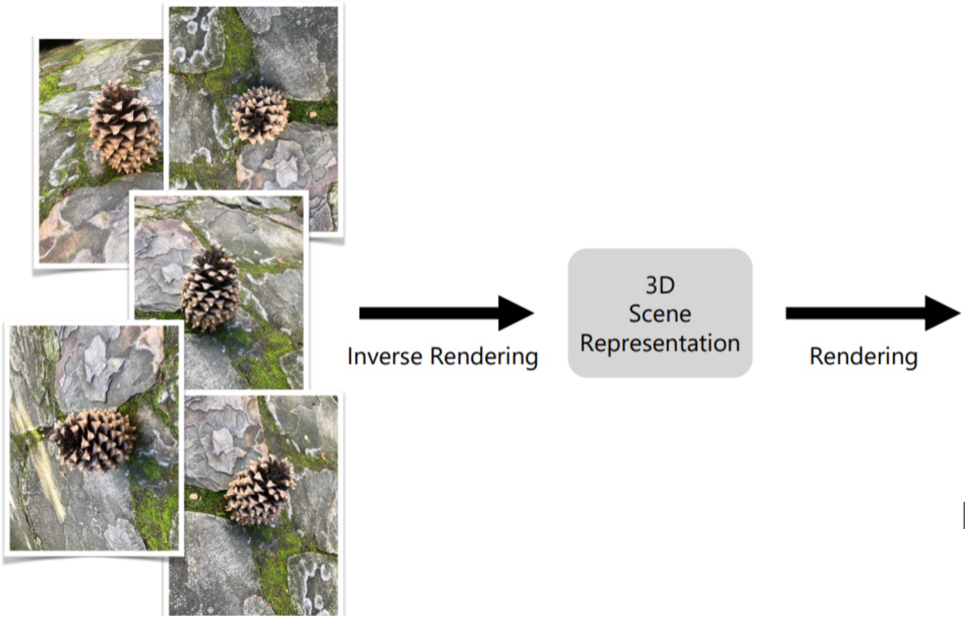
Radiance Fields:
NeRF to 3D Gaussian Splatting

Youngju Na
M.S. Student @ SGVR Lab

yjna2907@kaist.ac.kr

Background: Novel View Synthesis

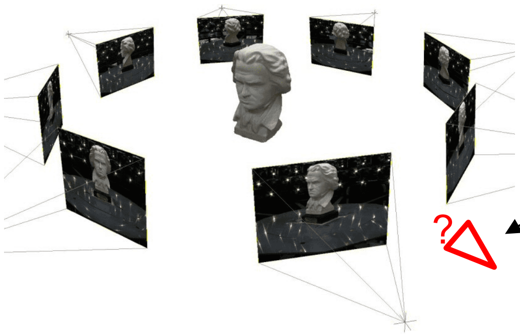


Images from Novel Viewpoints

Images from multiple camera viewpoints

Neural Radiance Fields ECCV 2020 Oral - Best Paper Honorable Mention

Input: images from various camera viewpoints



Output: images from novel camera viewpoints

Source: https://theaisummer.com/nerf/

Examples (synthesized from novel views)

Videos: https://www.youtube.com/watch?v=JuH79E8rdKc&t=191s

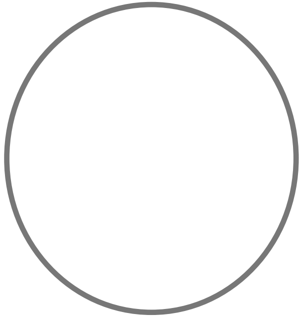
https://www.matthewtancik.com/nerf

Implicit Representation

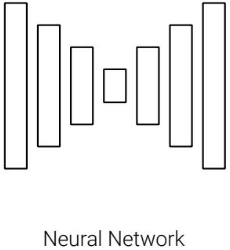
$f(\cdot)$ is a parameterized 2D/3D scalar field

x: coordinate

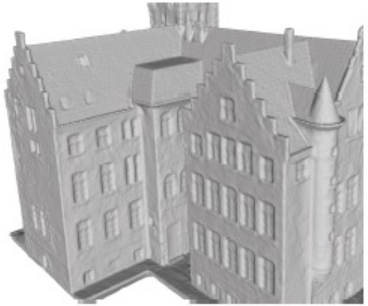
$$f(x) = \|x^2\| - 1$$



x

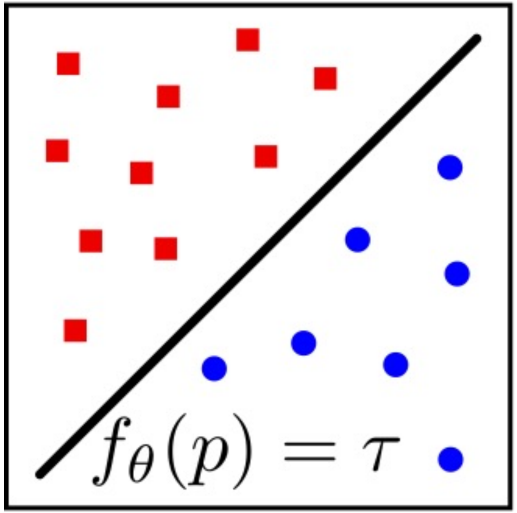


$f(x) = ?$



Represent 3D Scene as Continuous functions

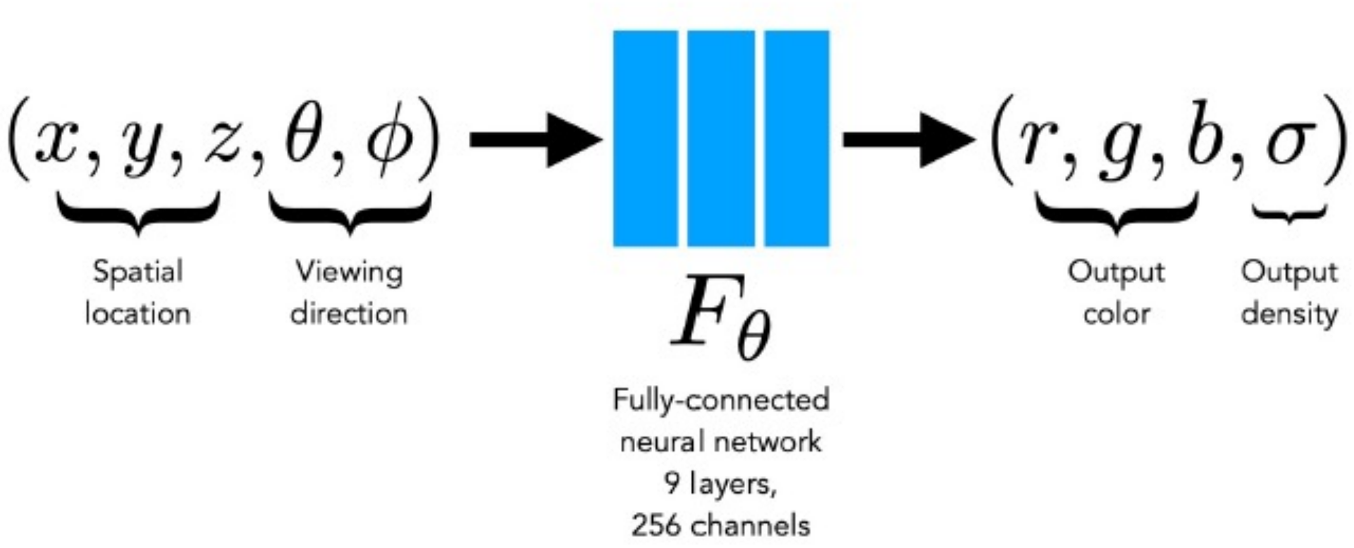
Signed Distance Function (SDF) or Occupancy Fields



Occupancy networks, Mescheder et al.

NeRF 3D Representations

Neural Network as a continuous shape representaiton.

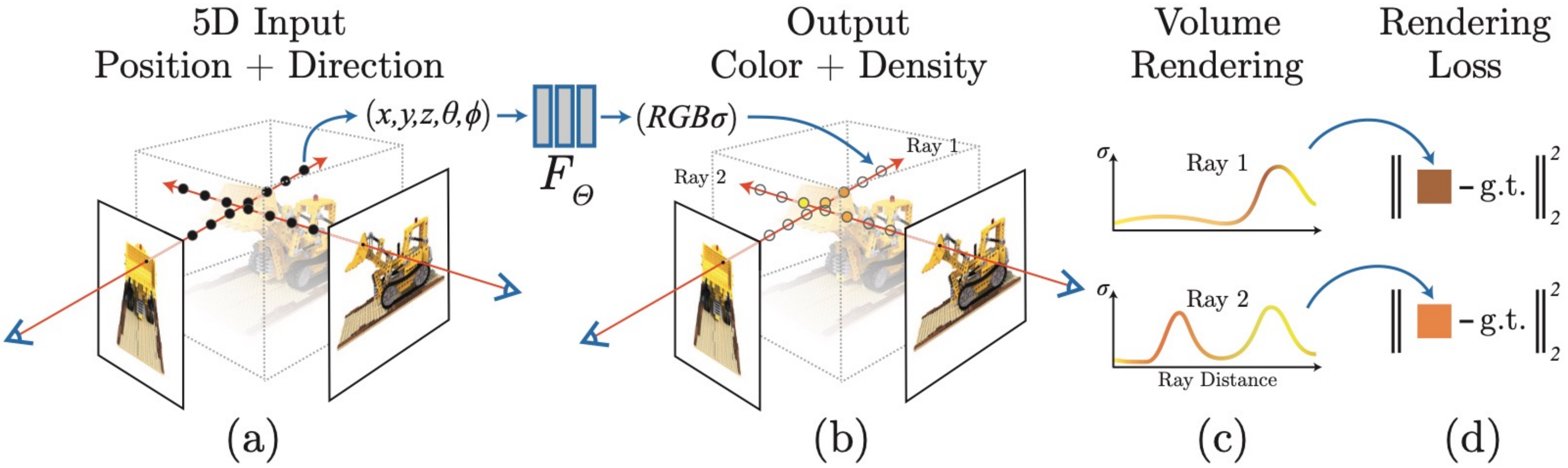


How do we learn 3D representations from 2D images?

https://www.matthewtancik.com/nerf

Method Overview

Cast Rays => Estimate 3D Representations => **Volume Rendering** => 2D Photometric Loss



https://www.matthewtancik.com/nerf

Neural Volumetric Rendering

Neural Volumetric **Rendering**

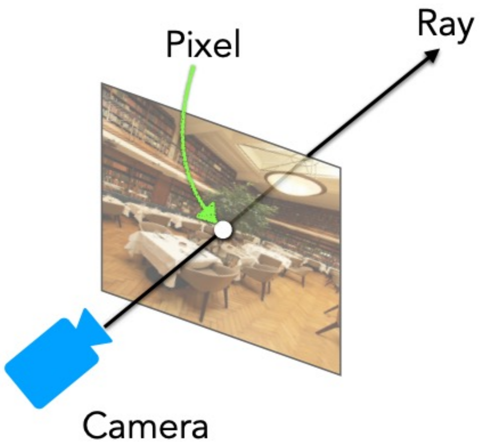
computing color along rays
through 3D space

What color is this pixel?

https://sites.google.com/berkeley.edu/nerf-tutorial/home

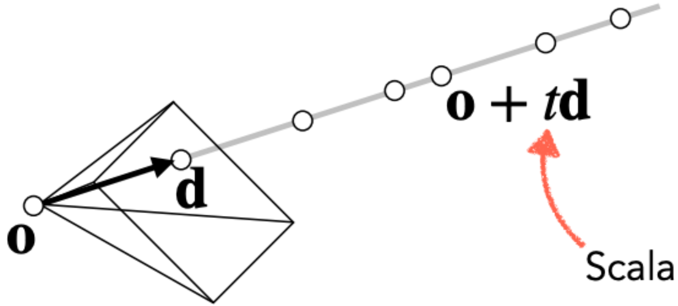
Cameras and rays

- We need the mathematical mapping from (*camera*, *pixel*) → *ray*
- Then can abstract underlying problem as learning the function *ray* → *color* (the "plenoptic function")



https://sites.google.com/berkeley.edu/nerf-tutorial/home

Calculating points along a ray



Scalar t controls distance along the ray

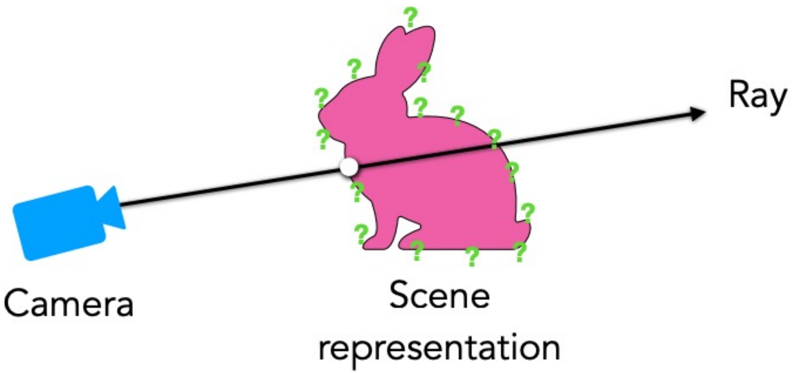
https://sites.google.com/berkeley.edu/nerf-tutorial/home

Neural **Volumetric** Rendering

continuous, differentiable
rendering model without
concrete ray/surface intersections

https://sites.google.com/berkeley.edu/nerf-tutorial/home

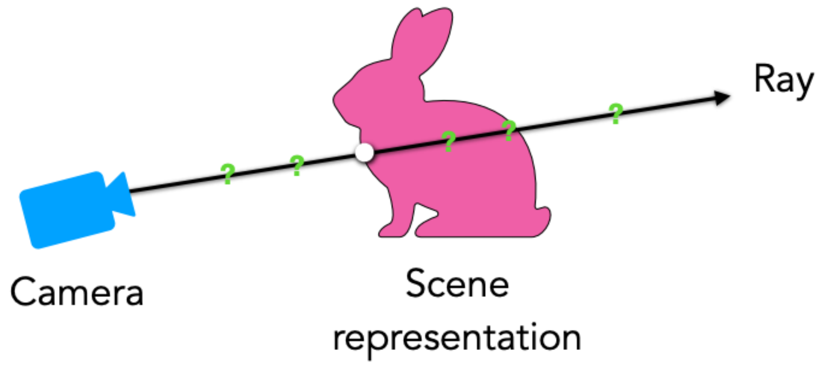
Surface vs. volume rendering



Surface rendering — loop over geometry, check for ray hits

https://sites.google.com/berkeley.edu/nerf-tutorial/home

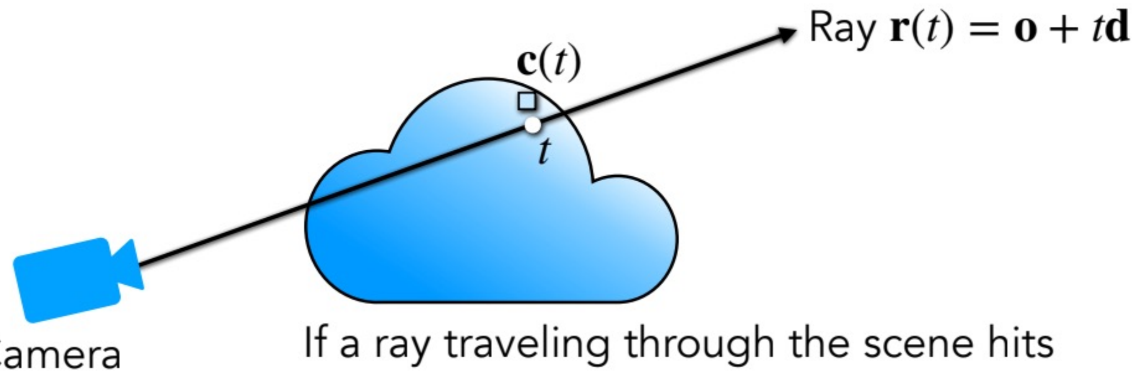
Surface vs. volume rendering



Volume rendering — loop over ray points, query geometry

https://sites.google.com/berkeley.edu/nerf-tutorial/home

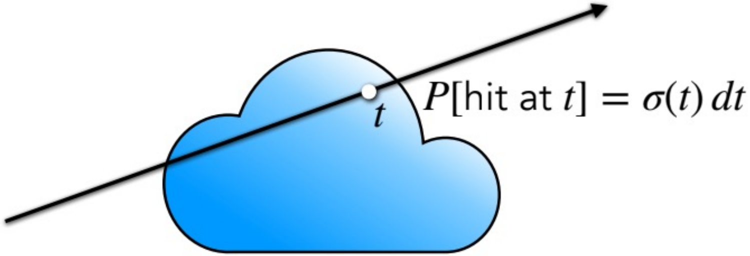
Volumetric formulation for NeRF



If a ray traveling through the scene hits a particle at distance t along the ray, we return its color $\mathbf{c}(t)$

https://sites.google.com/berkeley.edu/nerf-tutorial/home

What does it mean for a ray to "hit" the volume?

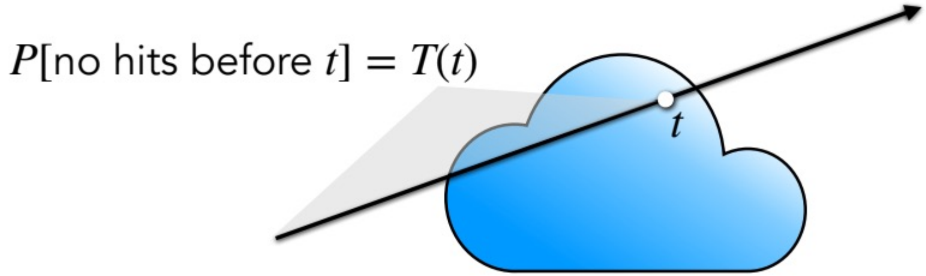


This notion is *probabilistic:* chance that ray hits a particle in a small interval around t is $\sigma(t)\, dt$.

σ is called the "volume density"

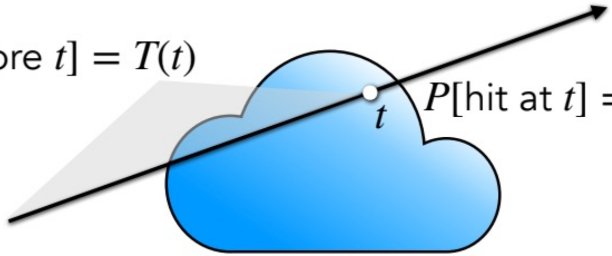
https://sites.google.com/berkeley.edu/nerf-tutorial/home

Probabilistic interpretation



To determine if t is the *first* hit along the ray, need to know $T(t)$: the probability that the ray makes it through the volume up to t.
$T(t)$ is called "transmittance"

PDF for ray termination



Finally, we can write the probability that a ray terminates at t as a function of only sigma

$$P[\text{first hit at } t] = P[\text{no hit before } t] \times P[\text{hit at } t]$$

$$= T(t)\sigma(t)dt$$

$$= \exp\left(-\int_{t_0}^{t} \sigma(s)\,ds\right)\sigma(t)\,dt$$

https://sites.google.com/berkeley.edu/nerf-tutorial/home

Expected value of color along ray

This means the expected color returned by the ray will be

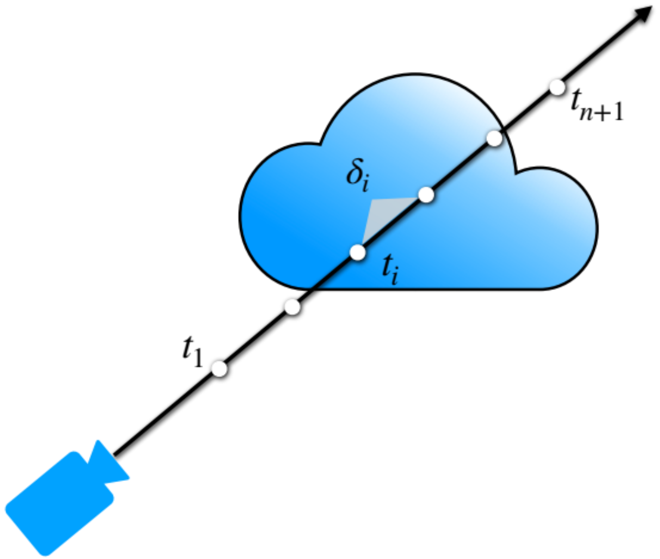
color

$$\int_{t_0}^{t_1} T(t)\sigma(t)\boxed{\mathbf{c}(t)}\,dt$$

Note the nested integral!

https://sites.google.com/berkeley.edu/nerf-tutorial/home

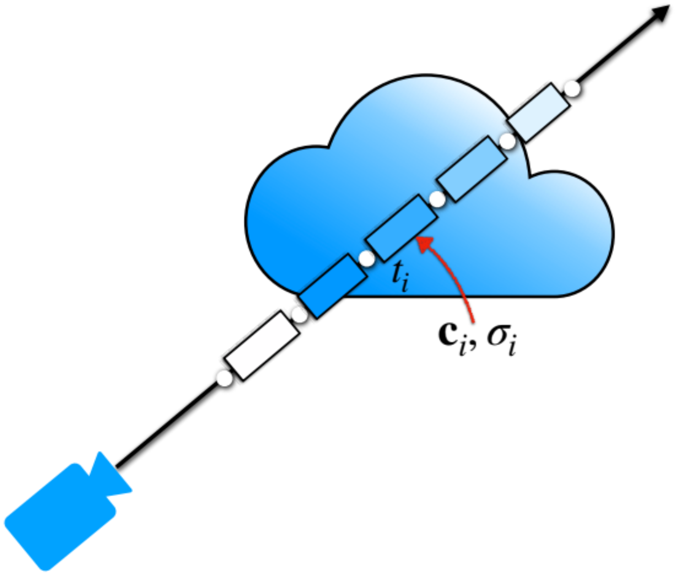
Approximating the nested integral



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

https://sites.google.com/berkeley.edu/nerf-tutorial/home

Approximating the nested integral



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

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

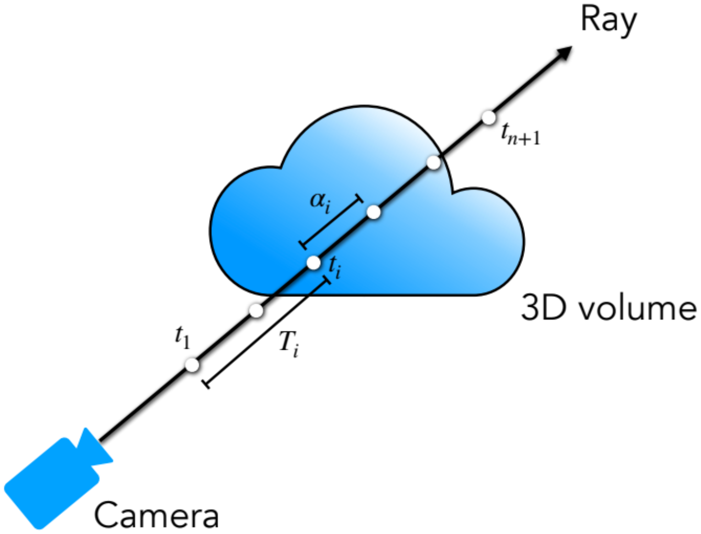
weights

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:

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



[Detailed derivation]

https://sites.google.com/berkeley.edu/nerf-tutorial/home

Volume rendering is trivially differentiable

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

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

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

colors

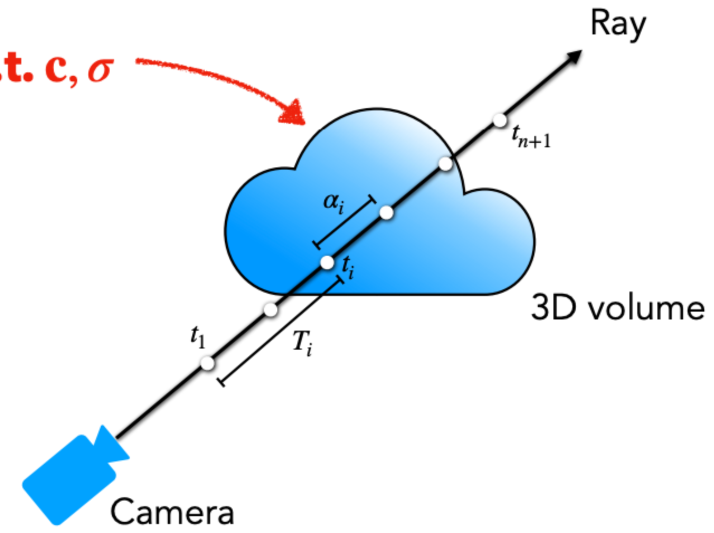
weights

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*:

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



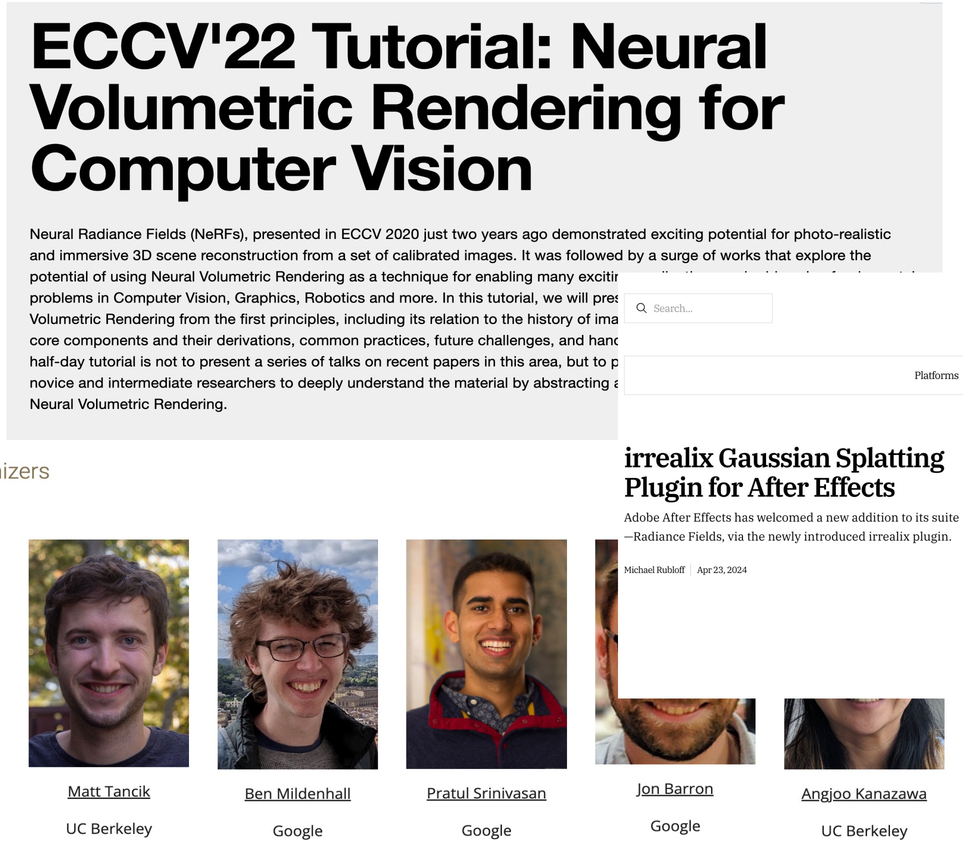
https://sites.google.com/berkeley.edu/nerf-tutorial/home

Video

Mildenhall et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020

Novel View Synthesis & View Dependency

Resources



ECCV'22 Tutorial: Neural Volumetric Rendering for Computer Vision

Neural Radiance Fields (NeRFs), presented in ECCV 2020 just two years ago demonstrated exciting potential for photo-realistic and immersive 3D scene reconstruction from a set of calibrated images. It was followed by a surge of works that explore the potential of using Neural Volumetric Rendering as a technique for enabling many excitir
problems in Computer Vision, Graphics, Robotics and more. In this tutorial, we will pres
Volumetric Rendering from the first principles, including its relation to the history of ima
core components and their derivations, common practices, future challenges, and hanc
half-day tutorial is not to present a series of talks on recent papers in this area, but to p
novice and intermediate researchers to deeply understand the material by abstracting a
Neural Volumetric Rendering.

Organizers

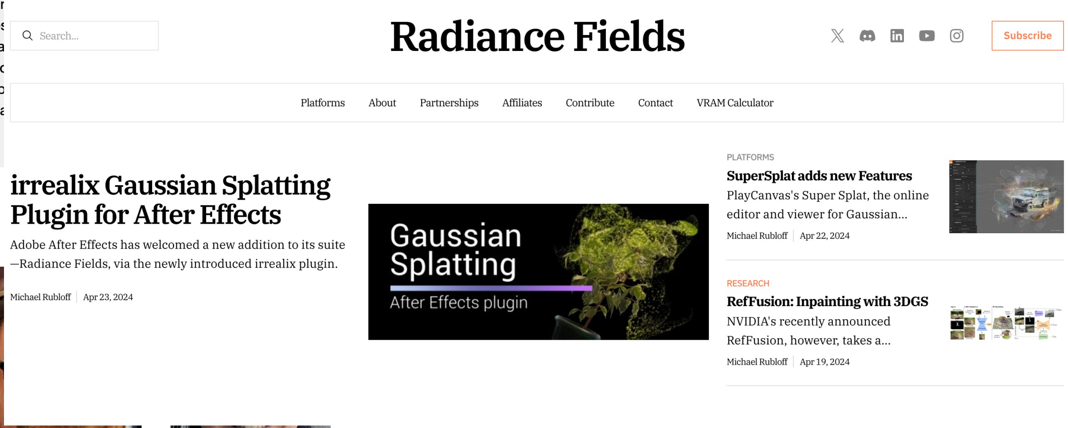
Matt Tancik
UC Berkeley

Ben Mildenhall
Google

Pratul Srinivasan
Google

Jon Barron
Google

Angjoo Kanazawa
UC Berkeley



Search...

Radiance Fields

Subscribe

Platforms | About | Partnerships | Affiliates | Contribute | Contact | VRAM Calculator

irrealix Gaussian Splatting Plugin for After Effects

Adobe After Effects has welcomed a new addition to its suite—Radiance Fields, via the newly introduced irrealix plugin.

Michael Rubloff | Apr 23, 2024

PLATFORMS

SuperSplat adds new Features

PlayCanvas's Super Splat, the online editor and viewer for Gaussian...

Michael Rubloff | Apr 22, 2024

RESEARCH

RefFusion: Inpainting with 3DGS

NVIDIA's recently announced RefFusion, however, takes a...

Michael Rubloff | Apr 19, 2024

https://sites.google.com/berkeley.edu/nerf-tutorial/home

https://radiancefields.com/

KAIST

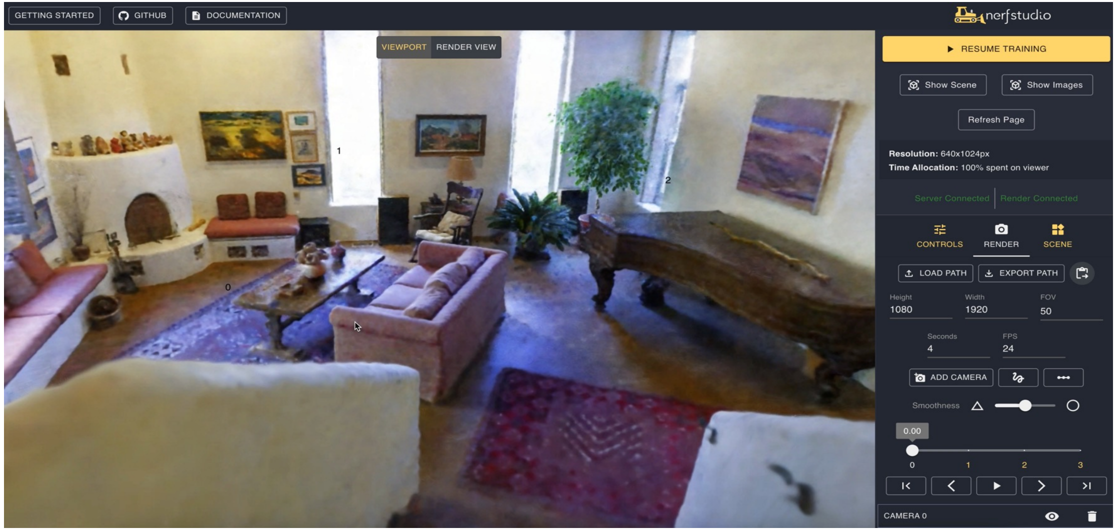
Resources

docs passing | pypi package 1.0.3 | Core Tests. passing | License Apache 2.0

nerfstudio

A collaboration friendly studio for NeRFs

Documentation | Viewer | Colab



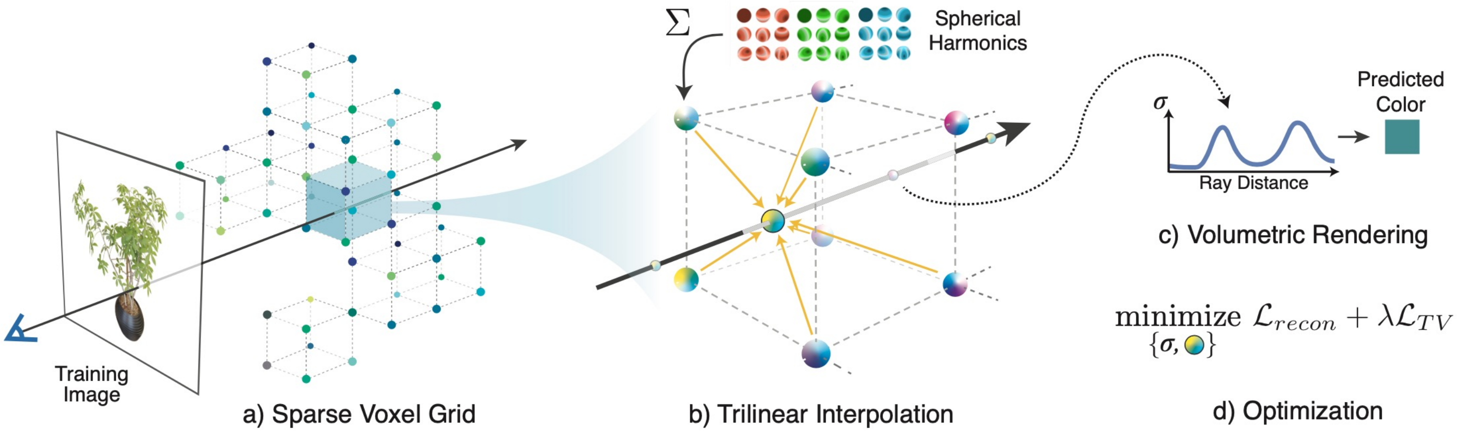
https://docs.nerf.studio/

KAIST

3D Gaussian Splatting

SIGGRAPH 2023 best paper award

Plenoxel [CVPR’22]: Fast Optimization / Rendering



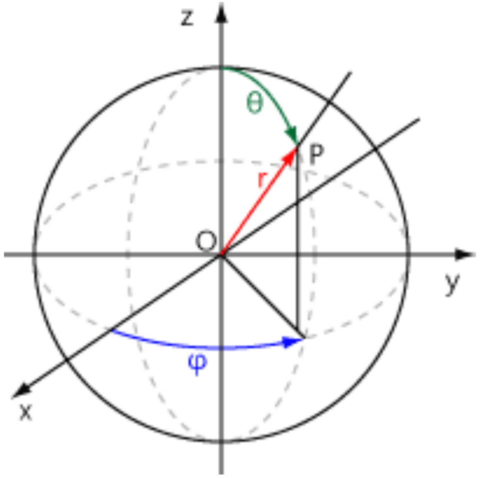
What does it optimize?

- **Spherical Harmonics (SH) Coefficients**
- **Volume Density**

=> No Neural Network

https://alexyu.net/plenoxels/

Spherical Harmonics (SH)



(θ : polar angle, φ : azimuthal angle)

If we solve Laplace equation in surface point,

$$Y_l^m(\theta,\phi) = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}}\,P_l^{|m|}(\cos\theta)e^{im\phi}$$

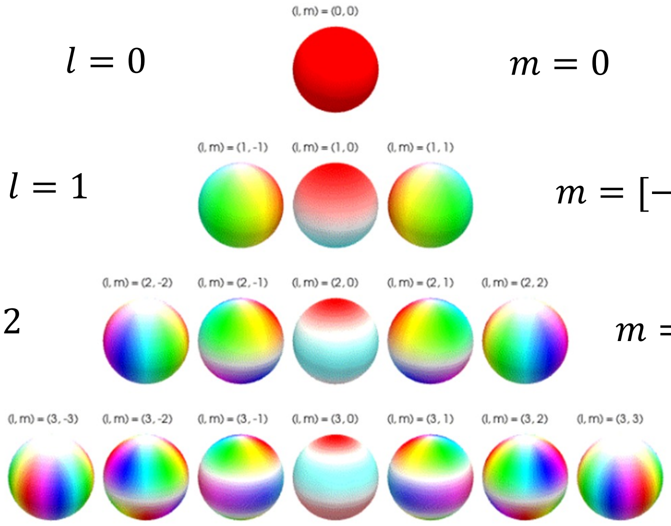
$$P_\ell^{(|m|)}(\cos\theta) = (-1)^m \frac{(\ell+|m|)!}{(\ell-|m|)!} P_\ell^{(-|m|)}(\cos\theta)$$

$l = 0$ $m = 0$

$l = 1$ $m = [-1, 0, 1]$

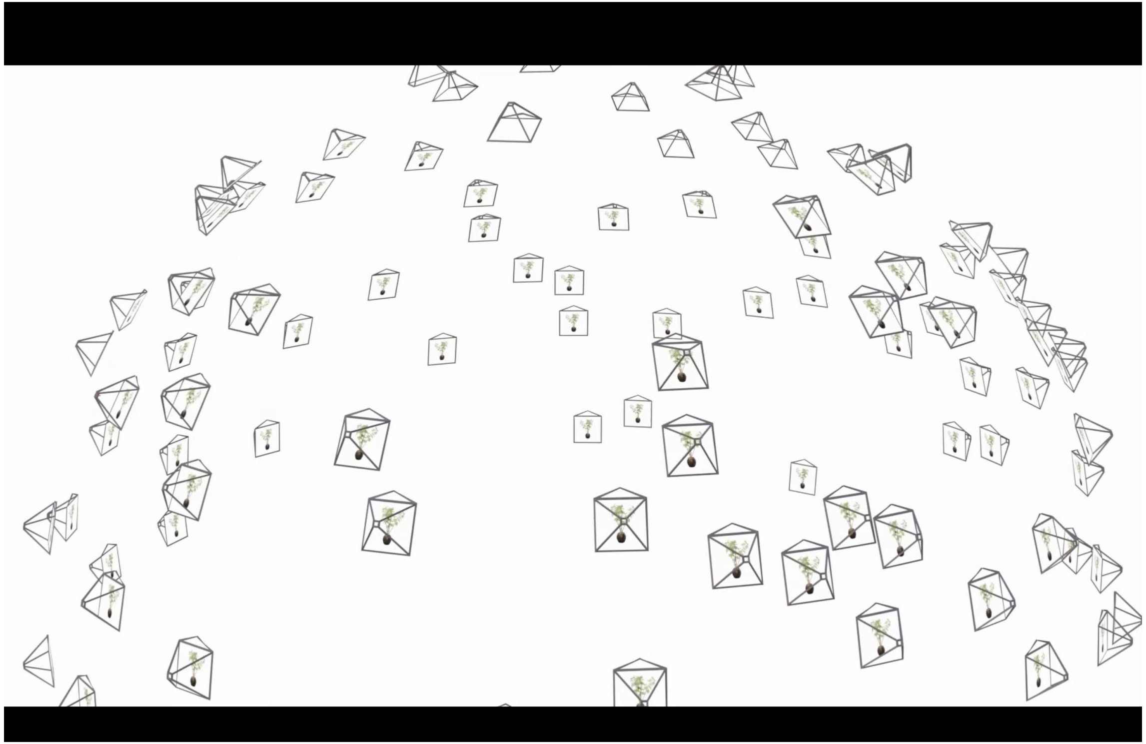
$l = 2$ $m = [-2, -1, 0, 1, 2]$

$l = 3$



KAIST

Fast Optimization / Rendering : Plenoxel [CVPR’22]



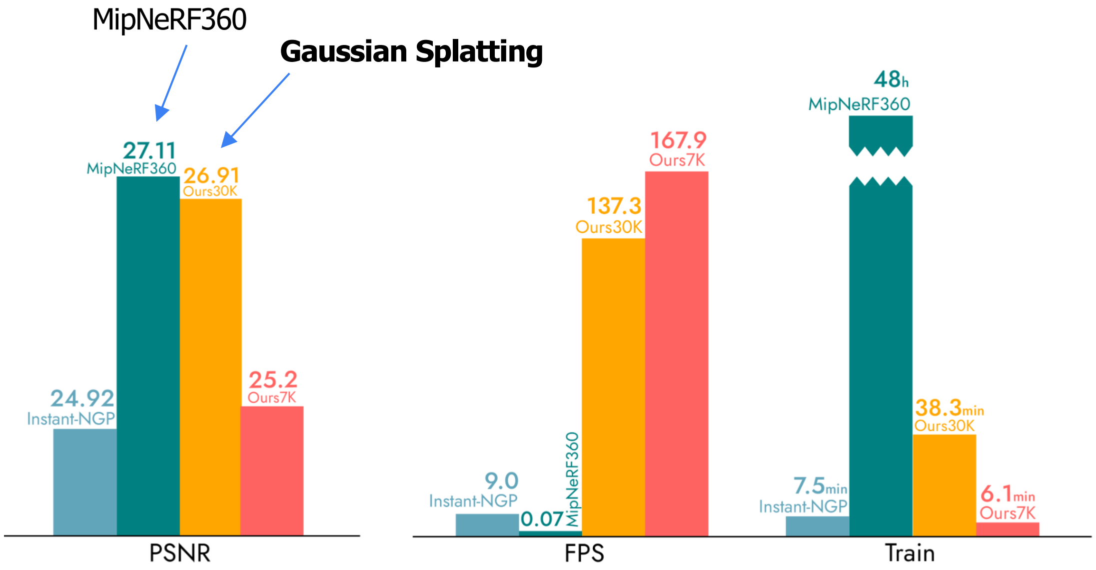
Mip-NeRF360 vs Plenoxels

Train	1.6 days	**~ 30 mins**
FPS	0.06s	**6.8s**
Memory	**8.6MB**	2.1GB

https://alexyu.net/plenoxels/

Gaussian Splatting: Fast 3D Reconstruction and Rendering (3DGS)

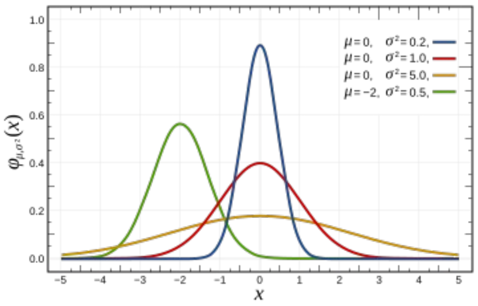
- Gaussian Splatting is a fast training and real-time rendering framework.
- Takes ~1hour training and achieves >**120 FPS.**



Gaussian Splatting: Fast 3D Reconstruction and Rendering

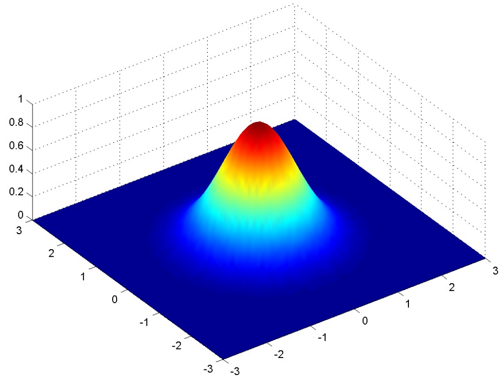
- **Representation: 3D Gaussians**

1D Gaussians

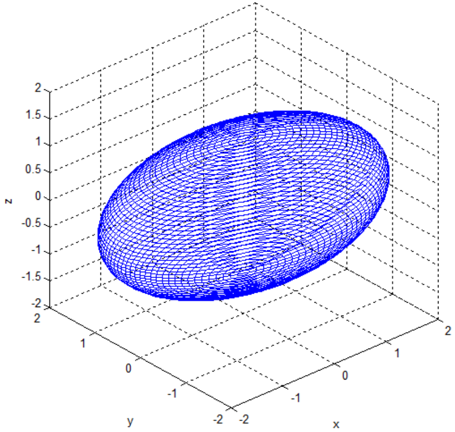


$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

2D Gaussians



3D Gaussians



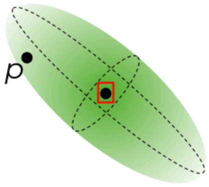
$$g(x) = exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Generalized multivariate gaussian distribution (without normalization)

Gaussian Splatting: Fast 3D Reconstruction and Rendering

- **Representation: 3D Gaussians**

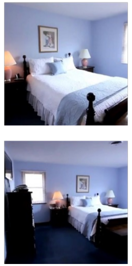
$$f_i(p) = \sigma(\alpha_i) \exp\left(-\frac{1}{2}(p - \boxed{\mu_i})\Sigma_i^{-1}(p - \boxed{\mu_i})\right)$$



Each 3D Gaussian is parametrized by:

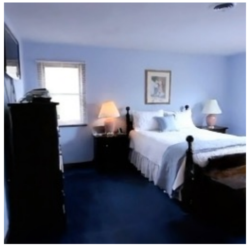
- **Mean μ:** 3D position (x, y, z)
- **Covariance Σ = RSS^TR^T**; (Scale S, Rotation R)
- **Opacity**: $\sigma(\alpha)$
- **Color parameters:** spherical harmonics (SH) coefficients.

Input Views



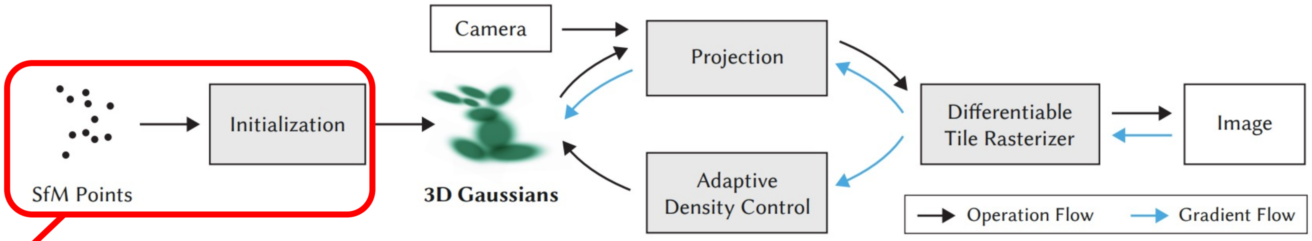
3D Gaussian

Novel Views

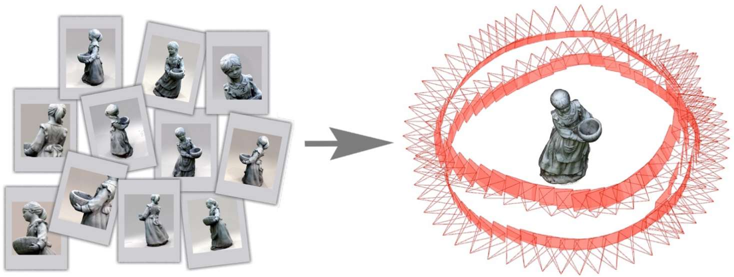


3D Gaussian Splatting (3D-GS)

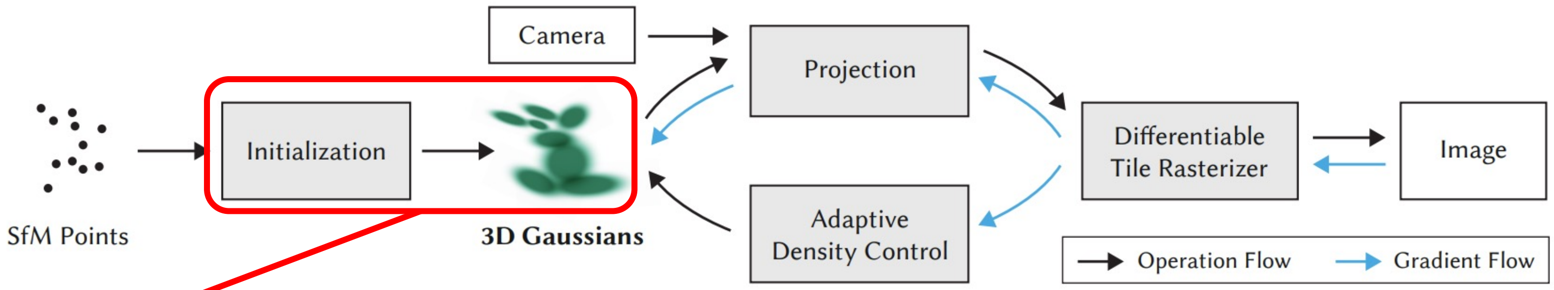
3D Gaussian Splatting for Real-Time Radiance Field Rendering [SIGGRAPH'23 Best Paper Award]



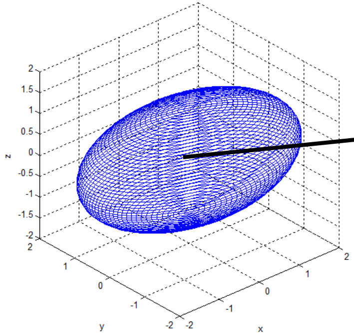
Step 1. Initialize points via Structure from Motion (SfM)



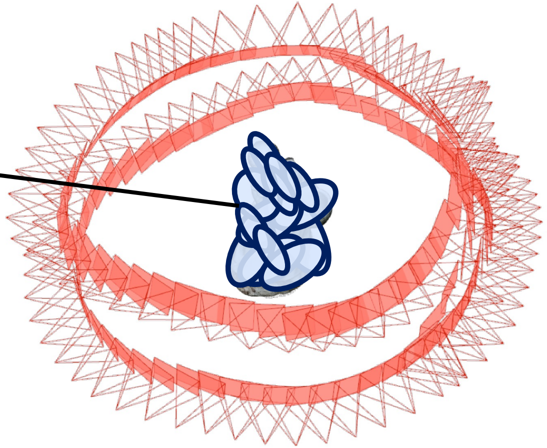
3D Gaussian Splatting (3D-GS)



Step 2. Represent points with multivariate Gaussians and assign parameters

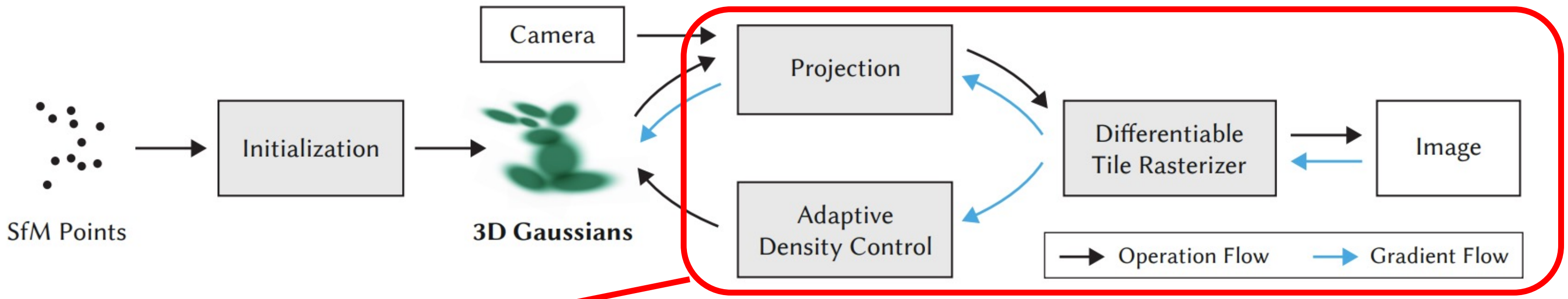


μ: Position
Σ: Covariance
S: Color
α: alpha

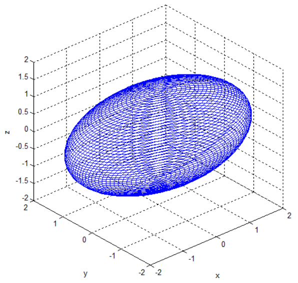


3D Gaussian Splatting for Real-Time Radiance Field Rendering [SIGGRAPH'23 Best Paper]

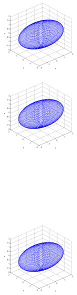
3D Gaussian Splatting (3D-GS)



Step 3. **Convert Gaussian primitives into an Image (Rasterize)**



Multivariate gaussians



$$\{\mu_1, \Sigma_1, S_1, \alpha_1\}$$
$$\{\mu_2, \Sigma_2, S_2, \alpha_2\}$$
$$\cdots$$
$$\{\mu_N, \Sigma_N, S_N, \alpha_N\}$$

Rasterization

1. Project into 2d according to the camera
2. Sort by depths
3. Aggregate gaussians (alpha-blending)

Training **(No Neural Network)**

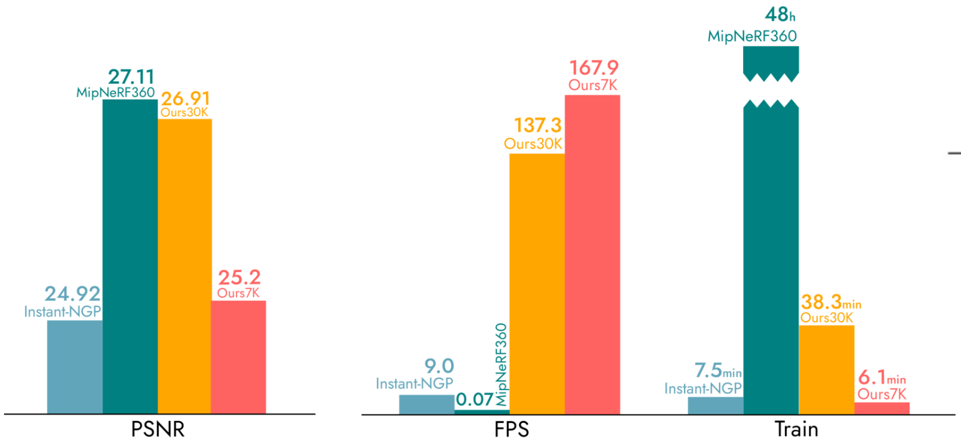
L {  ,  }

optimize

3D Gaussian Splatting for Real-Time Radiance Field Rendering [SIGGRAPH'23 Best Paper]

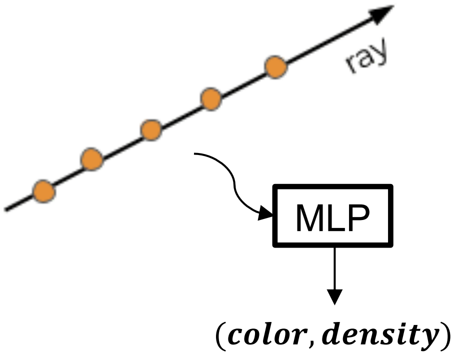
Experimental Results

3DGS has shown high quality rendering with 130+ FPS (real-time).



Dataset	Mip-NeRF360					
Method|Metric	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	Train	FPS	Mem
Plenoxels	0.626	23.08	0.463	25m49s	6.79	2.1GB
INGP-Base	0.671	25.30	0.371	5m37s	11.7	13MB
INGP-Big	0.699	25.59	0.331	7m30s	9.43	48MB
M-NeRF360	$0.792^{\dagger}$	$27.69^{\dagger}$	$0.237^{\dagger}$	48h	0.06	8.6MB
Ours-7K	0.770	25.60	0.279	6m25s	160	523MB
Ours-30K	0.815	27.21	0.214	41m33s	134	734MB

NeRF

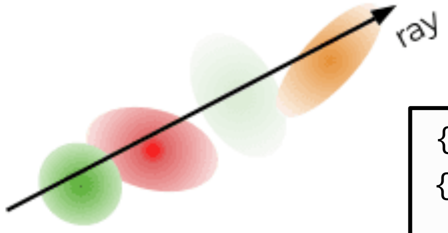


Implicit representation (Neural networks)

Volume Rendering

Slow rendering, low memory

Gaussian Splatting



$\{\mu_1, \Sigma_1, S_1, \alpha_1\}$
$\{\mu_2, \Sigma_2, S_2, \alpha_2\}$
$\cdots$
$\{\mu_N, \Sigma_N, S_N, \alpha_N\}$

Explicit representation (3D Gaussians)

Rasterization

Real-time rendering, higher memory

Applications

NeRF/3DGS Applications

1. Assume static scene => Dynamic Scene

2. Generative Models (Text-to-3D, Image-to-3D, etc.)

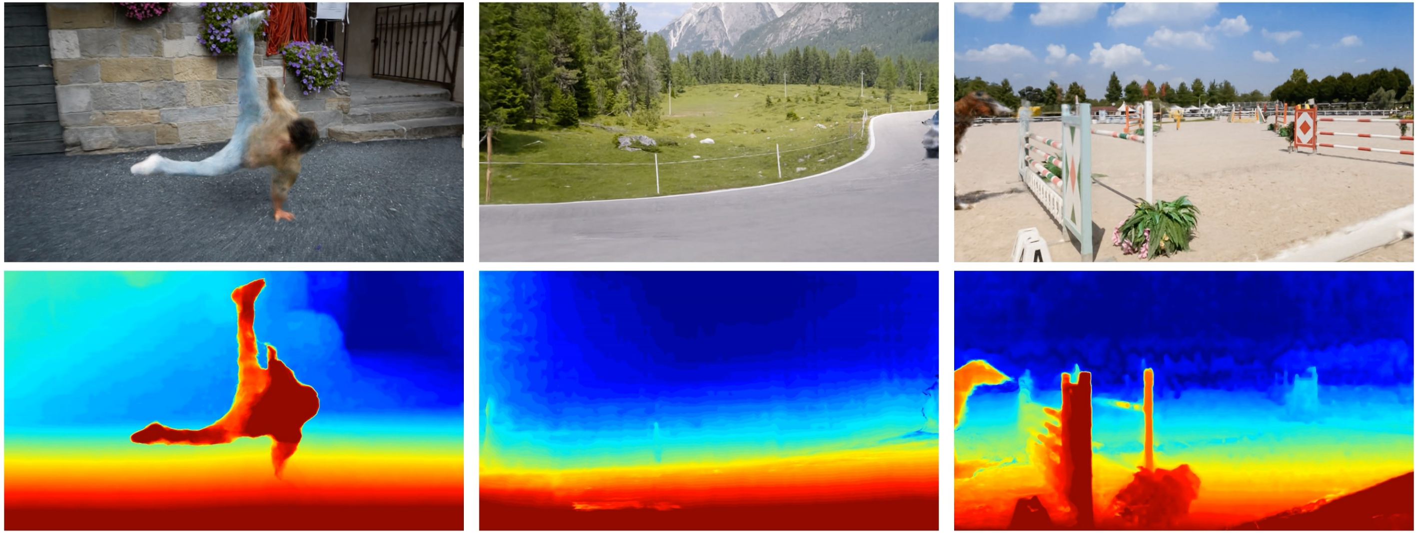
3. Relighting / Light Modeling

4. Navigation / Autonomous Driving

Etc.

List goes on and on…!
NeRF has been cited 6800+
3DGS, 1400+

The world we capture is usually Dynamic / Deformable



Dynamic NeRFs / 3DGS

RoDynRF, Liu et al. CVPR’23

DynIBaR, Li et al. CVPR’23

Bae et al. ECCV’24

KAIST

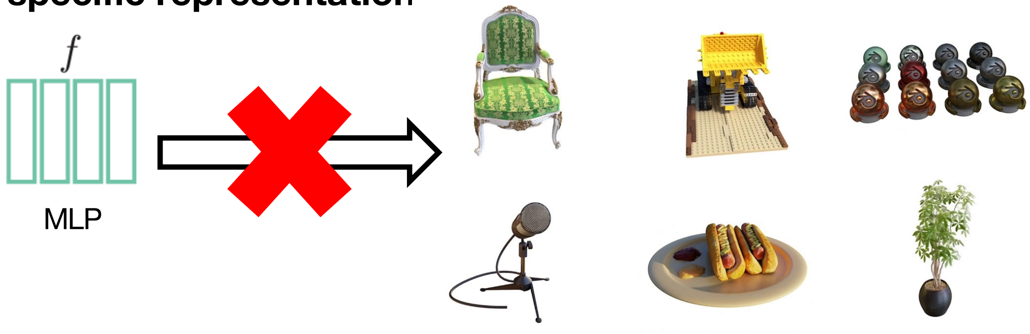
https://www.albertpumarola.com/research/D-NeRF

NeRF/3DGS requires Per-Scene Optimization

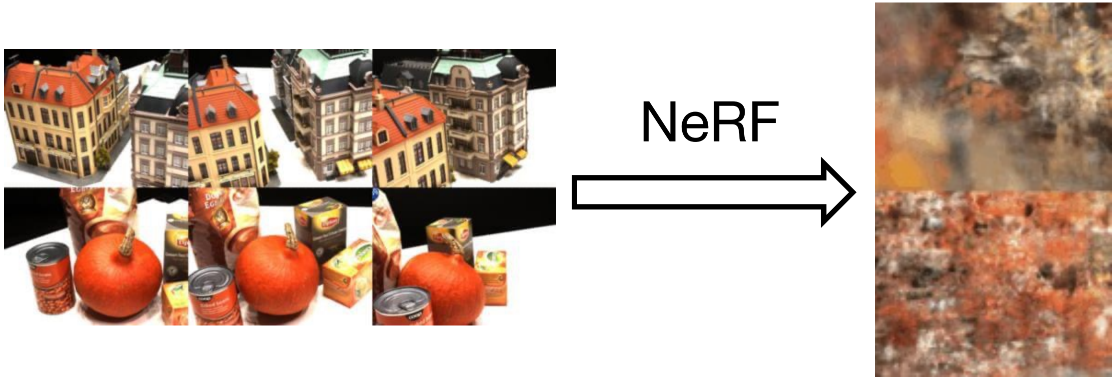
Generalizable Methods with Prior Knowledge

NeRF/3DGS requires Per-Scene Optimization with Dense Views

1. Scene-specific representation



2. Sparse input camera viewpionts



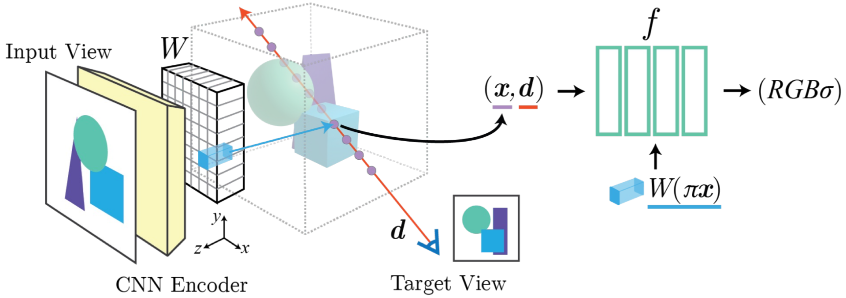
Not Generalizable

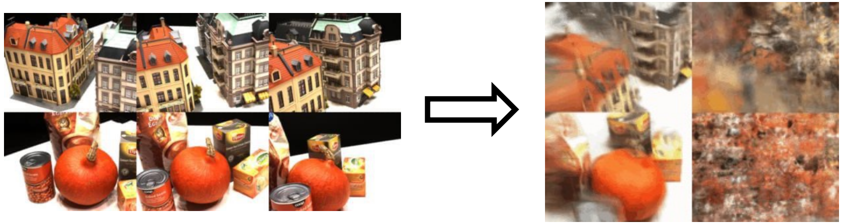
Cannot share representations across scenes or views

Generalizable NeRF / 3DGS

- Note 1. No Per Scene Optimization ❌, Generalizable ✅
- Note 2. No Dense Views ❌, Only 2-3 images ✅

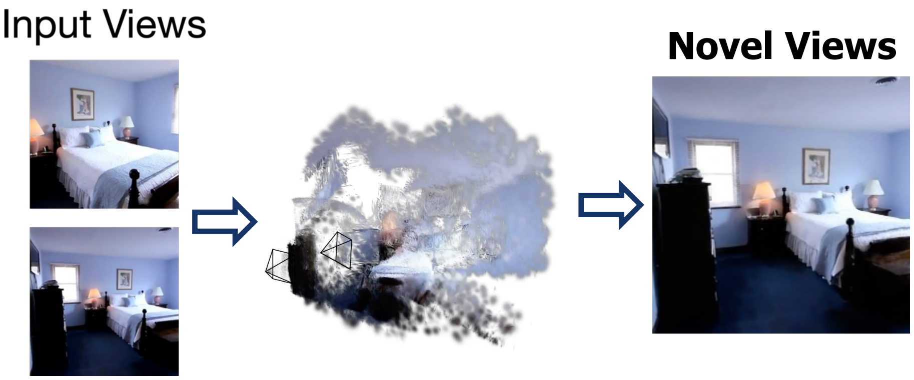
- **One-Shot NeRF (pixelNeRF [Yu et al. CVPR'21])**





PixelNeRF NeRF

- **One-Shot 3DGS (PixelSplat [Charatan et al. CVPR'24])**



Radiance Fields with Generative Models
Text-to-3D, Image-to-3D

LGM, ECCV’24

“motorcycle”

“mech suit”

“ghost lantern”

“furry fox head”

“dresser”

“swivel chair”

“astronaut”

“mushroom house”

DreamScenes, ECCV'24

A DSLR photo of a living room

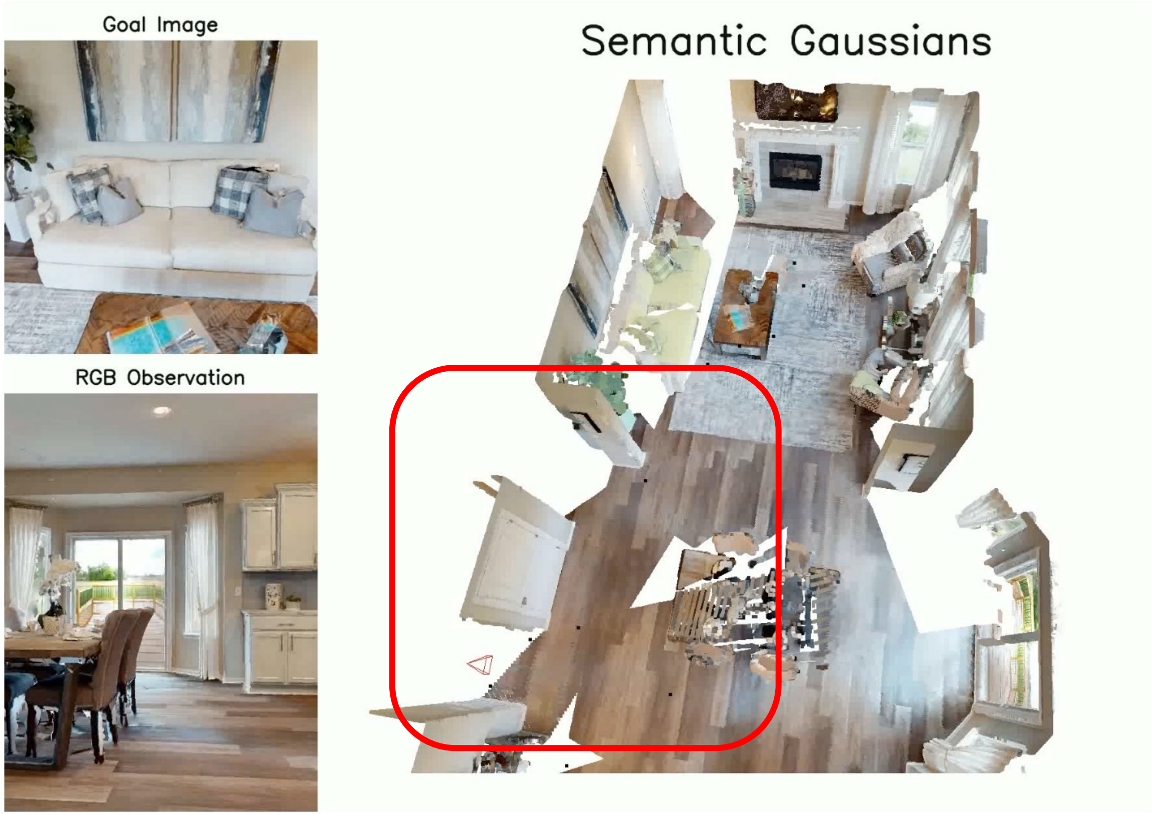
DSLR photo of a cyberpunk style bedroom, cyberpunk style

A minecraft cubes world with lake and mountains in the far distance and grass cubes in the near distance

DreamFusion [Poole et al. arXiv 2022]

SLAM
Localization and Mapping

GaussNav, ECCV'24



Other applications

Table of contents

Explore other applications that might interest you

https://github.com/MrNeRF/awesome-3D-gaussian-splatting

Thank you

yjna2907@kaist.ac.kr