6.91e-04	6.94e-02
100	1.930643e+02	9.51e-05	6.44e+00	2.75e+00	5.77e-01	3.77e+06	1	7.02e-04	7.01e-02

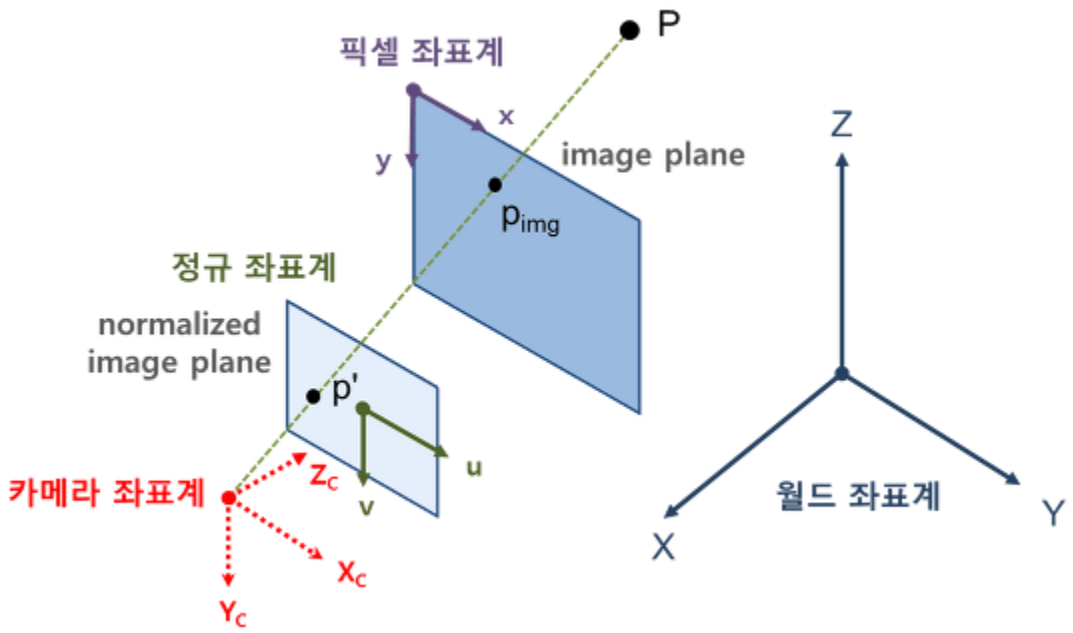
```
Bundle adjustment report
------------------------
    Residuals : 484
   Parameters : 374
   Iterations : 101
         Time : 0.070163 [s]
 Initial cost : 0.631598 [px]
   Final cost : 0.63158 [px]
  Termination : No convergence
```

NeRF: Camera Poses



Structure From Motion

NeRF: Camera Poses



$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \text{skew\_c}f_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= A[R \mid t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

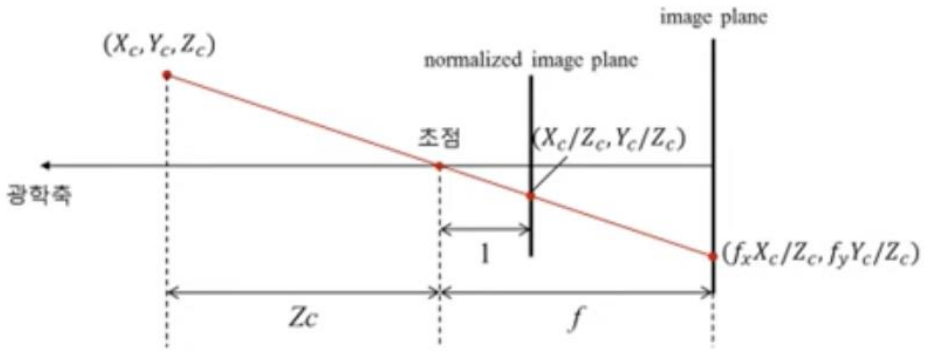
- (x, y, z) : World coordinate system
- $[R|t]$: Extrinsic parameter; Translate world coordinate system to camera coordinate system
- A : Intrinsic parameter

NeRF: Camera Poses

Intrinsic Parameter

$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \text{skew\_c}f_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= A[R \mid t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



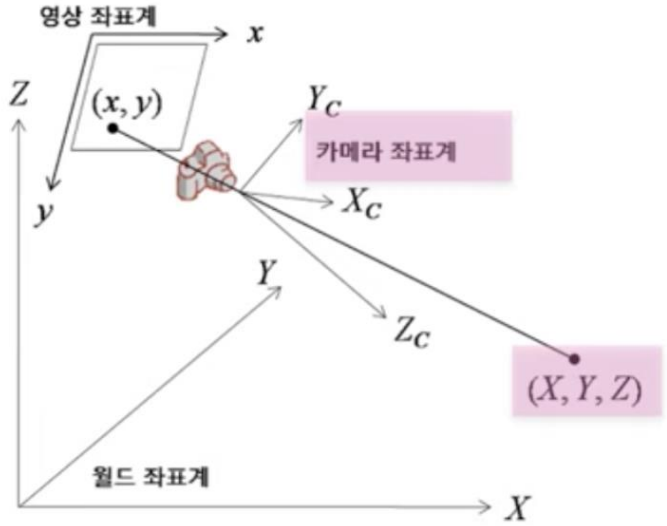
- f_x, f_y : focal length
- c_x, c_y : principal point

NeRF: Camera Poses

Extrinsic Parameter

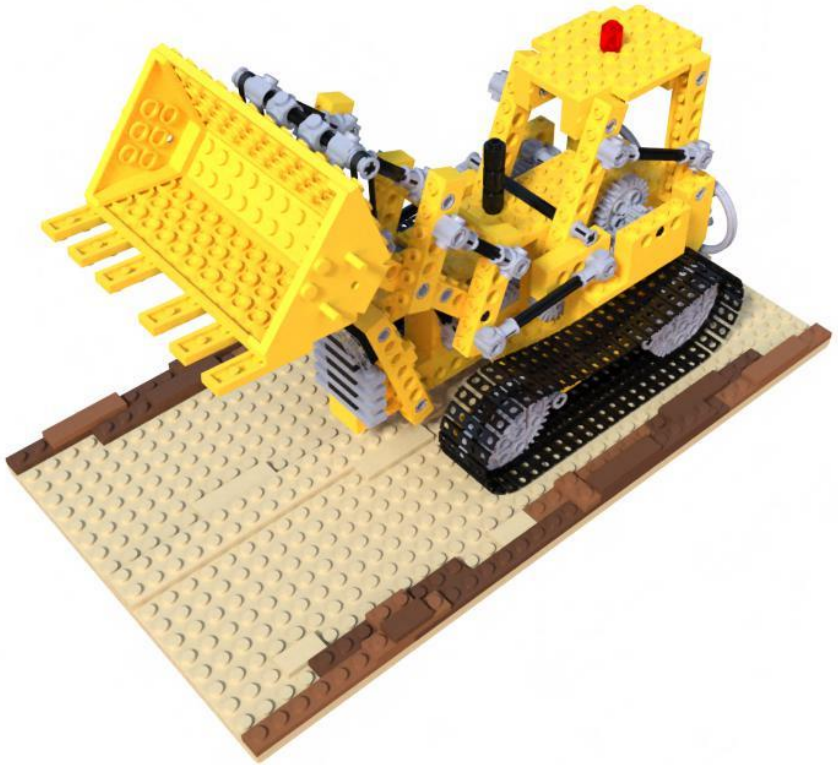
$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \text{skew\_c}f_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= A[R \mid t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

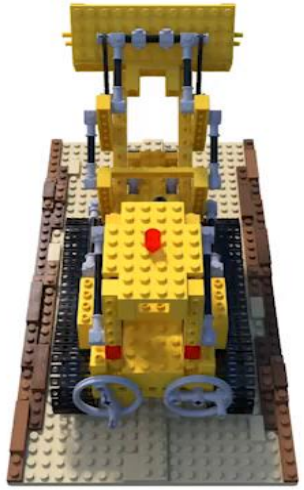


- It describes the position of the camera in a three-dimensional world coordinate system.
- Explain how far the camera is from the origin and at what angle it is rotated.
- That is, the transformation relationship between the camera coordinate system and the world coordinate system is described.
- It is expressed as a rotation and translation transformation between two coordinate systems.

NeRF: Results



NeRF: Results





NeRF: Results

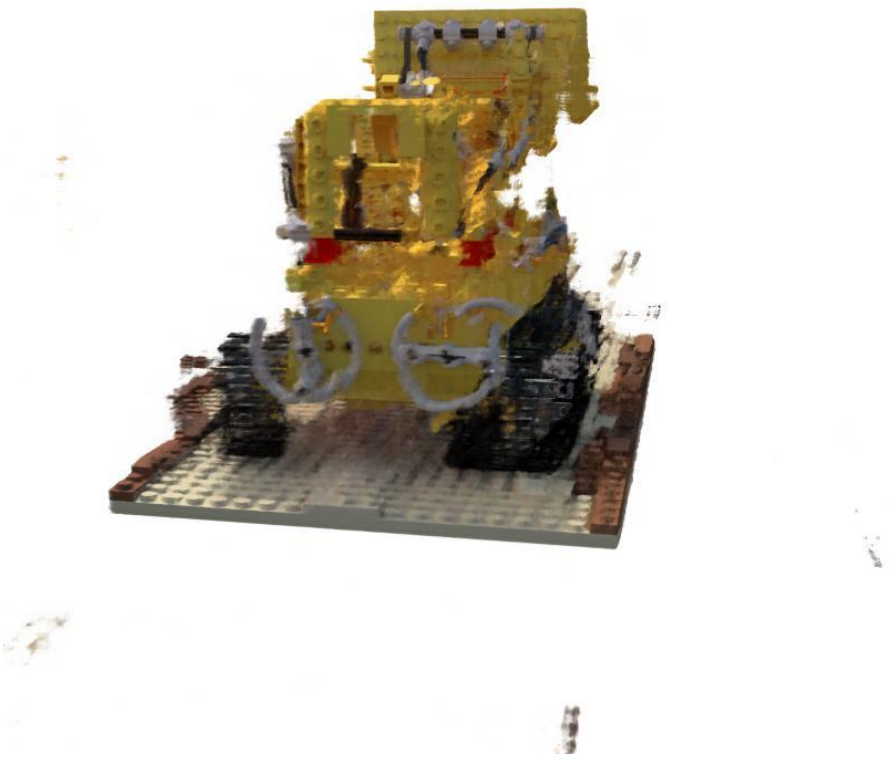
Weakness

- Slow
- Lots of input data
- Scene specific
- Fixed object
- Images taken in the same environment
- Camera parameters required
- Complex Representation

Weakness

Slow

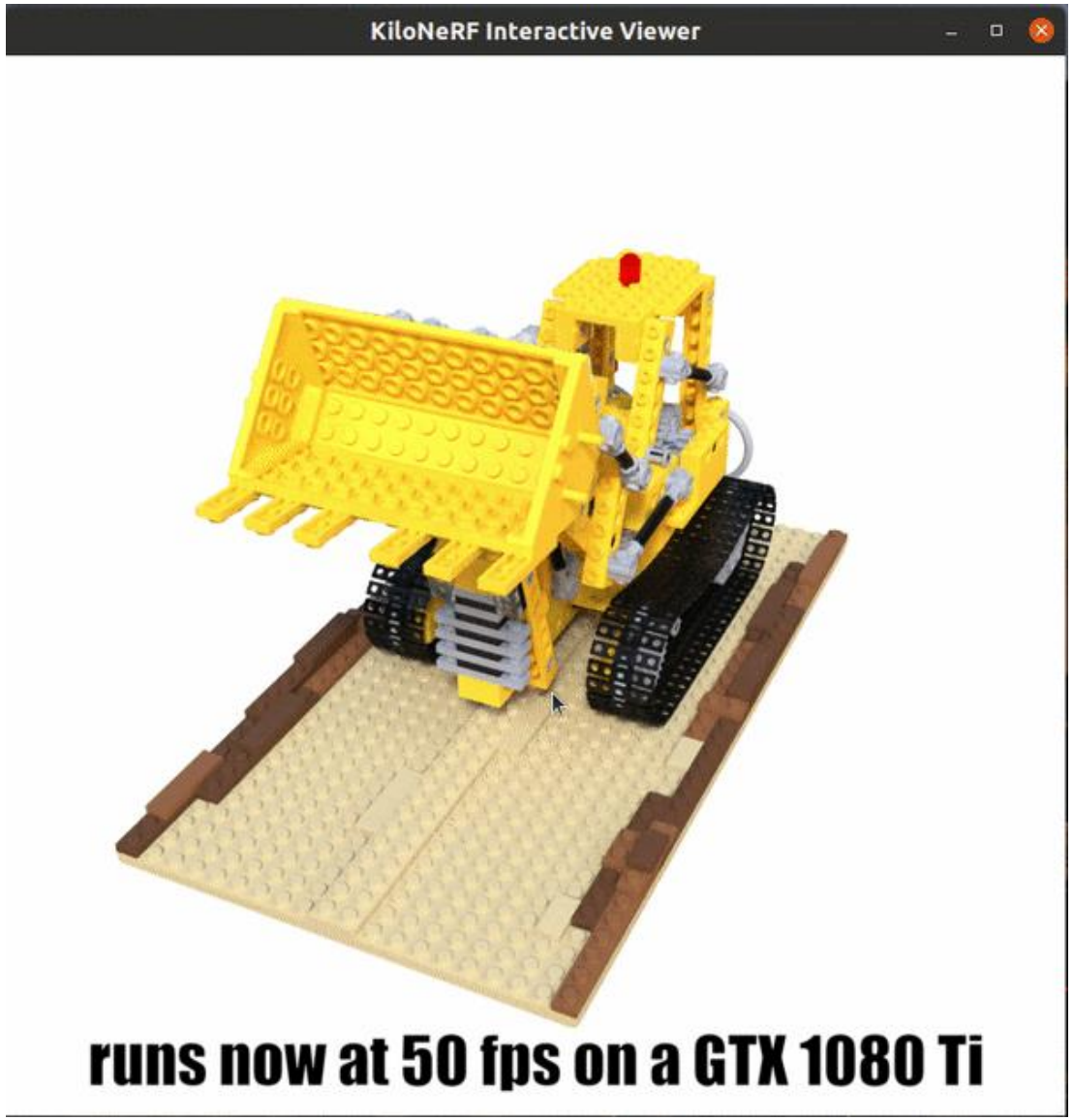
- NeRF takes more than a day to learn and 30 seconds to render a new image. Therefore, you cannot use NeRF in real time.



Weakness

Slow

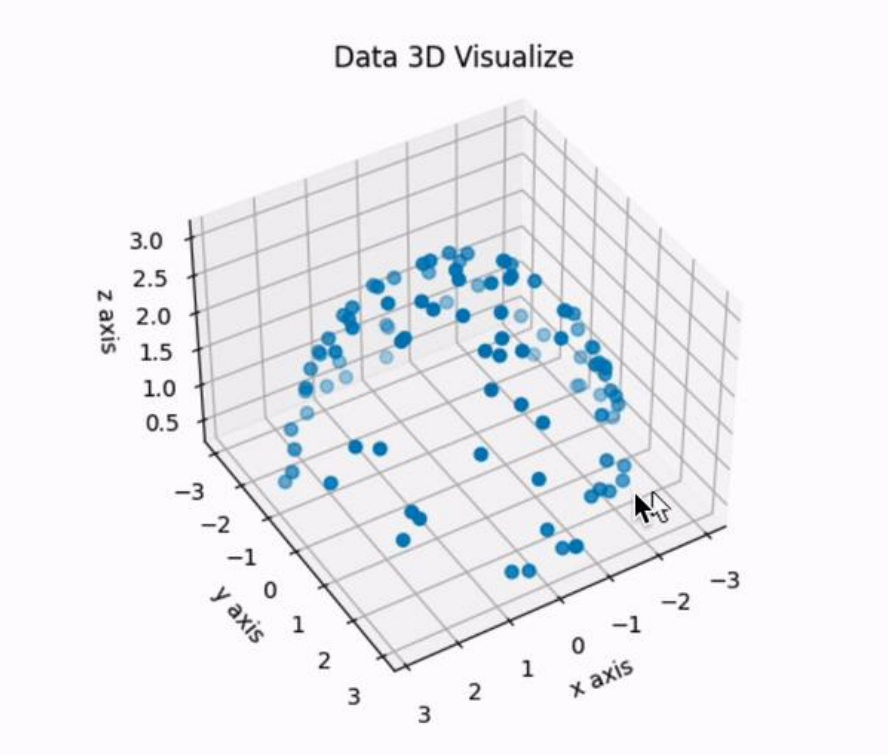
- PointNeRF (CVPR 2022)
- Plenoxels (CVPR2022)
- Instant NGP (SIGGRAPH 2022)
- EfficientNeRF(CVPR2022)
- Depth-supervised NeRF (CVPR2022)
- IBRNet (CVPR 2021)
- PlenOctrees (ICCV 2021)
- KiloNeRF (ICCV 2021)
- FastNeRF (ICCV 2021)
- MVSNeRF (ICCV 2021)
- NSVF (NeurIPS 2020)
- DVGO1, 2 (CVPR 2022)

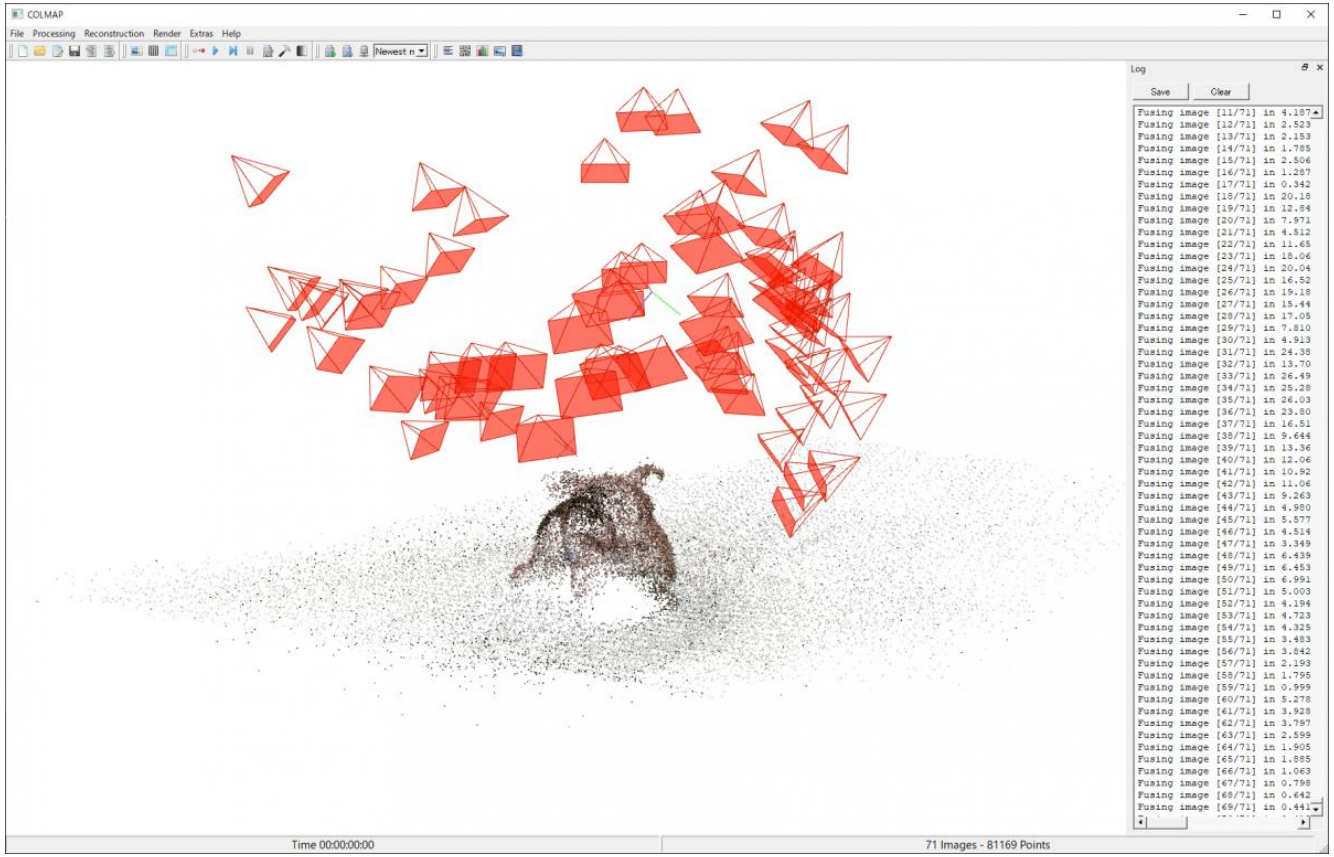


Weakness

Lots of input data

- NeRF requires a minimum of 50 pieces of learning data.RF. Realistically, it is not easy to take more than 50 pictures of the same object.

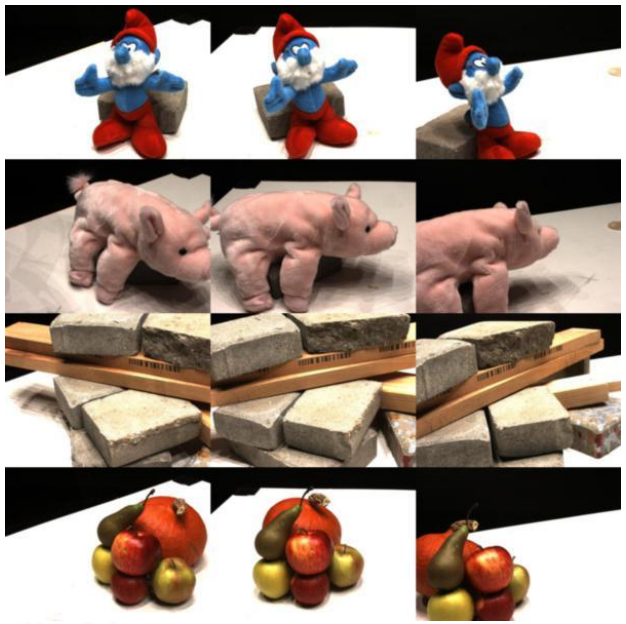


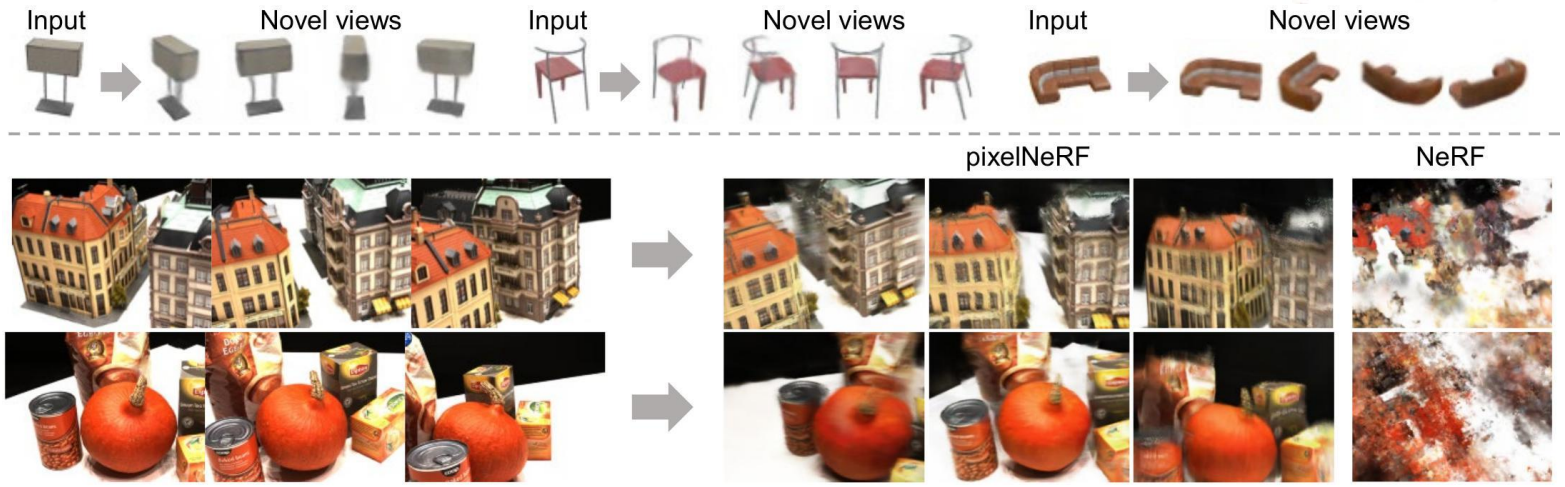


Weakness

Lots of input data

- Depth-supervised NeRF (CVPR2022)
- MVSNeRF (ICCV 2021)
- metaNeRF (CVPR 2021)
- PixelNeRF (CVPR 2021)
- DietNeRF (ICCV 2021)





Weakness

Scene specific

- NeRF must learn one model to generate an image of one object. For example, you cannot create an image of a building with a tree-learned model. To create images of so many objects in the world, you have to learn so many models.

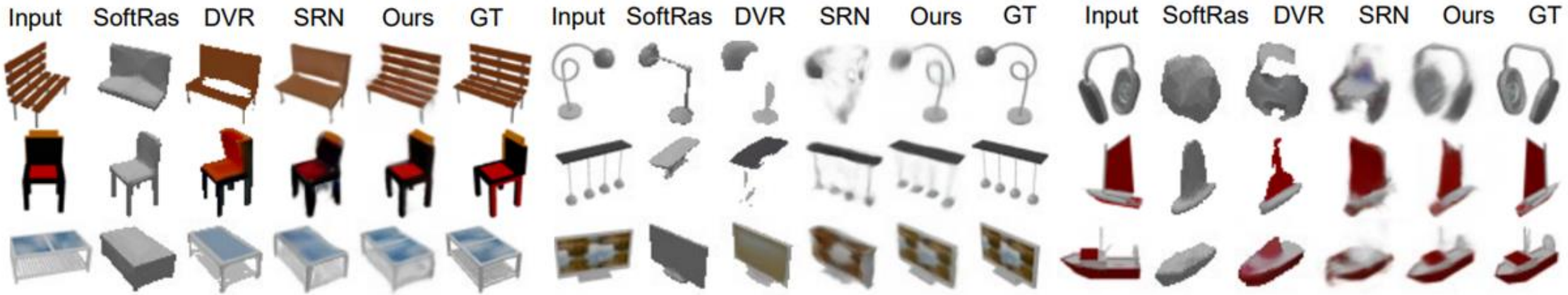


Figure 5: **Category-agnostic single-view reconstruction.** Going beyond the SRN benchmark, we train a single model to the 13 largest ShapeNet categories; we find that our approach produces superior visual results compared to a series of strong baselines. In particular, the model recovers fine detail and thin structure more effectively, even for outlier shapes. Quite visibly, images on monitors and tabletop textures are accurately reproduced; baselines representing the scene as a single latent vector cannot preserve such details of the input image. SRN's test-time latent inversion becomes less reliable as well in this setting. The corresponding quantitative evaluations are available in Table 4. Due to space constraints, we show objects with interesting properties here. Please see the supplemental for sampled results.

Weakness

Scene specific

- PixelNeRF (CVPR 2021)
- IBRNet (CVPR 2021)
- MVSNeRF (ICCV 2021)

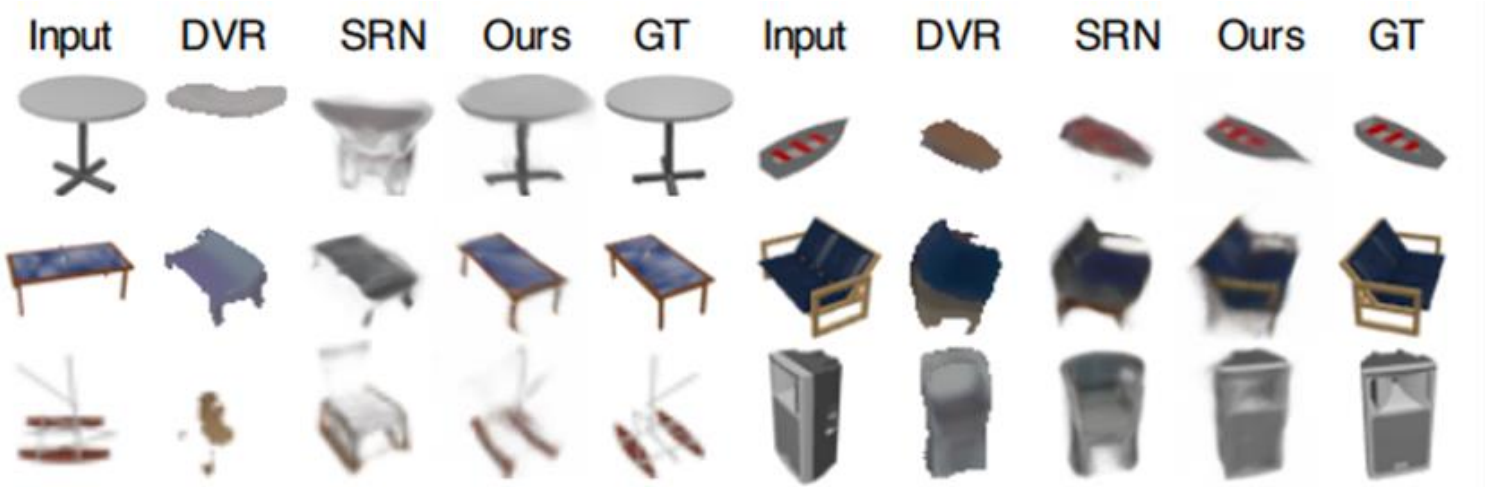


Figure 6: **Generalization to unseen categories.** We evaluate a model trained on planes, cars, and chairs on 10 unseen ShapeNet categories. We find that the model is able to synthesize reasonable views even in this difficult case.

Weakness

Fixed object

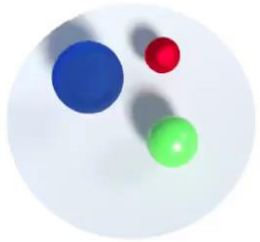
- NeRF can only be learned for motionless objects.

(a) Capture Process (b) Input (c) Nerfie (d) Nerfie Depth

Weakness

Fixed object

- BANMo (CVPR 2022)
- Nerfies (ICCV 2021)
- HyperNeRF (SIGGRAPH 2021)
- D-NeRF (CVPR 2021)



Weakness

Images taken in the same environment

- In reality, even the same object may look different depending on time or season. NeRF performs well only when learning various images taken in the same environment. Learning is difficult for objects taken in various environments.
- Photometric variation
- Transient objects

Weakness

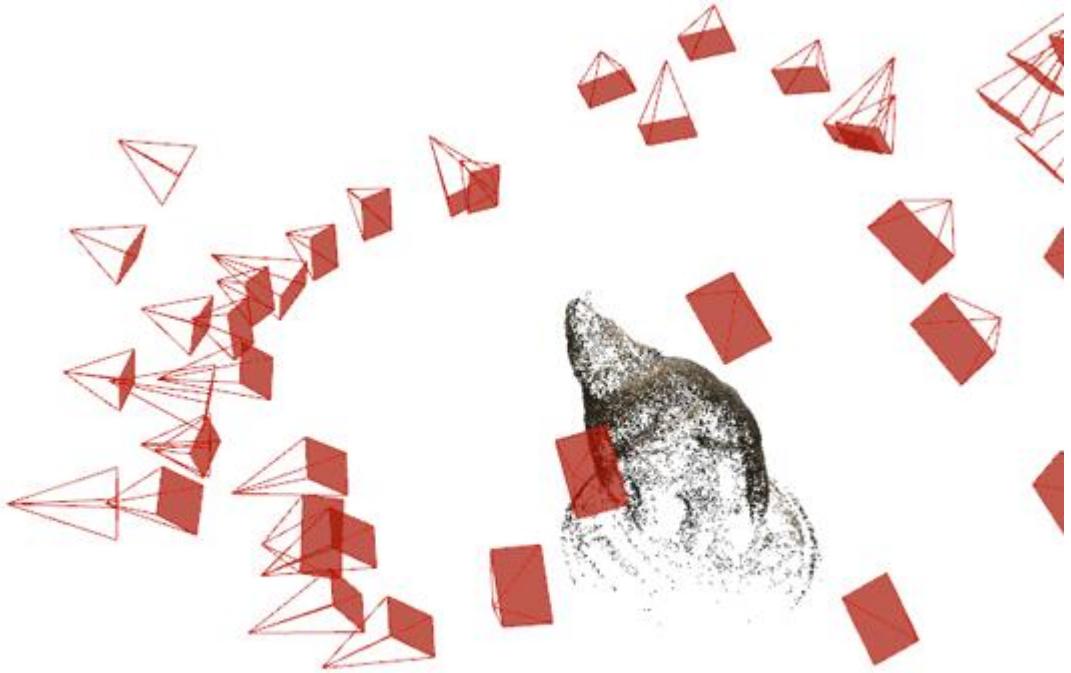
Images taken in the same environment

- NeRFPlayer: A Streamable Dynamic Scene Representation with Decomposed Neural Radiance Fields
- ShadeGAN (NeurIPS 2021)
- NeRS (NeuralPS 2021)
- NeRV (CVPR 2021)
- NeRF in the wild (CVPR 2021)
- NeRD (ICCV 2021)

Weakness

Camera parameters required

- NeRF requires a variety of information, including not only images but also parameters and directions of the camera that took the images. In general, it's not easy to know that information when we're photographing an object.



Weakness

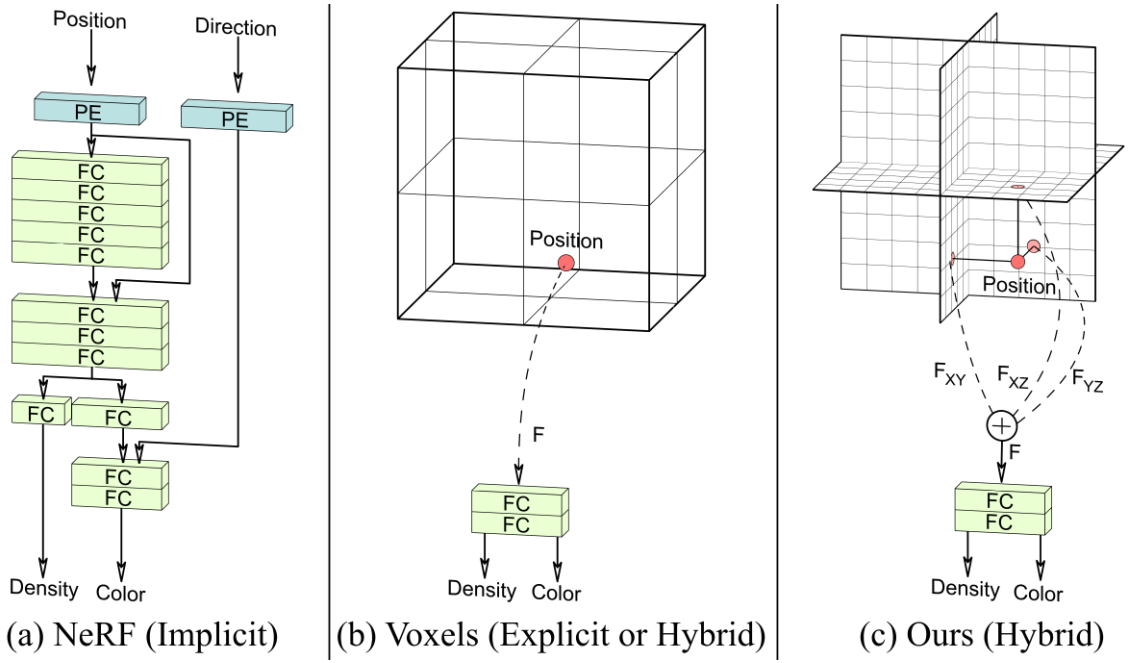
Camera parameters required

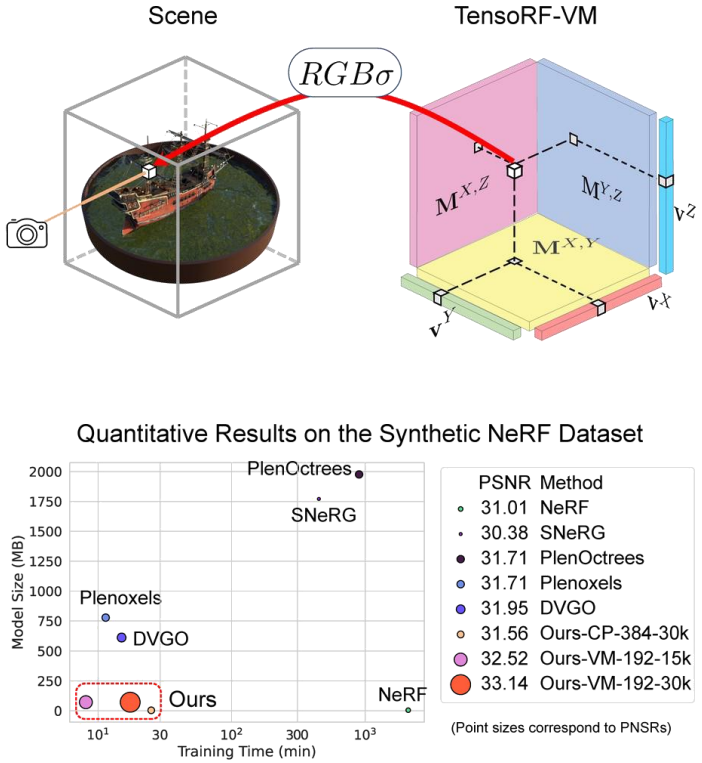
- iNeRF (IROS 2021)
- NeRF– (2021)
- GNeRF (ICCV2021)
- BARF (ICCV 2021)
- SCNeRF (ICCV 2021)
- Structure-Aware NeRF without Posed Camera via Epipolar Constraint
- GARF: Gaussian Activated Radiance Fields for High Fidelity Reconstruction and Pose Estimation
- Robustifying the Multi-Scale Representation of Neural Radiance Fields (BMVC 2022)

Weakness

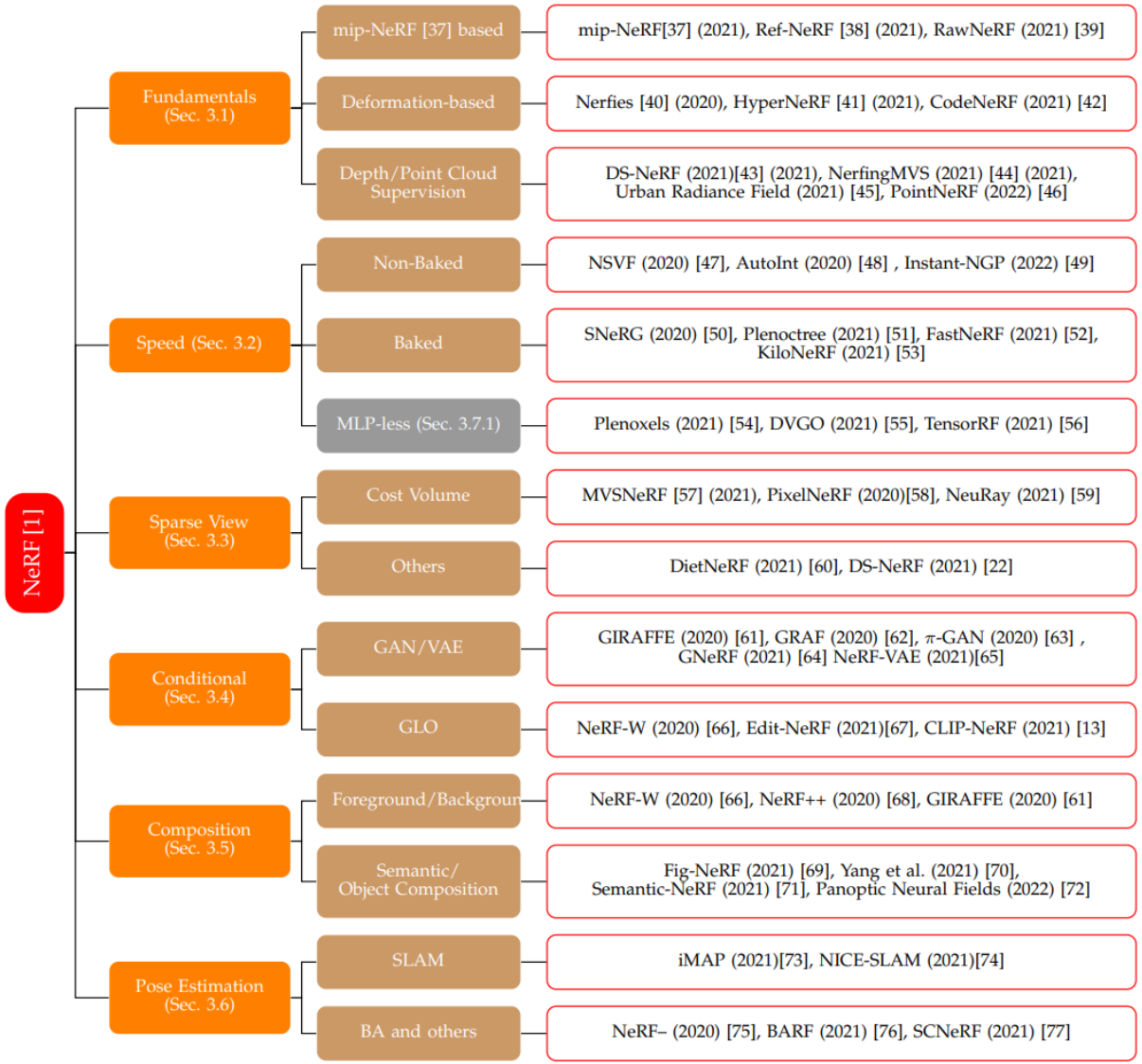
Complex Representation

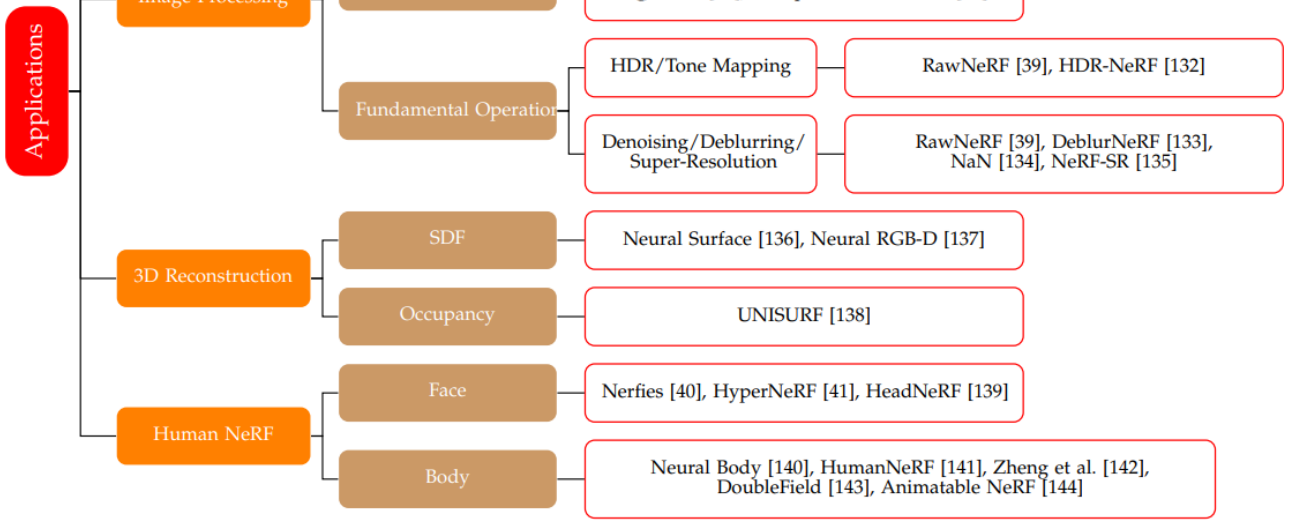
- TensoRF (ECCV2022)
- EG3D: Efficient Geometry-aware 3D Generative Adversarial Networks (CVPR 2022)





What now..


NeRF [1]
Fundamentals (Sec. 3.1)
mip-NeRF [37] based
mip-NeRF[37] (2021), Ref-NeRF [38] (2021), RawNeRF (2021) [39]
Deformation-based
Nerfies [40] (2020), HyperNeRF [41] (2021), CodeNeRF (2021) [42]
Depth/Point Cloud Supervision
DS-NeRF (2021)[43] (2021), NerfingMVS (2021) [44] (2021), Urban Radiance Field (2021) [45], PointNeRF (2022) [46]
Speed (Sec. 3.2)
Non-Baked
NSVF (2020) [47], AutoInt (2020) [48] , Instant-NGP (2022) [49]
Baked
SNeRG (2020) [50], Plenoctree (2021) [51], FastNeRF (2021) [52], KiloNeRF (2021) [53]
MLP-less (Sec. 3.7.1)
Plenoxels (2021) [54], DVGO (2021) [55], TensorRF (2021) [56]
Sparse View (Sec. 3.3)
Cost Volume
MVSNeRF [57] (2021), PixelNeRF (2020)[58], NeuRay (2021) [59]
Others
DietNeRF (2021) [60], DS-NeRF (2021) [22]
Conditional (Sec. 3.4)
GAN/VAE
GIRAFFE (2020) [61], GRAF (2020) [62], π-GAN (2020) [63] , GNeRF (2021) [64] NeRF-VAE (2021)[65]
GLO
NeRF-W (2020) [66], Edit-NeRF (2021)[67], CLIP-NeRF (2021) [13]
Composition (Sec. 3.5)
Foreground/Backgroun
NeRF-W (2020) [66], NeRF++ (2020) [68], GIRAFFE (2020) [61]
Semantic/ Object Composition
Fig-NeRF (2021) [69], Yang et al. (2021) [70], Semantic-NeRF (2021) [71], Panoptic Neural Fields (2022) [72]
Pose Estimation (Sec. 3.6)
SLAM
iMAP (2021)[73], NICE-SLAM (2021)[74]
BA and others
NeRF– (2020) [75], BARF (2021) [76], SCNeRF (2021) [77]


Applications
Urban
Street Level
Urban Radiance Field [45], BlockNeRF [127]
Remote Sensing/Aerial
MegaNeRF [128], BungeeNeRF [129], S-NeRF [130]
Image Processing
Editing
ClipNeRF [13], EditNeRF [67], CodeNeRF [42], Yang et al. [70], CoNeRF [84]
Labelling
Semantic-NeRF [71], NeSF [131], Fig-NeRF [69], Panoptic Neural Fields [72]
Fundamental Operatio
HDR/Tone Mapping
RawNeRF [39], HDR-NeRF [132]
Denoising/Deblurring/ Super-Resolution
RawNeRF [39], DeblurNeRF [133], NaN [134], NeRF-SR [135]
3D Reconstruction
SDF
Neural Surface [136], Neural RGB-D [137]
Occupancy
UNISURF [138]
Human NeRF
Face
Nerfies [40], HyperNeRF [41], HeadNeRF [139]
Body
Neural Body [140], HumanNeRF [141], Zheng et al. [142], DoubleField [143], Animatable NeRF [144]

What now..

7. arXiv:2212.13056 [pdf, other] cs.CV

MonoNeRF: Learning a Generalizable Dynamic Radiance Field from Monocular Videos

Authors: Fengrui Tian, Shaoyi Du, Yueqi Duan

Abstract: In this paper, we target at the problem of learning a generalizable dynamic radiance field from monocular videos. Different from most existing **NeRF** methods that are based on multiple views, monocular videos only contain one view at each timestamp, thereby suffering from ambiguity along the view direction in estimating point features and scene flows. Previous... ▽ More

Submitted 26 December, 2022; **originally announced** December 2022.

8. arXiv:2212.12871 [pdf, other] cs.CV cs.GR

PaletteNeRF: Palette-based Color Editing for NeRFs

Authors: Qiling Wu, Jianchao Tan, Kun Xu

Abstract: Neural Radiance Field (**NeRF**) is a powerful tool to faithfully generate novel views for scenes with only sparse captured images. Despite its strong capability for representing 3D scenes and their appearance, its editing ability is very limited. In this paper, we propose a simple but effective extension of vanilla... ▽ More

Submitted 25 December, 2022; **originally announced** December 2022.

Comments: 12 pages, 10 figures

9. arXiv:2212.11966 [pdf, other] cs.CV

Removing Objects From Neural Radiance Fields

Authors: Silvan Weder, Guillermo Garcia-Hernando, Aron Monszpart, Marc Pollefeys, Gabriel Brostow, Michael Firman, Sara Vicente

Abstract: Neural Radiance Fields (**NeRFs**) are emerging as a ubiquitous scene representation that allows for novel view synthesis. Increasingly,... ▽ More

Submitted 22 December, 2022; **originally announced** December 2022.

10. arXiv:2212.10950 [pdf, other] cs.CV

Incremental Learning for Neural Radiance Field with Uncertainty-Filtered Knowledge Distillation

Authors: Mengqi Guo, Chen Li, Gim Hee Lee

Abstract: Recent neural radiance field (**NeRF**) representation has achieved great success in the tasks of novel view synthesis and 3D reconstruction. However, they suffer from the catastrophic forgetting problem when continuously learning from streaming data without revisiting the previous training data This limitation prohibits the application of existing... ▽ More

Submitted 21 December, 2022; **originally announced** December 2022.

11. arXiv:2212.10699 [pdf, other] cs.CV cs.GR

PaletteNeRF: Palette-based Appearance Editing of Neural Radiance Fields

Authors: Zhengfei Kuang, Fujun Luan, Sai Bi, Zhixin Shu, Gordon Wetzstein, Kalyan Sunkavalli

Abstract: ...can be efficiently edited while maintaining photorealism. In this work, we present PaletteNeRF, a novel method for photorealistic appearance editing of neural radiance fields (**NeRF**) based on 3D color decomposition. Our method decomposes the appearance of each 3D point into a linear combination of palette-based bases (i.e., 3D segmentations defined by a gro... ▽ More

Submitted 20 December, 2022; **originally announced** December 2022.

12. arXiv:2212.09735 [pdf, other] cs.CV

Correspondence Distillation from NeRF-based GAN

Authors: Yushi Lan, Chen Change Loy, Bo Dai

Abstract: The neural radiance field (**NeRF**) has shown promising results in preserving the fine details of objects and scenes. However, unlike mesh-based representations, it remains an open problem to build dense correspondences across different... ▽ More

Submitted 19 December, 2022; **v1** submitted 19 December, 2022; **originally announced** December 2022.

Comments: Project page: https://nirvanalan.github.io/projects/DDF/index.html

13. arXiv:2212.09330 [pdf, other] cs.CV doi 10.1145/3571600.3571643

StyleTRF: Stylizing Tensorial Radiance Fields

Authors: Rahul Goel, Sirikonda Dhawal, Saurabh Saini, P. J. Narayanan

Abstract: ...method is used to stylize the captured appearance by training its network jointly or iteratively with the structure capture network. The state-of-the-art SNeRF method trains the **NeRF** and stylization network in an alternating manner. These methods have high training time and require joint optimization. In this work, we present StyleTRF, a compact, quick-to-op... ▽ More

Submitted 19 December, 2022; **originally announced** December 2022.

Comments: Accepted at ICVGIP-2022

What now..

랭킹뉴스

한경 금융

'너프 폭락'...존바론은 알고 있었다?

유하늘 기자

입력 2017.07.18 17:29 수정 2017.07.19 01:01 지면 A18

가가

뉴스카페

"NeRF, 거품 붕괴 단계"
일부선 "조정장세" 반론도

너프(NeRF) 가격 폭락을 미리 예견한 듯한 그래프가 온라인에서 화제다.

오늘의 주요뉴스

단독 생산량·고용 '勞 뜻대로'... 기아 전기차 新공장 타결

힘받는 '美 긴축 속도 조절론'에...주식·채권 거침없는 ...

단독 '강소로펌' LKB·린, 합병 추진...대형로펌 도약 기틀

What now..

너프 붐은 온다...

Thank you!
Q & A