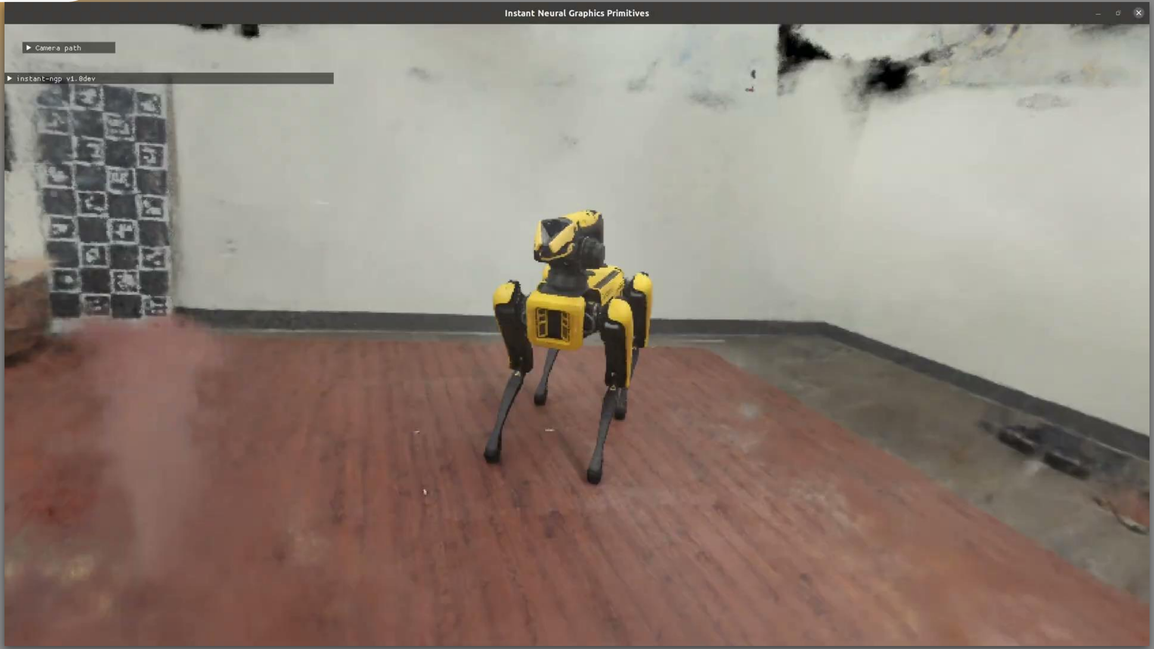
DeepRob

[Group 2] Lecture 3

by Harshavardhan, Raj Surya, Vaibhav

NeRFs, Gaussian Splatting and Manipulation

University of Minnesota

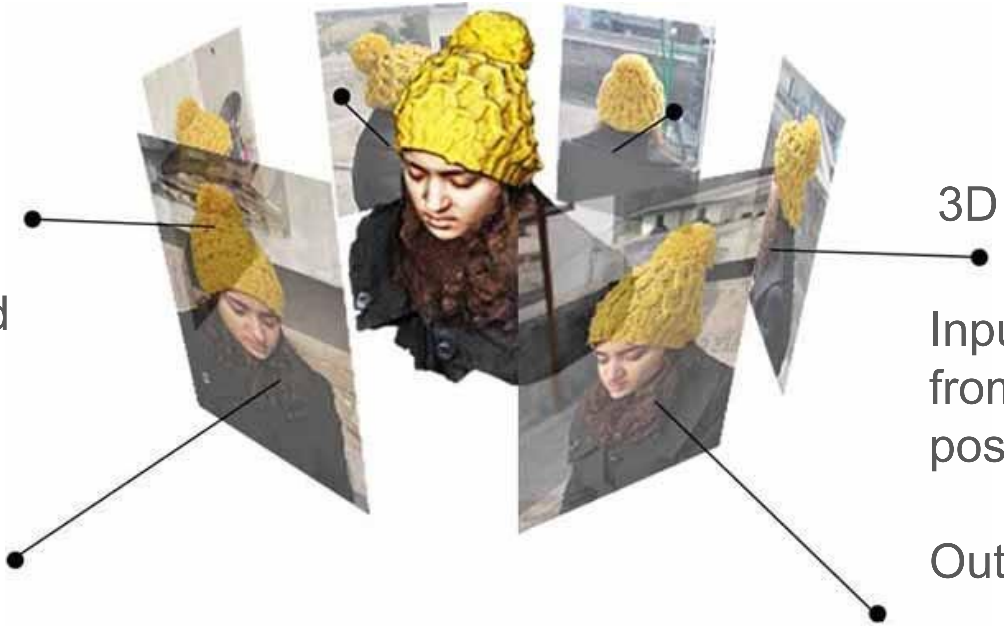


View Synthesis vs 3D Reconstruction

View Synthesis

Input: 3D Model and viewing direction

Output: Image



3D Reconstruction

Input: Multiple images from different positions

Output: 3D model

Image Source: Neitra 3D Pro

3D Representation (RECAP!)

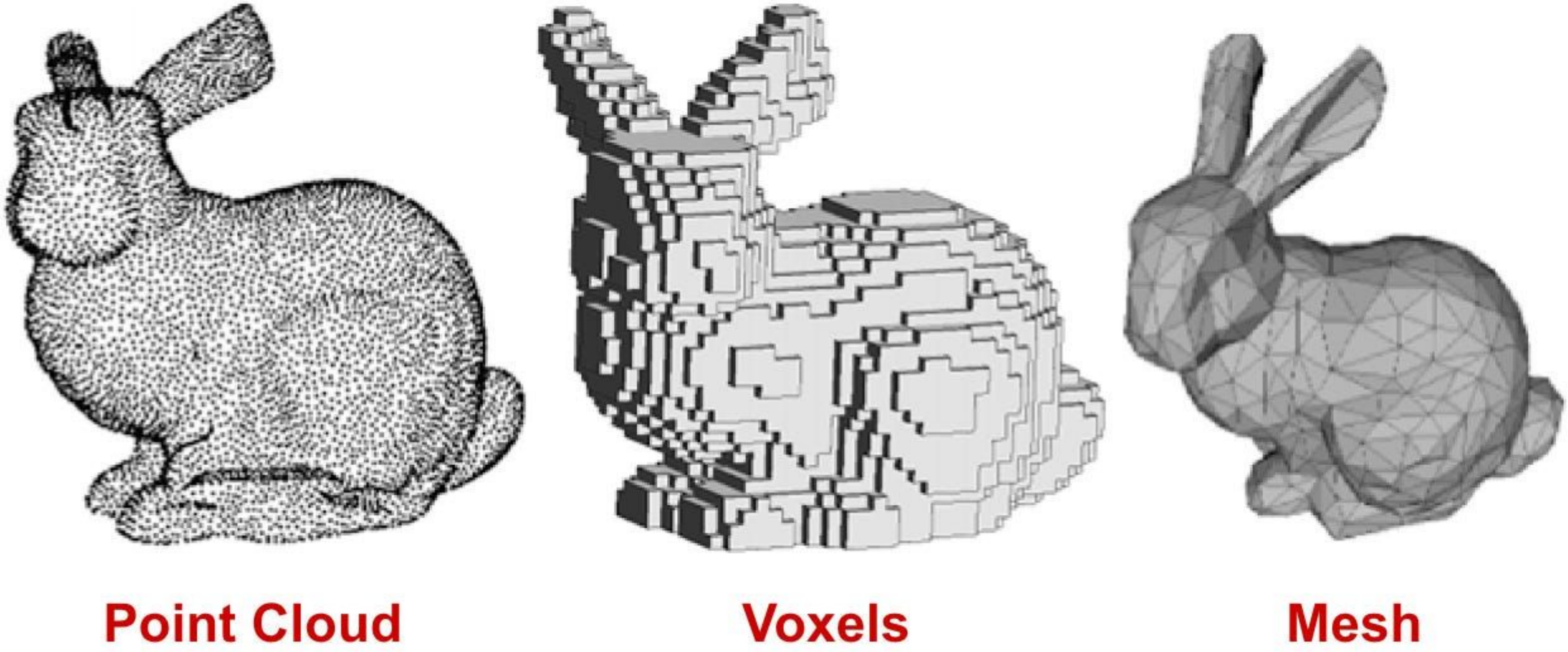


Image Source: Antonie Toisoul, 3D Data Representations

Voxel

Stores
- Occupancy
- Density
- Color
- Opacity

Pros
- Simplicity
- Uniform Resolution

Cons
- Memory
- Resolution
- Scalability

Volume Rendering - Ray Marching

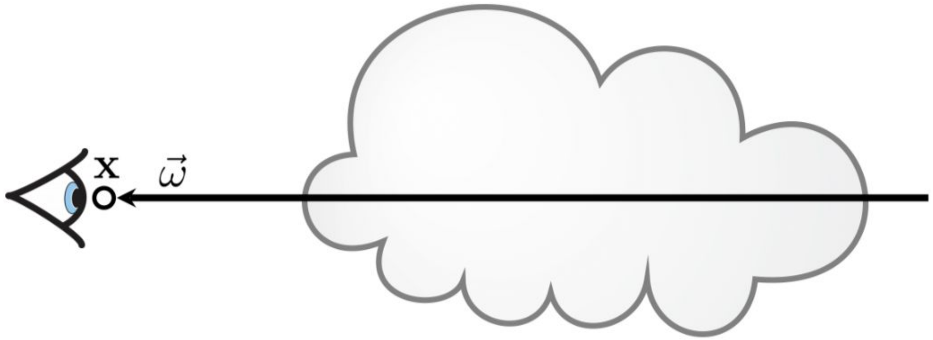
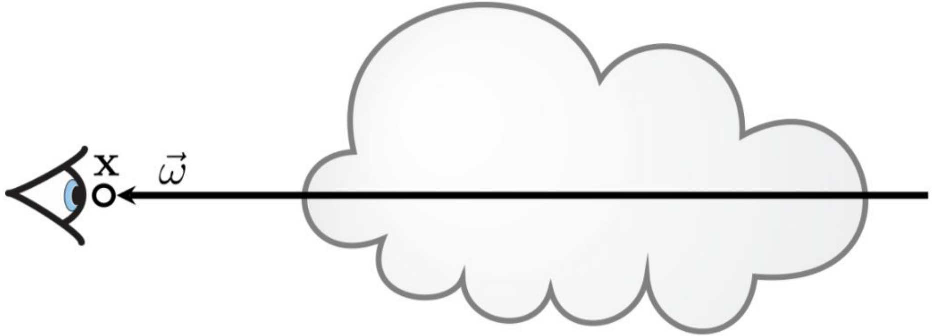


Image Credits: Prof. Shubham Tulsiani, CMU - Learning for 3D Vision
Kajiya, J.T., Herzen, B.P.V.: Ray tracing volume densities. Computer Graphics (SIGGRAPH) (1984)

Volume Rendering - Ray Marching

Radiance
(predicted color)

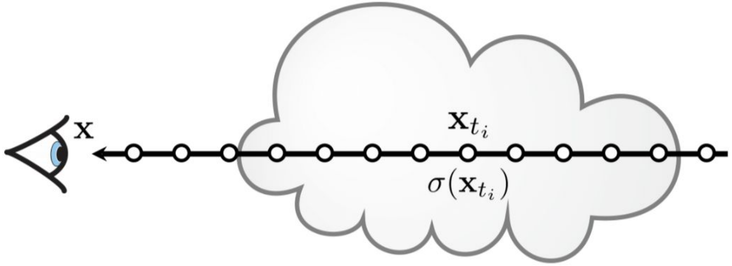


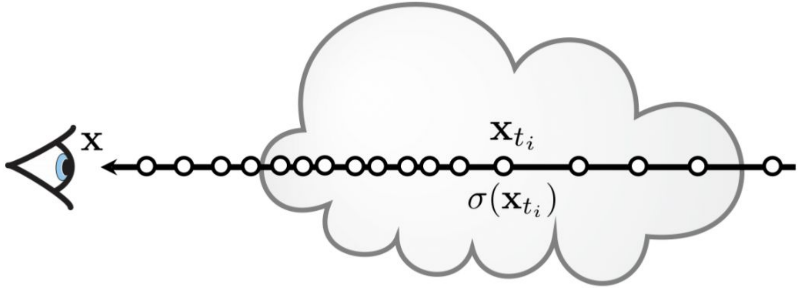
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

σ(x) = Volume Density, c = Color, T = Transmittance
t = Distance along the ray, d = Direction of ray

Image Credits: Prof. Shubham Tulsiani, CMU - Learning for 3D Vision
Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, & Ren Ng. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.

Computational Volume Rendering - Ray Marching





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

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

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

Image Credits: Prof. Shubham Tulsiani, CMU - Learning for 3D Vision
Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, & Ren Ng. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.

Computational Volume Rendering - Ray Marching

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

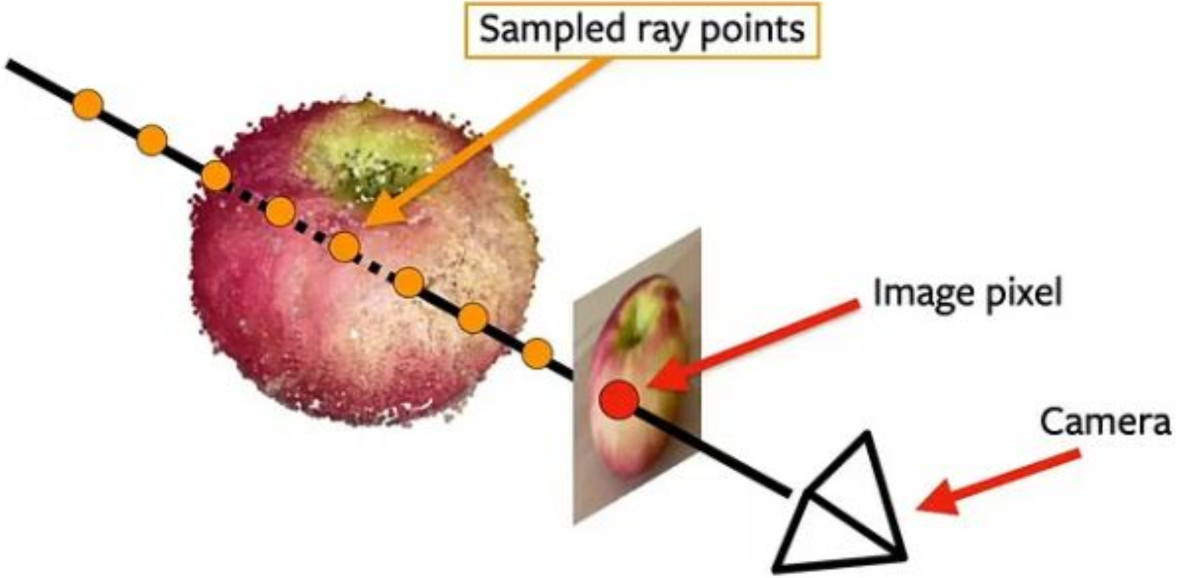
$$\delta_i = t_{i+1} - t_i$$

Algorithm

- Sample points (uniform/non-uniform) along the ray
- Compute C at each point/segment
- Sum up contributions across all segments

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, & Ren Ng. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.

Image Rendering



Algorithm
- Shoot a ray through every pixel
- Compute radiance (color)

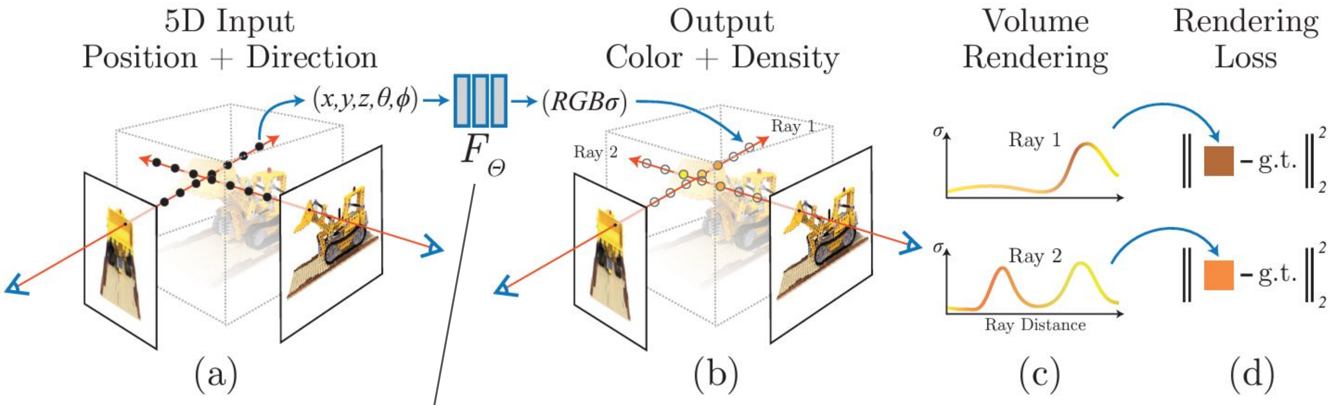
Image Source: David Novotny, pytorch3d tutorial

Drawbacks

- Memory usage
- Computational Cost
- Resolution Limitation

QUESTIONS?

Neural Radiance Fields (NeRFs)



F = Multi Layer Perceptron

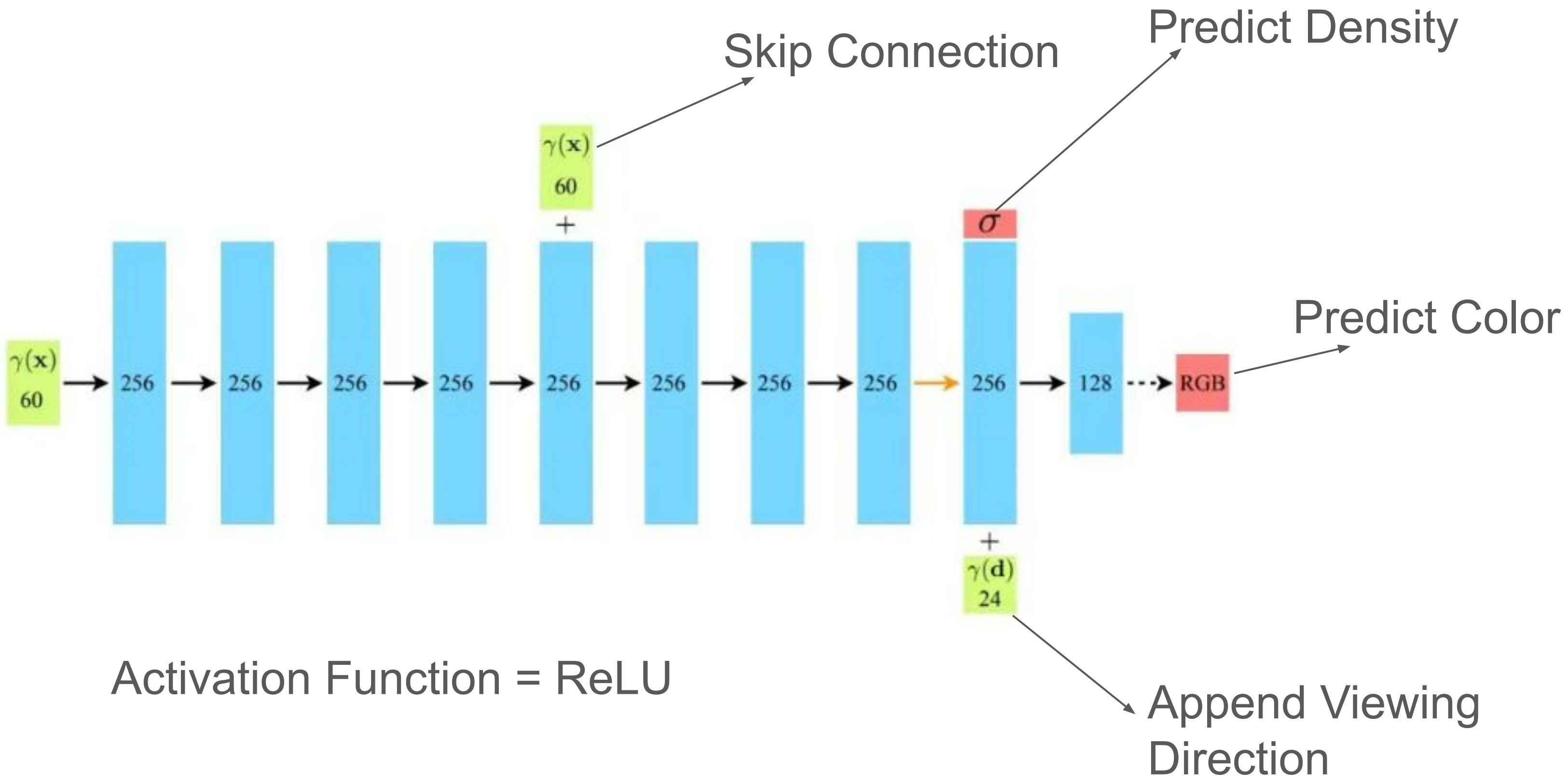
Maps 5D coordinate to volume density and directional emitted color

5 numbers in, 4 numbers out

$(x,y,z,\theta,\phi) \longrightarrow F\theta \longrightarrow (R,G,B,\sigma)$

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, & Ren Ng. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.

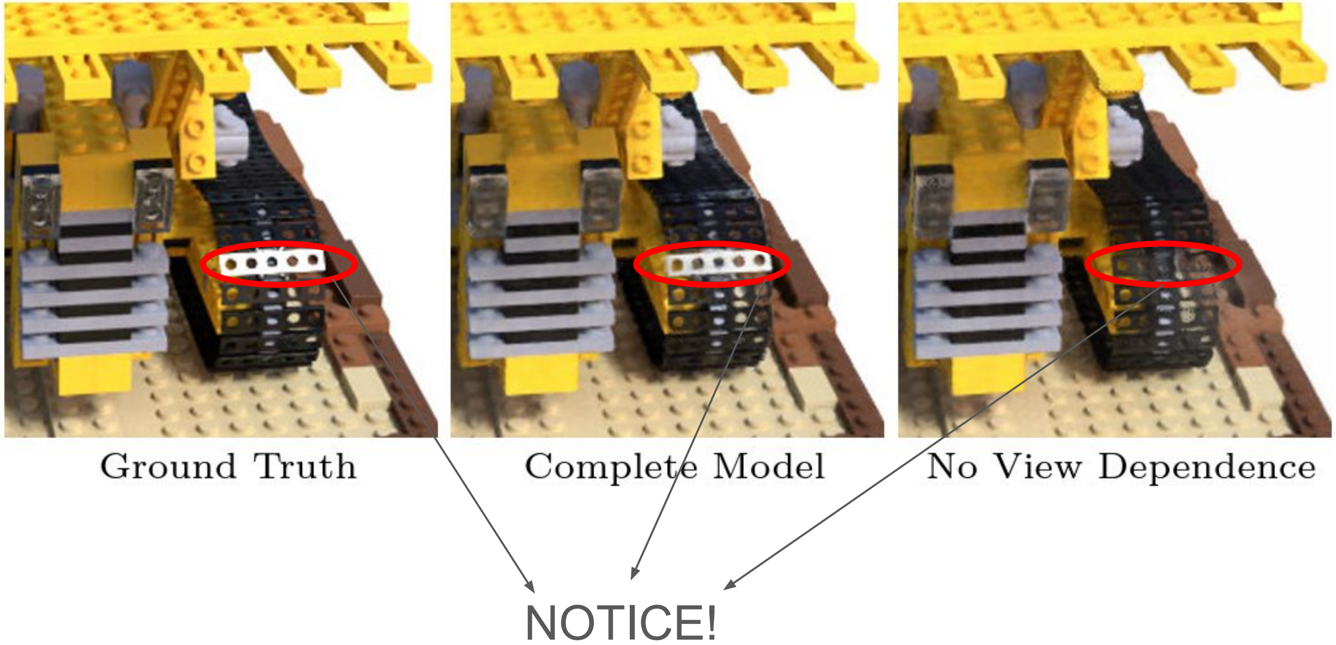
NeRF Network Architecture



Activation Function = ReLU

Image Source: Neural Radiance Fields (NeRFs): A Technical Exploration by Gaudenz Boesch

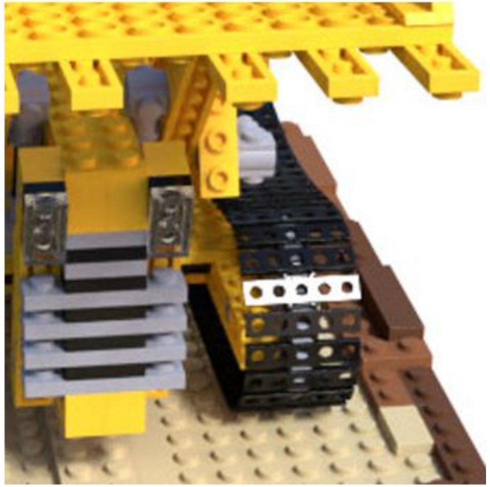
View Dependence



Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, & Ren Ng. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.

Optimization Techniques

1. Modelling High Frequency - Positional Encoding
2. Hierarchical Volume Sampling

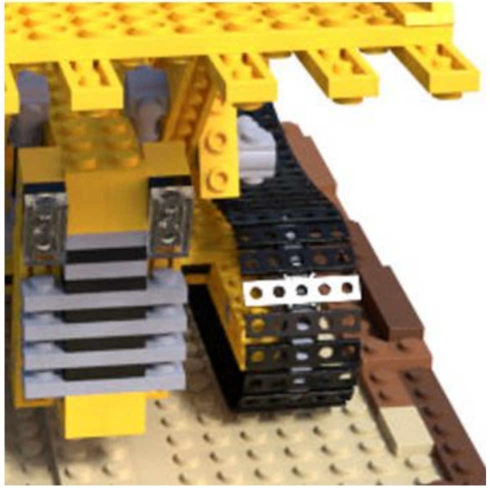


Ground Truth

No Positional Encoding

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, & Ren Ng. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.

Modeling High Frequency



Ground Truth

No Positional Encoding

Neural Networks are biased towards learning low frequency functions.

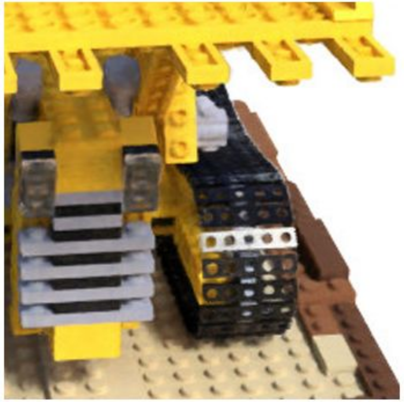
Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, & Ren Ng. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.

Rahaman, N., Baratin, A., Arpit, D., Dr¨ axler, F., Lin, M., Hamprecht, F.A., Bengio, Y., Courville, A.C.: On the spectral bias of neural networks. In: ICML (2018)

Positional Encoding

Ground Truth

No Positional Encoding



Complete Model

With positional encoding

Map input into higher dimensional input.

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

This function $\gamma(\cdot)$ is applied separately to each of the input variables.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, & Ren Ng. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.

Hierarchical Volume Sampling

Algorithm

- Perform uniform sampling (coarse sampling)
- Calculate PDF of each point
- Perform importance sampling using the PDF (fine sampling)

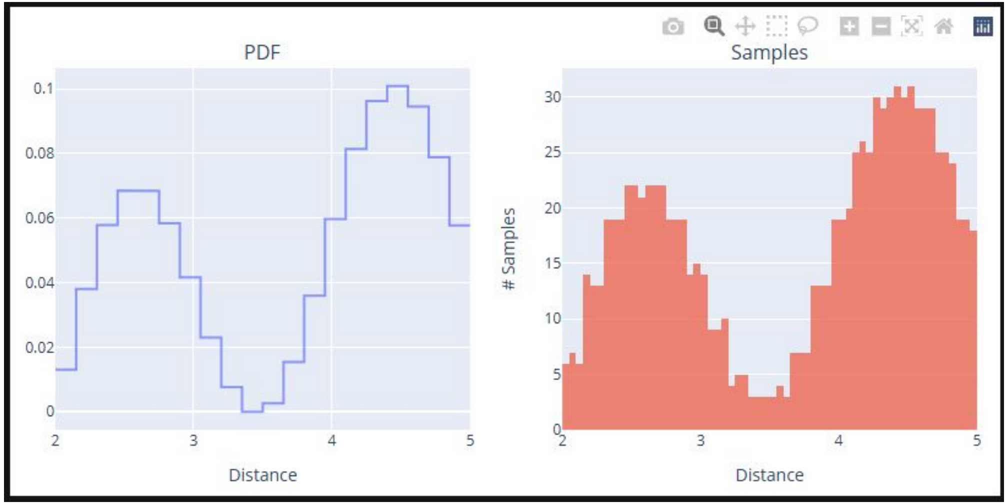
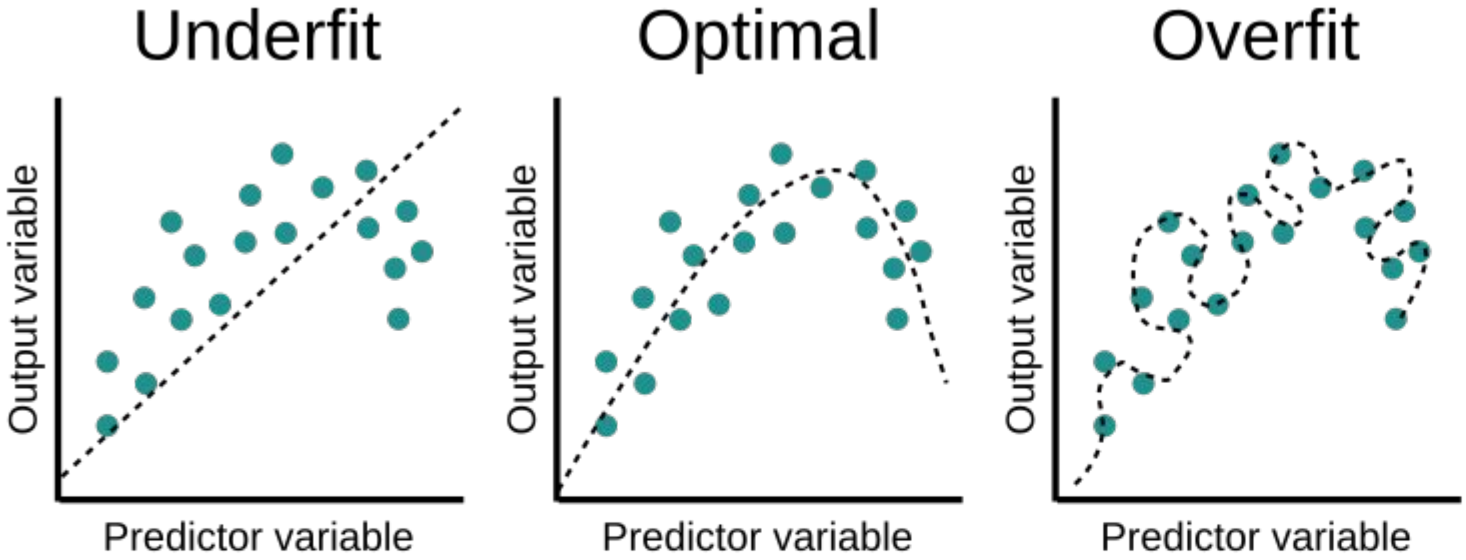


Image Source: NeRF Studio

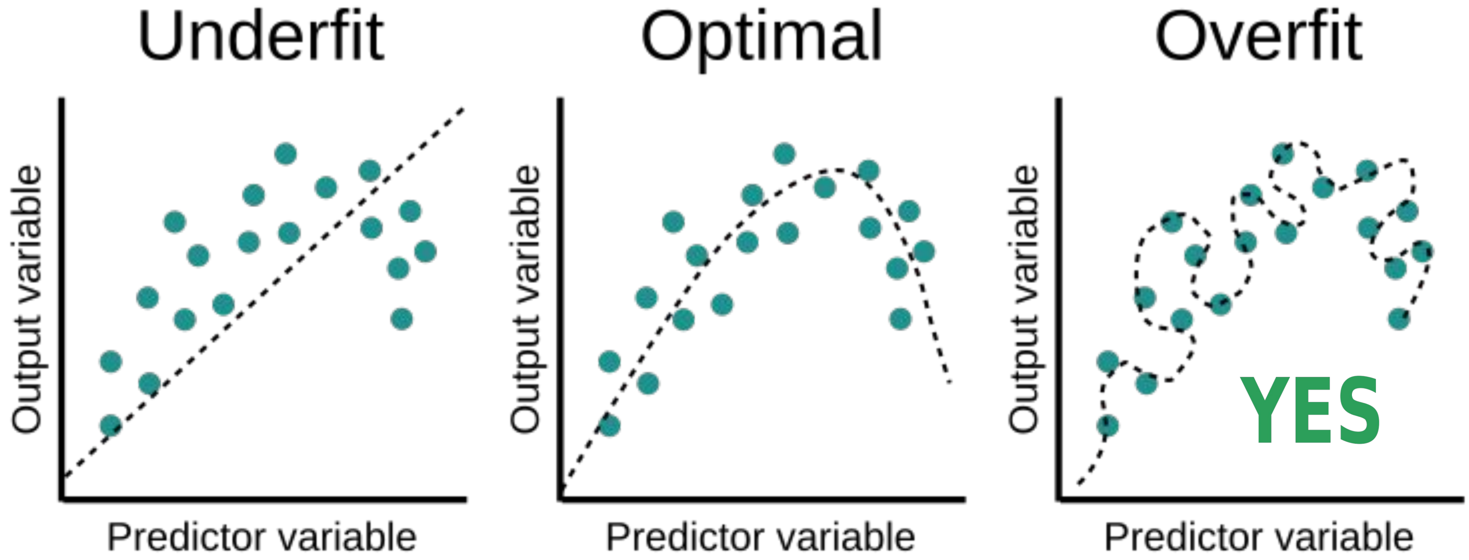
NeRF Implementation Details



What is the expected behaviour of a well trained MLP in general?

Image Source: https://thecorrelation.in/overfitting-and-underfitting/

NeRF Implementation Details



One Scene, One MLP

NO GENERALIZATION

Image Source: https://thecorrelation.in/overfitting-and-underfitting/

NeRF Implementation Details

Dataset

- RGB Images, corresponding camera pose and intrinsics along with scene bounds.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, & Ren Ng. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.

NeRF Implementation Details

Dataset

- RGB Images, corresponding camera pose and intrinsics along with scene bounds.

Batch

- Sample camera rays (batch size = 4096) from set of all pixels

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, & Ren Ng. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.

NeRF Implementation Details

Dataset
- RGB Images, corresponding camera pose and intrinsics along with scene bounds.

Batch
- Sample camera rays (batch size = 4096) from set of all pixels

Algorithm
- Sample Nc (=64) coarse points.
- Perform ray marching. Use MLP to compute color and density at each point.
- Compute PDF and sample Nf (=128) fine points. Perform ray marching again.
- Compute loss and optimize (Optimizer = Adam).

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

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

C_C = Radiance Calculated using coarse points
C_f = Radiance Calculated using fine points

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, & Ren Ng. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.

NeRF++

Need: Unbounded Scene

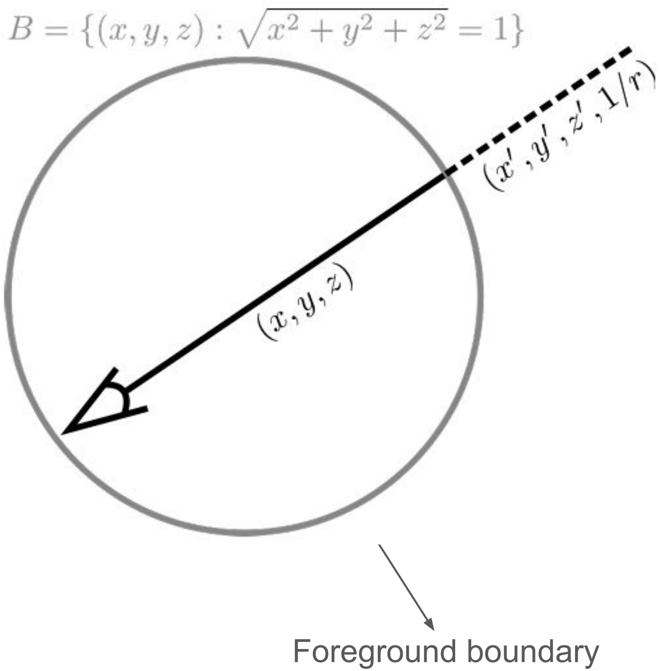
Method: Train two NeRF models, one for foreground, one for background

(a) NeRF++ prediction (b) predicted foreground (c) predicted background

F_θ G_θ

Kai Zhang, Gernot Riegler, Noah Snavely, & Vladlen Koltun (2020). NeRF++: Analyzing and Improving Neural Radiance Fields. arXiv:2010.07492.

NeRF++ Sampling



$(x,y,z,\theta,\phi) \longrightarrow F_\theta \longrightarrow (R,G,B,\sigma)$

$(x,y,z,\theta,\phi,1/r) \longrightarrow G_\theta \longrightarrow (R,G,B,\sigma)$

Background NeRF uses normalised coordinates.

Kai Zhang, Gernot Riegler, Noah Snavely, & Vladlen Koltun (2020). NeRF++: Analyzing and Improving Neural Radiance Fields. arXiv:2010.07492.

Try NeRF

- Instant NGP - NVLabs
  https://github.com/NVlabs/instant-ngp
- NeRF Studio
  https://docs.nerf.studio/

Let's go back to some NeRF limitations

- We need more powerful GPUs to complete training
- They take a lot of time to complete training
- They are scene and view dependent

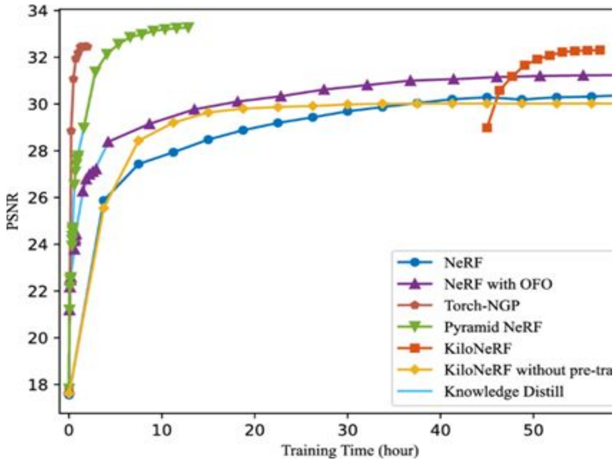


Image source - https://link.springer.com/article/10.1007/s11263-023-01829-3

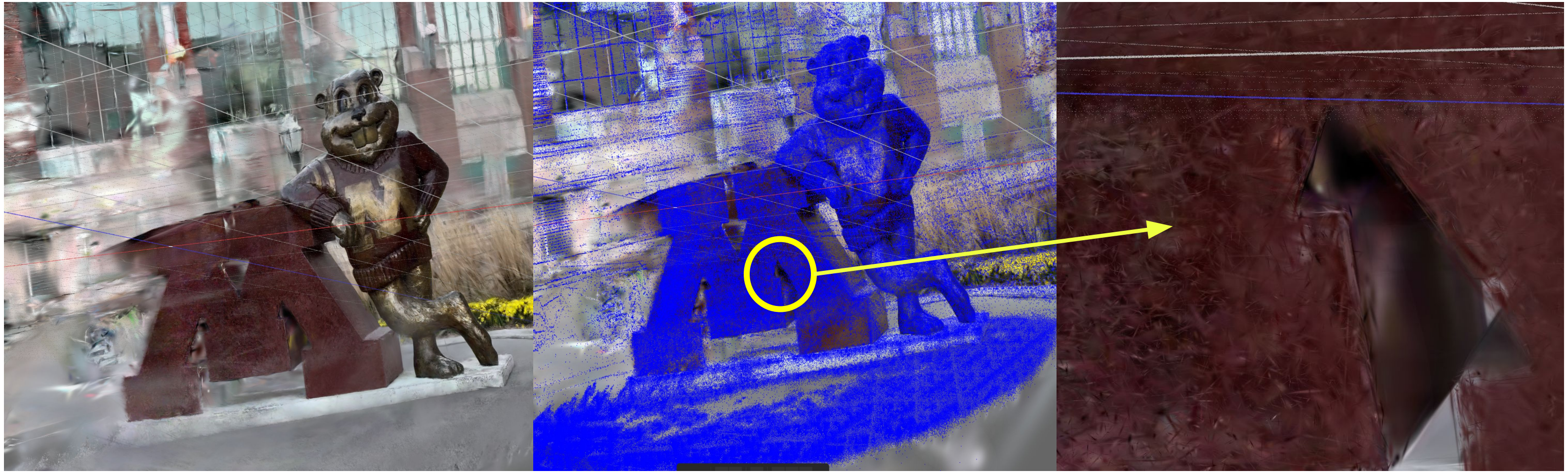
Image Source - https://www.nicehash.com/blog/post/nvidia-rtx-4090-specs-and-mining-hashrate

How would you address the limitations ?

What if we remove the neural network from NeRFs ?

- How do we learn the 3D information ?
- Data structure for representation
- How to project from 3D to 2D in an efficient way ?
- What makes training (optimization) faster ?
- How do we address view dependent effects and colors?

3D Gaussian Splatting



Reconstructed View Individual Splats A close up

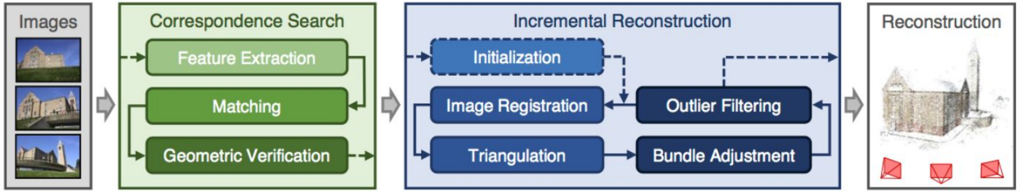
Viewer -https://github.com/playcanvas/supersplat

3D Gaussian Splatting

- How do we learn the 3D information ?
- What is the data structure for representation?
- How to project from 3D to 2D in an efficient way ?
- What makes training (optimization) faster ?
- How do we address view dependent effects and colors?

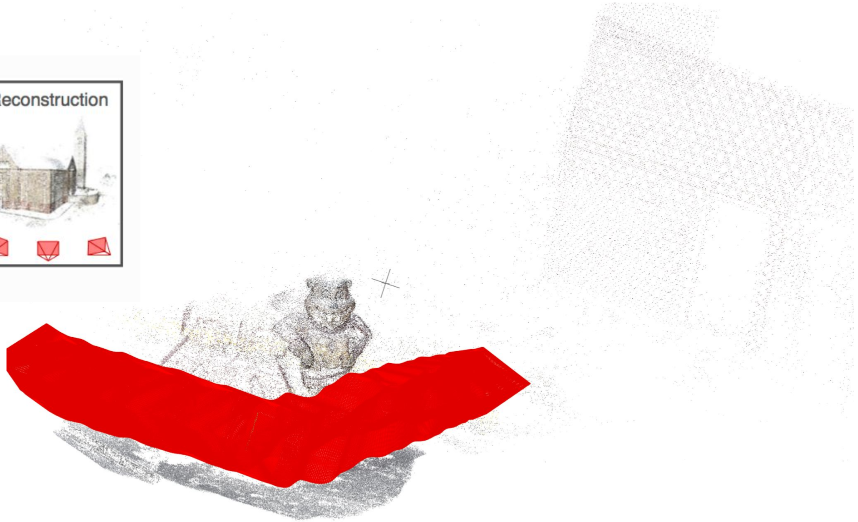
Point Clouds using SFM

3D initialization using SFM

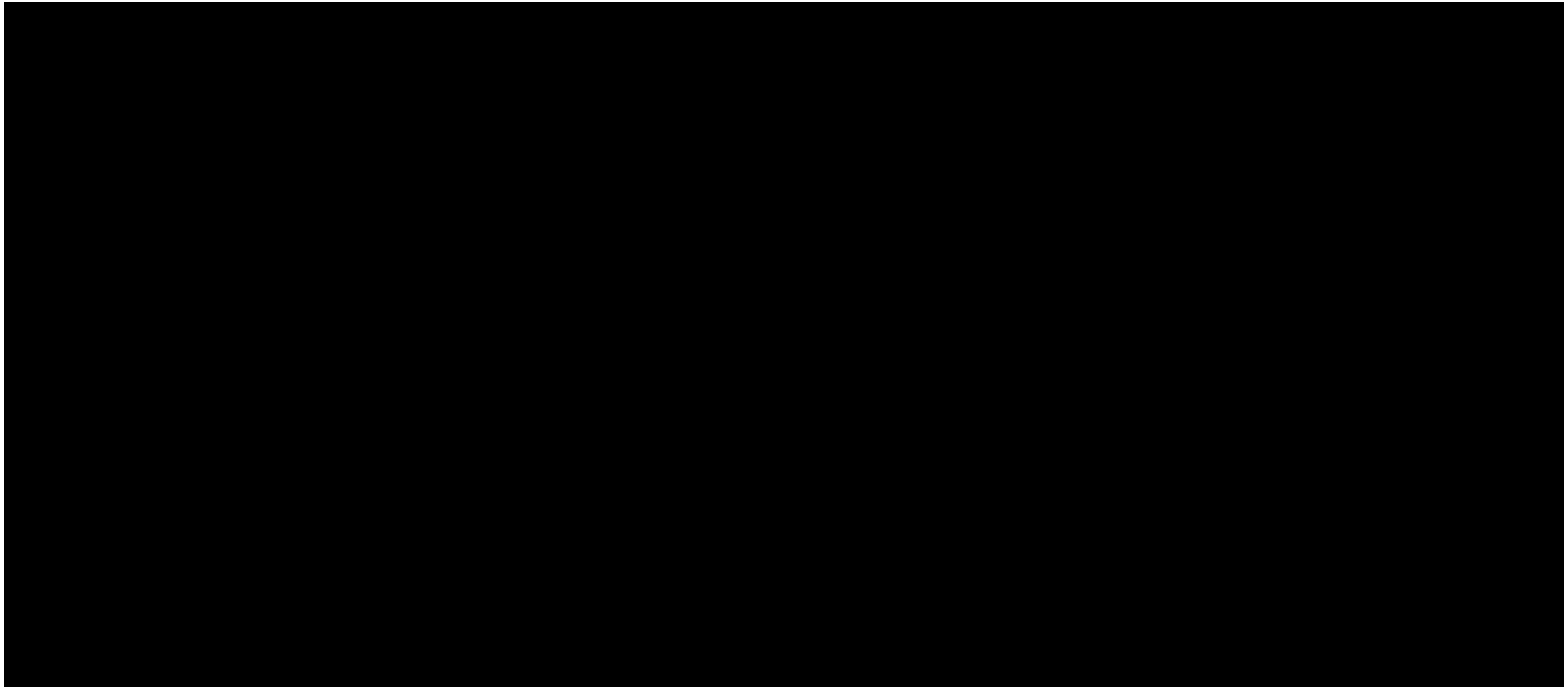


COLMAP's incremental Structure-from-Motion pipeline.

Source : https://colmap.github.io/tutorial.html#structure-from-motion



Structure from Motion



3D Gaussian Splatting

- How do we learn the 3D information ?
- What is the data structure for representation?
- How to project from 3D to 2D in an efficient way ?
- What makes training (optimization) faster ?
- How do we address view dependent effects and colors?

Point Clouds using SFM

Anisotropic Gaussians

Understanding 3D Gaussians

- Anisotropic(Elliptical) gaussians are closed for affine transformations.
- A single elliptical gaussian splat can cover a large distance, thus influencing the set of pixels it covers

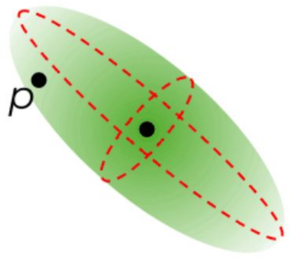


Image Source - https://towardsdatascience.com/a-comprehensive-overview-of-gaussian-splatting-e7d570081362

Parameters:

- Mean μ given by (x,y,z)
- Covariance Matrix Σ
- Opacity α
- Color from spherical harmonics (and rgb interchangeably)

3D -> 2D Gaussian projection.

- From the 2001 EWA Splatting paper, we can project to the image space as follows:

  $$\Sigma' = J W \Sigma W^T J^T$$

  W - viewing transformation
- EWA paper also shows that upon removing the third row and column of Σ' , we should get a 2x2 covariance matrix with the same properties and structure

- Since covariances only have physical meaning when they are positive and semi definitie :

  $$\Sigma = R S S^T R^T$$

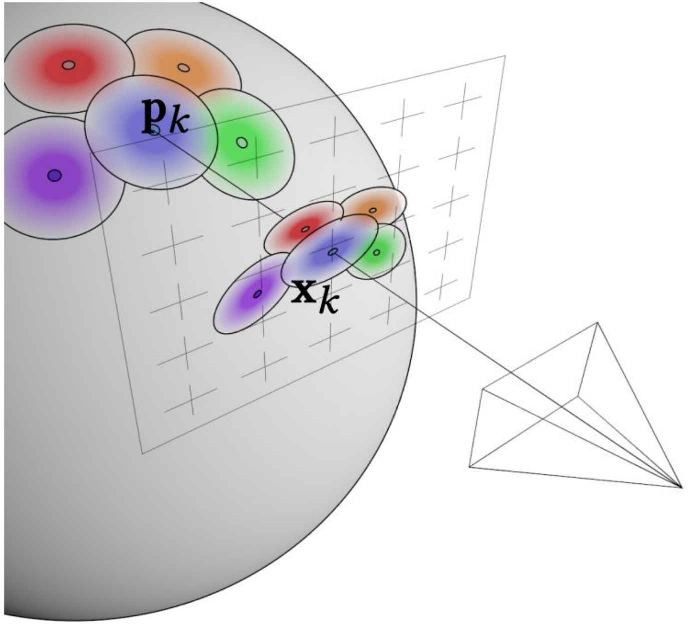


Image Source - https://www.researchgate.net/figure/Illustration-of-forward-splatting-using-EWA-Zwicker-et-al-2001_fig1_333717475

A more detailed overview of the Math in the previous slide

Alpha Compositing

The RGB value of each pixel is computed by α blending the gaussians from front-to-back. The rgb values of each pixel can be computed with:

$$C(u, v) = \sum_{i=1}^{N} \alpha_i c_i w_i$$

$$w_i = (1 - \sum_{j=0}^{i-1} \alpha_j w_j)$$

$$w_0 = 1$$

$$\alpha_i = o_i g_i(u, v)$$

Where c_i and o_i are the color and opacity of the i^{th} gaussian and $g_i(u, v)$ is the probabilty of the the i^{th} gaussian at the pixel coordinates u, v.

Source - https://github.com/joeyan/gaussian_splatting/blob/main/MATH.md

Why 3D anisotropic gaussians ? (Part 1)

Differentiable volumetric representation

Unstructured and explicit to allow fast rendering

Why 3D anisotropic gaussians ? (Part 2)

Image Source - 3D Gaussian Splatting for Real-Time Radiance Field Rendering

3D Gaussian Splatting

- How do we learn the 3D information ?
- Data structure for representation
- How to project from 3D to 2D in an efficient way ?
- What makes training (optimization) faster ?
- How do we address view dependent effects and colors?

Point Clouds using SFM

Anisotropic Gaussians

Tile Rasterizer

3D Gaussian Splatting

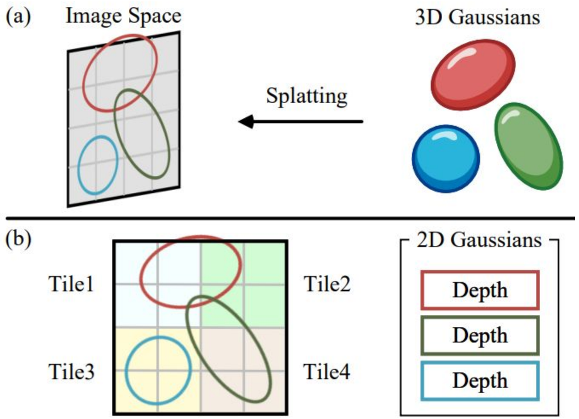
- How do we learn the 3D information ?
- What is the data structure for representation?
- How to project from 3D to 2D in an efficient way ?
- What makes training (optimization) faster ?
- How do we address view dependent effects and colors?

Point Clouds using SFM

Anisotropic Gaussians

Tile Rasterizer

Differentiable Tile Rasterizer



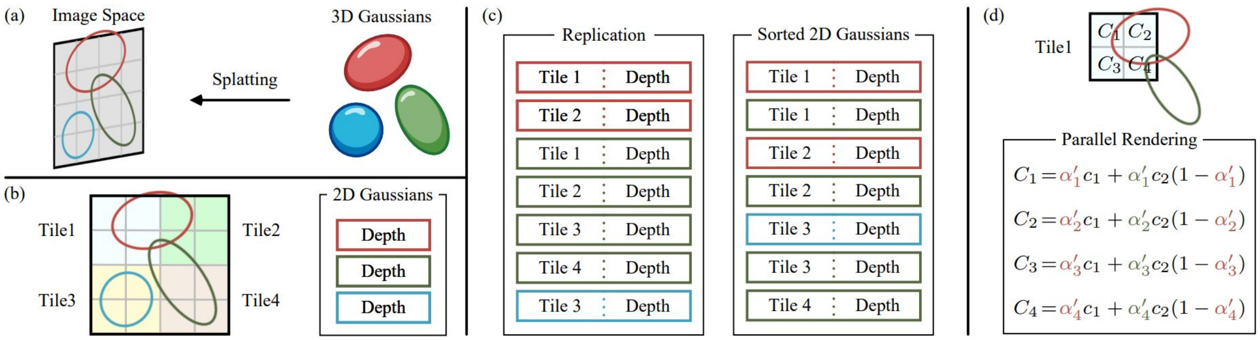
Chen, Guikun, and Wenguan Wang. "A Survey on 3D Gaussian Splatting." arXiv preprint arXiv:2401.03890 (2024).

View Frustum Culling

- Gaussians are culled against the view frustum and each independent tile, only retaining 99% confidence rendering task
- Guard bands surrounding each tile is used to reject gaussians at extreme positions.

Image Source - claude.ai

Differentiable Tile Rasterizer



$$\alpha'_n = \alpha_n \times \exp\left(-\frac{1}{2}(\boldsymbol{x}' - \boldsymbol{\mu}'_n)^\top \boldsymbol{\Sigma}'^{-1}_n (\boldsymbol{x}' - \boldsymbol{\mu}'_n)\right),$$

$$C = \sum_{n=1}^{|\mathcal{N}|} c_n \alpha'_n \prod_{j=1}^{n-1} (1 - \alpha'_j),$$

X represents the position of a pixel on the screen , α□ represents the learned opacity and C represents the color at each pixel and c□ represents the learnt color

Chen, Guikun, and Wenguan Wang. “A Survey on 3D Gaussian Splatting.” arXiv preprint arXiv:2401.03890 (2024).

Lets Splat ?

Image Source - https://medium.com/@soyoungpark.psy/lets-crack-3d-gaussian-splatting-in-5mins-5777a7b735eb

Adaptive Density Control

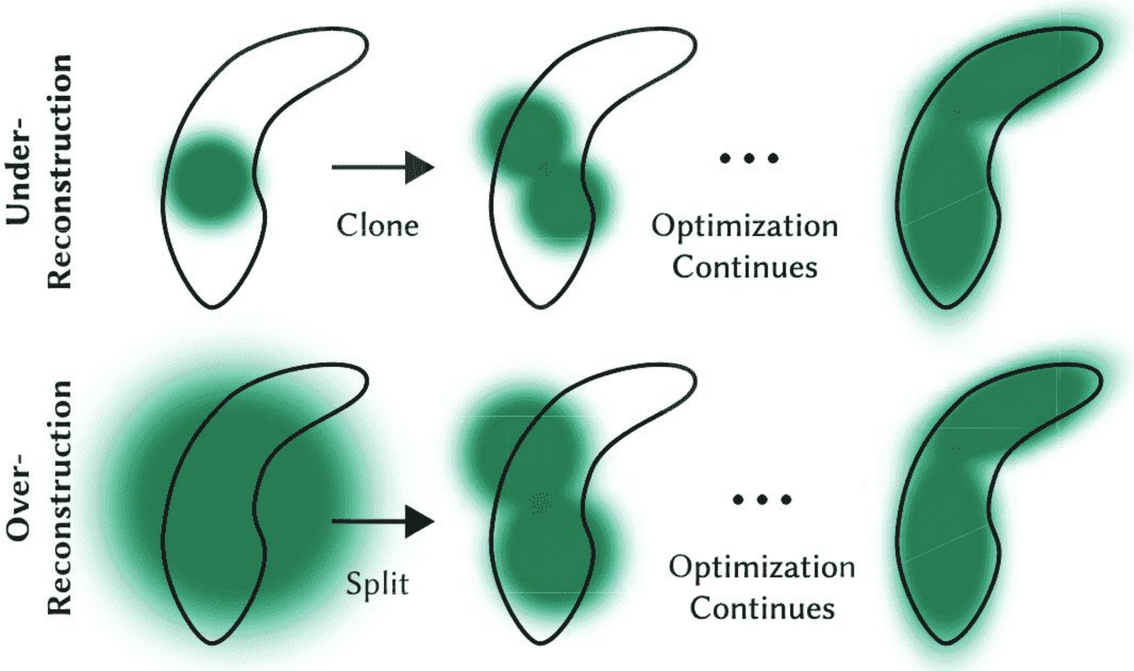


Image Source - 3D Gaussian Splatting for Real-Time Radiance Field Rendering

Overall Optimization Algorithm

Algorithm 1 Optimization and Densification
w, h: width and height of the training images

$M \leftarrow$ SfM Points ▷ Positions
$S, C, A \leftarrow$ InitAttributes() ▷ Covariances, Colors, Opacities
$i \leftarrow 0$ ▷ Iteration Count
while not converged **do**
  $V, \hat{I} \leftarrow$ SampleTrainingView() ▷ Camera V and Image
  $I \leftarrow$ Rasterize(M, S, C, A, V) ▷ Alg. 2
  $L \leftarrow Loss(I, \hat{I})$ ▷ Loss
  $M, S, C, A \leftarrow$ Adam(∇L) ▷ Backprop & Step
  if IsRefinementIteration(i) **then**
    for all Gaussians (μ, Σ, c, α) **in** (M, S, C, A) **do**
      if $\alpha < \epsilon$ or IsTooLarge(μ, Σ) **then** ▷ Pruning
        RemoveGaussian()
      end if
      if $\nabla_p L > \tau_p$ **then** ▷ Densification
        if $\|S\| > \tau_S$ **then** ▷ Over-reconstruction
          SplitGaussian(μ, Σ, c, α)
        else ▷ Under-reconstruction
          CloneGaussian(μ, Σ, c, α)
        end if
      end if
    end for
  end if
  $i \leftarrow i + 1$
end while

Algorithm Source - 3D Gaussian Splatting for Real-Time Radiance Field Rendering

GPU Rasterizer

Algorithm 2 GPU software rasterization of 3D Gaussians
w, h: width and height of the image to rasterize
M, S: Gaussian means and covariances in world space
C, A: Gaussian colors and opacities
V: view configuration of current camera

```
function RASTERIZE(w, h, M, S, C, A, V)
    CullGaussian(p, V)                              ▷ Frustum Culling
    M′, S′ ← ScreenspaceGaussians(M, S, V)          ▷ Transform
    T ← CreateTiles(w, h)
    L, K ← DuplicateWithKeys(M′, T)                 ▷ Indices and Keys
    SortByKeys(K, L)                                ▷ Globally Sort
    R ← IdentifyTileRanges(T, K)
    I ← 0                                           ▷ Init Canvas
    for all Tiles t in I do
        for all Pixels i in t do
            r ← GetTileRange(R, t)
            I[i] ← BlendInOrder(i, L, r, K, M′, S′, C, A)
        end for
    end for
    return I
end function
```

Algorithm Source - 3D Gaussian Splatting for Real-Time Radiance Field Rendering

3D Gaussian Splatting

- How do we learn the 3D information ? Point Clouds using SFM
- What is the data structure for representation? Anisotropic Gaussians
- How to project from 3D to 2D in an efficient way ? Tile Rasterizer
- What makes training (optimization) faster ?
- How do we address view dependent effects and colors? Spherical Harmonics

View Dependent Color

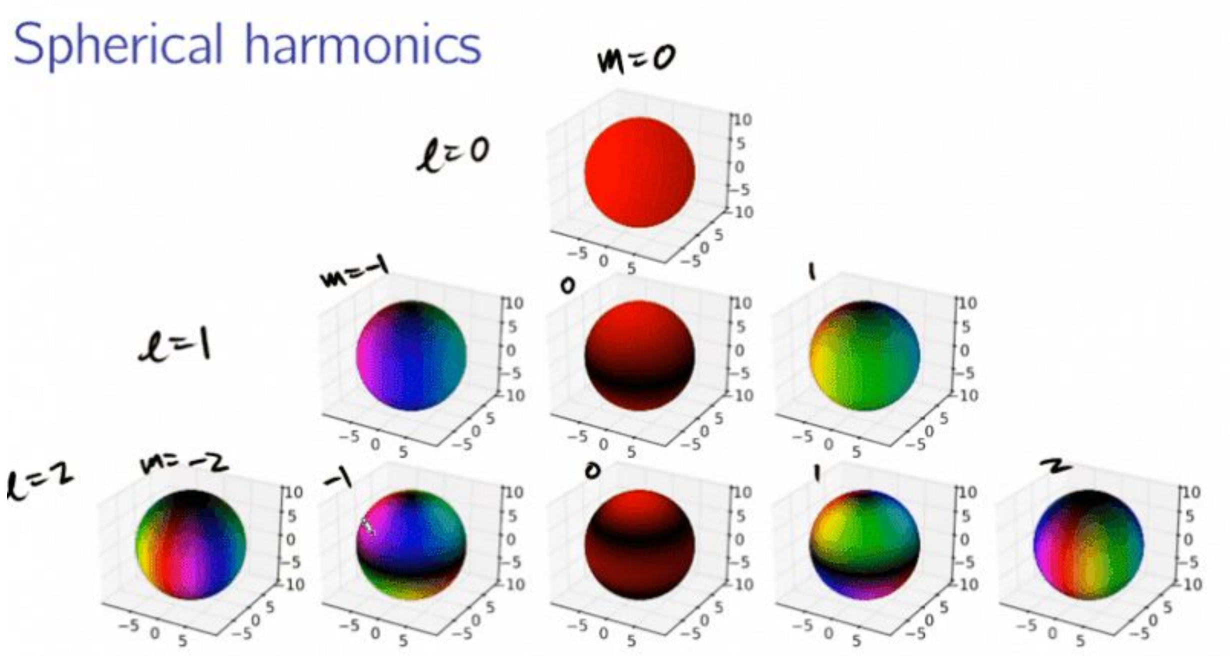


Image Source :https://www.physicsforums.com/threads/what-do-the-color-maps-in-spherical-harmonics-represent.805216/

How is SH calculated by the 3Dgs code ?

22 SPHERICAL HARMONIC LIGHTING

Equation 10. Cartesian version of the first few real SH functions.

	$m=-2$	$m=-1$	$m=0$	$m=1$	$m=2$
$l=0$			$\frac{1}{2}\sqrt{\frac{1}{\pi}}$		
$l=1$		$\frac{1}{2}\sqrt{\frac{3}{\pi}}\frac{y}{r}$	$\frac{1}{2}\sqrt{\frac{3}{\pi}}\frac{z}{r}$	$\frac{1}{2}\sqrt{\frac{3}{\pi}}\frac{x}{r}$	
$l=2$	$\frac{1}{2}\sqrt{\frac{15}{\pi}}\frac{yx}{r^2}$	$\frac{1}{2}\sqrt{\frac{15}{\pi}}\frac{yz}{r^2}$	$\frac{1}{4}\sqrt{\frac{5}{\pi}}\frac{2z^2-x^2-y^2}{r^2}$	$\frac{1}{2}\sqrt{\frac{15}{\pi}}\frac{zx}{r^2}$	$\frac{1}{2}\sqrt{\frac{15}{\pi}}\frac{x^2-y^2}{r^2}$

where

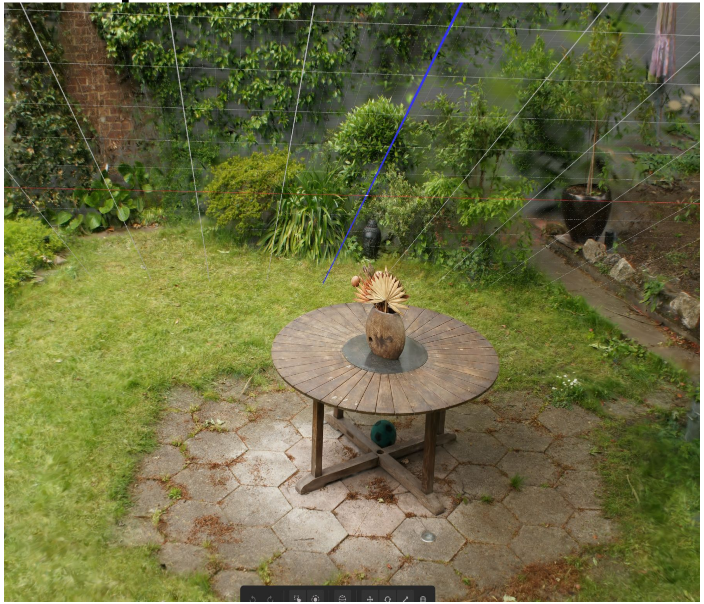
$$r=\sqrt{x^2+y^2+z^2} \quad (\text{n.b. usually } r=1)$$

Source :Spherical Harmonic Lighting: The Gritty Details

The source code for computing color from SH can be found here and the general data structure for SH seems to be arrays after a brief overview of the code.

Spherical Harmonics Comparisons

0th Degree(Band)



1st Degree(Band)

Images generated using 0 band SH should have uniform light representation in all direction , as the number of SH bands increase , the ability to handle different lighting from different directions increases. The image on the right should be a bit sharper and more representative of the actual image.

Images Generated using supersplat

3D Gaussian Splatting

- How do we learn the 3D information ? Point Clouds using SFM
- Data structure for representation
- How to project from 3D to 2D in an efficient way ? Anisotropic Gaussians
- What makes training (optimization) faster ? Tile Rasterizer
- How do we address view dependent effects and colors? Spherical Harmonics

3D Gaussian Splatting

• How do we learn the 3D information ?	Point Clouds using SFM
• What is the data structure for representation?	
• How to project from 3D to 2D in an efficient way ?	Anisotropic Gaussians
• What makes training (optimization) faster ?	Tile Rasterizer
• How do we address view dependent effects and colors?	Spherical Harmonics

Overall Pipeline

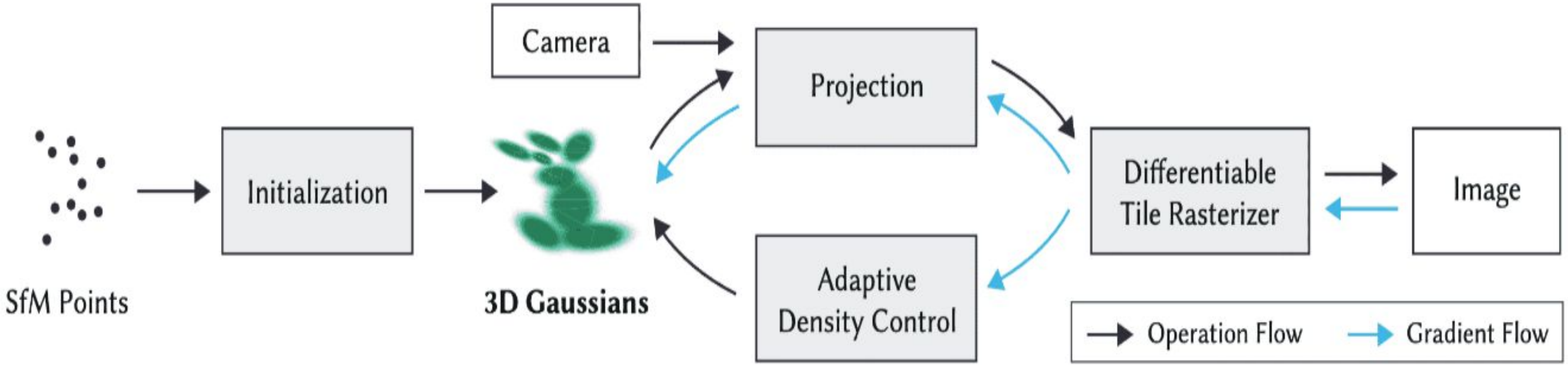


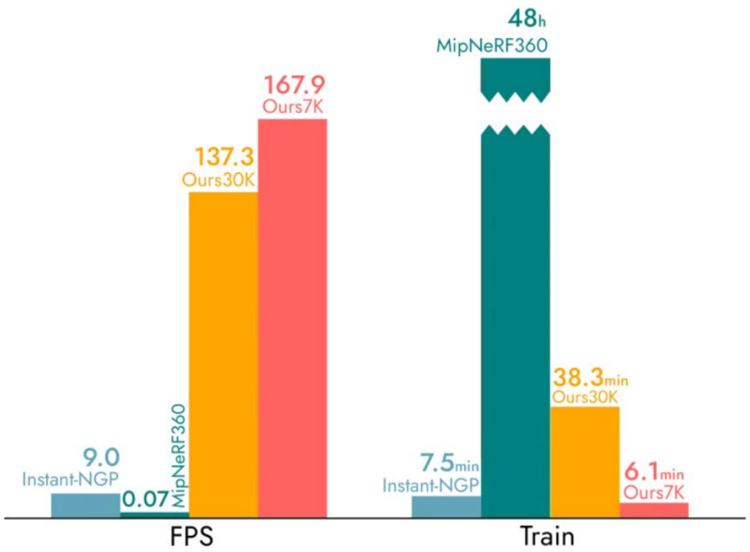
Image Source - 3D Gaussian Splatting for Real-Time Radiance Field Rendering

Optimization Methodologies

- Makes use of stochastic gradient descent for optimization
- Sigmoid activation function to constrain α between [0,1) and obtain smooth gradients
- Loss is defined by - $\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}$

How does 3DGS compare against NeRFs?

- 3DGS is significantly faster than NeRFs for both training and rendering purposes, although the memory it uses might be a relatively high as show in the table.



Limitations of Gaussian Splatting

- Sparse Point Cloud Dependency
- Static nature
- Floaters and other inherited artifacts

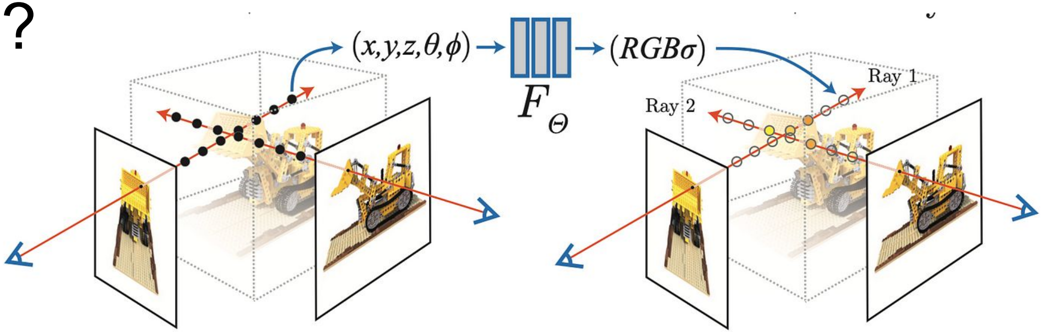
Progress and Developments in Gaussian Splatting Since the Seminal 2022 Paper

- Using sparse images
- Memory efficiency
- Photorealism(Raytracing)
- 3DGS with structured information
- Dynamic Scenes
- Moving away from colmap

and many more ….

Where do I introduce NeRF and Gaussian Splatting tin Robotics?

How is NeRF and Gaussian Splatting used for manipulation tasks?



Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2021). Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, *65*(1), 99-106.

Simulations & Scene Understanding

- NeRF’s and Gaussian Splats are able to get a better grasp of the scene’s 3D information.
- Recent advancements in gaussian splatting have enabled semantic segmentation of individual objects in the 3D scene.
- **The ability to retain structure hidden within the input information distinguishes both NeRF and Gaussian Splatting from traditional vision input processing techniques.**
- A key real world problem that is being tackled right now with both NeRF and Gaussian Splatting is **overcoming the Sim2Real gap.**

NeRF in Robotics:

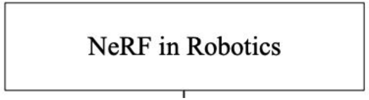


Fig. 1. A taxonomy of NeRF in robotics.

Wang, G., Pan, L., Peng, S., Liu, S., Xu, C., Miao, Y., ... & Wang, H. (2024). NeRF in Robotics: A Survey. *arXiv preprint arXiv:2405.01333*.

NeRF in Robotics:

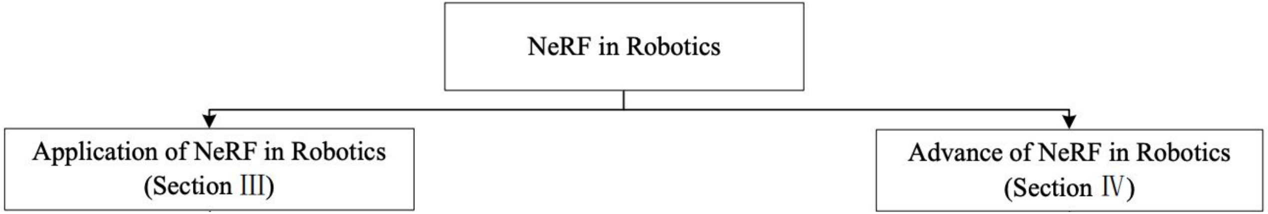


Fig. 1. A taxonomy of NeRF in robotics.

Wang, G., Pan, L., Peng, S., Liu, S., Xu, C., Miao, Y., ... & Wang, H. (2024). NeRF in Robotics: A Survey. *arXiv preprint arXiv:2405.01333*.

NeRF in Robotics:

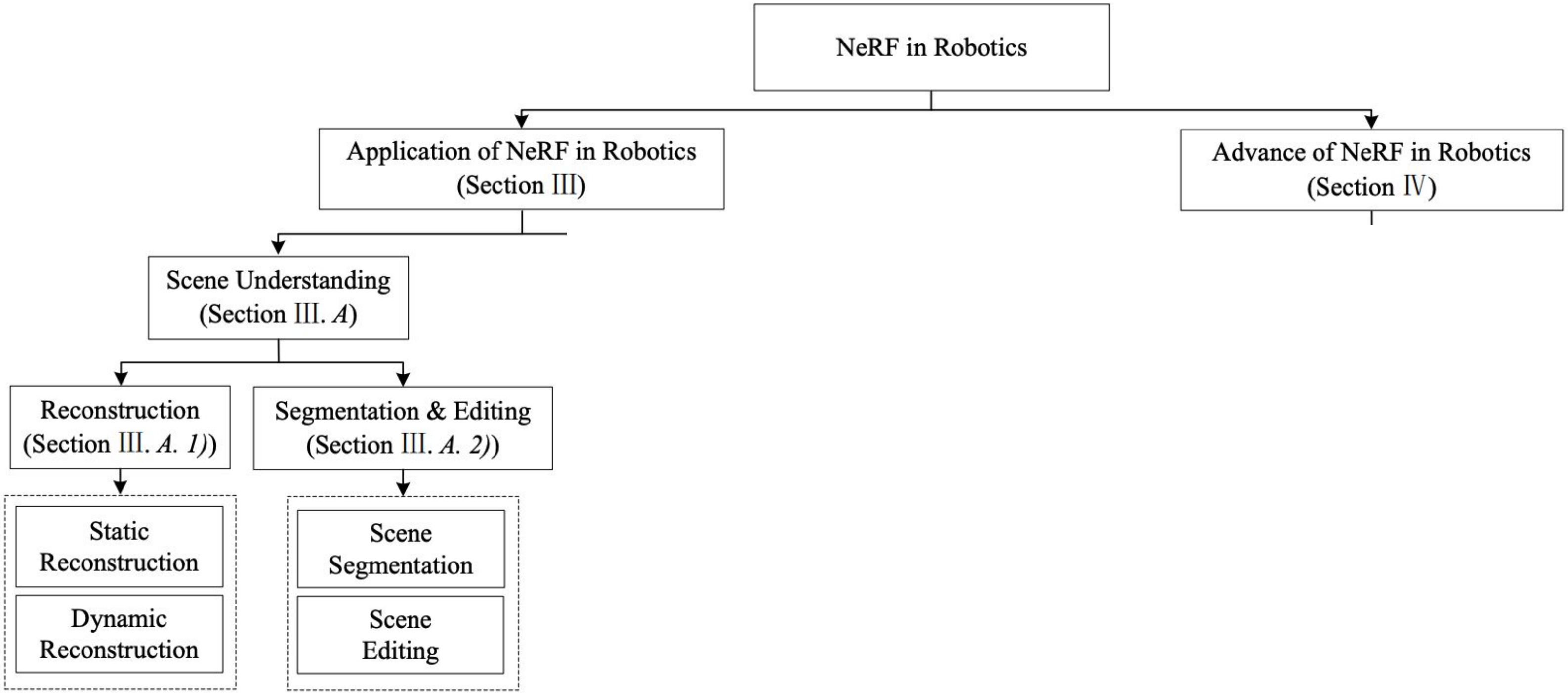


Fig. 1. A taxonomy of NeRF in robotics.

Wang, G., Pan, L., Peng, S., Liu, S., Xu, C., Miao, Y., ... & Wang, H. (2024). NeRF in Robotics: A Survey. *arXiv preprint arXiv:2405.01333*.

NeRF in Robotics:

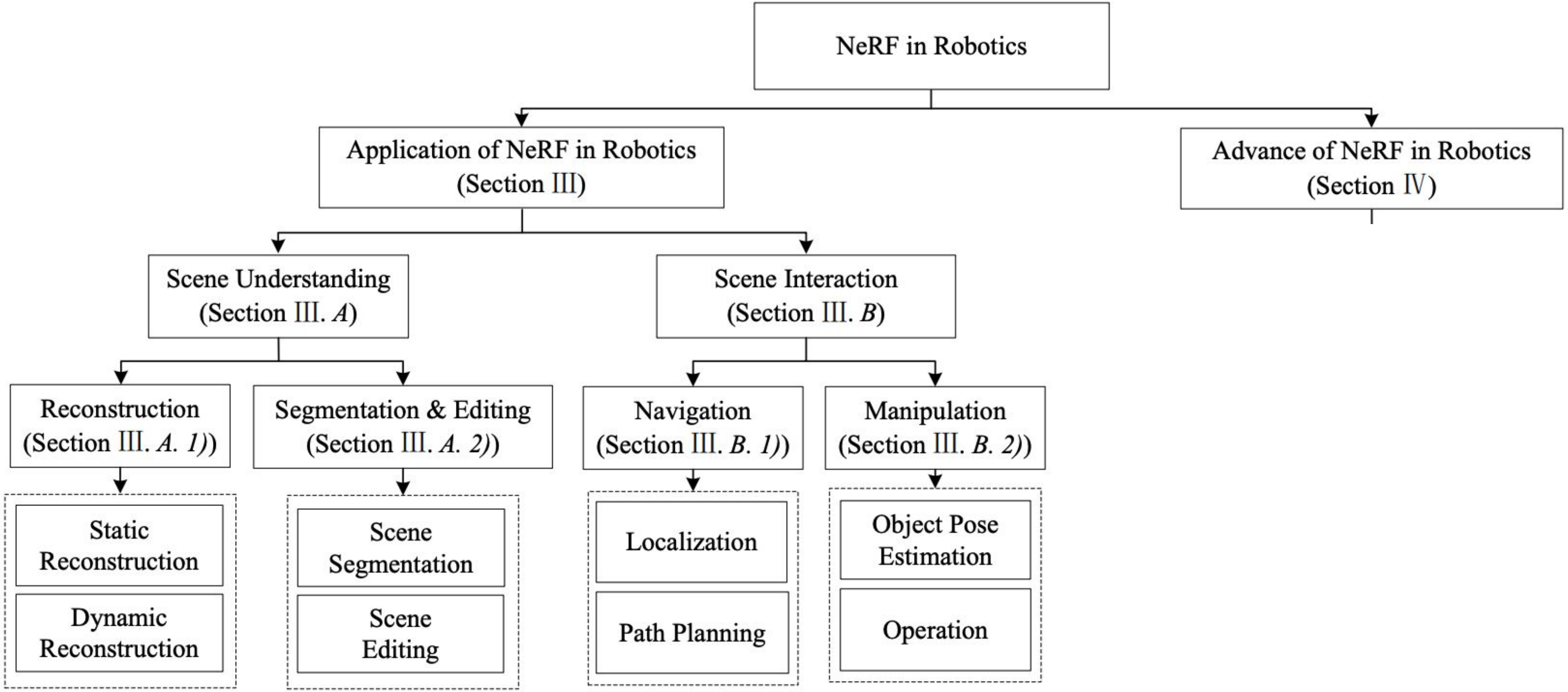


Fig. 1. A taxonomy of NeRF in robotics.

Wang, G., Pan, L., Peng, S., Liu, S., Xu, C., Miao, Y., ... & Wang, H. (2024). NeRF in Robotics: A Survey. *arXiv preprint arXiv:2405.01333*.

NeRF in Robotics:

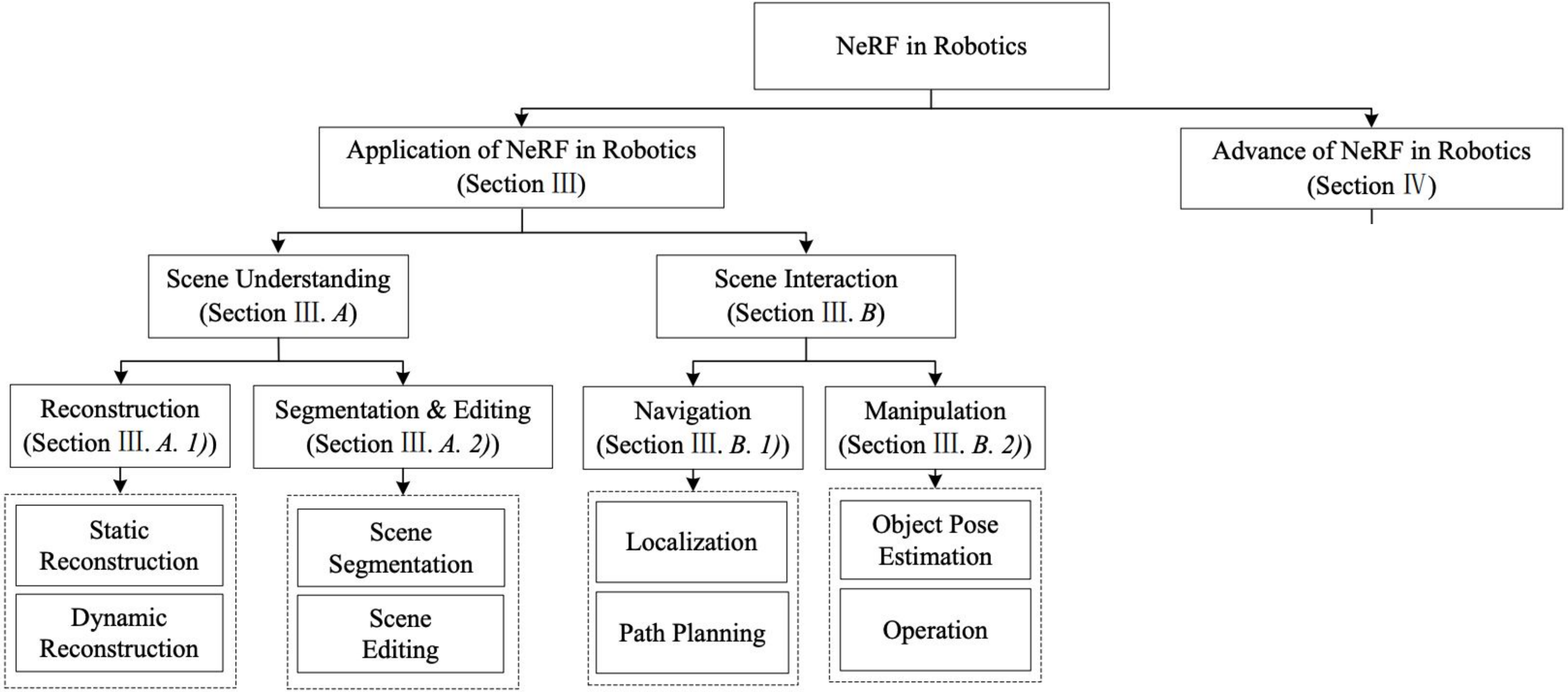


Fig. 1. A taxonomy of NeRF in robotics.

Wang, G., Pan, L., Peng, S., Liu, S., Xu, C., Miao, Y., ... & Wang, H. (2024). NeRF in Robotics: A Survey. *arXiv preprint arXiv:2405.01333*.

NeRF in Robotics:

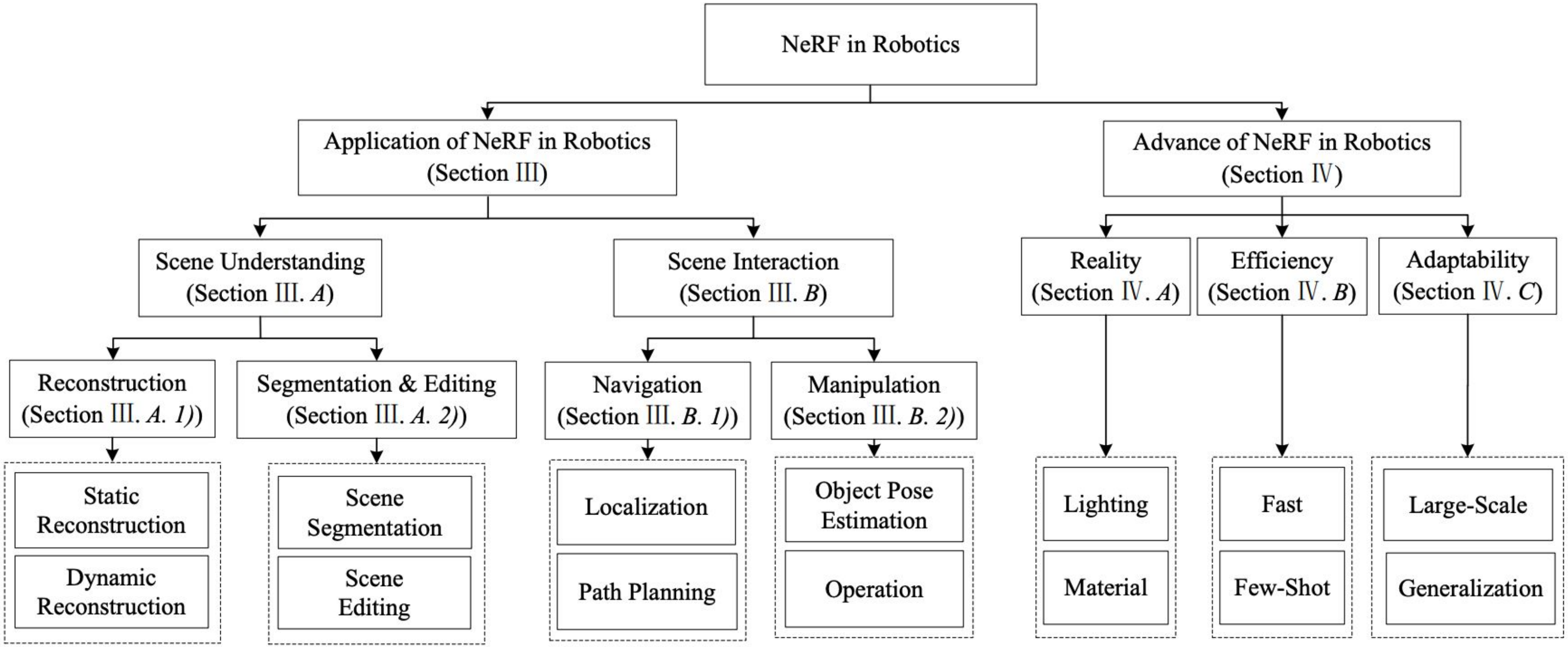


Fig. 1. A taxonomy of NeRF in robotics.

Wang, G., Pan, L., Peng, S., Liu, S., Xu, C., Miao, Y., ... & Wang, H. (2024). NeRF in Robotics: A Survey. *arXiv preprint arXiv:2405.01333*.

NeRF in Robotics:

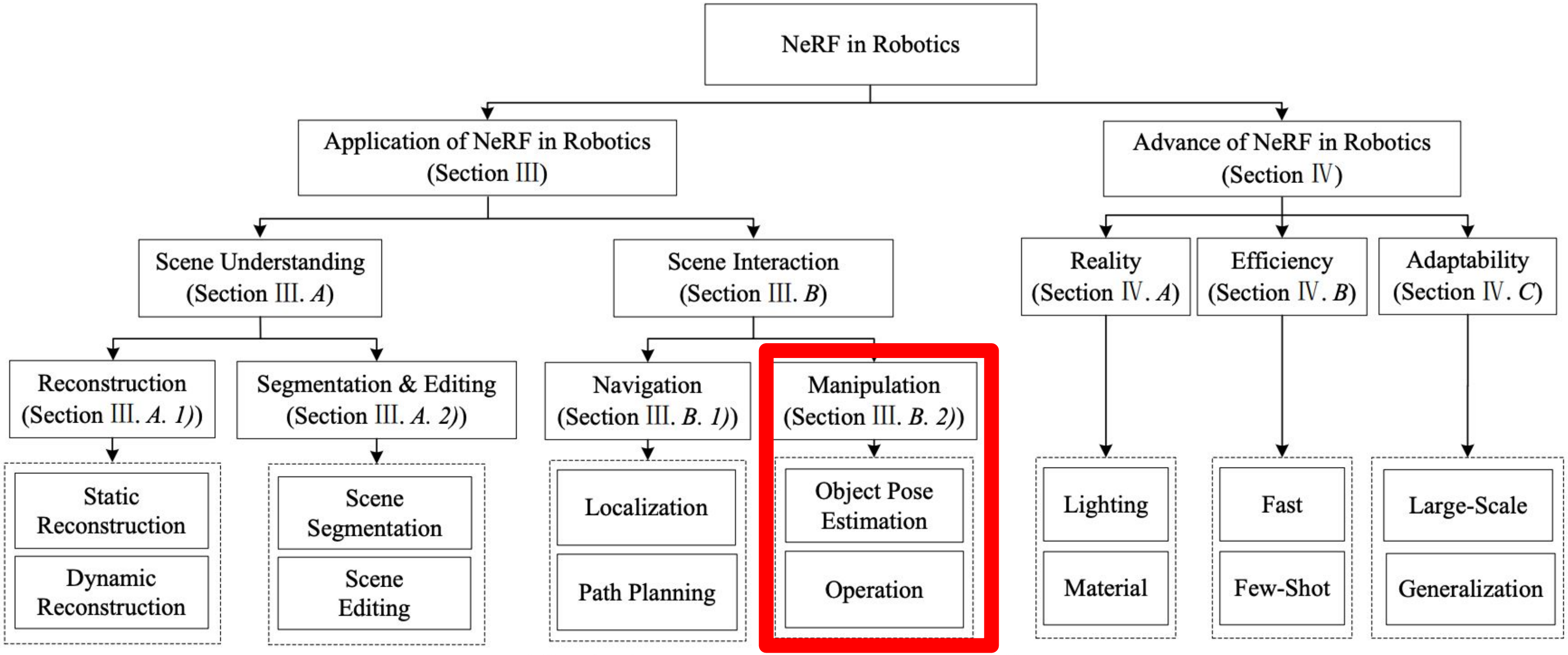


Fig. 1. A taxonomy of NeRF in robotics.

Wang, G., Pan, L., Peng, S., Liu, S., Xu, C., Miao, Y., ... & Wang, H. (2024). NeRF in Robotics: A Survey. *arXiv preprint arXiv:2405.01333*.

NeRF in Robotics:

Coming Up : NeRF2Real

NeRF in Robotics:

Coming Up : NeRF2Real

NeRF2Real is framework developed by DeepMind to incorporate the advantages of NeRF in creating vision processing models for robust applications

NeRF in Robotics: NeRF2Real

Byravan, A., Humplik, J., Hasenclever, L., Brussee, A., Nori, F., Haarnoja, T., ... & Heess, N. (2023, May). Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 9362-9369). IEEE.

NeRF in Robotics: NeRF2Real

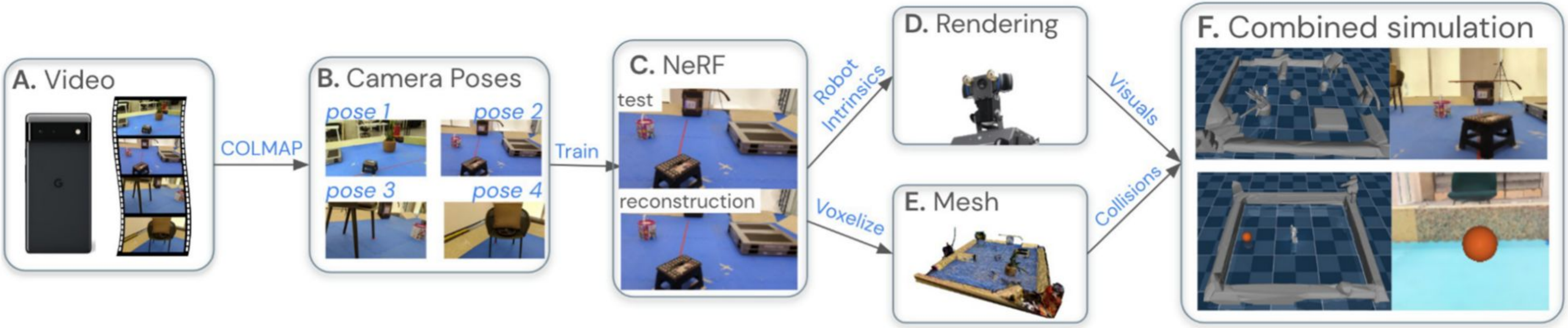


Fig. 2: Overview of our system for recreating a scene in a simulator. **A.** We collect a video of the scene using a generic phone. **B.** We use structure-from-motion software to label a subset of the video with camera poses. **C.** We train a NeRF on these labeled images. **D.** We render the scene from novel views using the calibrated intrinsics of the robot's head-mounted camera. **E.** We use the same NeRF to extract the scene geometry as a mesh. We coarsen the mesh and replace the floor with a flat primitive. **F.** We combine the simplified mesh with a model of a robot, and any other dynamic objects, in a physics simulator. See Fig. 3 for further details on this step.

Byravan, A., Humplik, J., Hasenclever, L., Brussee, A., Nori, F., Haarnoja, T., ... & Heess, N. (2023, May). Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 9362-9369). IEEE.

NeRF in Robotics: NeRF2Real

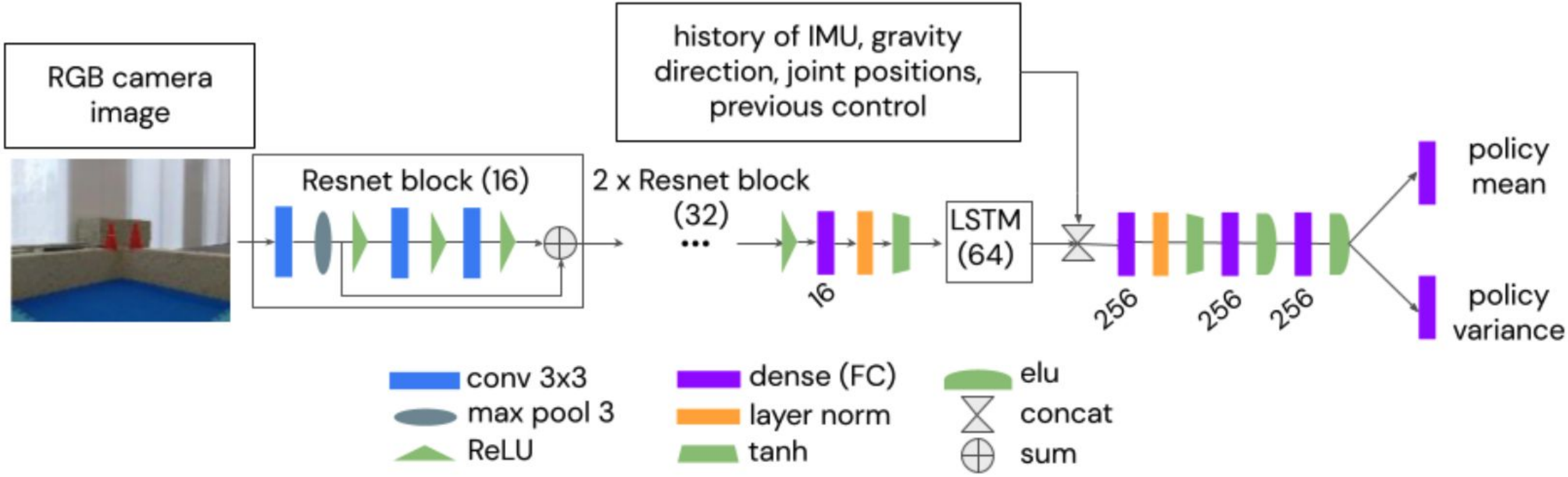


Fig. 4: The policy's network architecture.

Byravan, A., Humplik, J., Hasenclever, L., Brussee, A., Nori, F., Haarnoja, T., ... & Heess, N. (2023, May). Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 9362-9369). IEEE.

Architecture is inspired from the paper referenced below

Müller, T., Evans, A., Schied, C., & Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, *41*(4), 1-15.

NeRF in Robotics: NeRF2Real

The RL algorithm used is DMPO. (Deterministic Mixture Policy Optimization

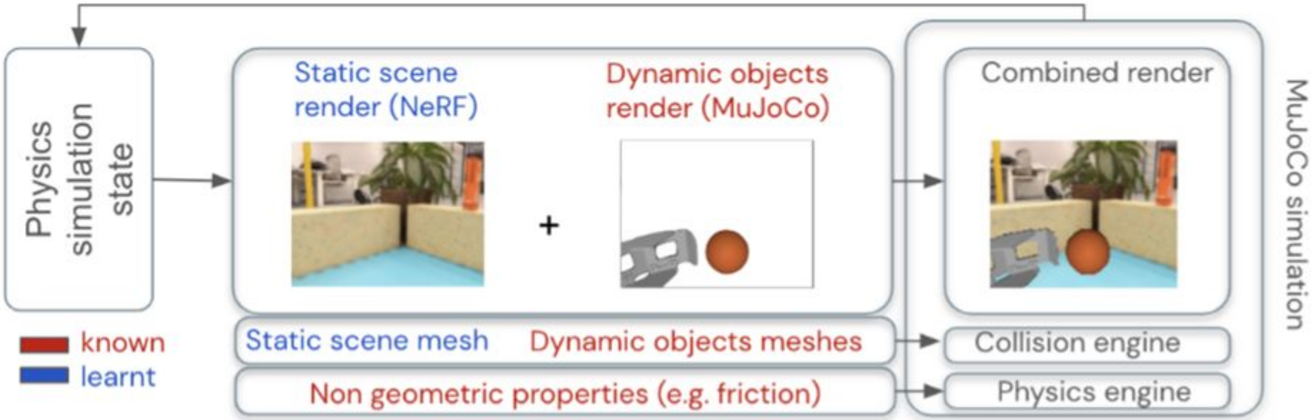


Fig. 3: Our MuJoCo simulation is created by combining: (1) the learnt static scene mesh (Section III-E), (2) the dynamic object meshes and (3) the learnt static scene NeRF rendering (Section III-D) on which (4) the Mujoco rendering of dynamic objects (a ball and robot's left arm in the camera image above) are overlaid. Other dynamic parameters (e.g. friction) are assumed known or measured.

Byravan, A., Humplik, J., Hasenclever, L., Brussee, A., Nori, F., Haarnoja, T., ... & Heess, N. (2023, May). Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 9362-9369). IEEE.
RL Training algorithm used is explained in the paper below
Osa, T., Hayashi, A., Deo, P., Morihira, N., & Yoshiike, T. (2023). Offline reinforcement learning with mixture of deterministic policies. *Transactions on Machine Learning Research*.

Gaussian Splatting in Robotics:

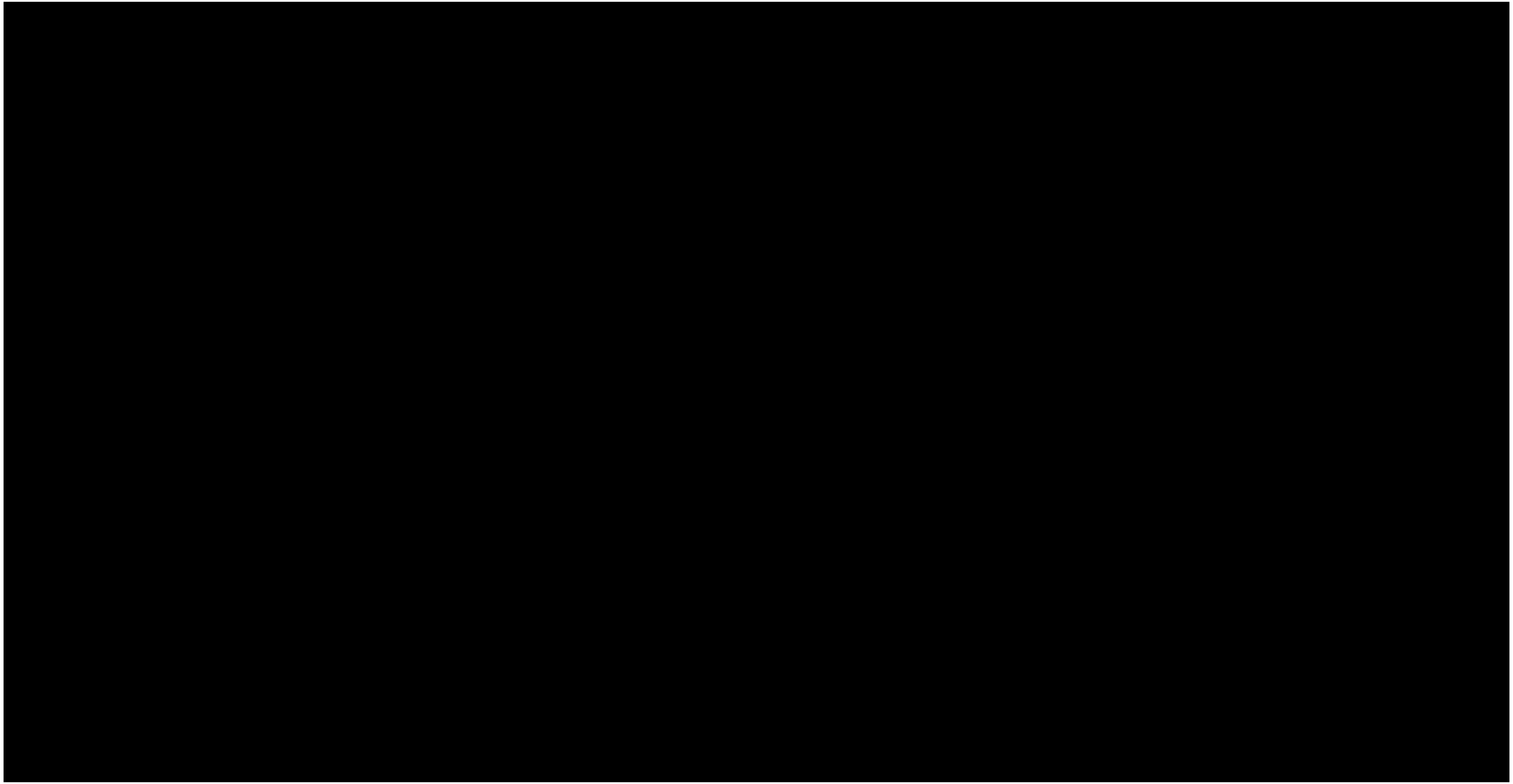
Coming Up : SplatSim

Gaussian Splatting in Robotics:

Coming Up : SplatSim

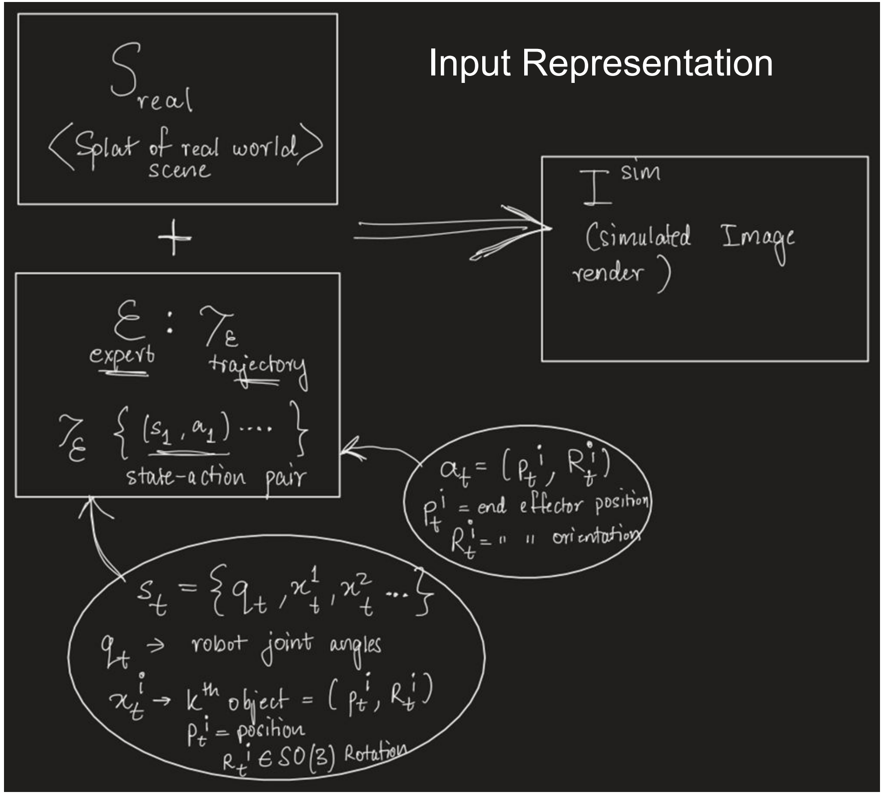
SplatSim is a very recent development, of bringing Gaussian Splatting of plain RGB Images to create high quality simulated renders to reduce the sim2Real gap in manipulation.

Gaussian Splatting in Robotics: Splat-Sim



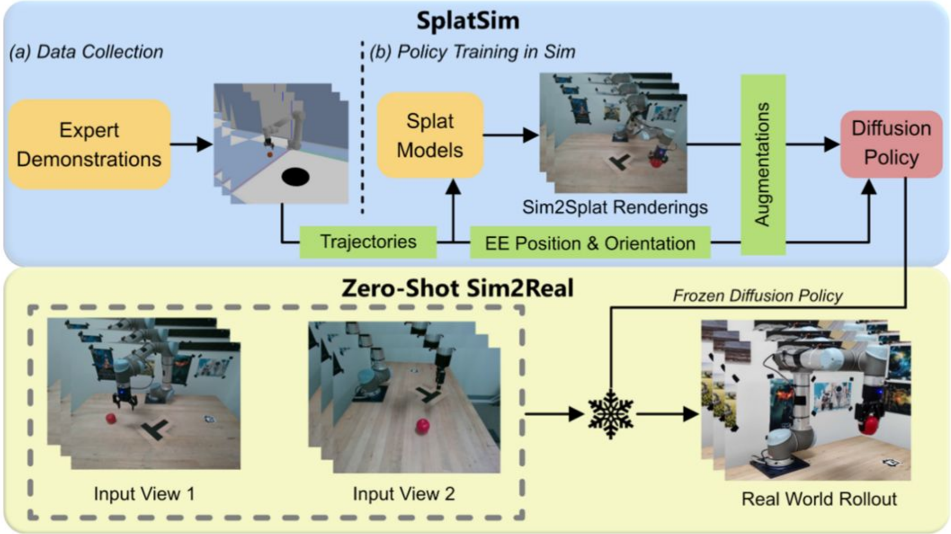
Qureshi, M. N., Garg, S., Yandun, F., Held, D., Kantor, G., & Silwal, A. (2024). Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting. *arXiv preprint arXiv:2409.10161*.

Gaussian Splatting in Robotics: Splat-Sim



1. The goal is to leverage expert demonstrations for obtaining a policy using Gaussian Splats

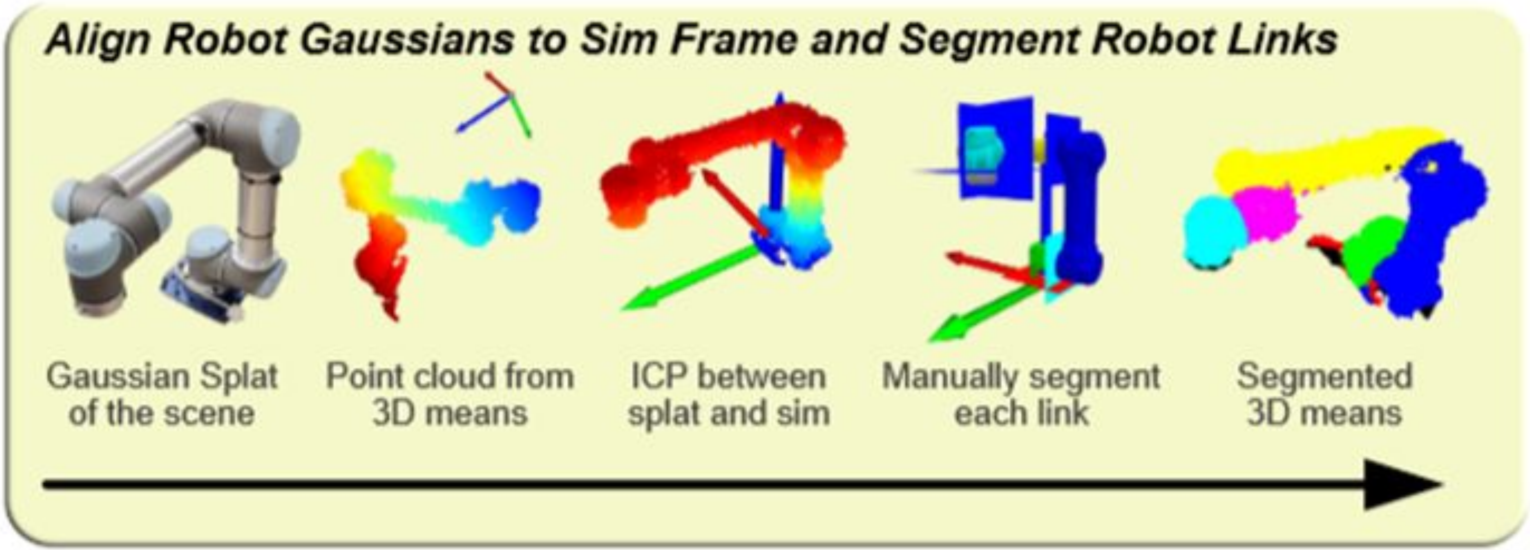
2. Providing defined Coordinate Frames for the Real, Simulated and robot enables clean transformation matrices



3. Given our Splat Model, it is important to align the Gaussian Splat with the real world robot links. Please see the next slide to understand this process

Qureshi, M. N., Garg, S., Yandun, F., Held, D., Kantor, G., & Silwal, A. (2024). Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting. *arXiv preprint arXiv:2409.10161*.

NeRF in Robotics: SplatSim



Qureshi, M. N., Garg, S., Yandun, F., Held, D., Kantor, G., & Silwal, A. (2024). Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting. *arXiv preprint arXiv:2409.10161*.

NeRF in Robotics: SplatSim

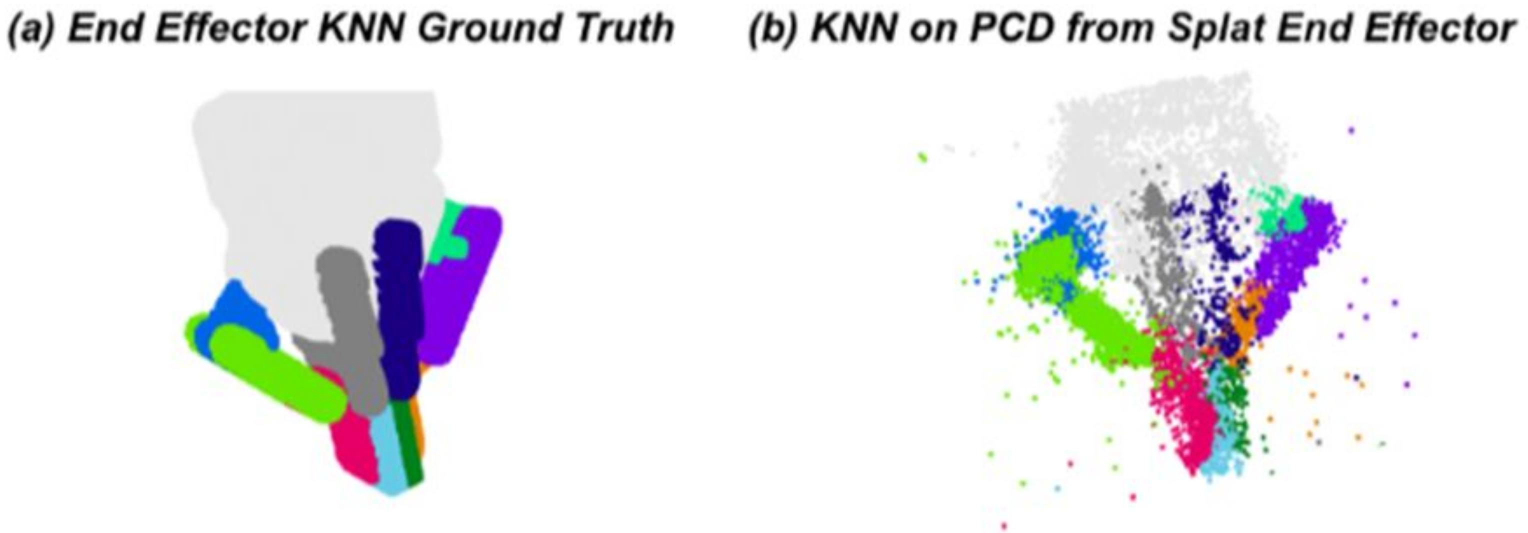
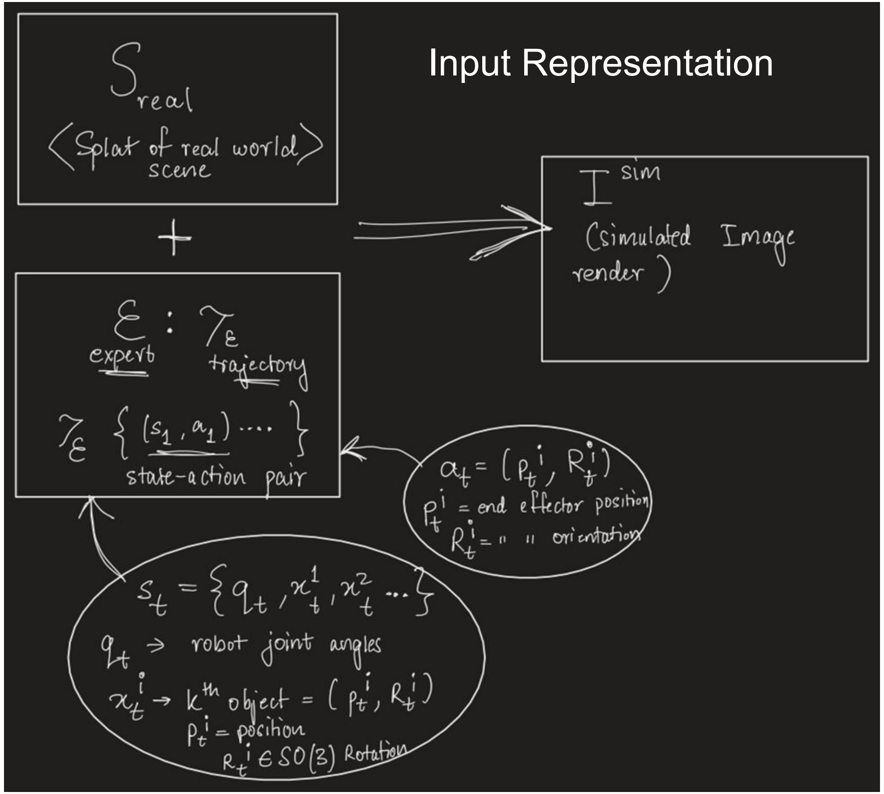


Fig. 4: We use a KNN-based classifier for segmenting links for articulated objects like parallel jaw grippers. We train a KNN model with the ground truth point labeling from the URDF model of the end effector.

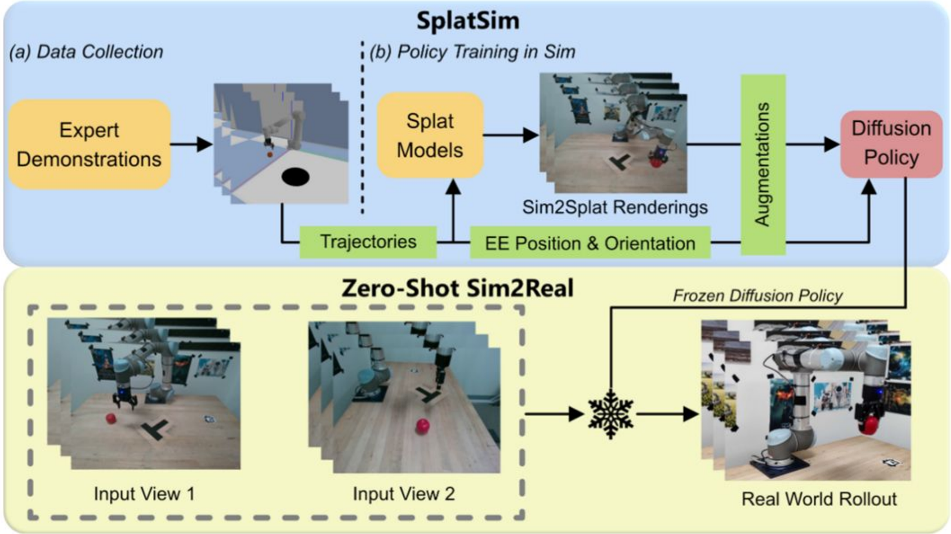
Qureshi, M. N., Garg, S., Yandun, F., Held, D., Kantor, G., & Silwal, A. (2024). Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting. *arXiv preprint arXiv:2409.10161*.

Gaussian Splatting in Robotics: Splat-Sim



1. The goal is to leverage expert demonstrations for obtaining a policy using Gaussian Splats

2. Providing defined Coordinate Frames for the Real, Simulated and robot enables clean transformation matrices



3. Given our Splat Model, it is important to align the Gaussian Splat with the real world robot links. Please see the next slide to understand this process

4. After the renders are finished, the Robot is trained using Reinforcement Learning, in particular the Diffusion Policy Algorithm is used

Qureshi, M. N., Garg, S., Yandun, F., Held, D., Kantor, G., & Silwal, A. (2024). Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting. *arXiv preprint arXiv:2409.10161*.
Diffusion Policy Paper referenced is
Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., ... & Song, S. (2023). Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 02783649241273668.

Next Lecture:
Student Lecture 4
Deformable Object Manipulation

DeepRob

[Group 2] Lecture 3

by Harshavardhan, Raj Surya, Vaibhav

NeRFs, Gaussian Splatting and Manipulation

University of Minnesota