Neural Graphics I

Multimodal Generative AI Theories and Applications

Lecture 8

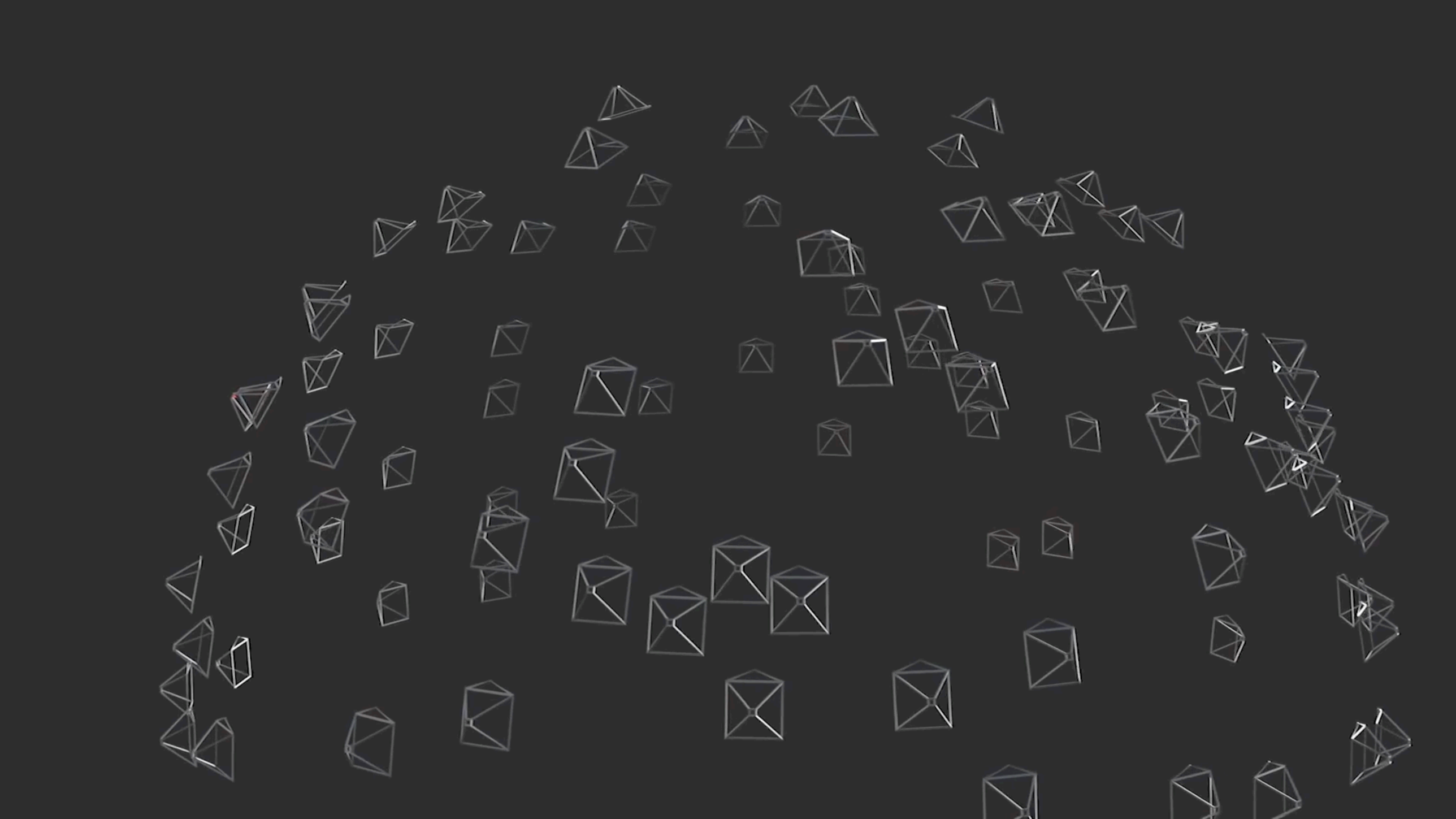
Jin-Hwa Kim and Sangdoo Yun

Movie credit: Matthew Tancik at https://www.youtube.com/watch?v=JuH79E8rdKc

NAVER AI LAB

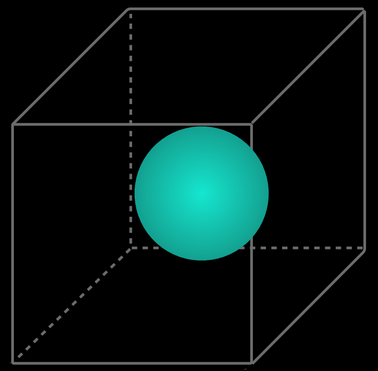
Neural Graphics series

- *Neural Graphics I, today*
  - NeRF and pinhole camera system
- *Neural Graphics II on Nov 14*
  - Camera pose refinement, Hash Grid, and surface modeling
- *Neural Graphics III on Nov 28*
  - 3DGS, spherical harmonics, and reflective modeling

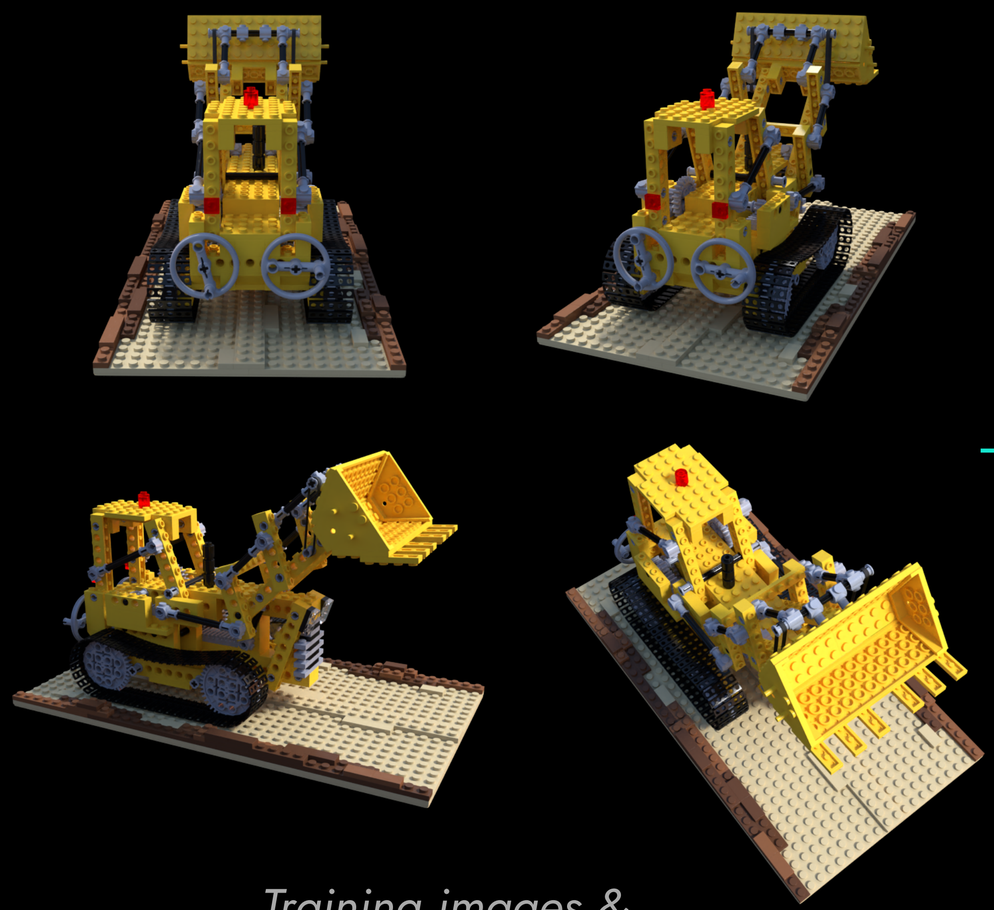


Neural radiance fields

- Neural radiance fields (NeRFs) represent a scene by neural networks and synthesize novel-views (Mildenhall et al., 2020).
- The *radiance* means "light or heat as emitted or reflected by something."
- The NeRFs is a learning framework for the *optics*.

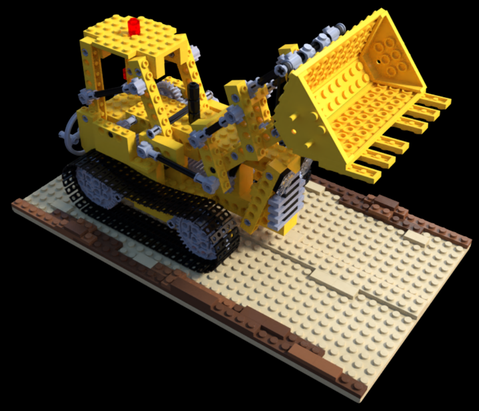


Novel-view synthesis

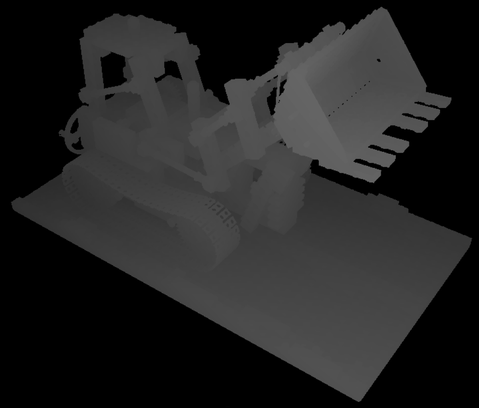


Training images & known camera poses

Novel-view synthesis with a camera pose



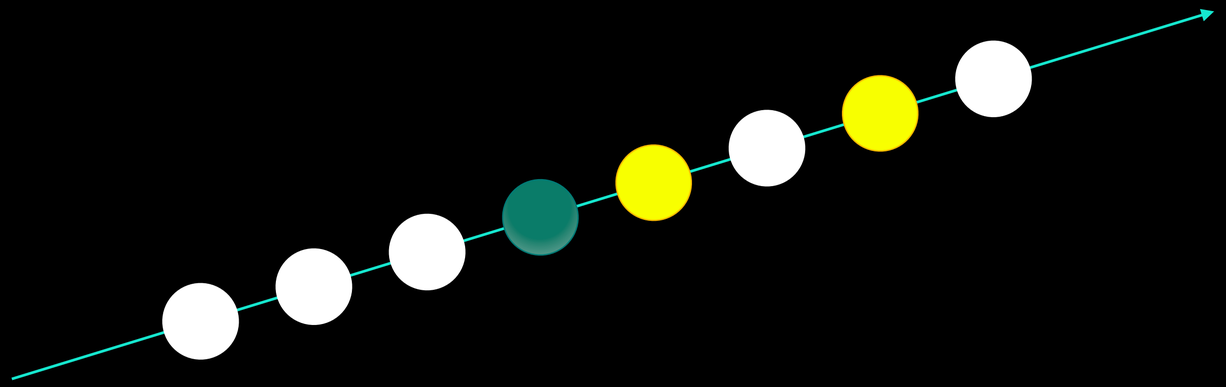
Rendered image



Depth map

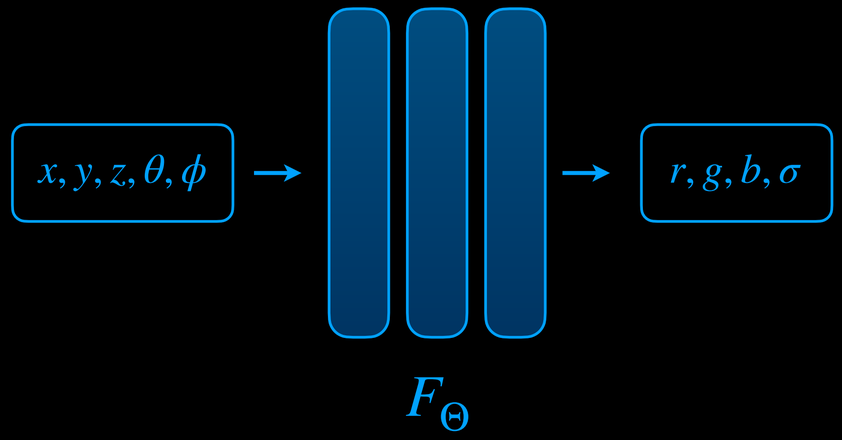
Radiance fields

- In 3D computer graphics, a *voxel* represents a value on a regular grid in continuous three-dimensional space.
- However, NeRFs represent each evaluated point using MLPs with two attributes: volume density σ and emitted color $\mathbf{c}$. *That's why we refer to it as radiance!*



The term "voxel" is a compound word combining volume and pixel.

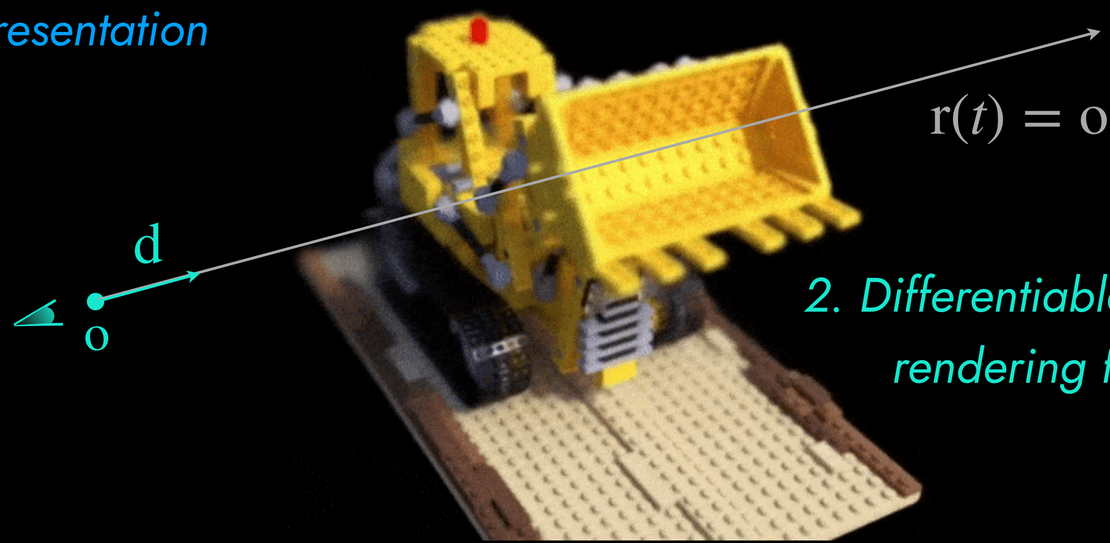
Three components



1. Neural volumetric 3D scene representation

$$\mathscr{L} = \sum_{\mathrm{r}\in\mathscr{R}} \|\hat{C}(\mathrm{r}) - C(\mathrm{r})\|_2^2$$

3. Optimization to synthesize all training images



2. Differentiable volumetric rendering function

Demonstration

- A vanilla NeRF can give …

https://www.matthewtancik.com/nerf

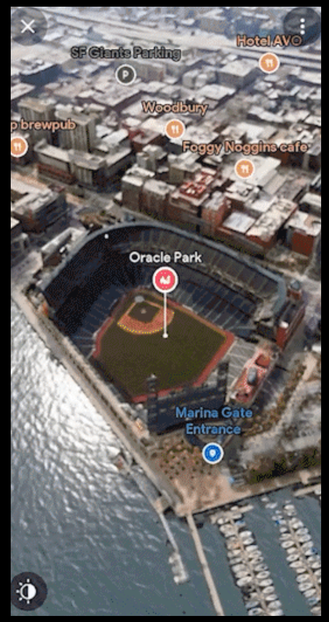
Commercial products

using NeRFs and classical methods

Digital Twin Worlds Created by ALIKE Solutions (https://www.youtube.com/watch?v=ZB37ljqovk8&t=69s)

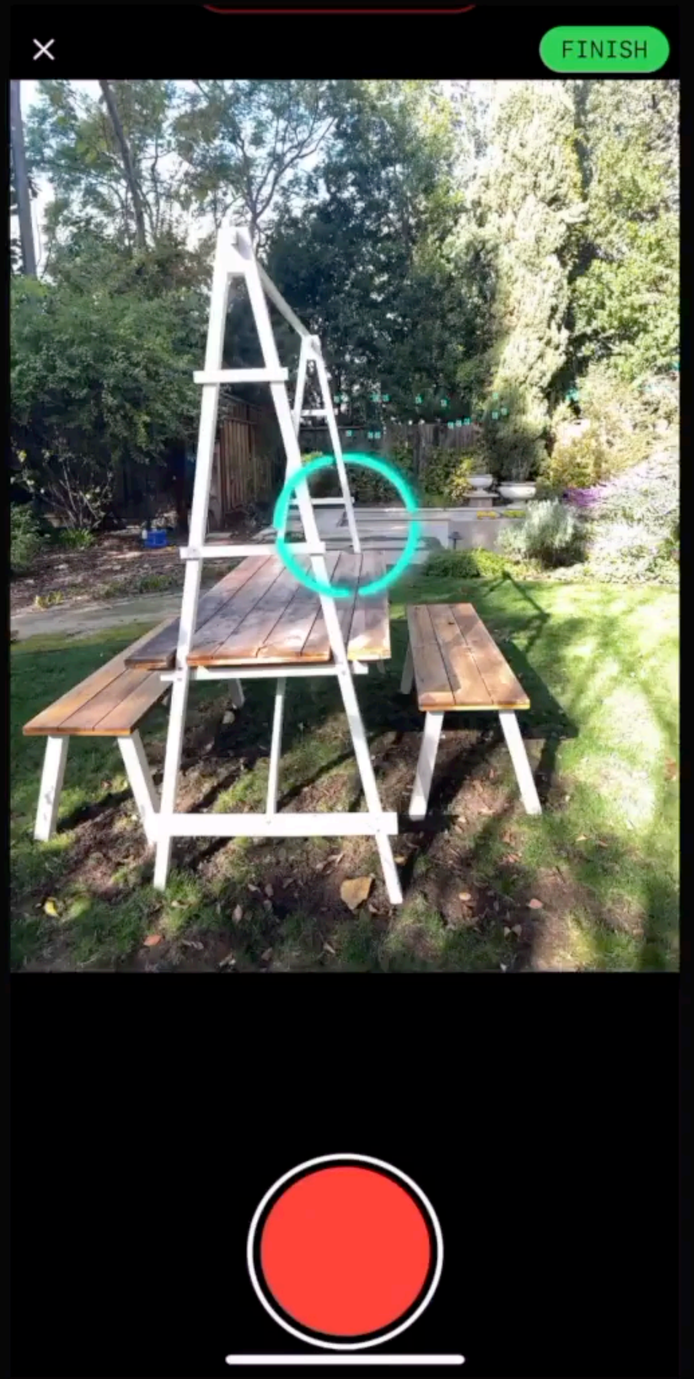
Google Maps

- *"New updates that make Maps look and feel more like the real world" —Sep 28, 2022*
  - Google Maps help you get the feel for a neighborhood before you go, explore *250+ landmarks in aerial view*, search for *nearby places with Live View* and more.
- *"Digital twin"*

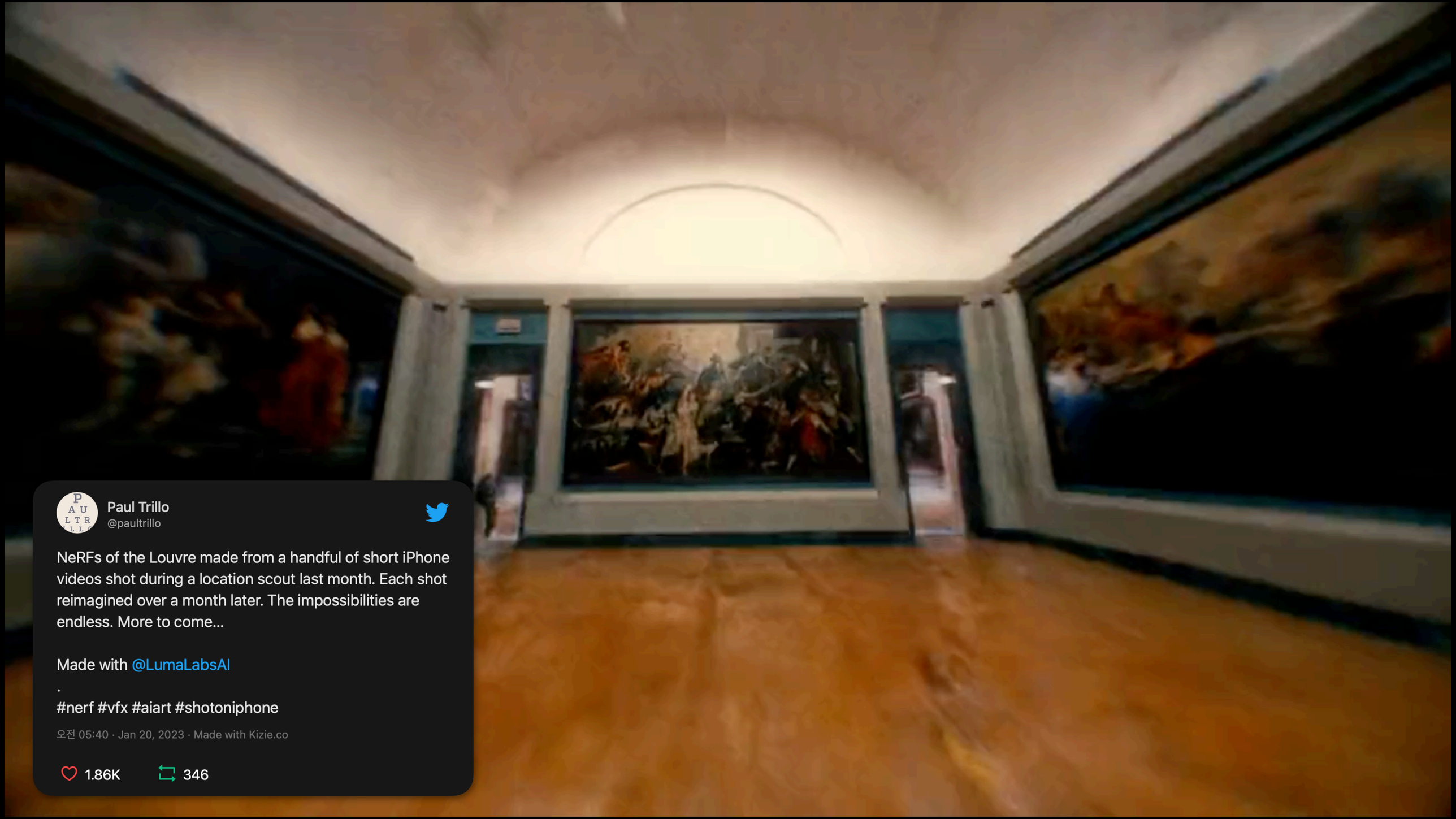


https://blog.google/products/maps/4-new-updates-maps-searchon-2022/


Brandenburg Gate, Germany



Movie credit: Luma AI at https://lumalabs.ai/



Paul Trillo
@paultrillo

NeRFs of the Louvre made from a handful of short iPhone videos shot during a location scout last month. Each shot reimagined over a month later. The impossibilities are endless. More to come...

Made with @LumaLabsAI
.
#nerf #vfx #aiart #shotoniphone

오전 05:40 · Jan 20, 2023 · Made with Kizie.co

1.86K 346

Balenciaga Track Sneaker

White Mesh and Nylon

https://www.balenciaga.com/en-kr/track-sneaker-white-542023W1GB19000.html

Reality Converter by Apple

Makes it easy to convert, view, and customize USDZ 3D objects on Mac. Customize material properties with your own textures, and edit file metadata. You can even preview your USDZ object under a variety of lighting and environmental conditions

https://developer.apple.com/news/?id=01132020a

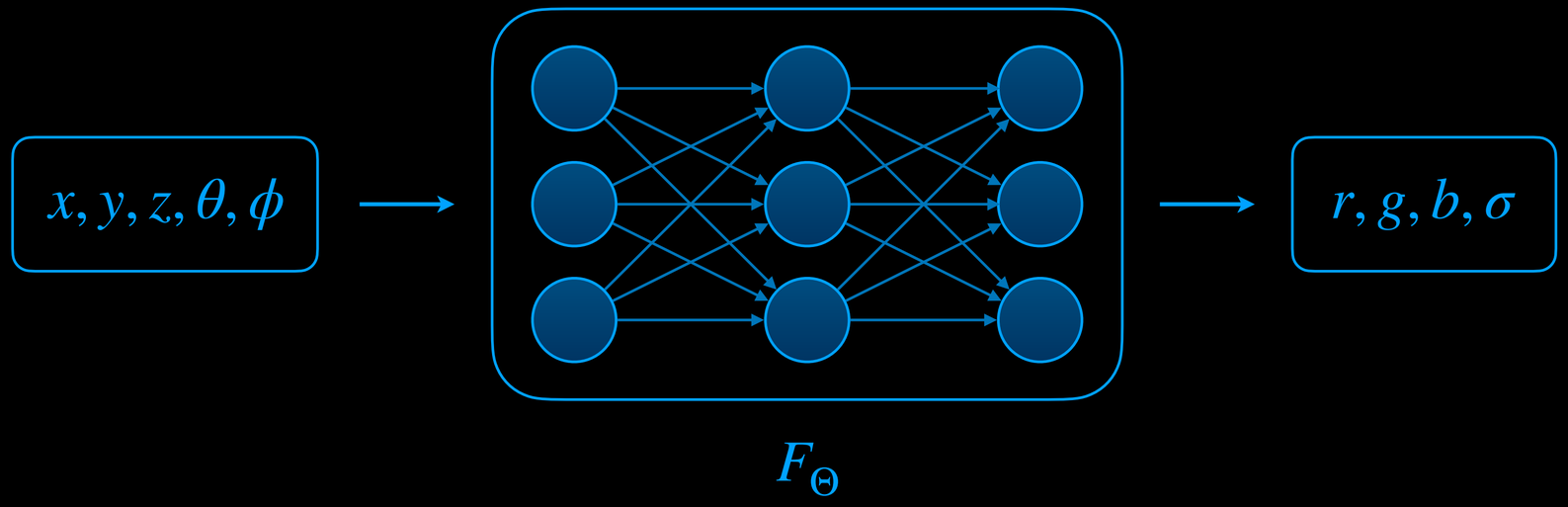
three.js webgl - animation - keyframes

Model: Littlest Tokyo by Glen Fox, CC Attribution.

Volume renderings

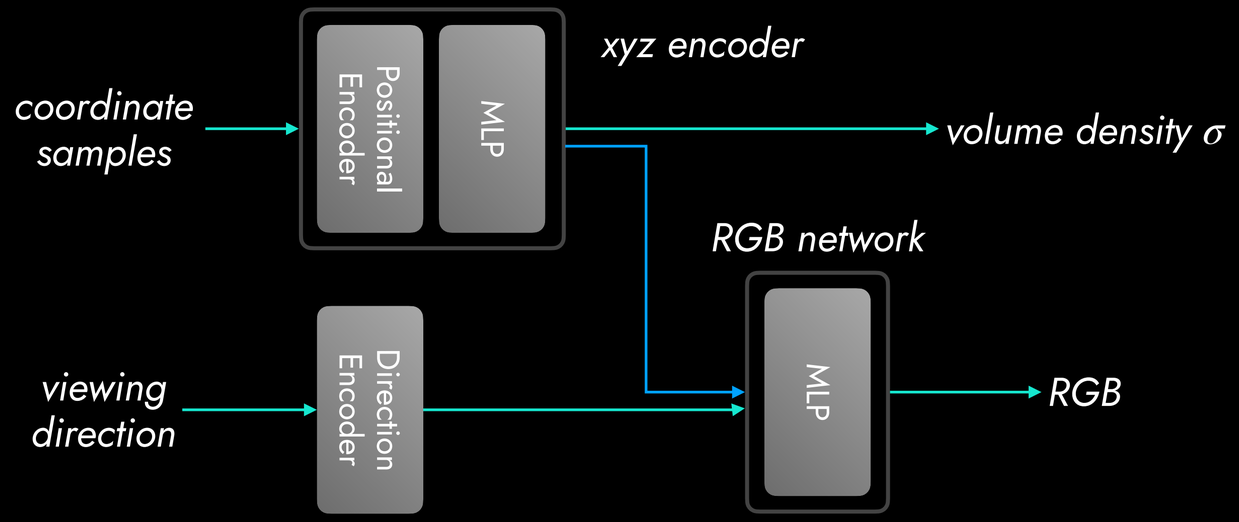
NeRF scene representation

- An MLP network $F_\Theta : (\mathrm{x}, \mathrm{d}) \rightarrow (\mathrm{c}, \sigma)$ takes 3D location $\mathrm{x} = (x, y, z)$ and 2D viewing direction $\mathrm{d} = (\theta, \phi)$, giving an emitted color $\mathrm{c} = (r, g, b)$ and volume density σ.
- A view-dependent modeling for physical world



Non-Lambertian NeRF

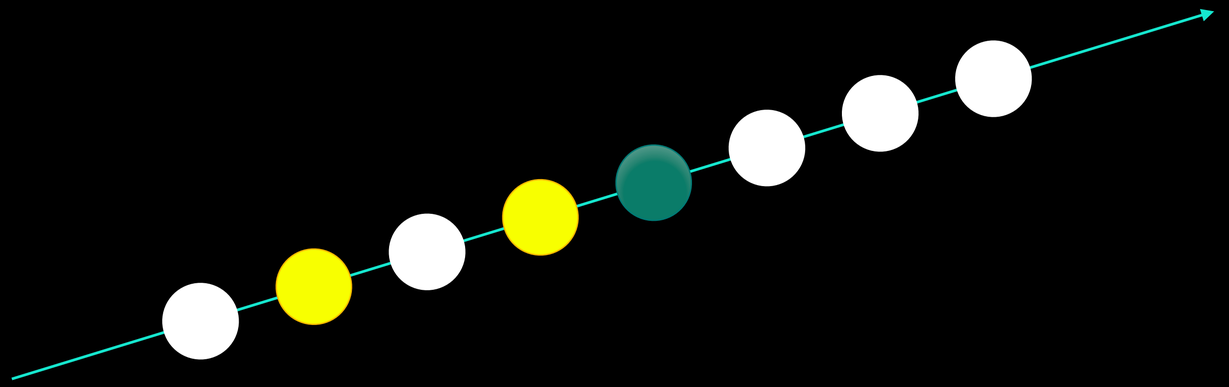
- The emitted color depends on *both* location and viewing direction.
- While, the brightness of a Lambertian surface to an observer is the same regardless of the viewing direction. The surface's luminance is isotropic, and the luminous intensity obeys Lambert's cosine law. *Yes, NeRF's non-Lambertian.*



Photometria (Lambert, 1760)

Volume rendering with radiance fields

- Classical volume rendering (Kajiya et al., 1984)
- Numerical estimation using quadrature
- Hierarchical volume sampling



Simplification

- The volume rendering used in NeRF does not consider *scattering* for computational efficiency, although it considers *absorption* and *emission*.

Absorption

Emission

Scattering

Credit: Novak et al., 2018, http://commons.wikimedia.org.

Simplification

- The volume rendering used in NeRF does not consider *scattering* for computational efficiency, although it considers *absorption* and *emission*.

Absorption

Emission

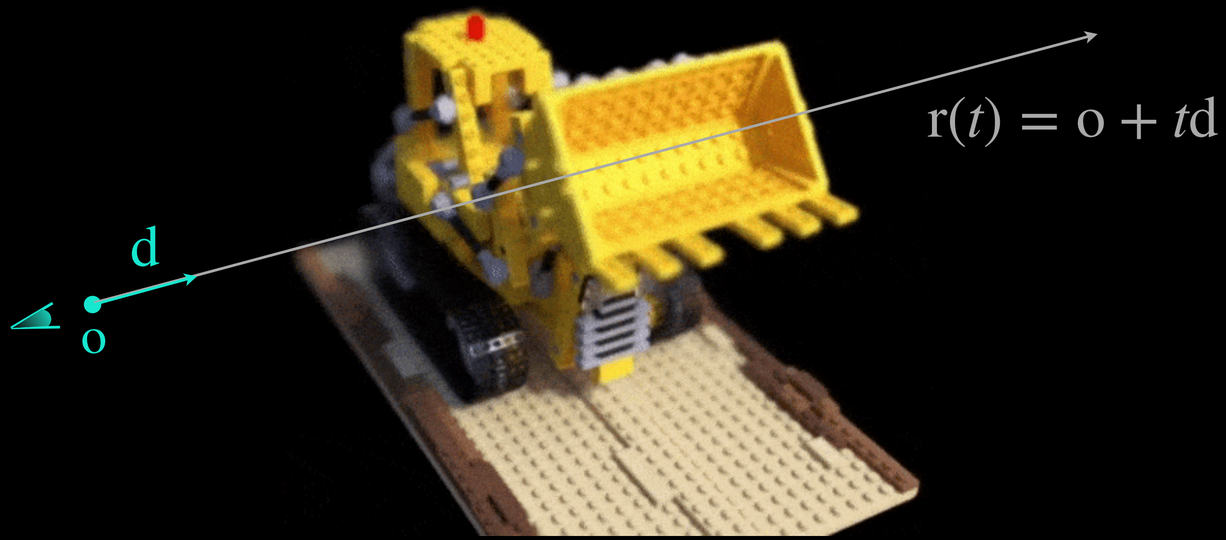
Scattering

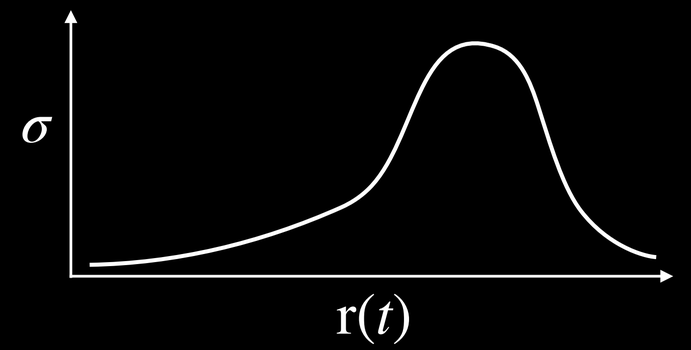
Credit: Novak et al., 2018, http://commons.wikimedia.org.

Classical volume rendering

- The volume density $\sigma(x)$ is the differential probability at an infinitesimal particle at x.
- The expected color $C(\mathrm{r})$ of camera ray $\mathrm{r}(t) = \mathrm{o} + t\mathrm{d}$ with near and far bounds t_n and t_f, respectively, is as follows:

$$C(\mathrm{r}) = \int_{t_n}^{t_f} T(t)\sigma\big(\mathrm{r}(t)\big)\mathrm{c}\big(\mathrm{r}(t), \mathrm{d}\big)dt, \quad \text{where} \quad T(t) = \exp\Big(- \int_{t_n}^{t} \sigma\big(\mathrm{r}(s)\big)ds \Big)$$





Classical volume rendering

- The volume density $\sigma(x)$ is the differential probability at an infinitesimal particle at x.
- The expected color $C(\mathrm{r})$ of camera ray $\mathrm{r}(t) = \mathrm{o} + t\mathrm{d}$ with near and far bounds t_n and t_f, respectively, is as follows:

$$C(\mathrm{r}) = \int_{t_n}^{t_f} T(t)\sigma\big(\mathrm{r}(t)\big)\mathrm{c}\big(\mathrm{r}(t), \mathrm{d}\big)dt, \quad \text{where} \quad T(t) = \exp\Big(- \int_{t_n}^{t} \sigma\big(\mathrm{r}(s)\big)ds \Big)$$

- $T(t)$ is the *accumulated transmittance* along the ray r, *i.e.*, the *probability* that the ray travels from t_n to t without hitting any other particle.

Transmittance

- The transmittance function $T(t)$ is the probability of a ray traveling over the interval $[0,t)$ without hitting any particles. $T(t+dt)$ is equal to $T(t)\big(1-dt\cdot\sigma(t)\big)$.

A hitting probability is the density.

$$T(t+dt) = T(t)\big(1-dt\cdot\sigma(t)\big)$$

$$\frac{T(t+dt)-T(t)}{dt} = T'(t) = -T(t)\sigma(t)$$

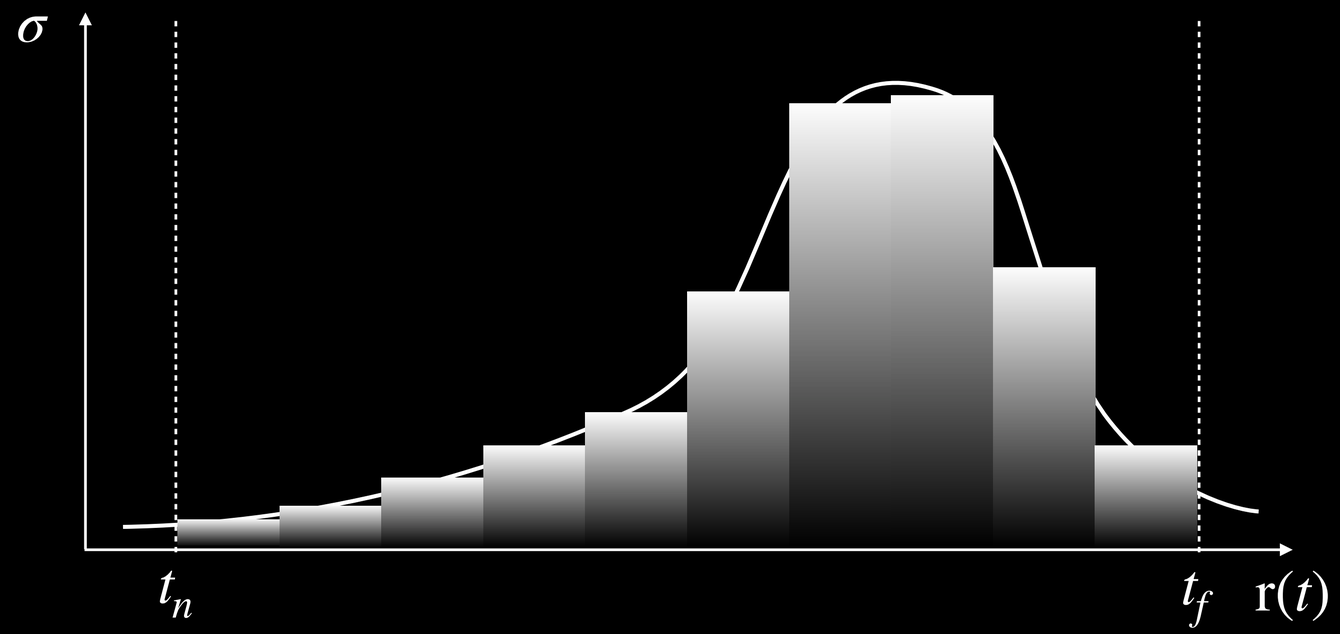
Ref. logarithmic differentiation

$$\frac{T(t)'}{T(t)} = -\sigma(t) \quad \Leftrightarrow \quad \int_a^b \frac{T(t)'}{T(t)}dt = \log T(t)\Big|_a^b = -\int_a^b \sigma(t)dt$$

$$\log T(b) - \log T(a) = -\int_a^b \sigma(t)dt \quad \Leftrightarrow \quad \frac{T(b)}{T(a)} = \exp\Big(-\int_a^b \sigma(t)dt\Big)$$

Numerical estimation using quadrature

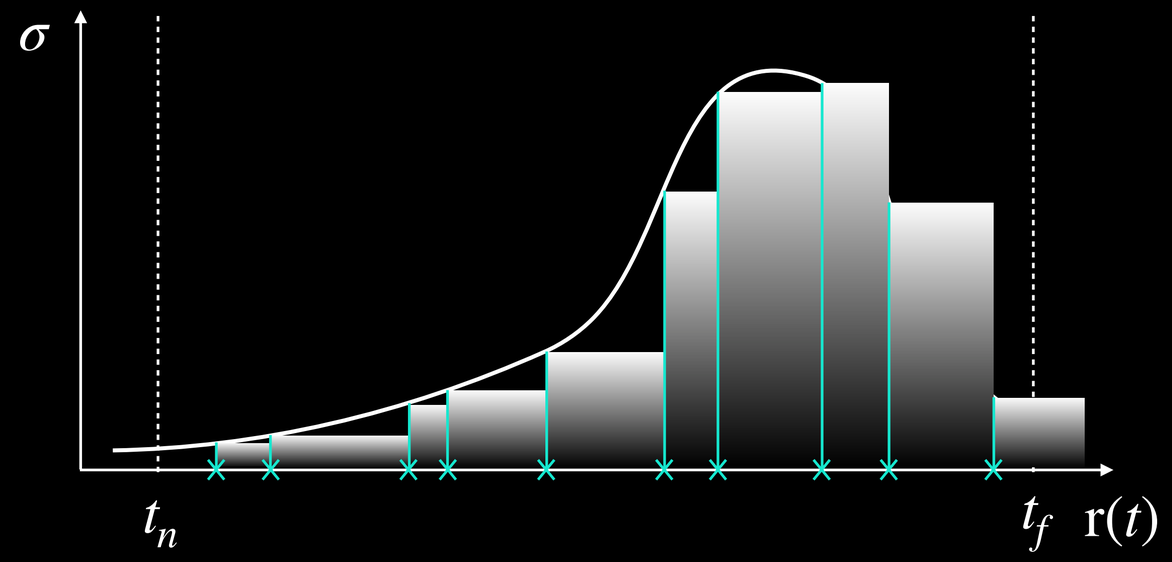
- The exact calculation of the integral is intractable and inefficient. One approximation is to estimate using quadrature, *compensating representation's resolution*.



Stratified sampling

- Stratified sampling partition $[t_n, t_f]$ into N evenly-spaced bins, and draw a sample from each bin, uniformly at random. Totally, N samples will be drawn.

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right] \quad \text{for } i \in [1,2,\ldots,N]$$



Stratified sampling

- Stratified sampling partition $[t_n, t_f]$ into N evenly-spaced bins, and draw a sample from each bin, uniformly at random. Totally, N samples will be drawn.

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right] \quad \text{for } i \in [1,2,\ldots,N]$$

- The volume rendering with the *stratified sampling* gives the estimate $C(\mathrm{r})$.

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

where $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples, and σ_i and c_i are the volume density and colors at t_i, which is cheap to evaluate using MLPs.

Hierarchical volume sampling

- An adaptive sampling strategy with two networks: the *coarse* and *fine*.
- The stratified sampling is for the coarse network, while a more informed sampling is for the fine network using the previous results.
- Then, the alpha-composited color and a piecewise constant PDF from this coarse network are the ground for the hierarchical volume sampling.

$$\hat{C}_c(\mathrm{r}) = \sum_{i=1}^{N_c} w_i c_i, \quad w_i = T_i\left(1 - \exp(-\sigma_i \delta_i)\right)$$

- $\hat{C}_c(\mathrm{r})$ enables a faster evaluation of a rendered color, while the normalized w_i, $\hat{w}_i = w_i / \Sigma_{j=1}^{N_c} w_j$, defines the PDF for the sampling for fine networks.

Hierarchical volume sampling (Cont'd)

- Then, compute the final rendered color of the ray $\hat{C}_f(\mathrm{r})$ using $N_c + N_f$ samples.
- Similarly to importance sampling, but it is a non-uniform discretization rather than an independent probabilistic estimation of the entire integral.

Positional encoding

- Spectral bias (Tancik et al., 2020) may interfere the learning of NeRF performing poorly at high-frequency variation in color and geometry (Rahaman et al., 2018).
- Formulating with a composition of two functions $F_\Theta = F'_\Theta \circ \gamma$ where $\gamma : \mathbb{R} \rightarrow \mathbb{R}^{2L}$

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

- Applied to each of the three coordinate values in $\mathrm{x} \in [-1,1]^3$.
- It is known that it makes MLP easily approximate a higher frequency function.

Optimization

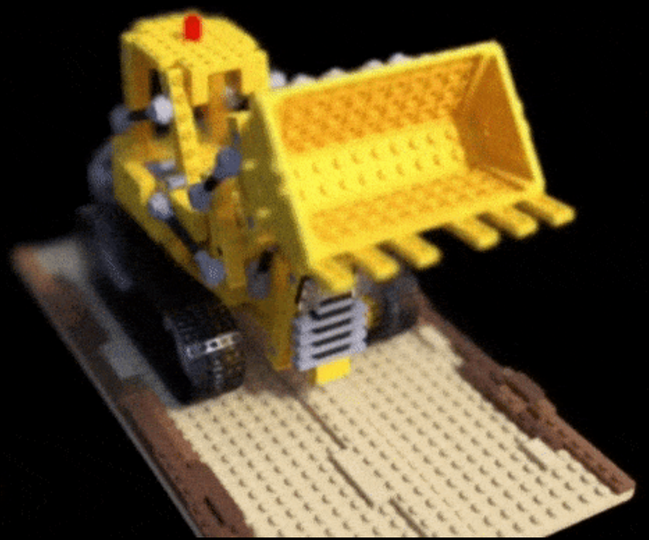
- Assume that we know ground-truth camera poses and intrinsics, and scene bounds. The COLMAP structure-from-motion (SfM) package provides a good initialization for them (Schönberger et al., 2016).
- The objective function is the total squared error between the rendered and true pixel colors for both the coarse and fine renderings:

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

- where $\mathscr{R}$ is the randomly sampled batch of camera rays from the set of all pixels in the dataset. The first term ensures that the estimation of coarse network is reliable during training.

Datasets

- NeRF-Synthesis (Mildenhall et al., 2020)
  - Originally, this is the third set of Realistic Synthetic 360° consists of *eight* objects of complicated geometry and realistic non-Labertian materials.
  - *Six* of them are rendered from the viewpoints on the *upper hemisphere*, while *the others* are from the viewpoints on a *full sphere* at 800x800 resolution.
  - 100 training views and 200 for testing
  - Drum, ficus, hotdog, lego, mic, etc.



Datasets (Cont'd)

- LLFF (Mildenhall et al., 2019, 2020)
  - Real images of complex scenes
  - Roughly *forward-facing* images of 8 scenes captured by a handheld cellphone
  - 20 to 62 images, 1/8 views among them for testing

https://github.com/Fyusion/LLFF

Datasets (Cont'd)

- IMC Phototourism
  - Collection of photos by tourists in a real-world scenario
  - The dataset is provided by the Image Matching Workshop (and Challenge)
  - COLMAP results are available.

https://www.cs.ubc.ca/research/image-matching-challenge/current/

Old Town Square
Prague, Czech Republic

Credit: Martin-Brualla et al. (2021); https://nerf-w.github.io/

Hyperparameters

- In the early work, they used a batch size of 4K rays, N_c=64, N_f=128.
- Adam optimizer with the learning rate of 5×10^{-4}, exponentially decaying to 5×10^{-5}, while the others are set to default.
- Typically, 100-300k iterations to converge on a single NVIDIA V100 GPU taking *1-2 days* (which only takes few minutes nowadays).

Preprocessing

- So, the problem is how to pre-process the dataset?
  - For each image, the camera position o.
  - For each pixel, the viewing direction d and the corresponding *colors*.
- Terminology & concepts:
  - World, camera, and image coordinate systems
  - Normalized device coordinate (NDC)

Coordinate Transforms

- Extrinsics and intrinsics parameters transform among world, camera, and image coordinates.

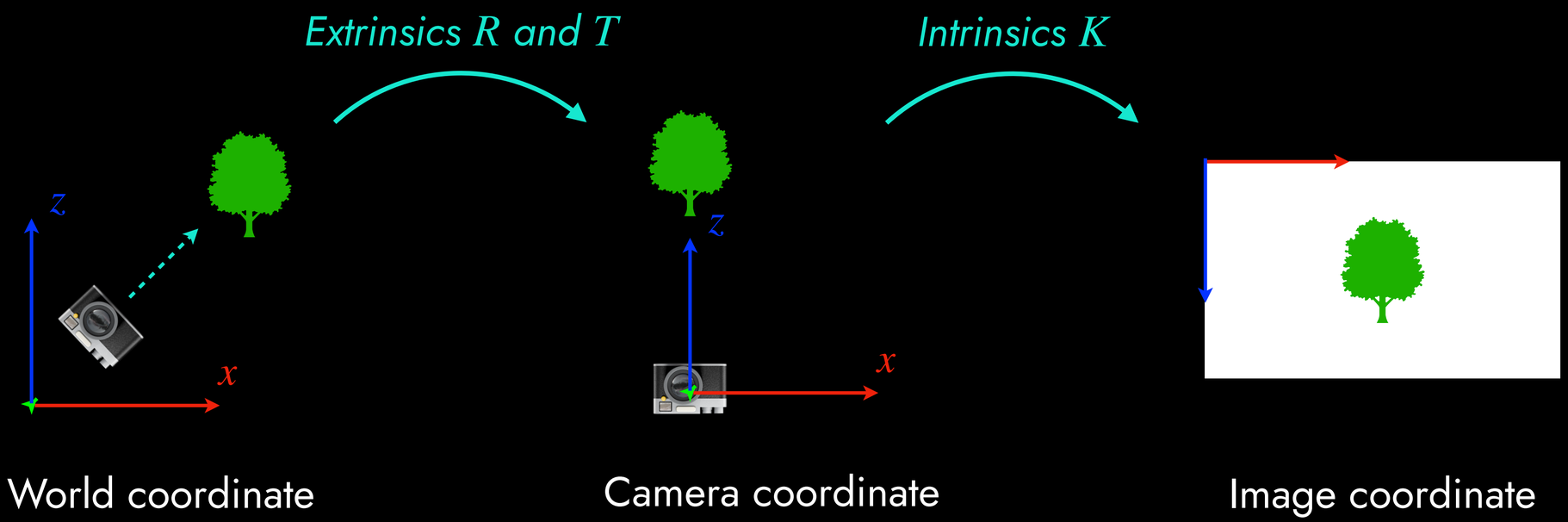


Figure modified from Peter Hedman's

Camera projection

$$z_c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

2D image coordinates — *Intrinsic properties (scaling, optic center)* — *Extrinsic properties ("inverse" of the camera pose matrix)* — *3D world coordinates*

where z_c is the z-coordinate of the camera relative to the world origin. c_x and c_y are the principal point, ideally the center of the image.

- We assume that we have the access to camera pose and intrinsics by COLMAP (Schönberger et al., 2016).
- Usually, we assume the same focal length for x and y, $f = f_x = f_y$.

Extrinsic vs camera pose

- The extrinsic matrix is the inverse of the camera pose matrix.

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathrm{R} & \mathrm{t} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathrm{R_c} & \mathrm{C} \\ 0 & 1 \end{pmatrix}^{-1}$$

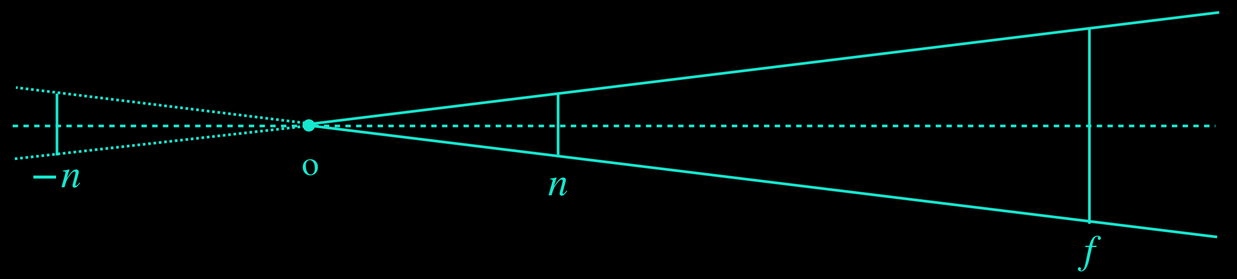
where R_c is the *camera rotation matrix* and C is the *camera position* in the world coordinate system. Note that one may show that:

$$\begin{pmatrix} \mathrm{R_c} & \mathrm{C} \\ 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \mathrm{R_c}^\intercal & -\mathrm{R_c}^\intercal \mathrm{C} \\ 0 & 1 \end{pmatrix}$$

Ref. $R_c^\intercal = R_c^{-1}$ for a valid rotation matrix.

Focal length

- Focal length $\mathbf{f} := (f_x, f_y)$ measures how strongly the system converges light, which is the inverse of the system's *optical power*.
- In our notation, it satisfies $1/f_x = 1/n + 1/f$. But, $1/f$ can be ignored due to $f \gg n$, leading to $f_x = f_y \approx n$, which is the distance to the near plane.
- In our context, the unit of focal length is *pixel*.



Ref. iPhone 16's main camera has the (effective) focal length of 26mm.

Get ray directions

- The ray $\mathrm{r} = \mathrm{o} + t\mathrm{d}$ for a pixel in the position of (u, v) of an image can be defined as:

$$\mathrm{d} = \left(\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \right)^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}, \quad \mathrm{o} = - \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

Inverse of intrinsic x extrinsic camera matrix image space to world space

coordinate in image space

camera position

- *Now, we have preprocessed the dataset!* We have d and o, and $C(\mathrm{r})$ is the colors of the pixel in the position of (u, v) in the image

Get ray directions

- The ray $\mathrm{r} = \mathrm{o} + t\mathrm{d}$ for a pixel in the position of (u, v) of an image can be defined as:

$$\mathrm{d} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}^{-1} \begin{pmatrix} \frac{u - c_x}{f_x} \\ \frac{v - c_y}{f_y} \\ 1 \end{pmatrix}, \quad \mathrm{o} = - \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

Inverse of the extrinsic rotation matrix camera space to world space (c2w)

direction in camera space

camera position

$$\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1/f_x & 0 & -c_x/f_x \\ 0 & 1/f_y & -c_y/f_y \\ 0 & 0 & 1 \end{pmatrix}$$

See the block matrix inversion.

Evaluation metrics

- Peak Signal-to-Noise Ratio (PSNR)
- Structural Similarity Index Map (SSIM)
- Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018)

Peak Signal-to-Noise Ratio (PSNR)

- The ratio between the maximum power of a signal and the power of corrupting noise that affects the fidelity of a generated image.
- For the pixels $\in [0,1]$, PNSR is a logarithm of MSE defined as:

$$PSNR = 10 \log_{10}\left(\frac{MAX_I}{MSE}\right)$$

$$MSE = \frac{\sum_{i=1}^{H} \sum_{j=1}^{W} \left(I_0(i,j) - I_1(i,j)\right)^2}{HW}$$

where MAX_I denotes the maximum value of pixel color. For the RGB color space, we take the average over color channel.

Structural Similarity Index Map (SSIM)

- SSIM assesses perceptual differences, luminance, contrast, and structural.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

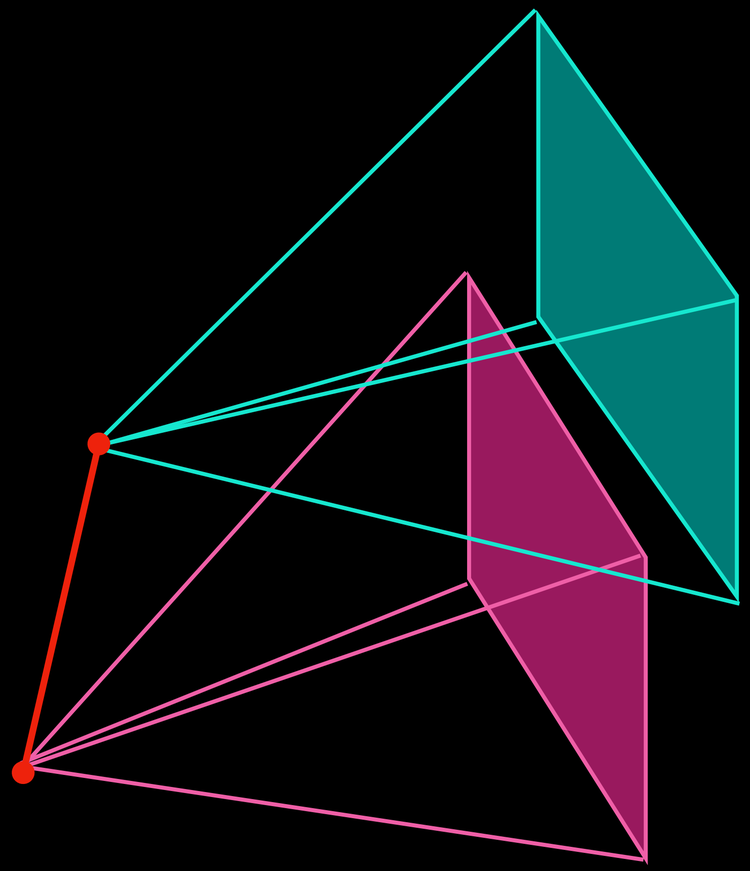
- where $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ to prevent division by near-zero. L is the max pixel-value. Some uses $k_1 = 0.01$ and $k_2 = 0.03$ or $c_1 = c_2 = \epsilon$, a very small value.

Learned Perceptual Image Patch Similarity (LPIPS)

- A distance d between two patches x and x_0, given a network $\mathscr{F}$. The d is calculated using $\hat{y}^l$ and $\hat{y}_0{}^l$ in $\mathbb{R}^{H_l \times W_l \times C_l}$ for the layer l of $\mathscr{F}$.
- Before that it channel-wisely scales the activations by a vector of $w^l \in \mathbb{R}^{C_l}$, we l2-normalize for the channel dimension. The d is defined as follows:

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \| w_l \circ (\hat{y}^l_{hw} - \hat{y}_0{}^l_{hw}) \|_2^2$$

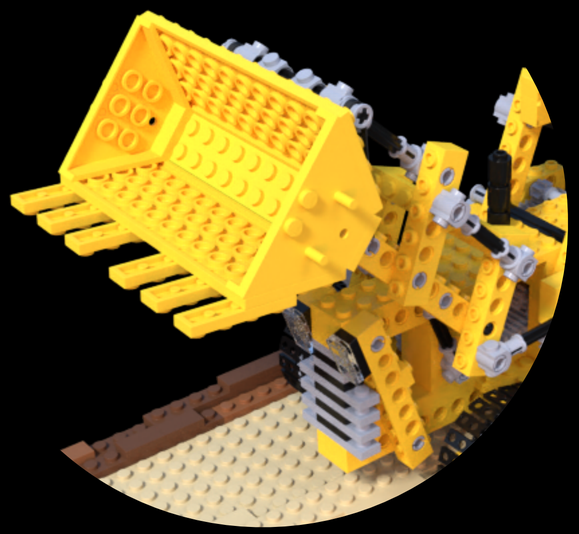
- where the linear weights w_l is fine-tuned ($\mathscr{F}$ is fixed), all fine-tuned, or learning from scratch.



Camera Pose Refinement

Assumption of perfect camera poses

- NeRFs assume that it knows the *perfect* camera poses for training images to get accurate training rays in the world coordinate system for photometric supervision.
- Imperfect camera poses are the source of blurred images or artifactual floatings from inaccurate depth estimation.



Camera pose errors

- Most benchmarks utilizes the COLMAP (SfM) library to estimate camera poses, which used to have some degree of errors.
- COLMAP estimates the camera poses from a set of (un)ordered images. It calculates camera intrinsics and extrinsics (position and orientation).
- With studio capturing, camera calibration is crucial to determine camera intrinsic parameters such as focal length, principal point, and lens distortion coefficients.

Ray directions with pose errors

- The noised ray $\tilde{\mathrm{r}}(t) = \tilde{\mathrm{o}} + t\tilde{\mathrm{d}}$ for a pixel in the position of (u, v) of an image:

$$\tilde{\mathrm{d}} = \begin{pmatrix} \tilde{r}_{11} & \tilde{r}_{12} & \tilde{r}_{13} \\ \tilde{r}_{21} & \tilde{r}_{22} & \tilde{r}_{23} \\ \tilde{r}_{31} & \tilde{r}_{32} & \tilde{r}_{33} \end{pmatrix}^{-1} \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}, \quad \tilde{\mathrm{o}} = -\begin{pmatrix} \tilde{r}_{11} & \tilde{r}_{12} & \tilde{r}_{13} \\ \tilde{r}_{21} & \tilde{r}_{22} & \tilde{r}_{23} \\ \tilde{r}_{31} & \tilde{r}_{32} & \tilde{r}_{33} \end{pmatrix}^{-1} \begin{pmatrix} \tilde{t}_1 \\ \tilde{t}_2 \\ \tilde{t}_3 \end{pmatrix}$$

Inverse of extrinsic rotation matrix

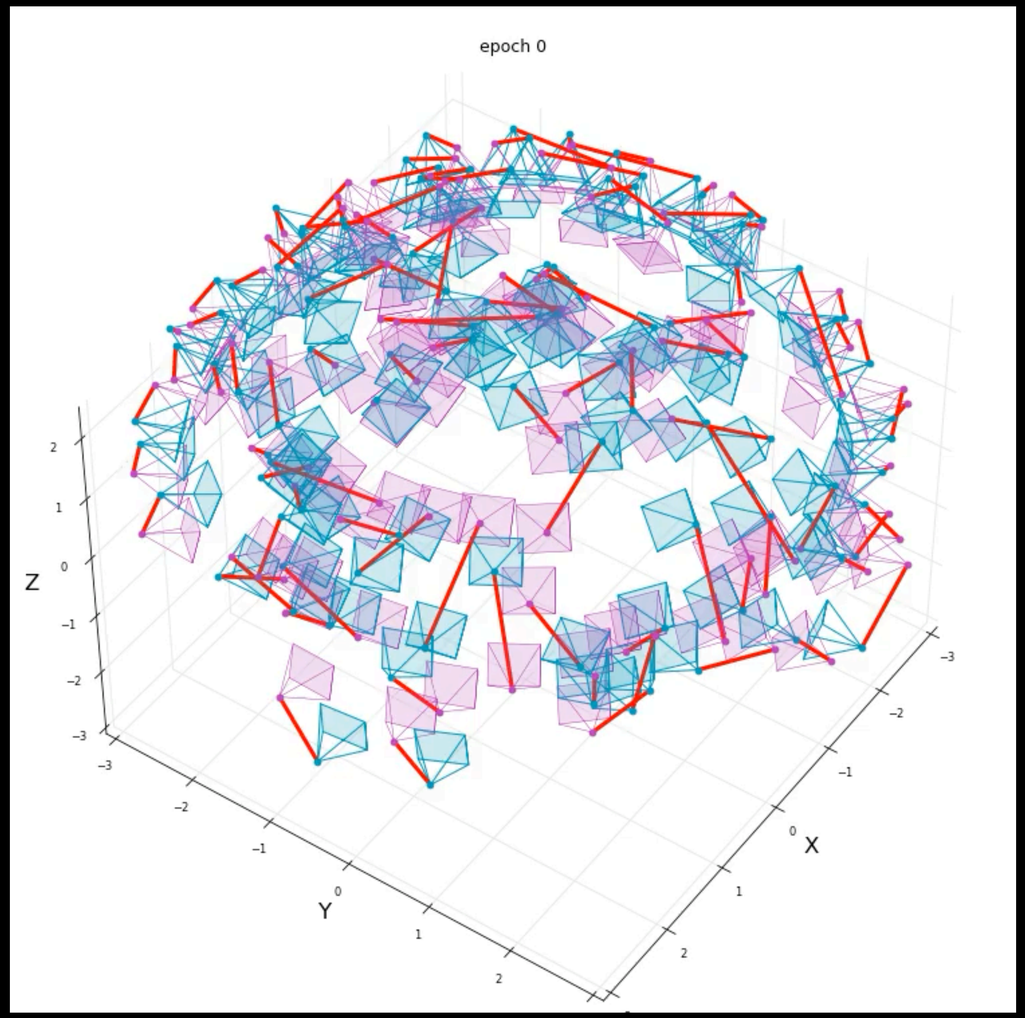
Inverse of intrinsic camera matrix

coordinate in image space

image space to world space

camera position

- Here, we assume that the pose errors are tolerable for local refinement methods using error back-propagation. (*e.g.*, < 60°)



A major pose error is refined within 5k iterations (not epoch, sorry for that).

Gradient analysis

- The volume rendering with the *stratified sampling* gives the estimated $C(\mathrm{r})$.

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

- Let's focus on a unit of photometric supervision, the pair of a ray and its corresponding pixel colors. The gradient with respect to a sample $\mathrm{r}(t_i)$ is:

$$\frac{\partial \hat{C}(\mathrm{r})}{\partial \mathrm{r}(t_i)} = \frac{\partial \hat{C}(\mathrm{r})}{\partial \sigma_\mathrm{i}} \frac{\partial \sigma_i}{\partial \mathrm{r}(t_i)} + \frac{\partial \hat{C}(\mathrm{r})}{\partial \mathrm{c}_\mathrm{i}} \frac{\partial c_i}{\partial \mathrm{r}(t_i)}$$

- The network outputs σ_i and c_i are subjects to calculate the input gradient.

Gradient analysis (cont'd)

- In turn, the gradient of a sample $\mathrm{r}(t_i)$ on the ray r with respect to $\tilde{\mathrm{o}}$ and $\tilde{\mathrm{d}}$ is:

$$\frac{\partial \mathrm{r}(t_i)}{\partial \tilde{\mathrm{o}}} = \frac{\partial(\tilde{\mathrm{o}} + t_i \tilde{\mathrm{d}})}{\partial \tilde{\mathrm{o}}} = 1$$

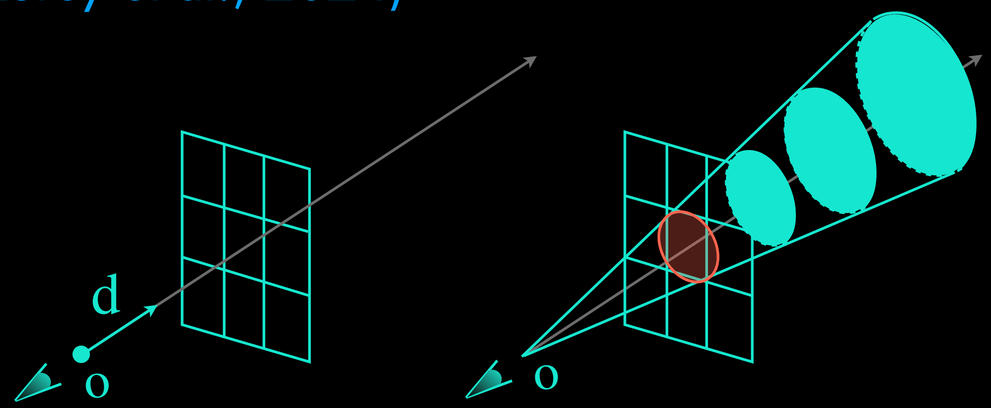
$$\frac{\partial \mathrm{r}(t_i)}{\partial \tilde{\mathrm{d}}} = \frac{\partial(\tilde{\mathrm{o}} + t_i \tilde{\mathrm{d}})}{\partial \tilde{\mathrm{d}}} = t_i$$

- The farther from the ray origin, the more the camera orientation changes.
- Note that the *scale problem* arises between the relative importance of camera location and camera orientation since $E[t_i]$ can be less than 1 or greater than 1.

The intrinsic parameters can also be optimized similarly.

Recent works may include...

- Voxel-based method
  - Plenoxels (Yu et al., 2021), TensoRF (Chen et la., 2022), Instant-NGP (Müller et al., 2022)
- Camera pose refinement
  - BARF (Lin et al., 2021), GARF (Chng et al., 2022), ✨Instant-Pose (Heo et al., 2023)
  - DUSt3R (Wang et al., 2023), MASt3R (Leroy et al., 2024)
- Mip-NeRF (Barron et al., 2021)
- DreamField (Jain et al., 2022)
- DreamFusion (Poole et al., 2022)



Conclusion

- Novel views are synthesized by optimizing the volume rendering function through neural networks, using a sparse set of input images with known poses.
- The whole procedure is differentiable; however, it requires known camera poses to convert image coordinates to world coordinates for training rays.
- We will explore how to learn camera poses through end-to-end optimization, especially for *fast* voxel-based methods, which are known to be challenging.