3D Volumetric Rendering

Lalit Manam

COMPUTER VISION LABORATORY
DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF SCIENCE
BENGALURU
INDIA

April 3, 2024

3D Reconstruction

- Recovering the 3D structure of the scene given images
- Classified based on the nature of input and application
  - Structure-from-Motion (SfM)
  - Simultaneous Localization and Mapping (SLAM)
  - Visual Odometry
  - Volumetric Rendering

3D Reconstruction

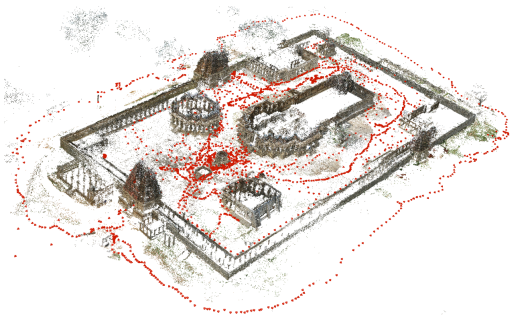


Figure: Reconstruction of Vitthala Temple, Hampi

- Recovering the 3D structure of the scene given images
- Classified based on the nature of input and application
  - Structure-from-Motion (SfM)
  - Simultaneous Localization and Mapping (SLAM)
  - Visual Odometry
  - Volumetric Rendering

3D Reconstruction

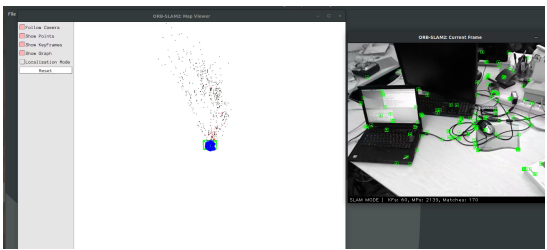
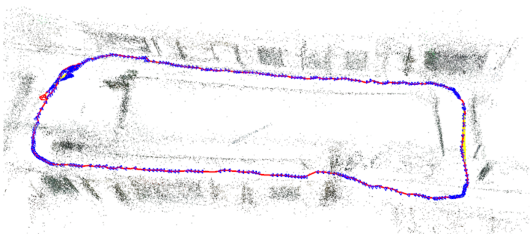


Figure: Animation showing Visual SLAM by ORB-SLAM2

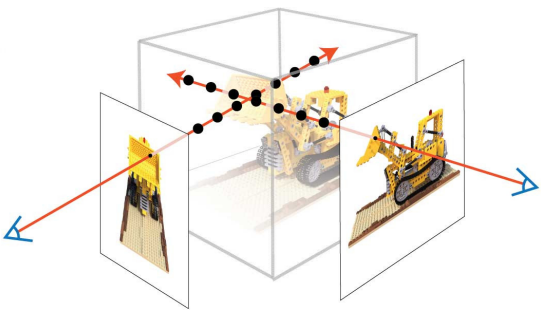
- Recovering the 3D structure of the scene given images
- Classified based on the nature of input and application
  - Structure-from-Motion (SfM)
  - Simultaneous Localization and Mapping (SLAM)
  - Visual Odometry
  - Volumetric Rendering

3D Reconstruction



- Recovering the 3D structure of the scene given images
- Classified based on the nature of input and application
  - Structure-from-Motion (SfM)
  - Simultaneous Localization and Mapping (SLAM)
  - Visual Odometry
  - Volumetric Rendering

3D Reconstruction



- Recovering the 3D structure of the scene given images
- Classified based on the nature of input and application
  - Structure-from-Motion (SfM)
  - Simultaneous Localization and Mapping (SLAM)
  - Visual Odometry
  - Volumetric Rendering

Structure-from-Motion

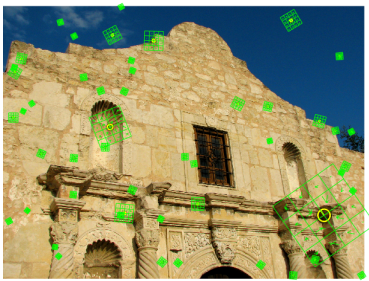
Problem Setup

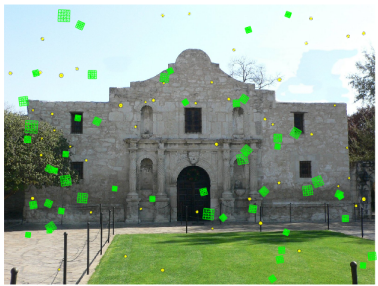
Image Collection

Problem Setup

Image Matching

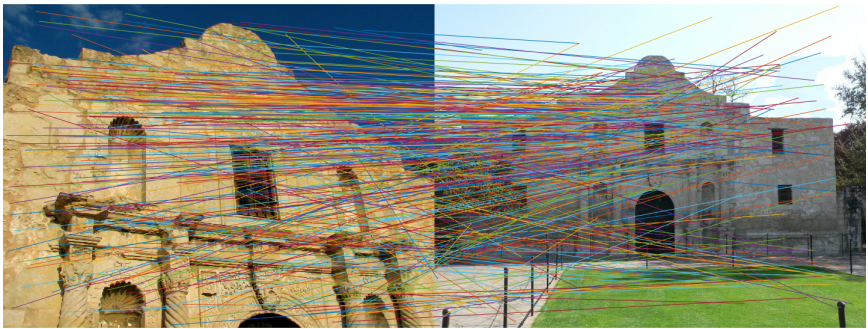
Problem Setup





Keypoint/Feature Extraction

Problem Setup



Keypoint/Feature Correspondence Search

Problem Setup

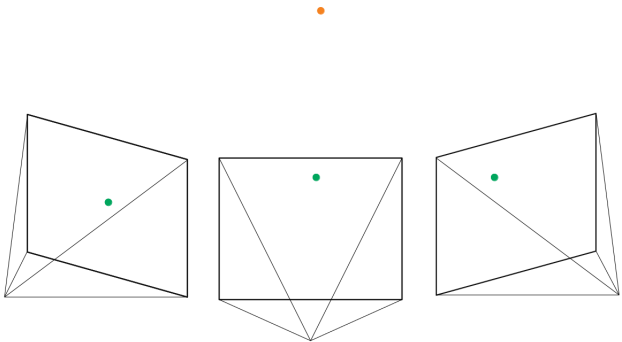
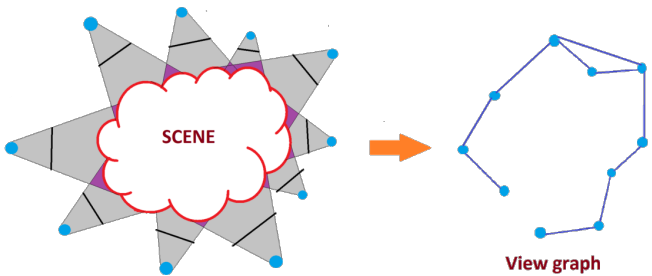
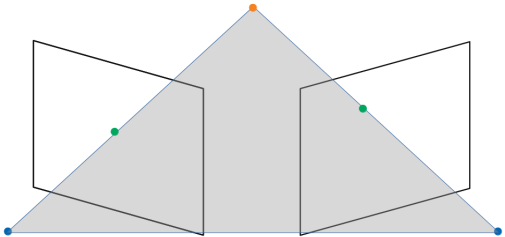


Image point correspondences are known

Problem Setup

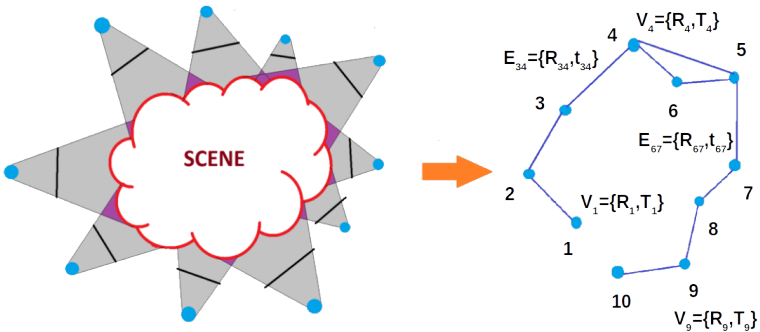


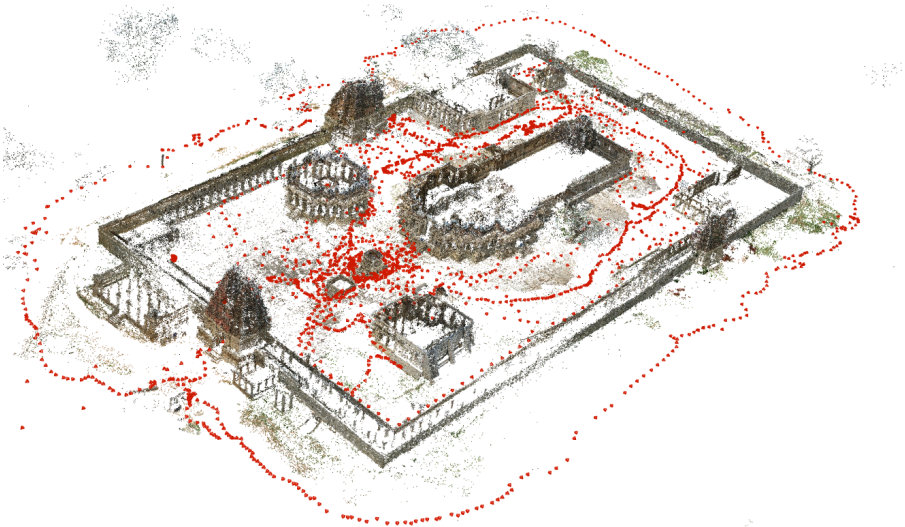
Epipolar Geometry



- Coplanarity of the camera centers and 3D point
- Constraint: $\mathbf{p}_2^T \mathbf{E} \mathbf{p}_1 = 0$
- $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$: relative rotation ($\mathbf{R}$) and translation direction ($\mathbf{t}$)

View Graph Abstraction



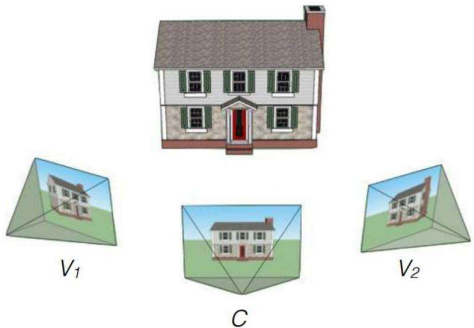


Reconstruction of Vitthala Temple, Hampi

Volumetric Rendering Methods

What is Rendering?

Rendering/View Synthesis



- Given camera parameters, estimate the image captured by the camera
- Some form of scene information should be known apriori

Flynn et al. DeepStereo: Learning to Predict New Views from the World's Imagery

Class of Methods

Methods Classified based on Storage of Scene Information

- Image based rendering
- Depth image based rendering
- Volumetric rendering

Image based Rendering

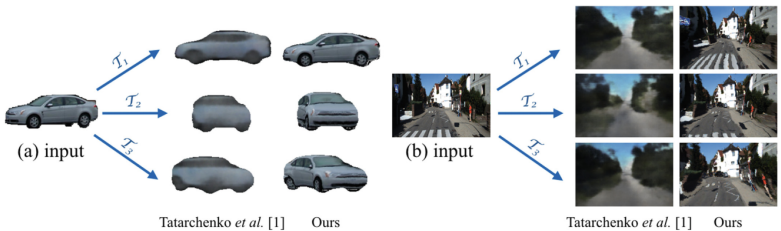


Fig. 1. Given an input image, our goal is to synthesize novel views of the same object (left) or scene (right) corresponding to various camera transformations (T_i). Our approach, based on learning appearance flows, is able to generate higher-quality results than the previous method that directly outputs pixels in the target view [1].

Zhou et al. View Synthesis by Appearance Flow

Depth Image based Rendering

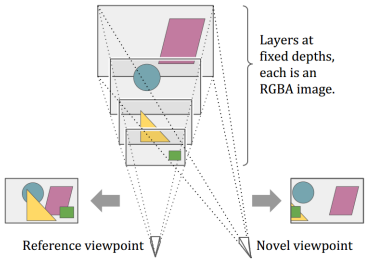
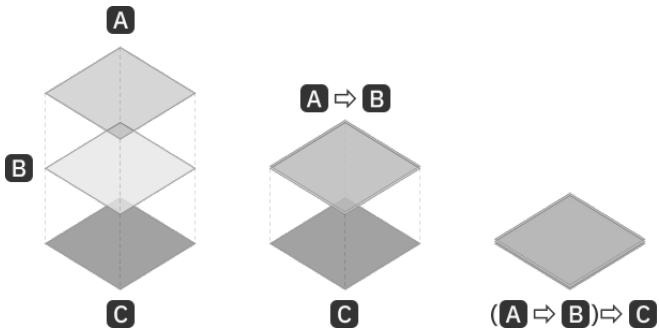


Fig. 2. An illustration of the multiplane image (MPI) representation. An MPI consists of a set of fronto-parallel planes at fixed depths from a reference camera coordinate frame, where each plane encodes an RGB image and an alpha map that capture the scene appearance at the corresponding depth. The MPI representation can be used for efficient and realistic rendering of novel views of the scene.

Zhou et al. Stereo Magnification: Learning view synthesis using multiplane images

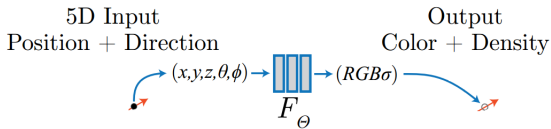
Depth Image based Rendering

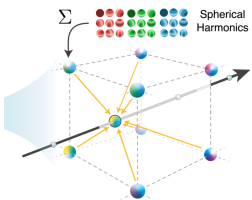
Alpha Composition

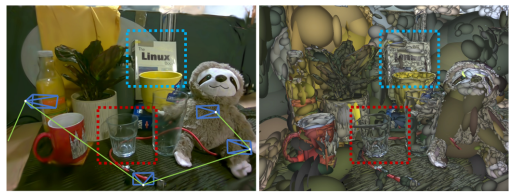


https://ciechanow.ski/alpha-compositing/

3D Volumetric Rendering







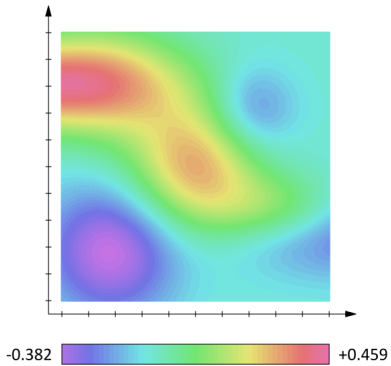
Mildenhall *et al.* NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Yu *et al.* Plenoxels: Radiance Fields without Neural Networks

Matsuki *et al.* Gaussian Splatting SLAM

What is a Field?

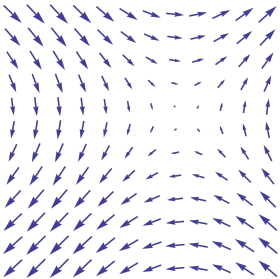
Scalar Field



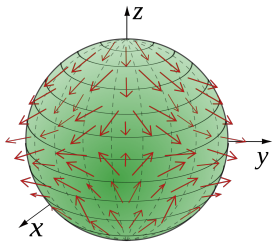
Scalar field on a plane

Credit: wikipedia.org

Vector Field



Plane



Sphere

Vector field on different surfaces

Credit: wikipedia.org

Vector Field

Optic flow field

Credit: fzheng.me/2015/03/25/optical-flow/

Is RENDERING same as RECONSTRUCTION?

Aim of the Problem

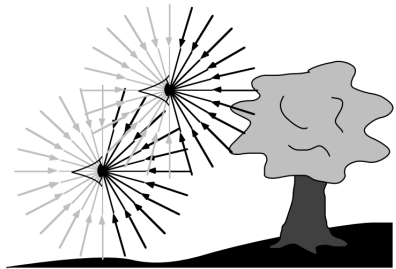


Fig. 1.3
The plenoptic function describes the information available to an observer at any point in space and time. Shown here are two schematic eyes-which one should consider to have punctate pupils-gathering pencils of light rays. A real observer cannot see the light rays coming from behind, but the plenoptic function does include these rays.

Adleson *et al.* The Plenoptic Function and the Elements of Early Vision

Courtesy for the next few slides: NeRF ECCV 2022 Tutorial
sites.google.com/berkeley.edu/nerf-tutorial/home

Plenoptic Function

Figure by Leonard McMillan

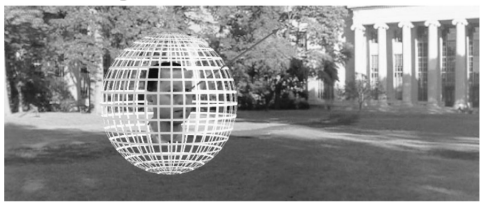
Q: What is the set of all things that we can ever see?

A: The Plenoptic Function (Adelson & Bergen '91)

Let's start with a stationary person and try to parameterize everything that they can see…

Slide credit: Alyosha Efros

Grayscale Snapshot



$P(\theta,\phi)$

- is intensity of light
  - Seen from a single position (viewpoint)
  - At a single time
  - Averaged over the wavelengths of the visible spectrum

Slides from Alyosha Efros

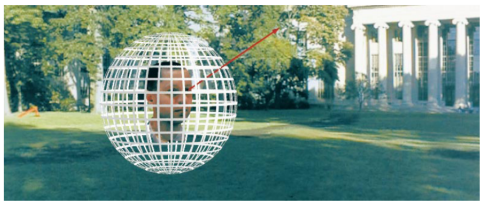
Color snapshot

$P(\theta,\phi,\lambda)$

- is intensity of light
  - Seen from a single position (viewpoint)
  - At a single time
  - As a function of wavelength

Slides from Alyosha Efros

A movie

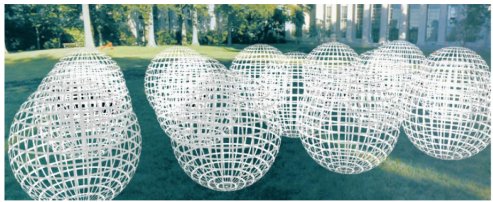


$P(\theta,\phi,\lambda,t)$

- is intensity of light
  - Seen from a single position (viewpoint)
  - Over time
  - As a function of wavelength

Slides from Alyosha Efros

A holographic movie

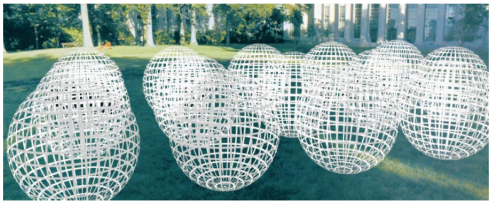


$$P(\theta,\phi,\lambda,t,V_X,V_Y,V_Z)$$

- is intensity of light
  - Seen from ANY position and direction
  - Over time
  - As a function of wavelength

Slides from Alyosha Efros

The plenoptic function

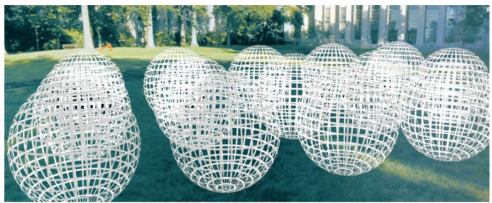


$$P(\theta,\phi,\lambda,t,V_X,V_Y,V_Z)$$

7D function, that can reconstruct every position & direction,
at every moment, at every wavelength
= it recreates the entirety of our visual reality!

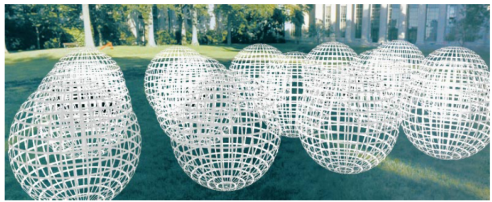
Slides from Alyosha Efros

Goal: Plenoptic Function from a set of images



- Objective: Recreate the visual reality
- All about recovering photorealistic pixels, not about recording 3D point or surfaces
  —Image Based Rendering aka **Novel View Synthesis**

Plenoptic Function



Look familiar 🙂?

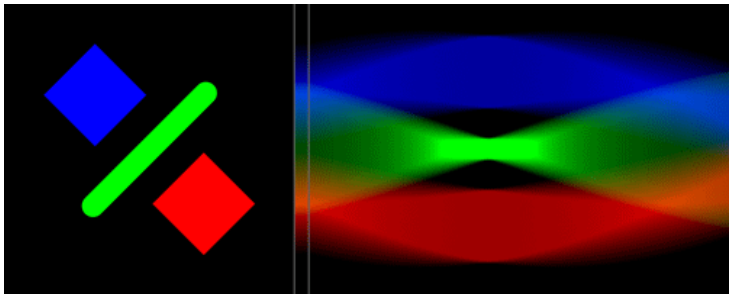
7D function:
2 – direction
1 – wavelength
1 – time
3 – location

$$P(\theta,\phi,\lambda,t,V_X,V_Y,V_Z) \longrightarrow P(\theta,\phi,V_X,V_Y,V_Z)$$

Let's simplify:

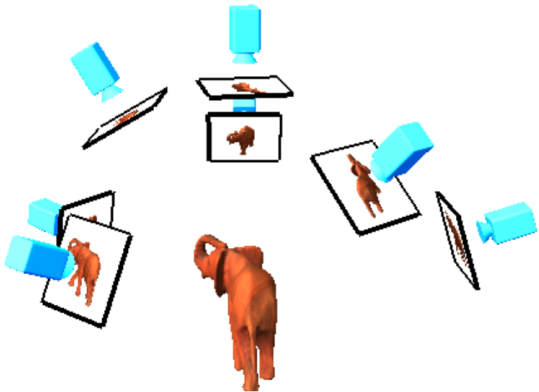
1. Remove the time
2. Remove the wavelength & let the function output RGB colors

Physics of the Problem



Credit: en.wikipedia.org/wiki/Radon_transform

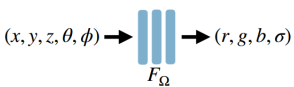
Physics of the Problem



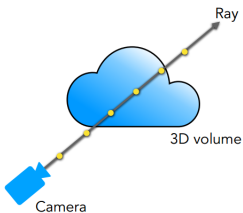
Credit: markboss.me/post/nerf_at_eccv22/

Overview

Three Key Components

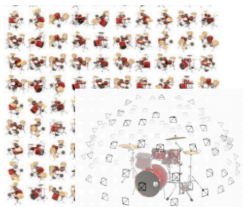


Volumetric 3D Scene Representation



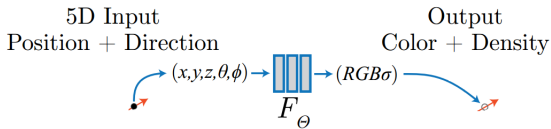
Differentiable Volumetric Rendering Function

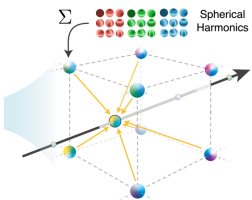
Objective: Synthesize all training views

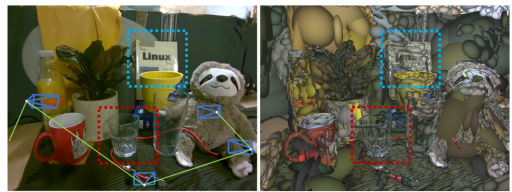


Optimization via Analysis-by-Synthesis

3D Volumetric Representation







Mildenhall *et al.* NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Yu *et al.* Plenoxels: Radiance Fields without Neural Networks

Matsuki *et al.* Gaussian Splatting SLAM

Volumetric Rendering

Rasterization vs. Ray Tracing

Rasterization loop:

For each object

For each pixel—closer?

Ray tracing loop:

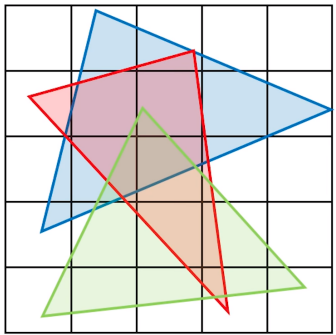
For each pixel

For each object—closest?

https://www.youtube.com/watch?v=ynCxnR1i0QY

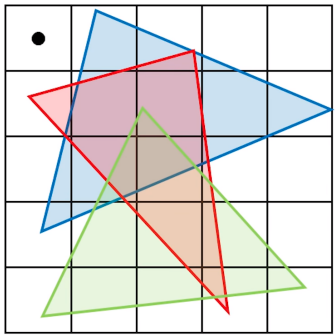
Volumetric Rendering

Ray Tracing



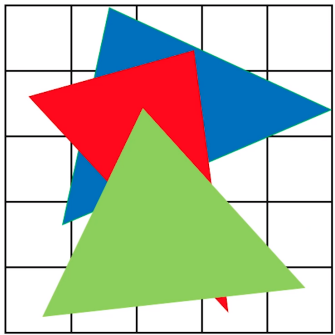
Volumetric Rendering

Ray Tracing



Volumetric Rendering

Rasterization



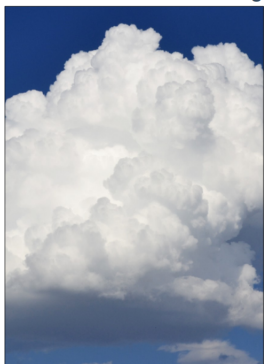
Ray Tracing Derivation

Absorption

Scattering

Emission

http://commons.wikimedia.org



http://wikipedia.org

Simplify

Absorption

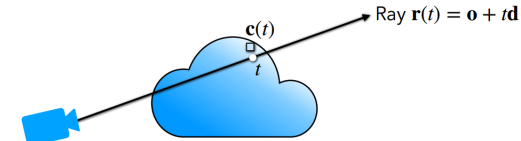
http://commons.wikimedia.org

Scattering

Emission

http://wikipedia.org

Volumetric Rendering Equation



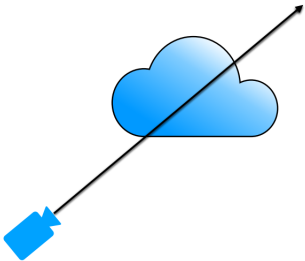
$$C = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Derivation on Board

Mildenhall et al. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

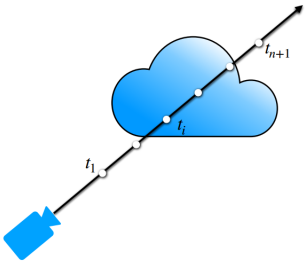
Approximating the integral

Approximating the nested integral



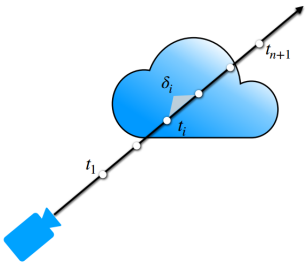
We use quadrature to approximate the nested integral,

Approximating the nested integral



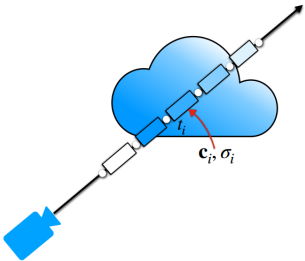
We use quadrature to approximate the nested integral,
splitting the ray up into n segments with endpoints $\{t_1, t_2, \ldots, t_{n+1}\}$

Approximating the nested integral



We use quadrature to approximate the nested integral, splitting the ray up into n segments with endpoints $\{t_1, t_2, \ldots, t_{n+1}\}$ with lengths $\delta_i = t_{i+1} - t_i$

Approximating the nested integral



We assume volume density and color are roughly constant within each interval

Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx$$

This allows us to break the outer integral

Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i\mathbf{c}_i\,dt$$

This allows us to break the outer integral
into a sum of analytically tractable integrals

Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i\mathbf{c}_i\,dt$$

Caveat: piecewise constant density and color
do not imply constant transmittance!

Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i\mathbf{c}_i\,dt$$

Caveat: piecewise constant density and color
do not imply constant transmittance!

Important to account for how early part of a
segment blocks later part when σ_i is high

Evaluating T for piecewise constant density

$$\text{For } t \in [t_i, t_{i+1}],\ T(t) = \exp\left(-\int_{t_1}^{t_i} \sigma_i\, ds\right) \exp\left(-\int_{t_i}^{t} \sigma_i\, ds\right)$$

We need to evaluate at continuous t values
that can lie *partway through* an interval

Evaluating T for piecewise constant density

$$\text{For } t \in [t_i, t_{i+1}],\ T(t) = \exp\left(-\int_{t_1}^{t_i} \sigma_i\, ds\right) \exp\left(-\int_{t_i}^{t} \sigma_i\, ds\right)$$

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

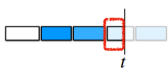
"How much light is blocked by all previous segments?"

Evaluating T for piecewise constant density

For $t \in [t_i, t_{i+1}]$, $T(t) = \exp\left(-\int_{t_1}^{t_i} \sigma_i \, ds\right) \exp\left(-\int_{t_i}^{t} \sigma_i \, ds\right)$

"How much light is blocked partway through the current segment?"

$\exp\left(-\sigma_i(t - t_i)\right)$



Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i\mathbf{c}_i\,dt$$

Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i\mathbf{c}_i\,dt$$

Substitute $$= \sum_{i=1}^{n} T_i\sigma_i\mathbf{c}_i \int_{t_i}^{t_{i+1}} \exp\left(-\sigma_i(t-t_i)\right)\,dt$$

Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i\mathbf{c}_i\,dt$$

$$= \sum_{i=1}^{n} T_i\sigma_i\mathbf{c}_i \int_{t_i}^{t_{i+1}} \exp\left(-\sigma_i(t-t_i)\right)\,dt$$

Integrate

$$= \sum_{i=1}^{n} T_i\sigma_i\mathbf{c}_i \frac{\exp\left(-\sigma_i(t_{i+1}-t_i)\right)-1}{-\sigma_i}$$

Deriving quadrature estimate

$$
\begin{aligned}
\int T(t)\sigma(t)\mathbf{c}(t)\,dt &\approx \sum_{i=1}^{n}\int_{t_i}^{t_{i+1}} T(t)\sigma_i\mathbf{c}_i\,dt \\
&= \sum_{i=1}^{n} T_i\sigma_i\mathbf{c}_i\int_{t_i}^{t_{i+1}} \exp\left(-\sigma_i(t-t_i)\right)\,dt \\
&= \sum_{i=1}^{n} T_i\sigma_i\mathbf{c}_i\,\frac{\exp\left(-\sigma_i(t_{i+1}-t_i)\right)-1}{-\sigma_i} \\
\text{Cancel } \sigma_i \quad &= \sum_{i=1}^{n} T_i\mathbf{c}_i\left(1-\exp(-\sigma_i\delta_i)\right)
\end{aligned}
$$

Connection to alpha compositing

$$= \sum_{i=1}^{n} T_i \mathbf{c}_i \underbrace{(1 - \exp(-\sigma_i \delta_i))}_{\substack{\text{segment} \\ \text{opacity } \alpha_i}}$$

Connection to alpha compositing

$$= \sum_{i=1}^{n} T_i \mathbf{c}_i \underbrace{(1 - \exp(-\sigma_i \delta_i))}_{\text{segment opacity } \alpha_i}$$

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

$$\text{color} = \sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i$$

Summary: volume rendering integral estimate

Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

$$\mathbf{c} \approx \sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i$$

colors

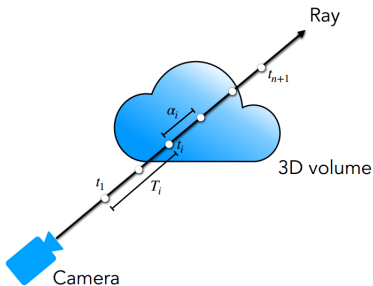
weights

How much light is blocked earlier along ray:

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

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

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



Volume rendering is trivially differentiable

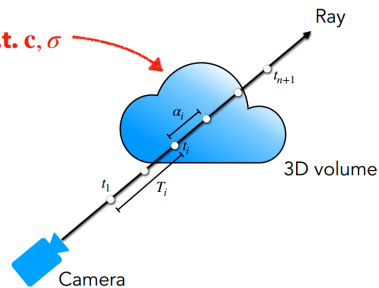
Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

$$\mathbf{c} \approx \sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i$$

colors

weights

differentiable w.r.t. $\mathbf{c}, \sigma$



How much light is blocked earlier along ray:

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

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

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

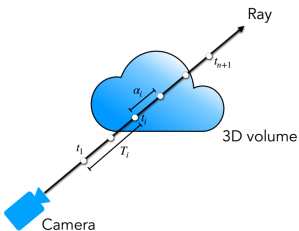
Volumetric Rendering Equation

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

$$\begin{aligned}
& C(p) = \\
&= \sum_{i=1}^{N} c_i (1 - \exp(-\sigma_i \delta_i)) T_i = \\
&= \sum_{i=1}^{N} c_i (1 - \exp(-\sigma_i \delta_i)) \exp(- \sum_{j=1}^{i-1} \sigma_j \delta_j) = \\
&= \sum_{i=1}^{N} c_i \underbrace{(1 - \exp(-\sigma_i \delta_i))}_{\alpha_i} \prod_{j=1}^{i-1} \underbrace{\exp(-\sigma_j \delta_j)}_{1-\alpha_j} = \\
&= \sum_{i=1}^{N} c_i \alpha_i \underbrace{\prod_{j=1}^{i-1} (1 - \alpha_j)}_{transmittance}
\end{aligned}$$

https://towardsdatascience.com/a-comprehensive-overview-of-gaussian-splatting-e7d570081362

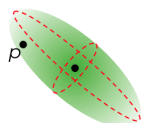
Volumetric Rendering Equation



$$C(p) = \sum_{i=1}^{N} c_i \alpha_i \underbrace{\prod_{j=1}^{i-1}(1-\alpha_j)}_{transmittance}$$

With Densities

$$f_i(p) = \alpha_i \exp(-\frac{1}{2}(p-\mu_i)\Sigma_i^{-1}(p-\mu_i))$$



$$C(p) = \sum_{i \in N} c_i f_i^{2D}(p) \underbrace{\prod_{j=1}^{i-1}(1-f_j^{2D}(p))}_{transmittance}$$

With Gaussians

Methods

NeRF: Basic Approach

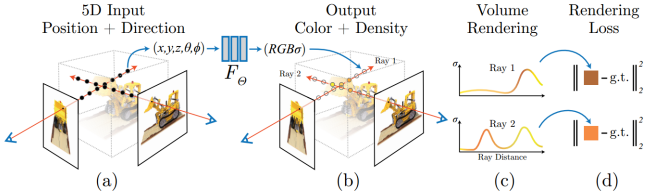


Fig. 2: An overview of our neural radiance field scene representation and differentiable rendering procedure. We synthesize images by sampling 5D coordinates (location and viewing direction) along camera rays (a), feeding those locations into an MLP to produce a color and volume density (b), and using volume rendering techniques to composite these values into an image (c). This rendering function is differentiable, so we can optimize our scene representation by minimizing the residual between synthesized and ground truth observed images (d).

Scene Representation: Neural Networks
Volumetric Rendering: Ray Tracing

Mildenhall *et al.* NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

NeRF: Basic Approach

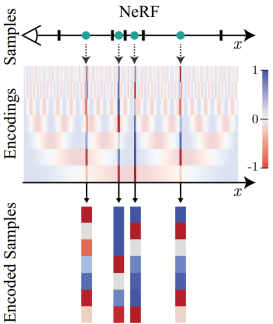
Training Loss:

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

- Uses coarse and fine network to get high resolution images
- Samples 1024 rays, uses 400 x 400 images
- Random sampling of pixels in each iteration
- Other tricks required to make it work

NeRF: Basic Approach

Positional Encoding



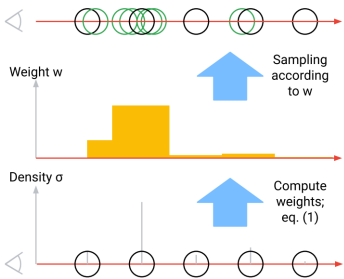
$$\gamma(p) = \big(\sin(2^0\pi p), \cos(2^0\pi p), \cdots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)\big)$$

- Coordinate values are normalized to $[-1, 1]$
- Applied separately to each dimension
- Done for both 3D point and direction vector

Barron *et al.* Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields

NeRF: Basic Approach

Hierarchical Sampling



Arandjelovic *et al.* NeRF in detail: Learning to sample for view synthesis

NeRF: Basic Approach

Architecture

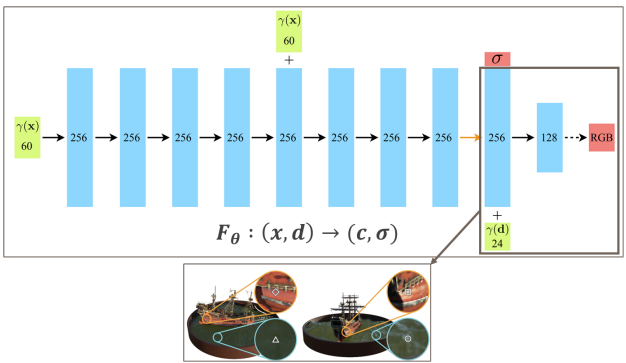


Fig. 5: Comparisons on test-set views for scenes from our new synthetic dataset generated with a physically-based renderer. Our method is able to recover fine

NeRF without Neural Network

Plenoxels: Overview

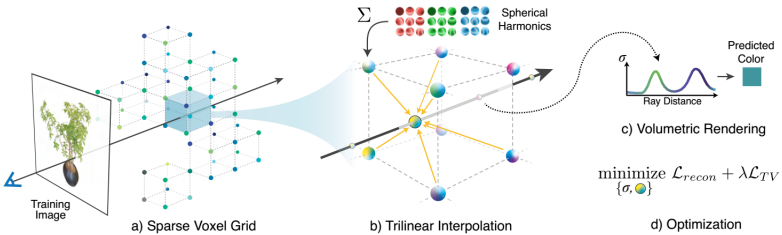


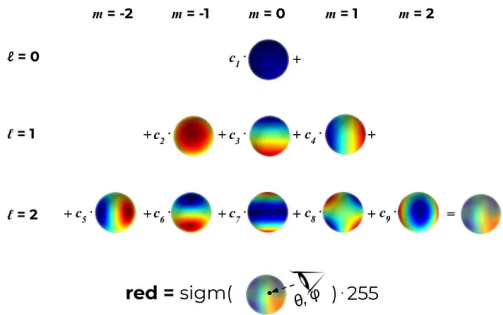
Figure 2. **Overview of our sparse Plenoxel model.** Given a set of images of an object or a scene, we optimize a (a) sparse voxel ("Plenoxel") grid with density and spherical harmonic coefficients at each voxel. To render a ray, we (b) compute the color and opacity of each sample point via trilinear interpolation of the neighboring voxel coefficients. We integrate the color and opacity of these samples using (c) differentiable volume rendering, following the recent success of NeRF [28]. The voxel coefficients can then be (d) optimized using the standard MSE reconstruction loss relative to the training images, along with a total variation regularizer.

Scene Representation: Voxel Grid
Volumetric Rendering: Ray Tracing

Yu *et al.* Plenoxels: Radiance Fields without Neural Networks

NeRF without Neural Network

Spherical Harmonics



NeRF without Neural Network

Plenoxels: Training loss

$$\mathcal{L} = \mathcal{L}_{recon} + \lambda_{TV}\,\mathcal{L}_{TV} \tag{3}$$

Where the MSE reconstruction loss $\mathcal{L}_{recon}$ and the total variation regularizer $\mathcal{L}_{TV}$ are:

$$\mathcal{L}_{recon} = \frac{1}{|\mathcal{R}|}\sum_{\mathbf{r}\in\mathcal{R}} \|C(\mathbf{r}) - \hat{C}(\mathbf{r})\|_2^2$$

$$\mathcal{L}_{TV} = \frac{1}{|\mathcal{V}|}\sum_{\substack{\mathbf{v}\in\mathcal{V}\\ d\in[D]}} \sqrt{\Delta_x^2(\mathbf{v},d) + \Delta_y^2(\mathbf{v},d) + \Delta_z^2(\mathbf{v},d)}$$

with $\Delta_x^2(\mathbf{v},d)$ shorthand for the squared difference between the dth value in voxel $\mathbf{v} := (i,j,k)$ and the dth value in voxel $(i+1,j,k)$ normalized by the resolution, and analogously for $\Delta_y^2(\mathbf{v},d)$ and $\Delta_z^2(\mathbf{v},d)$, where D is the total number of density and spherical harmonic (SH) coefficients stored at each voxel. In practice we use different weights for SH

- Uses coarse to fine strategy
- Uses multi sphere image background model
- Other regularizations required to make it work

Figure 11. **Synthetic scenes.** We show a random view from each of the synthetic scenes, comparing the ground truth, Neural Volumes [ ], JAXNeRF [,], and our Plenoxels.

3D Gaussian Splatting

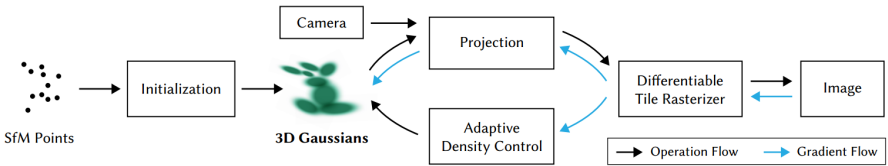
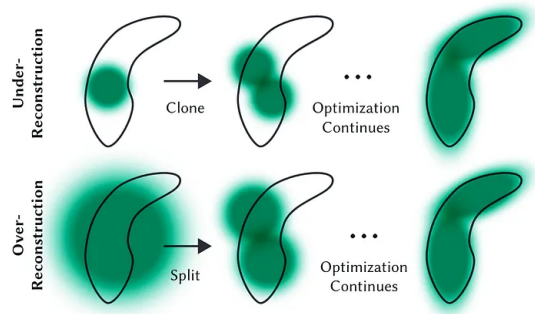


Fig. 2. Optimization starts with the sparse SfM point cloud and creates a set of 3D Gaussians. We then optimize and adaptively control the density of this set of Gaussians. During optimization we use our fast tile-based renderer, allowing competitive training times compared to SOTA fast radiance field methods. Once trained, our renderer allows real-time navigation for a wide variety of scenes.

Scene Representation: 3D Gaussians
Volumetric Rendering: Rasterization

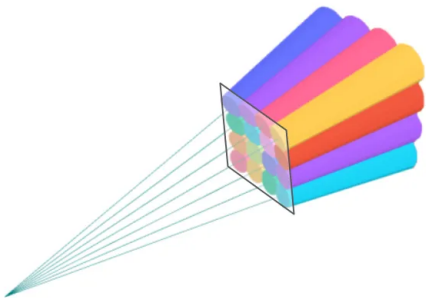
Kerbl *et al.* 3D Gaussian Splatting for Real-Time Radiance Field Rendering

3D Gaussian Splatting



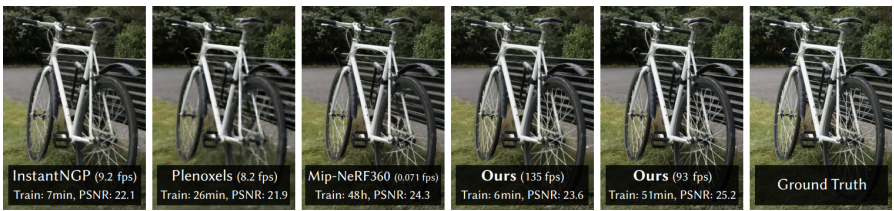
Adaptive control of Gaussians

3D Gaussian Splatting

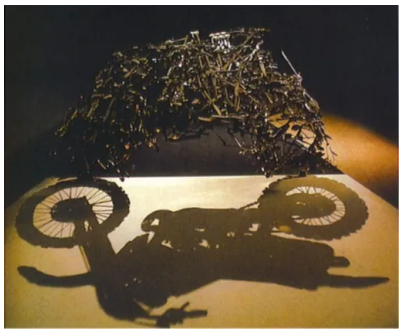


Rasterization

3D Gaussian Splatting



Can you trust your eyes?



Underconstrained Problem

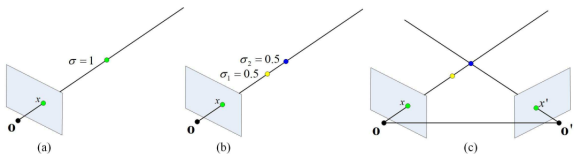
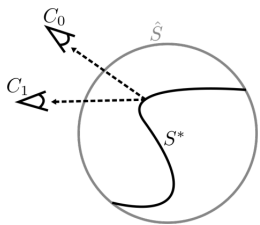


Fig. 1. (a) The correct solution; (b) one incorrect solution; (c) projecting the voxel to another view makes (b) impossible. x and x' are the two camera centers, respectively. $\mathbf{o}$ and $\mathbf{o}'$ are the projections of the voxel projecting onto two views, respectively. σ is the volume density.

Chen *et al.* Structure-Aware NeRF without Posed Camera via Epipolar Constraint

Underconstrained Problem

Shape-Radiance Ambiguity



Shape-Radiance Ambiguity

There exists a family of radiance fields that perfectly explains the training images in the absence of regularization.

Zhang *et al.* NeRF++: Analyzing and Improving Neural Radiance Fields

Underconstrained Problem

Shape-Radiance Ambiguity

GT training view | Prediction (PSNR: 25.4) | GT test view | Prediction (PSNR: 10.7)

Figure 2: To demonstrate the shape-radiance ambiguity, we pretrain NeRF on a synthetic dataset where the opacity field σ is optimized to model an incorrect 3D shape (a unit sphere, instead of a bulldozer shape), while the radiance field $\mathbf{c}$ is optimized to map the training rays' intersection with the sphere and view directions to their pixel color. In this example, we use 3 MLP layers to model the effects of view-dependence (see the MLP structure in Figure 3), and fit to 50 synthetic training images with viewpoints randomly distributed on a hemisphere. The resulting incorrect solution explains the training images very well (left two images), but fails to generalize to novel test views (right two images).

Zhang *et al.* NeRF++: Analyzing and Improving Neural Radiance Fields

The End?

Possibilities

Assumptions

- Known camera motion: NeRF- -, BARF, GNeRF, L2GNeRF, SPARF, Structure-Aware NeRF
- Known intrinsics: NeRF- -
- Same intrinsics for all cameras: Mip-NeRF, Mip-NeRF 360

Problems

- Slow training: Instant Neural Graphics Primitives, Plenoxels, DirectVoxGo, ReLU Fields, TensorRF
- Slow rendering: PlenOctrees, NeX, AdaNeRF

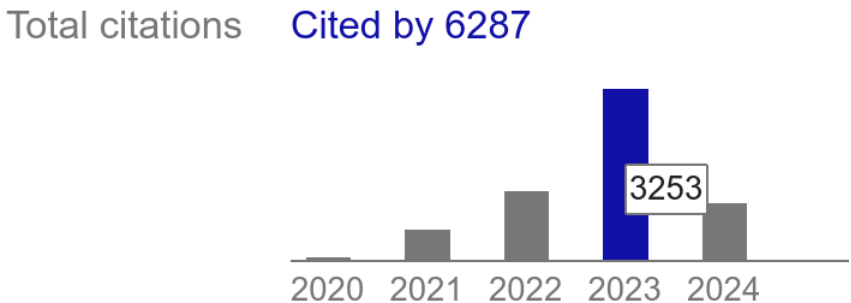
Possibilities

Extensions

- Sparse input: pixelNeRF, MVSNeRF, Depth-supervised NeRF, Dense depth priors
- Scene dependent training/optimization: MVSNeRF
- Large scale: Block-NeRF, NeRFusion
- Unknown camera pose estimation: iNeRF
- Pairwise registration: nerf2nerf

Applications (other than novel view synthesis)

- SLAM: iMAP, NICE-SLAM, NeRF-SLAM, Loc-NeRF
- SfM: Level-S^2fM



Mildenhall *et al.* NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis