3D Understanding & Novel View Synthesis

In the real World

Dr. Fabian Manhardt
Google

Goal of this lecture

- Understand the topic of novel view synthesis and getting a feeling for its applications
- Fully grasp the concept of NeRFs and Gaussian Splatting as the two most important techniques in this domaing
- Understand the most important challenges and get a rough understanding how those are tackled in literature
- Understand the task of semantic segmentation and how it can be employed within novel view synthesis

3D Novel View Synthesis & Scene Understanding - Ingredients

Reconstruction/Geometry

Planes
Point cloud
3D mesh
Voxel map
Neural representations

Semantics

Segments
Semantic Instance Seg
3D bounding boxes
Panoptic

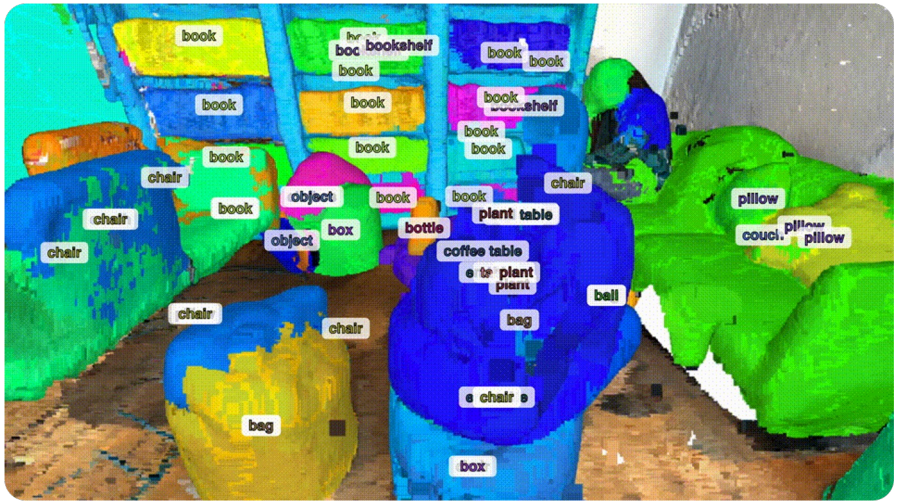
Layouts/Abstraction

Birds eye view
3D Scene graphs
Scene Captions

Novel View Synthesis

Zip-nerf [Barron23]

3D Semantic Understanding



Mask3D [Schult23]

3D Novel Synthesis & Scene Understanding - Applications

Augmented Reality

Smartphone

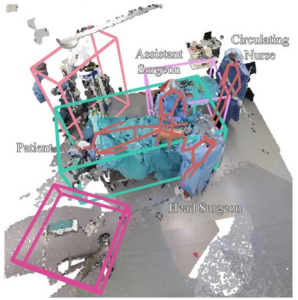
Headsets / Smart glasses

Robotics

Autonomous Driving

Interior design and architecture

Medical



...

Current Smartphone AR capabilities for Scene Understanding

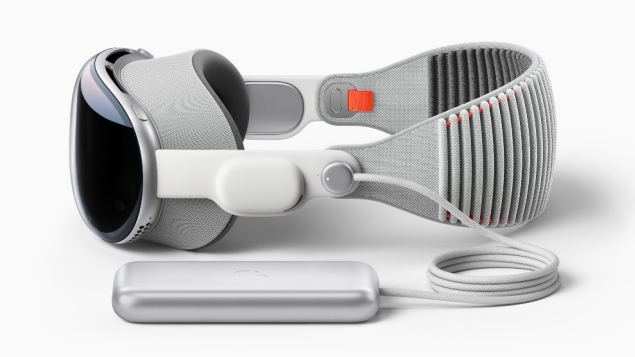
3D Semantic Mesh, ARKit (with Lidar)

Long range depth estimation /
Plane estimation, ARCore (monocular)

Niantic 3D Mapping (monocular)

Currently available features:
3D Planes, Depth prediction, Persistent anchors/objects,
SLAM and 3D Mapping, 3D semantic segmentation,
3D Layouts (with lidar)

Augmented Reality: from headsets to smart glasses



Apple VisionPro

Magic Leap 2

Xreal Air AR glasses

Vuzix Smart Glasses

Microsoft HoloLens 2

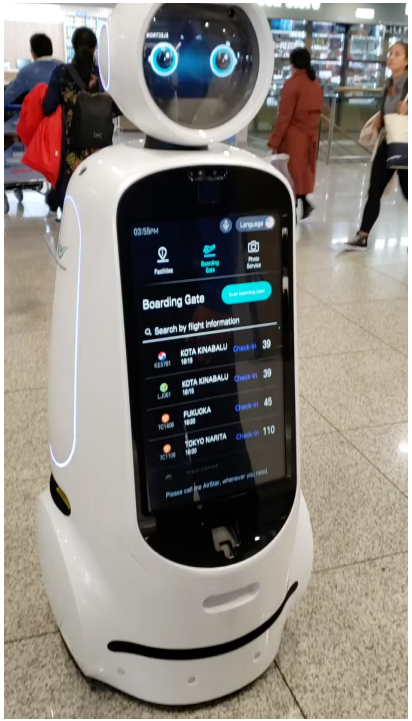
Project Aria, Meta

Immersivity / "smartness"

Glass form factor

Apple VisionPro Spatial Audio features (from VisionPro announcement)

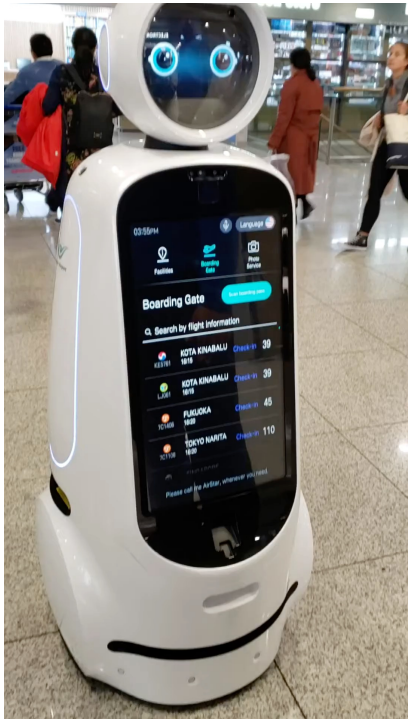
Scene understanding for household/service robotics



Incheon Airport Service AIRSTAR Robot

Going from this..

Scene understanding for household/service robotics



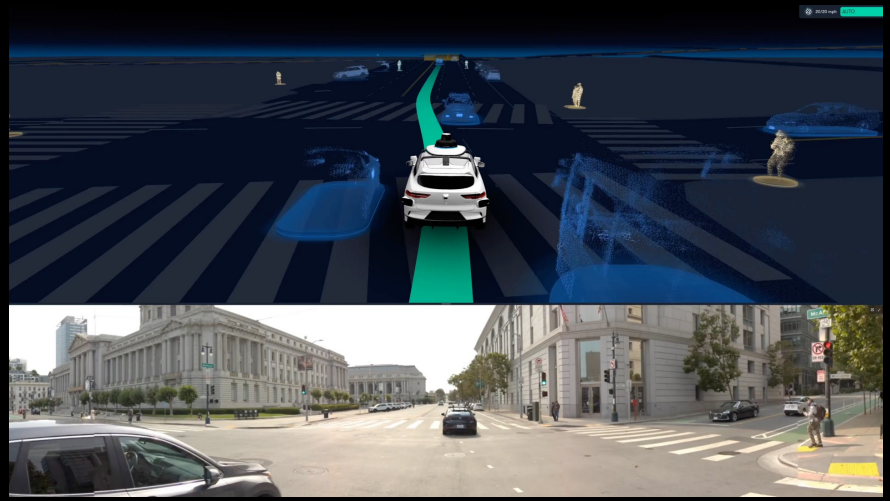
Incheon Airport Service AIRSTAR Robot

TRI home helping robot

Going from this..

..to this

Scene understanding for Autonomous Driving



Waymo Car in San Francisco

MobilEye Car in New York City

3D Representations

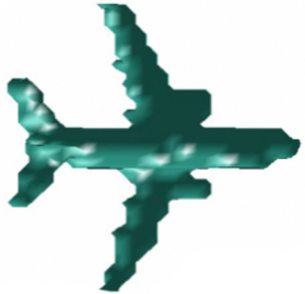
Explicit 3D data representation

Point Cloud

- unordered list of 3D coordinates
- sparse, no topology
- can handle **full 3D**

3D Mesh

- collection of 3D vertices and faces
- sparse, with topology
- can handle **full 3D**

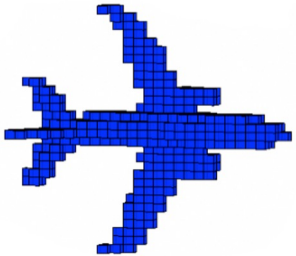


Sparse

Organized

Voxel map

- discretized 3D coordinates on a regular grid
- organized, no topology
- can handle **full 3D**

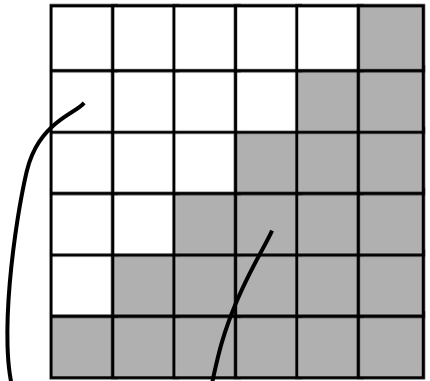


Range (depth) map

- 1-channel image encoding distances
- organized, no topology
- only **2.5D views**

Occupancy Field and Signed Distance Field

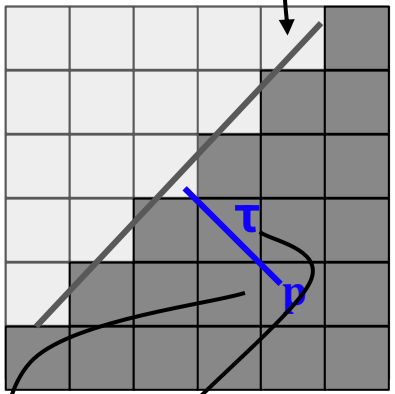
OF



$$f(p) = \begin{cases} 0, & \text{if p outside} \\ 1, & \text{otherwise} \end{cases}$$

(T)SDF

Surface implicitly defined as f(x) = 0



$$f(p) = \begin{cases} -\tau, & \text{if p outside} \\ \tau, & \text{otherwise} \end{cases}$$

- Special cases of a voxel map where each voxel stores:
  - OF: volume occupancy
  - SDF: distance to the nearest surface
- Common variations: Truncated SDF (TSDF), Unsigned DF (UDF)

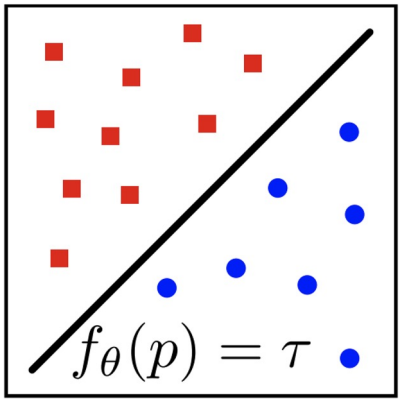
3D Implicit representations

Learn a **function f** via a non-linear classifier whose decision boundary is the desired 3D surface

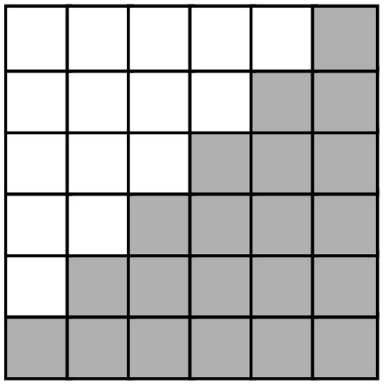
The function f approximates an **Occupancy Field [1] or a Signed Distance Field [2]**

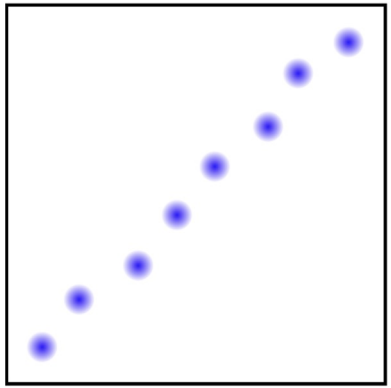
Why?

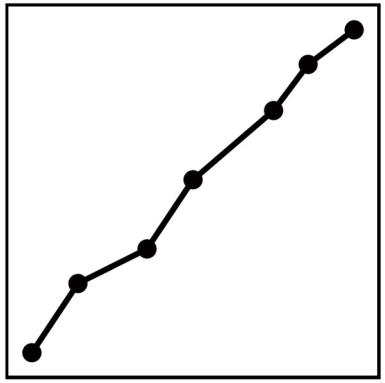
- No discretization
- Arbitrary topology & resolution
- Low memory footprint

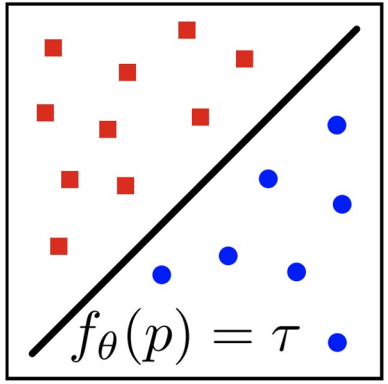


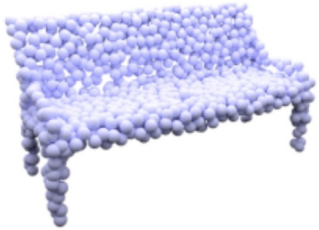
[1] L Mescheder et al, Occupancy Networks: Learning 3D Reconstruction in Function Space, CVPR 2019
[2] JJ Park et al, DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation, CVPR 2019







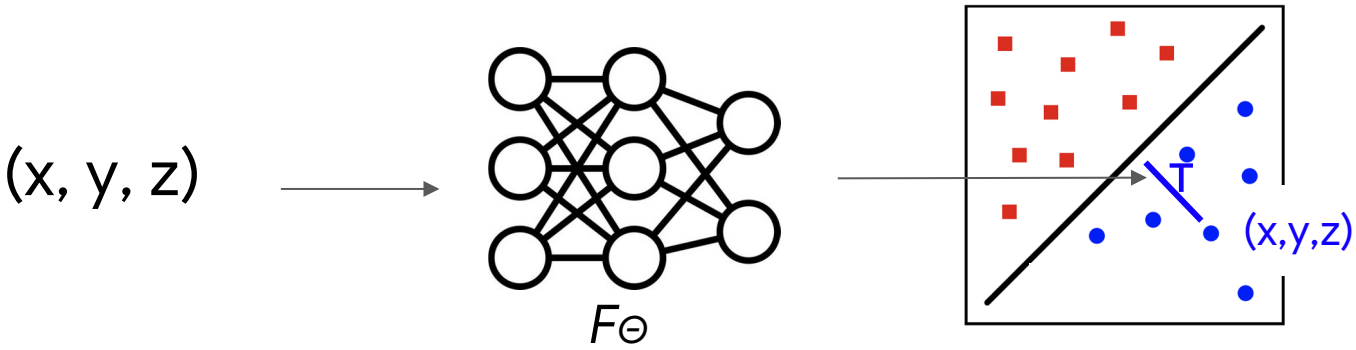




Voxel Map **Point Cloud** **3D Mesh** **Implicit**

L Mescheder et al, Occupancy Networks: Learning 3D Reconstruction in Function Space, CVPR 2019

Summary of 3D implicit representations



3D implicit representations are a type of data representation that uses a function to map from a domain to a range. The function is typically learned by overfitting a neural network.

They tend to be more compact and flexible than traditional explicit representations

Applications

- Data storage / compression
- 3D classification / segmentation
- 3D reconstruction from single view
- 3D generation

Novel View synthesis

Mildenhall et al. "Local Light Field Fusion." SIGGRAPH 2019

Novel View Synthesis

Input

- A set of input images capturing a scene from different viewpoints

Output

- A representation that
  - faithfully reproduces the input views
  - Enables rendering **novel viewpoints** of the scene

Challenges

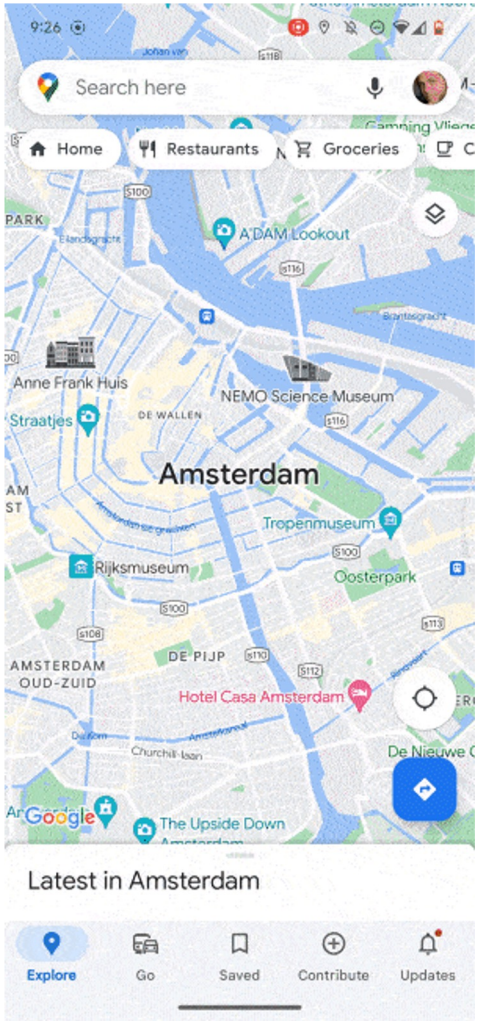
- Requires understanding of how the scene looks like in 3D
- Needs to model complex lighting effects (shadows, reflections, etc)
- Ill-posed problem

State-of-the-Art View Synthesis

What kind of view synthesis is possible today?

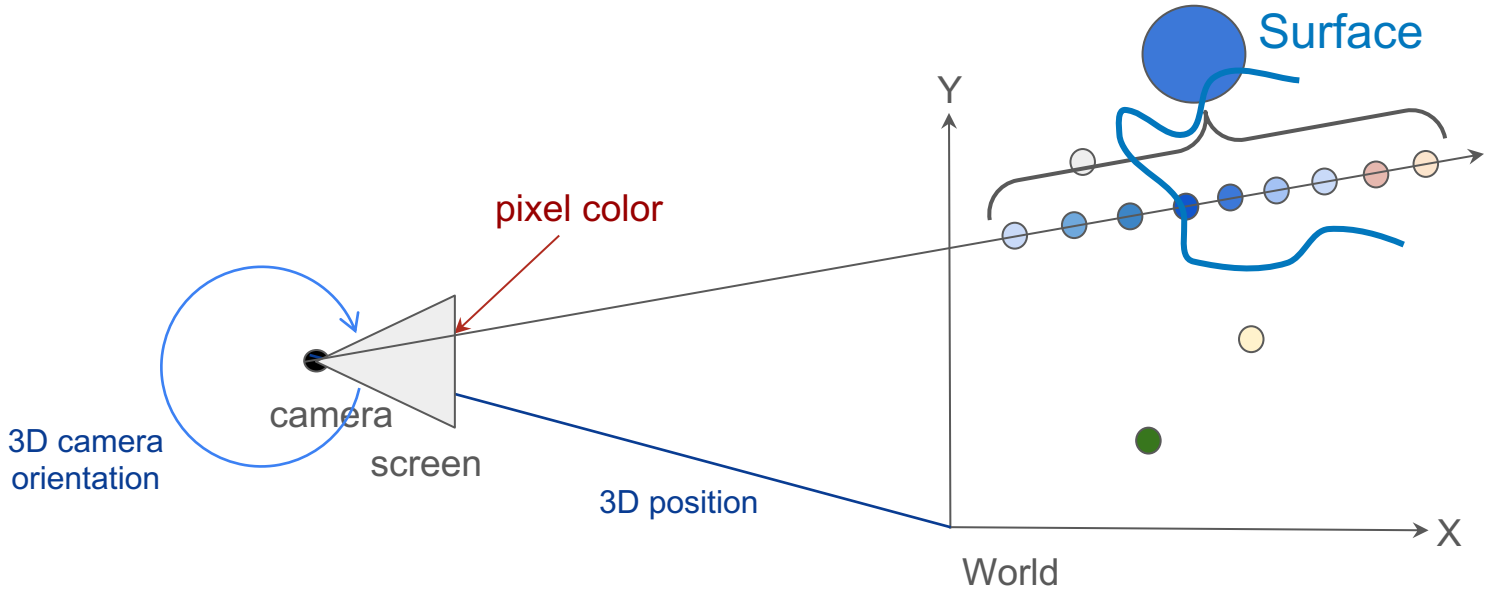
Barron et al. "Zip-nerf: Anti-aliased grid-based neural radiance fields." ICCV 2023

Some Google Products

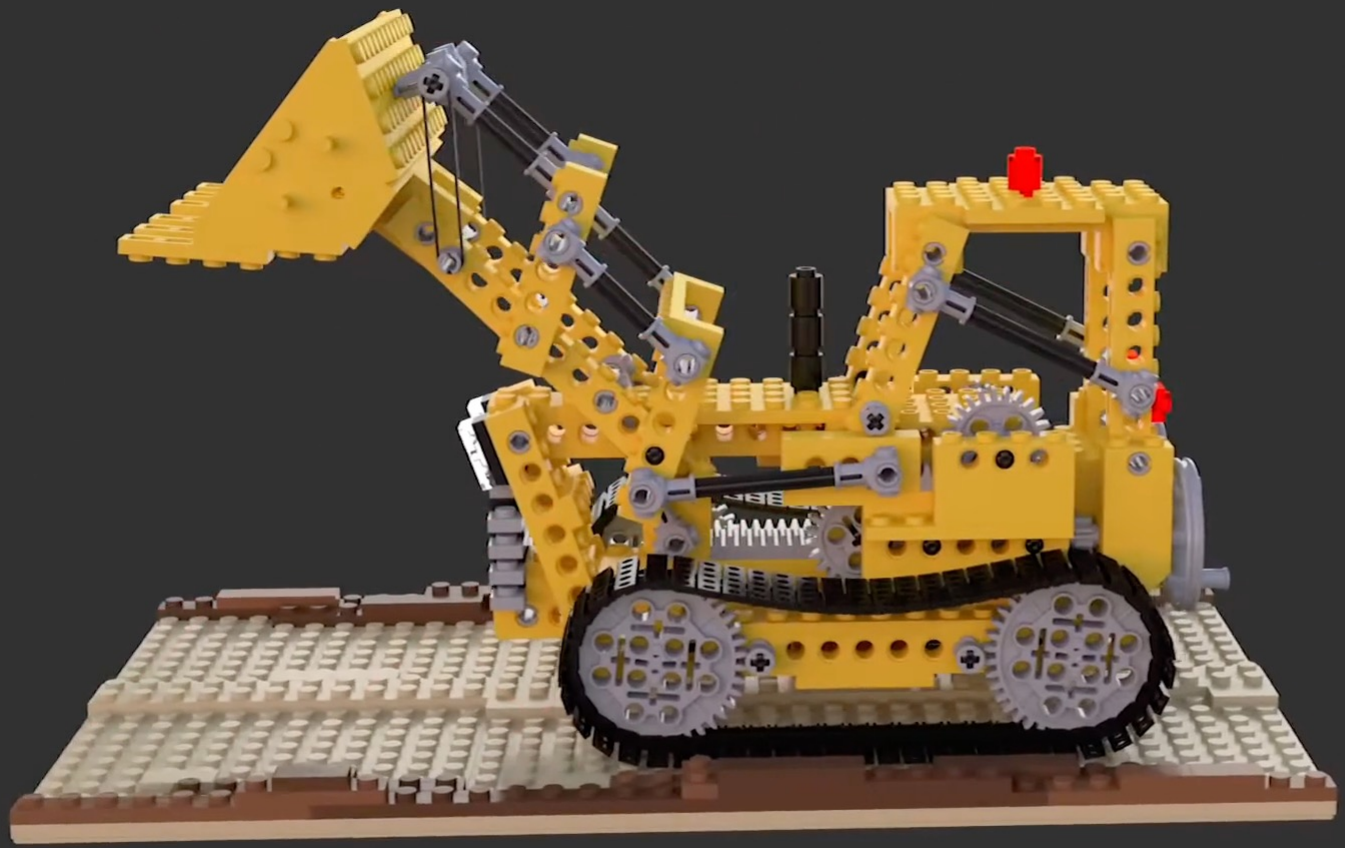


NeRFs: Neural Radiance Fields

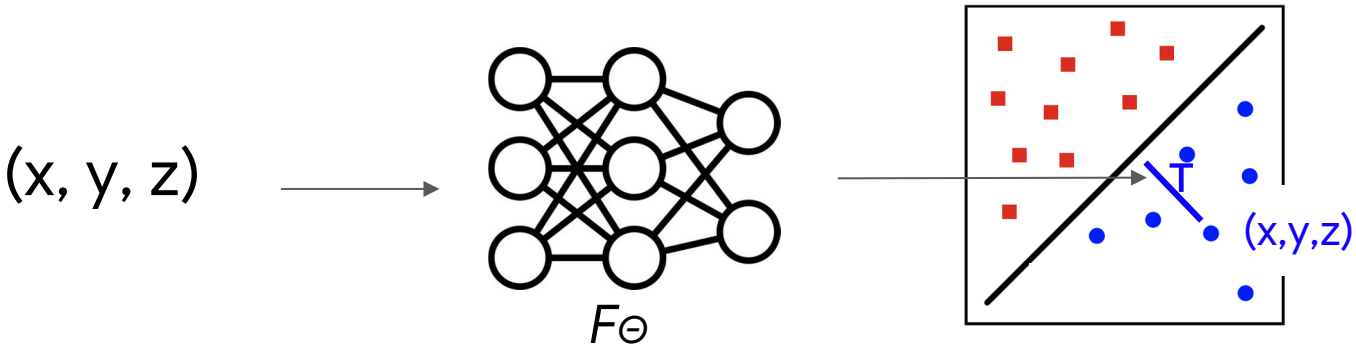
Neural Radiance Fields



Each point in space emits color



Summary of 3D implicit representations



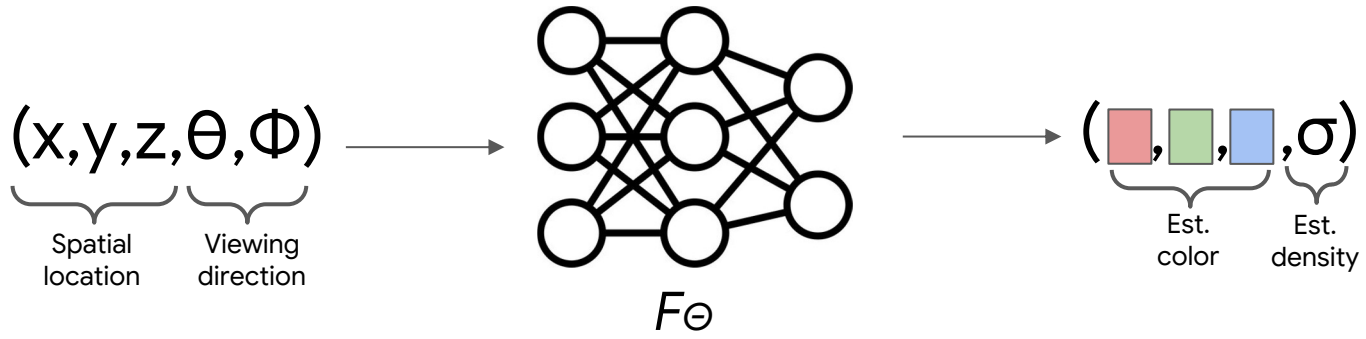
3D implicit representations are a type of data representation that uses a function to map from a domain to a range. The function is typically learned by overfitting a neural network.

They tend to be more compact and flexible than traditional explicit representations

Applications

- Data storage / compression
- 3D classification / segmentation
- 3D reconstruction from single view
- 3D generation

Neural Radiance Fields (NeRFs)



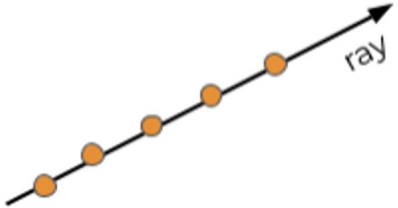
- The network is a simple ReLU MLP that maps from location/view direction to color/density
- Density σ describes how solid/transparent a 3D point is (can model, e.g., fog)
- Conditioning on view direction allows for modeling **view-dependent effects**

Input: posed images (no explicit 3D geometry or depth)

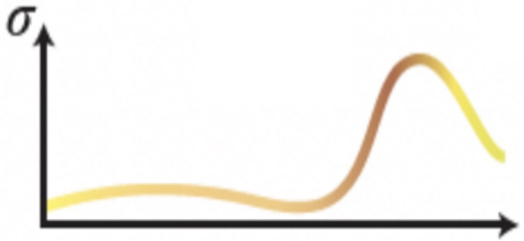
Volumetric Rendering

Secret of NeRFs:

Differentiable Everywhere



Sampled ray points

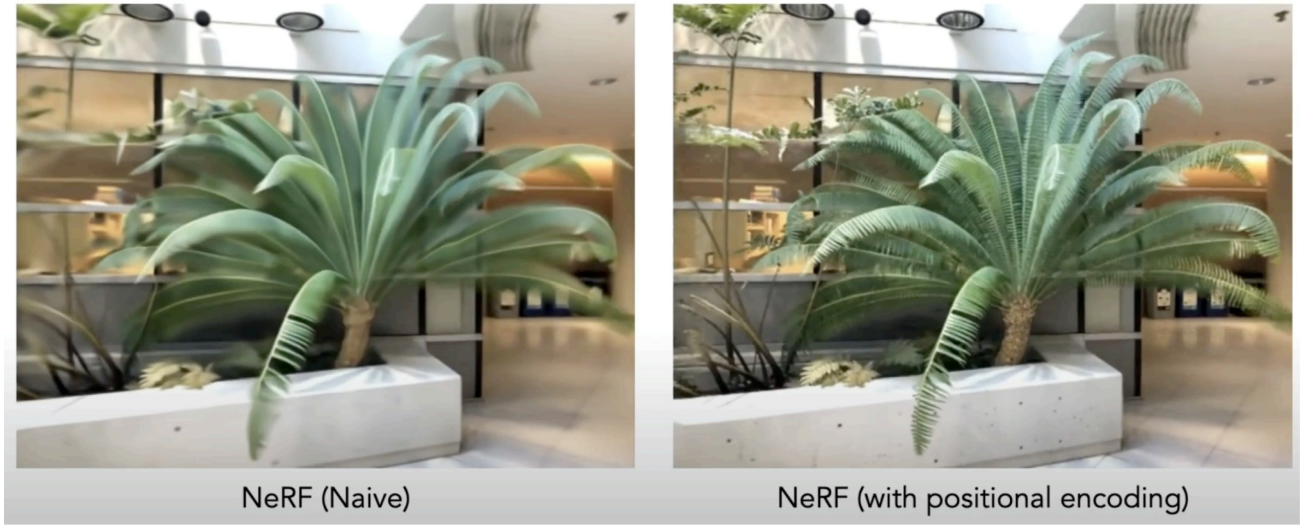


Underlying density distribution

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \alpha_i(\mathbf{x}_i) \prod_{j<i} \left(1 - \alpha_j(\mathbf{x}_j)\right) c(\mathbf{x}_i, \mathbf{d}) \qquad \alpha_i(\mathbf{x}) = 1 - \exp\left(-\sigma_\theta(\mathbf{x})\, \delta_i\right)$$

$\delta_i = t_{i+1} - t_i$: distance between sampled points

Positional encoding and Fourier features



$F_\Theta(x, y, z, \theta, \Phi)$

$F_\Theta(\gamma(x, y, z), \gamma(\theta, \Phi))$

Adding positional encodings to input coordinates (point and direction) helps recover fine details

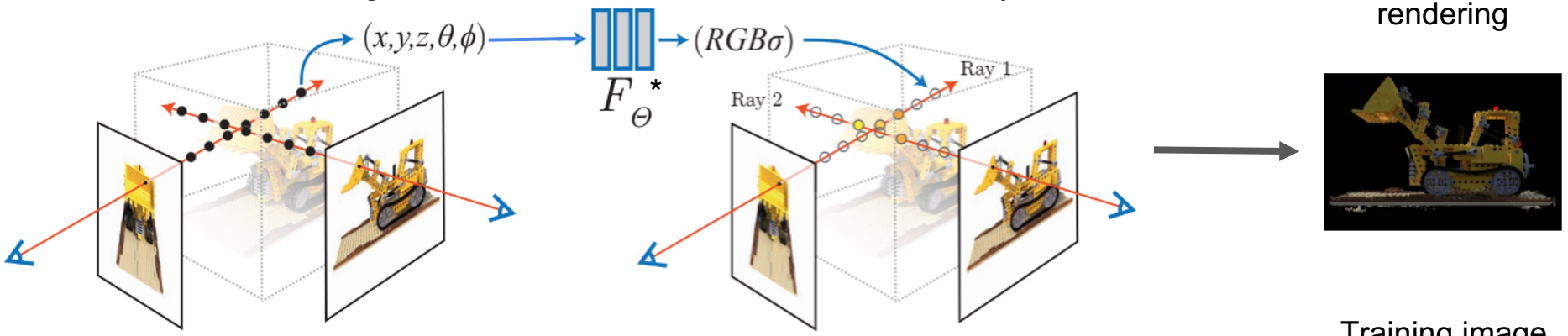
Tancik et al.: Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. NeurIPS, 2020

Training of NeRFs

5D Input
Position + Viewing direction

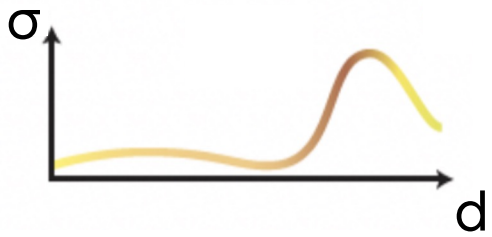
Output
Color + Density

Volumetric rendering



Training image

Volumetric Rendering
Single Ray



Rendering (or photometric) Loss

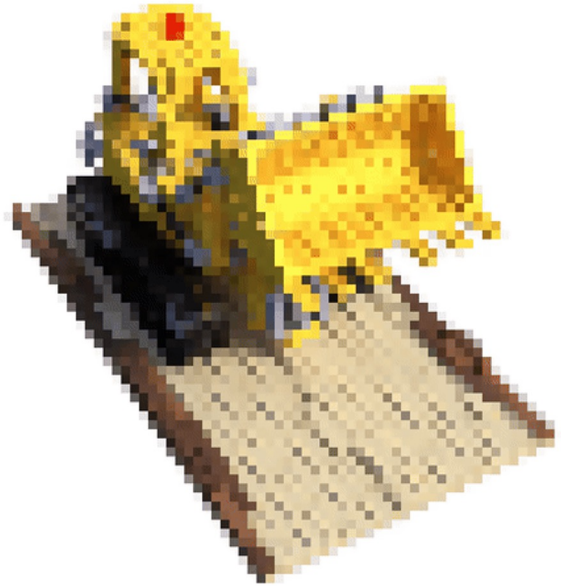
*This is done in practice with 2 MLPs: one non-view dependant that regresses the density, the other that takes also the viewing direction and computes the RGB (since density should not be view-dependant!)

Images from Mildenhall et al. NeRF. Commun. ACM.

MipNeRF

In NeRF rendering we effectively solve a multi-scale problem:

- Models are often trained at a given scale given the training data
- Output rendering, however, should be performed at different scales and distances



MipNeRF

Original Image

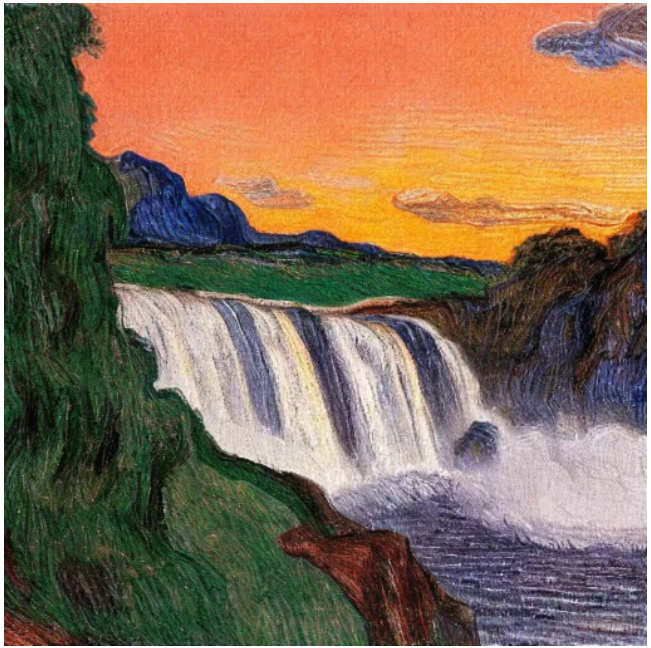
"Subsampling"

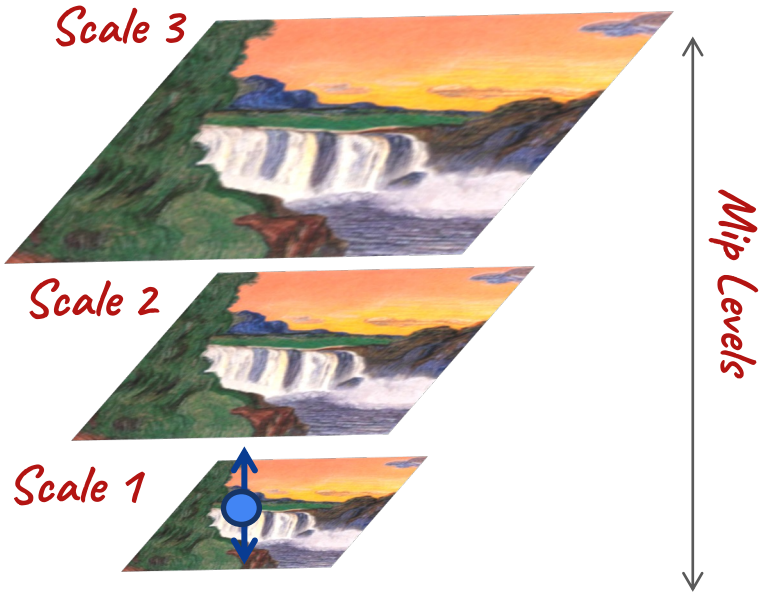
Blurred Image

"Subsampling"

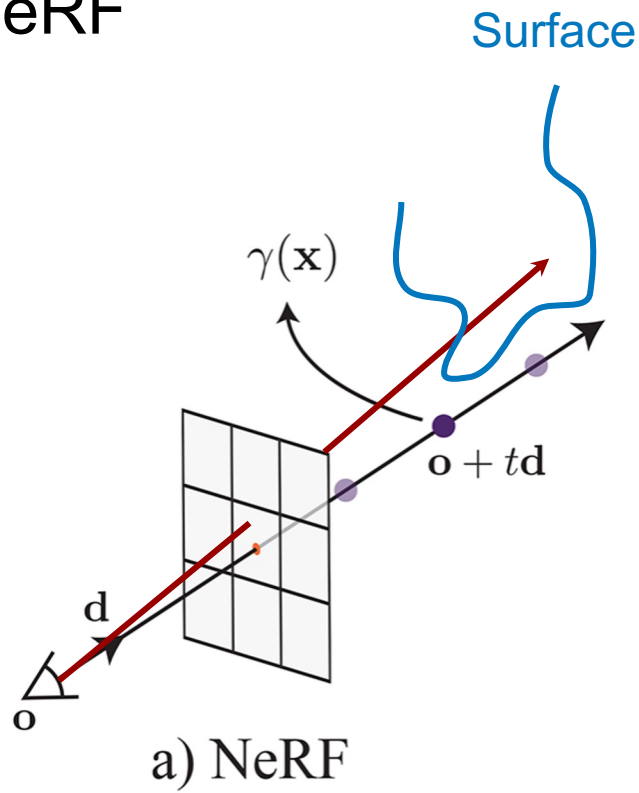
Downsampling an image, high frequency details can cause artifacts
When instead first blurring and then downsampling, results are again clean

MipNeRF





MipNeRF

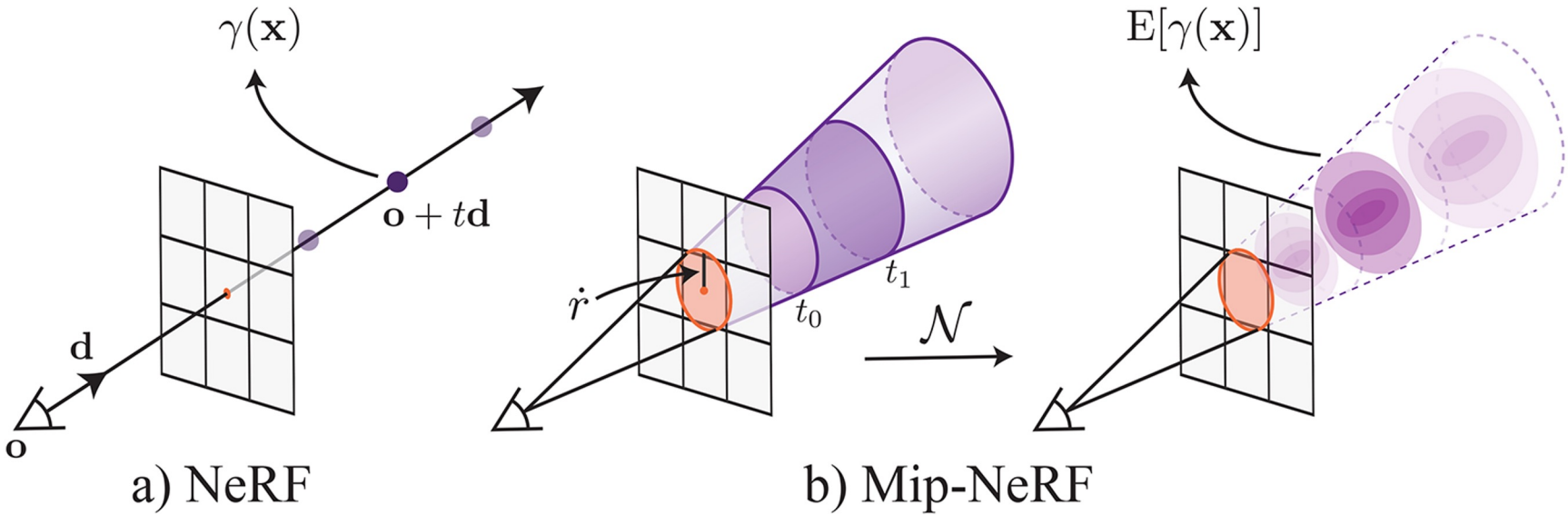


Dependent on distance or scale, we can miss the surface, leading to artifacts

In addition, high frequencies in the full resolution can lead to inconsistencies when subsampling

Similar effect is also in 2D im resampling.
Idea: Blur image before downsampling.

MipNeRF



MipNeRF replaces point samples with gaussians to model the whole cone.
This is done via the use of integrated positional encoding.

MipNeRF

Spatial position

Positional encoding features

Mean, variance of spatial region

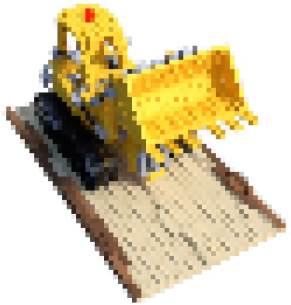
Integrated positional encoding features

Integrate positional encodings:

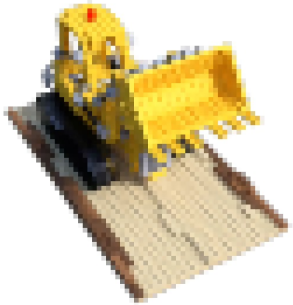
- Encodes mean and variance of the gaussian
- Allows the model to understand the size of space that is encoded

MipNeRF

NeRF



Mip - NeRF



NeRF

Mip - NeRF

Some Open Challenges

In the wild data

Nerf in the Wild

Expensive Training Time

Instant NGP

Surface Reconstruction

Neural Distance Fields

NeRF in the Wild

The **real world** is often very complex

Casually taken images differ in daytime, occlusion, cameras

NeRFs trained on **these images look bad** as they are trying to find a representation that satisfies all settings

NeRF in the Wild

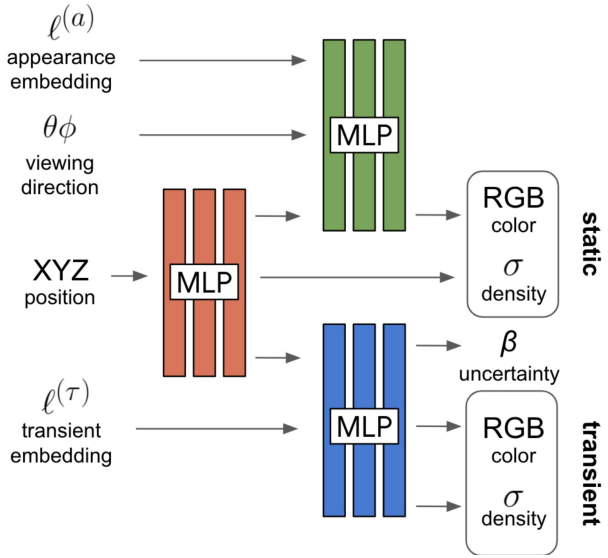
Proposes to use two different branches for **static** and **transient**

Static:

Appearance embedding (glo features) encode difference in appearance due to e.g. change of daytime

Transient

Optimized to understand object that are occluding / not part of the actual scene. Uncertainty loss to implicitly learn this



(a) Static (b) Transient (c) Composite (d) Image (e) Uncertainty

NeRF in the Wild

Input Views

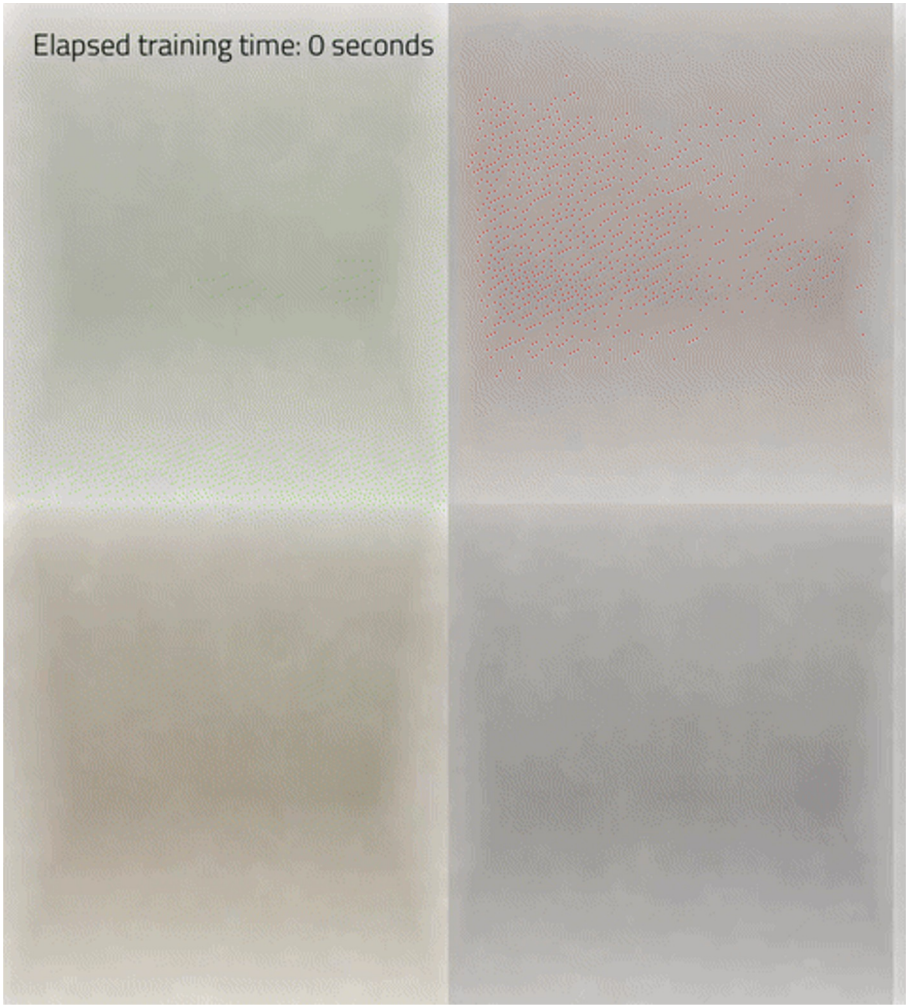
Clean Novel Views

NeRF in the Wild

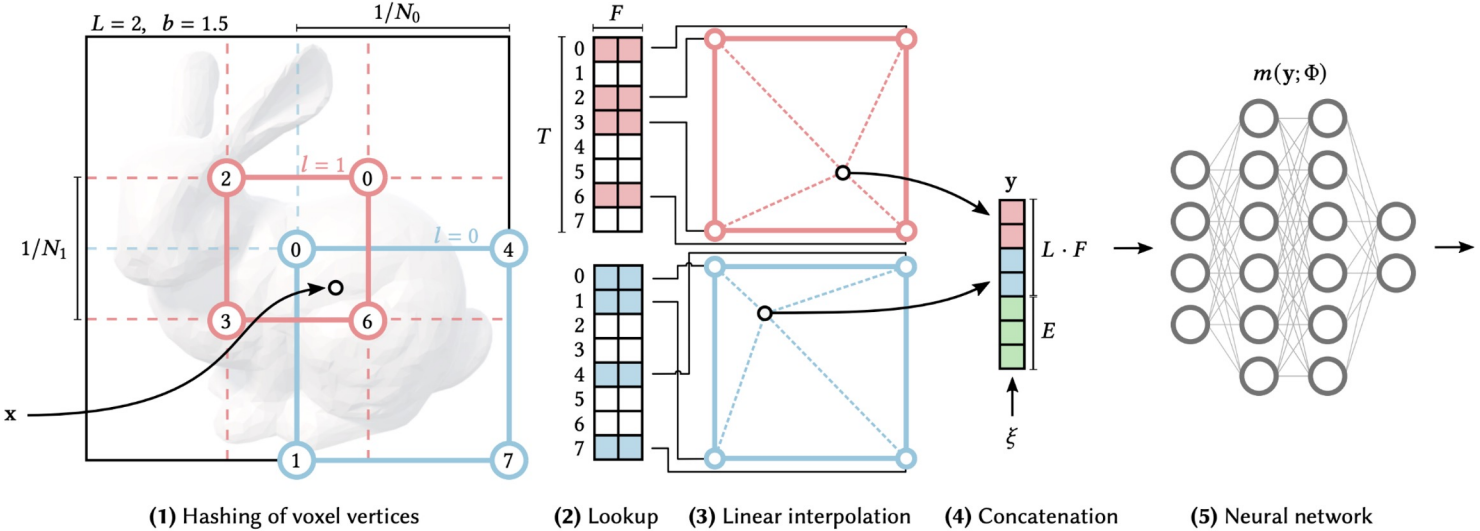
Instant NGP

Training of **(Mip) NeRF** takes **several hours** due to training of large MLPs for modelling the scene

Instant NGP unloads most of the heavy computations to trainable hash grids



Instant NGP



- Instant NGP unloads most of the heavy computations to trainable hash grids
- This comes for the cost of memory to store all these feature grids
- Effectively trading training speed for memory consumption

Instant NGP

(a) None	(b) Multiresolution grid	(c) Frequency	(d) Hashtable ($T=2^{14}$)	(e) Hashtable ($T=2^{19}$)
411k parameters 10:45 (mm:ss)	10k + 16.3M parameters 1:26 (mm:ss)	438k + 0 parameters 13:53 (mm:ss)	10k + 494k parameters 1:40 (mm:ss)	10k + 12.6M parameters 1:45 (mm:ss)

- Using hash grids training speed can be reduced from hours/days to a few minutes/second
- Larger hash tables can significantly increase rendering quality
- However, comes with the cost of needing significantly more parameters / GPU memory for training

Instant NGP

Neural Distance Fields

Neural radiance fields

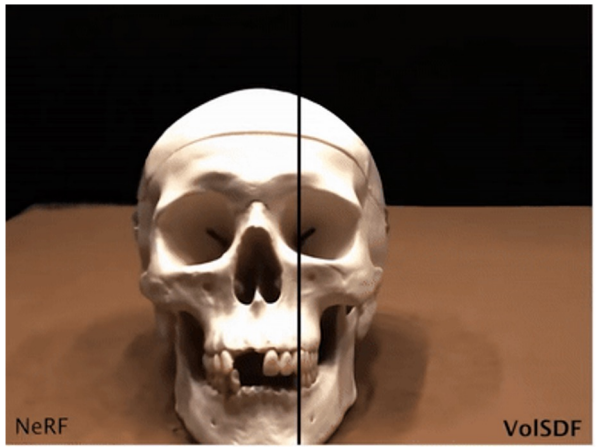
+ Achieves superb novel view synthesis, thanks to smooth optimization behavior
- Optimized for novel view synthesis, neglecting the actual underlying geometry

Neural distance / occupancy fields (NeuS, VolSDF, UniSurf, ..)

+ 3D representation aimed at keeping the power of SDF / occupancies for geometry extraction,
+ Can be well optimized by unifying the SDF / occupancy representations with standard volumetric rendering

Signed distance functions / models

+ Powerful representation for extraction of underlying geometry
- Not as trivial to render and thus to optimize using only a set of posed RGB images



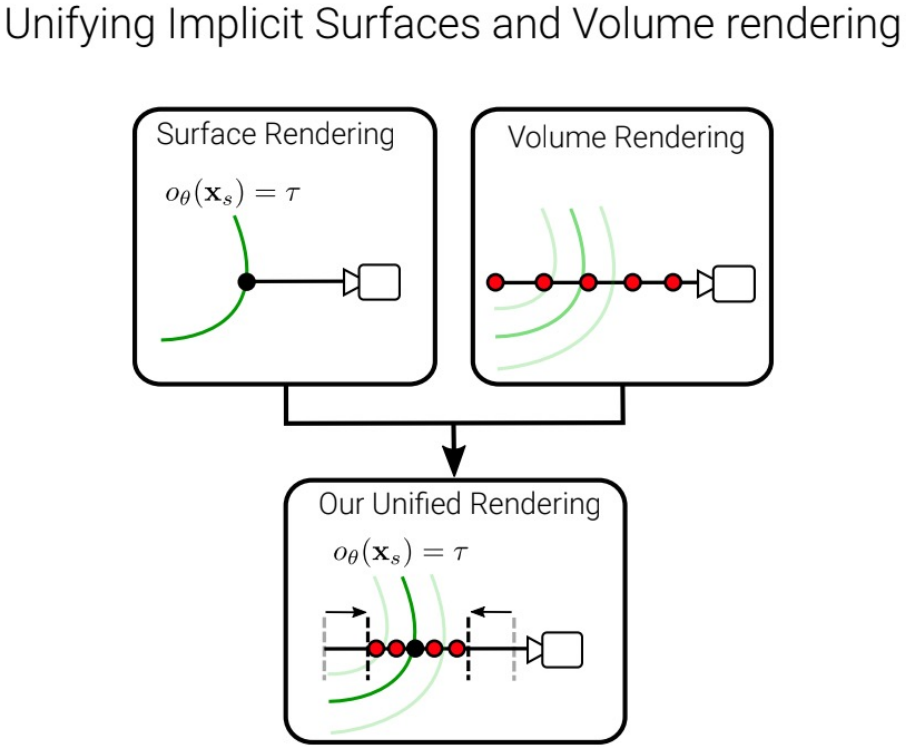
VolSDF [Yariv21] results on DTU

Neural Distance Fields / UNISURF

The **goal** of distance fields is to conduct **surface rendering** whilst keeping the power of **volumetric rendering**

Mapping SDF/occupancy values to densities enables the use of standard volumetric rendering (e.g. VolSDF)

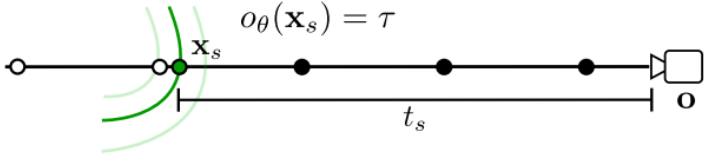
Incrementally decreasing the sampling range enforces the density to follow the underlying geometry (e.g. UNISURF)



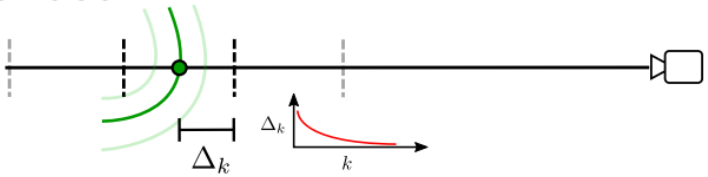
Oechsle, Peng, and Geiger: UNISURF: Unifying Implicit Surface Representations and Radiance Fields. ICCV, 2021.

Neural Distance Fields / UNISURF

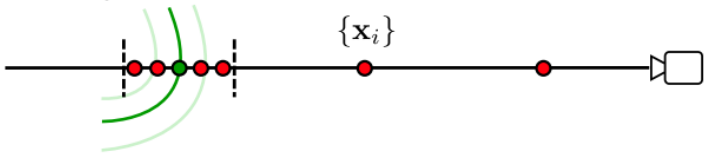
- Find surface along a ray: uniform sampling + iterative secant method



- Define interval at the surface



- Volume rendering with occupancies

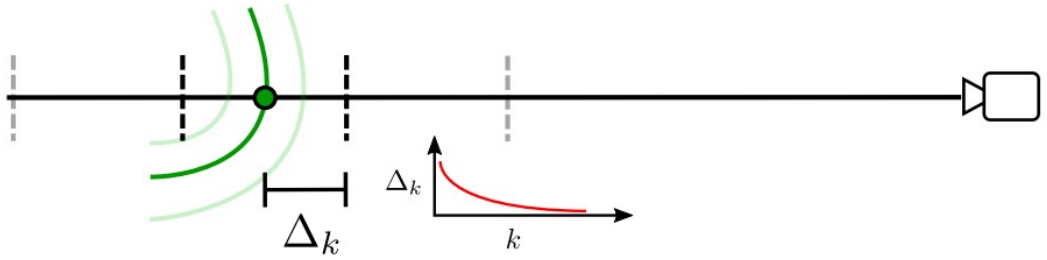


Oechsle, Peng, and Geiger: UNISURF: Unifying Implicit Surface Representations and Radiance Fields. ICCV, 2021.

Neural Distance Fields / UNISURF

Transition from Volume rendering to Surface rendering

- Exponential decay of interval Δ_k wrt. iterations k



Theorem

Volume and surface rendering become equivalent when reducing the interval and increasing the number of samples.

$$\lim_{\substack{\Delta \to 0 \\ N \to \infty}} \hat{C}_v(\mathbf{r}) = \hat{C}_s(\mathbf{r})$$

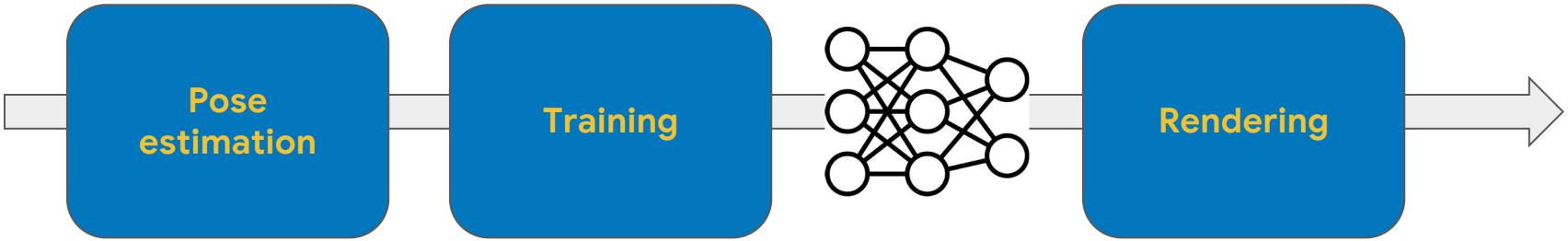
Oechsle, Peng, and Geiger: UNISURF: Unifying Implicit Surface Representations and Radiance Fields. ICCV, 2021.

A simple and powerful representation..

.. still limited for many real world applications

Nerf main limitations for real world applications

- **Pose estimation** is a critical step. Nerf requires dense pose coverage with accurate pose estimation. **Noise and sparsity** highly affect quality.
- **Rendering in real time is still a problem**, especially on embedded/mobile settings (e.g. smartphone)



SPARF: Better poses under sparse settings

RadSplat: real-time mobile rendering and novel view synthesis

Some Open Challenges

Noisy Camera Poses

"BARF"

Bundle-Adjusting Radiance Fields

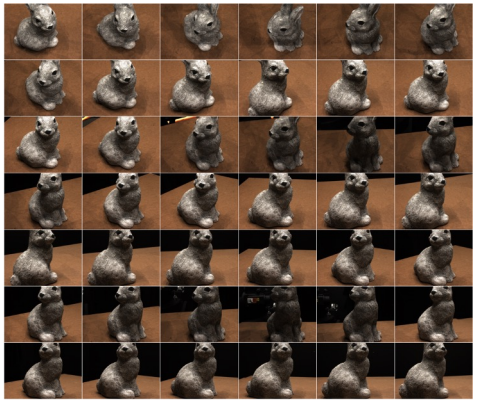
Only Few Images

"SPARF"

Sparse Pose-Adjusting Radiance Fields

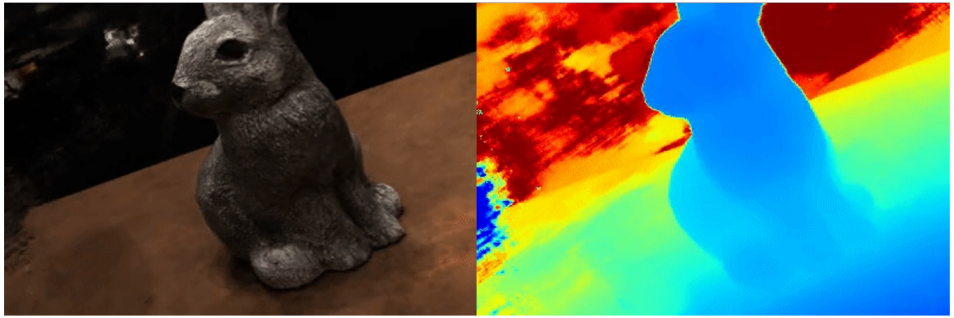
NeRFs: Challenge in the sparse-view setting

Input



+

Fixed **ground-truth** camera poses

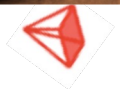


For realistic novel-view renderings, it requires:

- Lots of training images (dense coverage of the 3d space)
- Known and accurate camera poses for the training images

What happens??

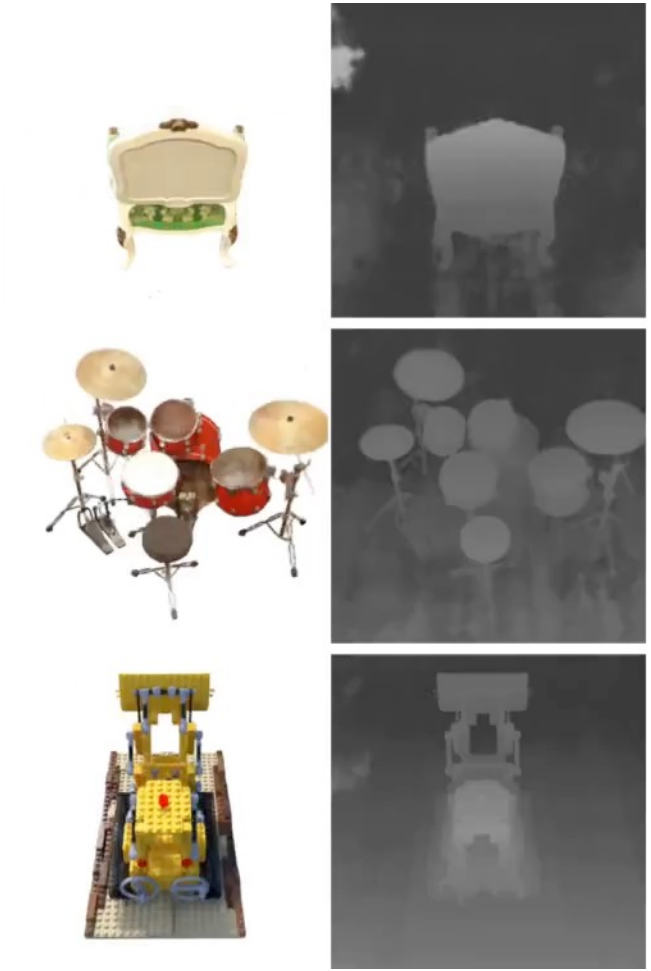
Input Sparse



Noisy camera poses

BARF

Images + **imperfect** **camera poses**

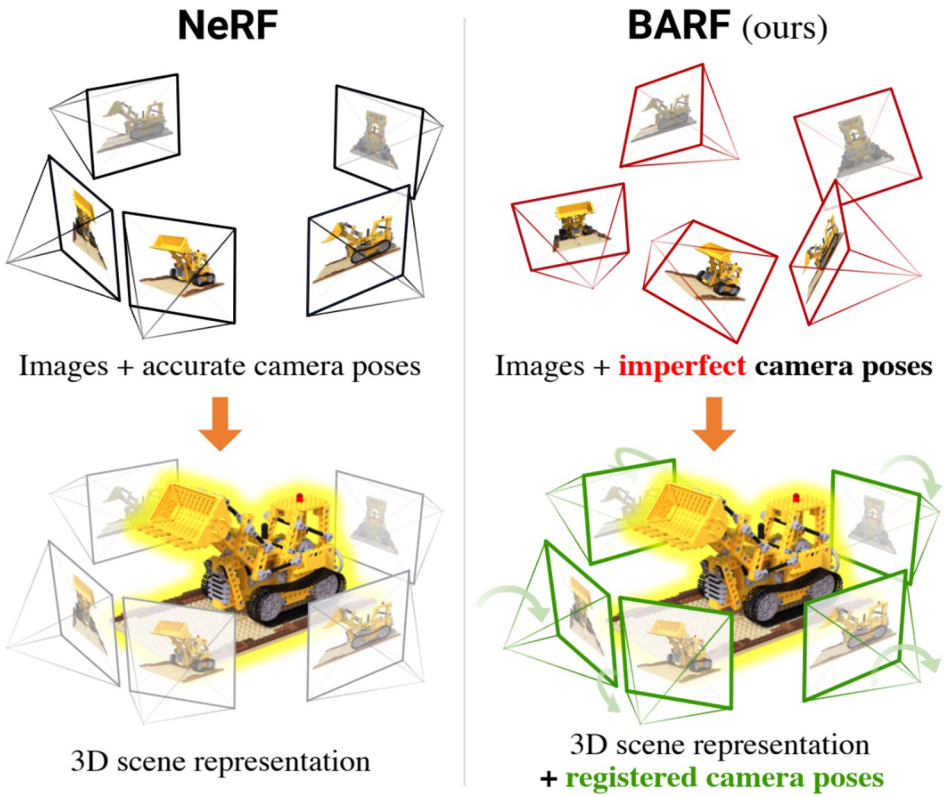


Neural Radiance Fields expect perfectly accurate camera poses

In the real world this is not always guaranteed, weaking the NeRF quality

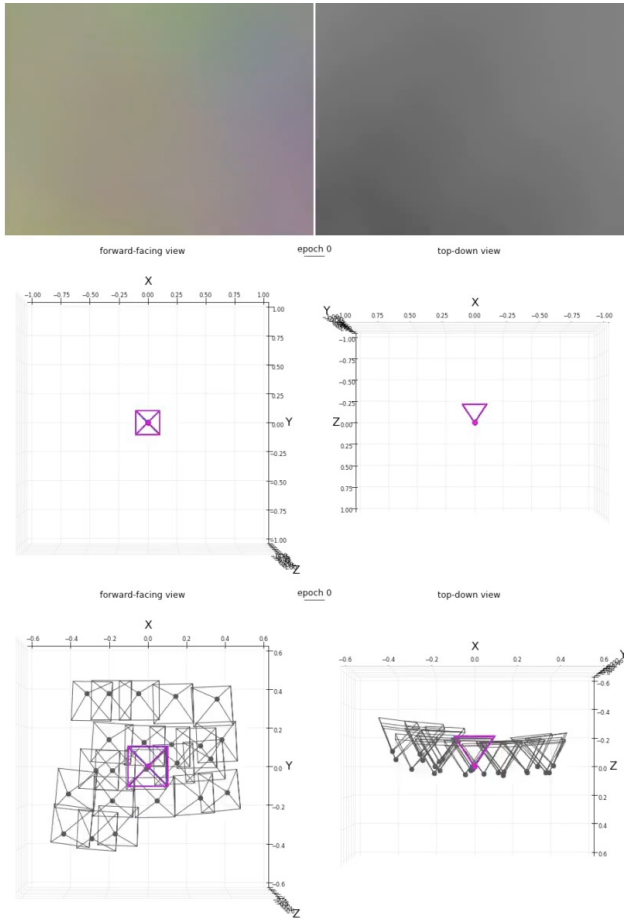
BARF

- Proposes to jointly finetune the camera poses within the NeRF
- This requires backpropagating the gradient through the ray casting up to the camera pose
- Follows a coarse to fine strategy to avoid too fast overfitting to a suboptimal solution



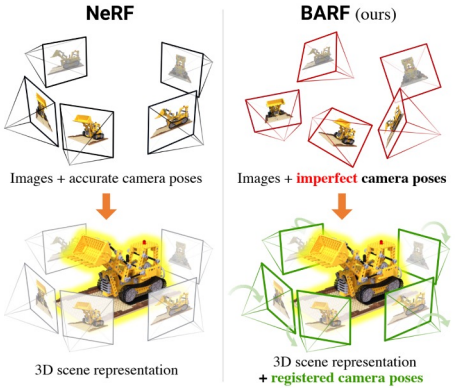
BARF

- Proposes to jointly finetune the camera poses within the NeRF
- This requires backpropagating the gradient through the ray casting up to the camera pose
- Follows a coarse to fine strategy to avoid too fast overfitting to a suboptimal solution



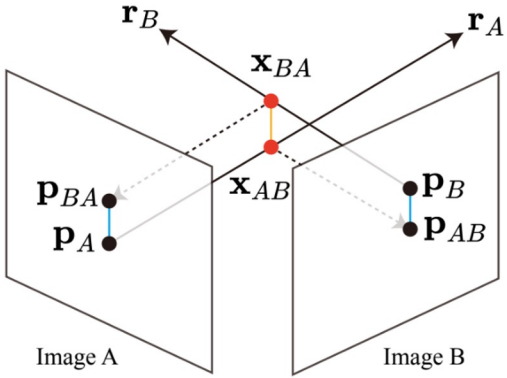
NeRFs from noisy poses

BARF (ICCV 2021) + Follow-ups



- Proposes to jointly finetune the camera poses with the NeRF
- Follows a coarse to fine strategy to avoid too fast overfitting to a suboptimal solution

SCNeRF (ICCV 2021)



- Proposes the *Projected Ray Distance loss* ⇒ Computes the intersection of the corresponding 2 rays, and measures the re-projection error.
- Proposed loss has a geometric basis but it impacts only the learnt poses

CamP (SIGGRAPH Asia 2023)

- CamP preconditions camera optimization in camera-optimizing Neural Radiance Field
- Proposes using a proxy problem to compute a whitening transform that eliminates the correlation between camera parameters and normalizes their effects

Some Open Challenges

Noisy Camera Poses

"BARF"

Bundle-Adjusting Radiance Fields

Only Few Images

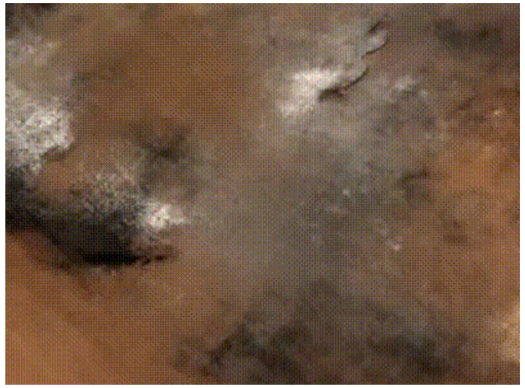
"SPARF"

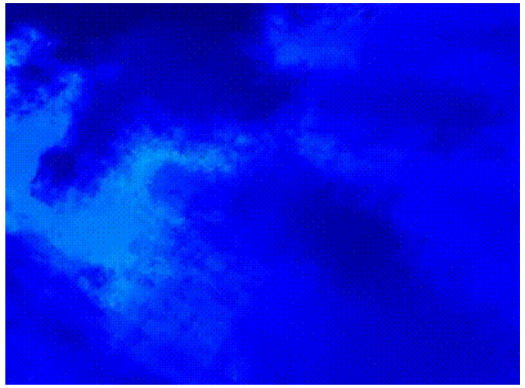
Sparse Pose-Adjusting Radiance Fields

NeRFs: Challenge in the sparse-view setting

Input

\+ Fixed **ground-truth** camera poses





Sparse input views with **fixed ground-truth poses**:

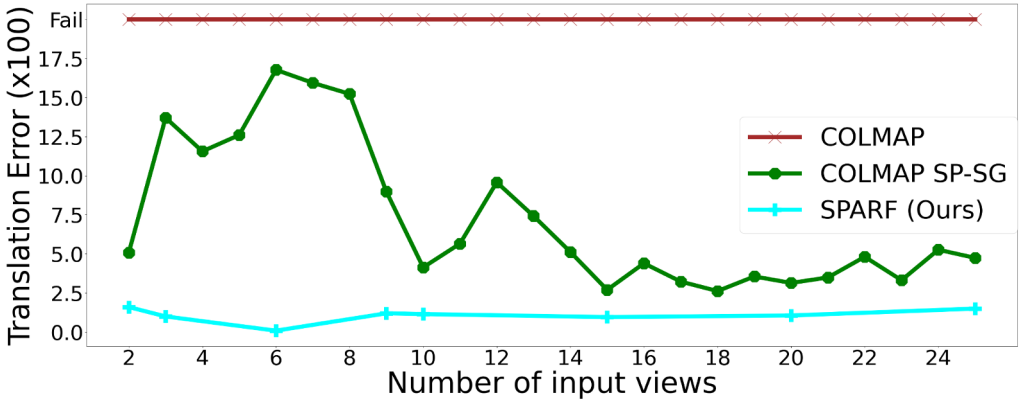
⇒ Overfit to the training views

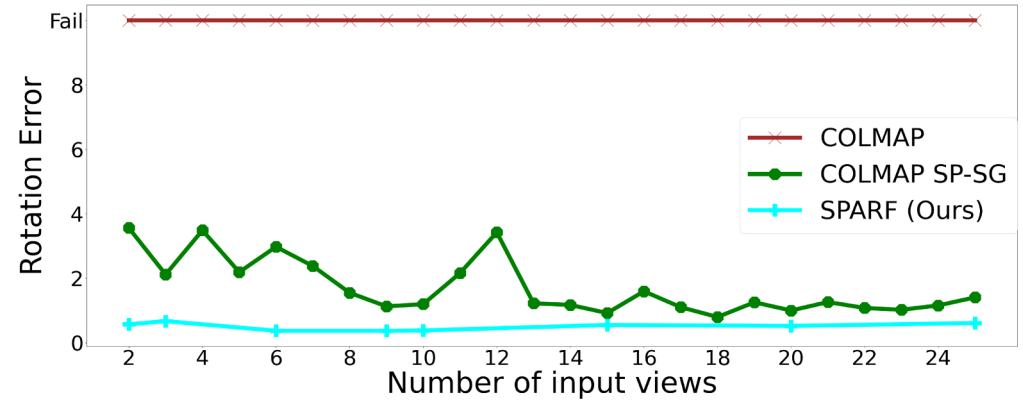
- Degenerate geometry
- Bad novel view rendering

NeRFs: Challenge in the sparse-view setting

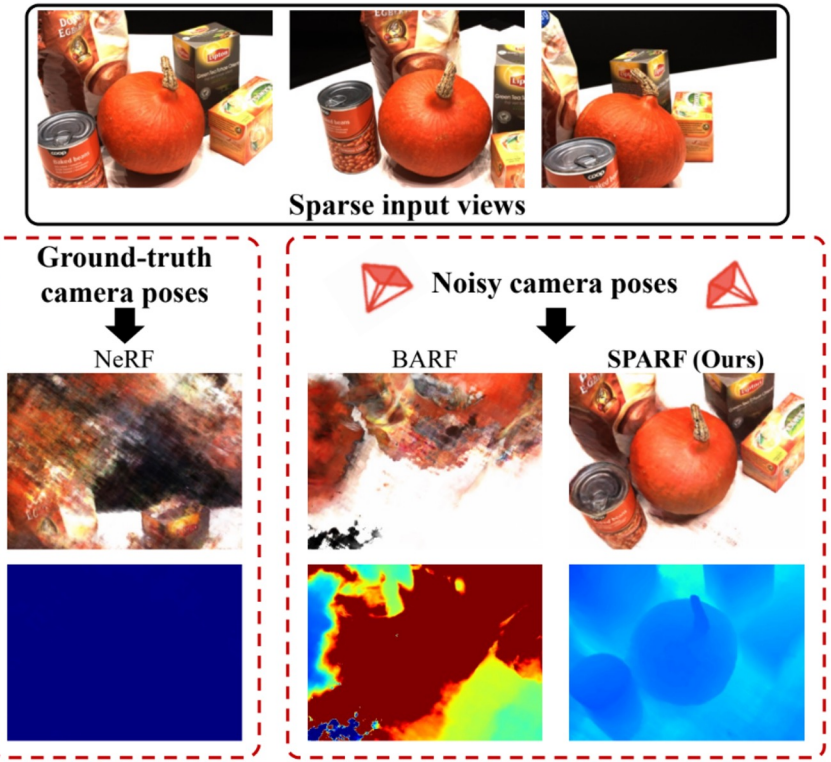
How do we get the poses?

- The standard is to use COLMAP, a structure-from-motion approach
- On few, wide-baseline images, COLMAP will most likely fail
- Even when using better matching, the performance of COLMAP degrades as the number of views decrease
- Pose errors will lead to errors in the learnt scene and therefore in the renderings





SPARF



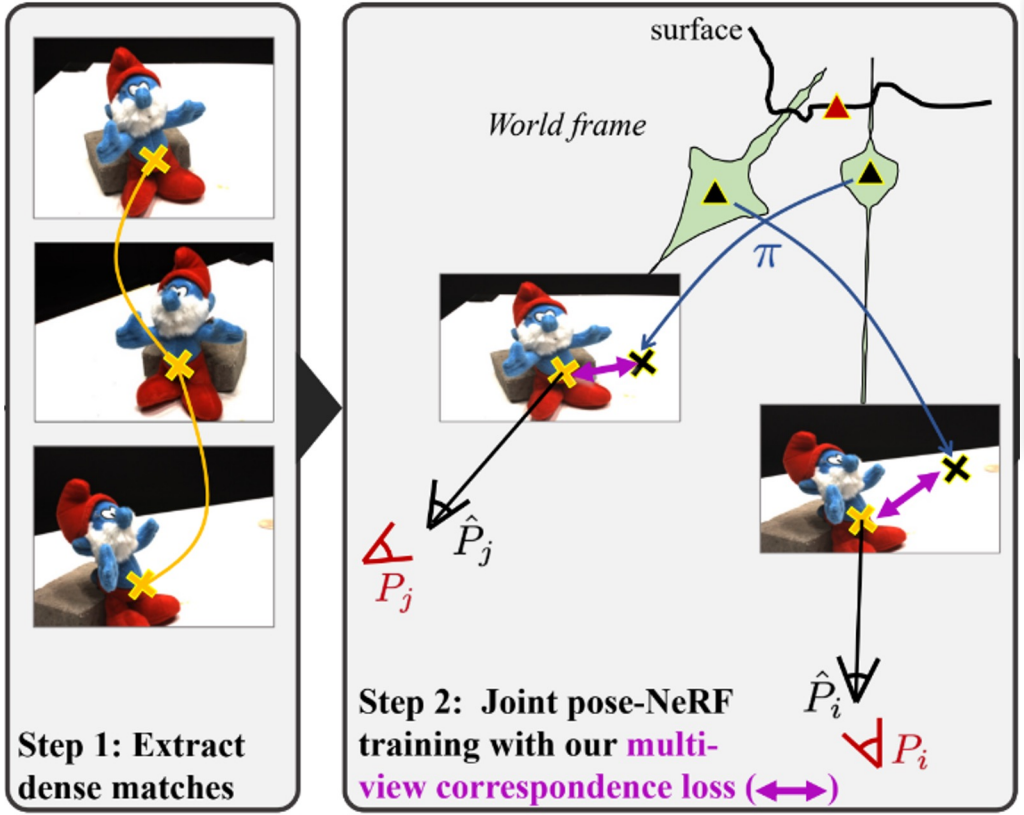
Main challenges:

- NeRF overfit to the few training images without learning a meaningful geometry
- Previous works use the photometric loss. It is applied to each image independently

We propose a joint pose-NeRF training strategy.

We add **two additional constraints into the NeRF optimization, which rely on multi-view geometry principles.**

Multi-view correspondence loss



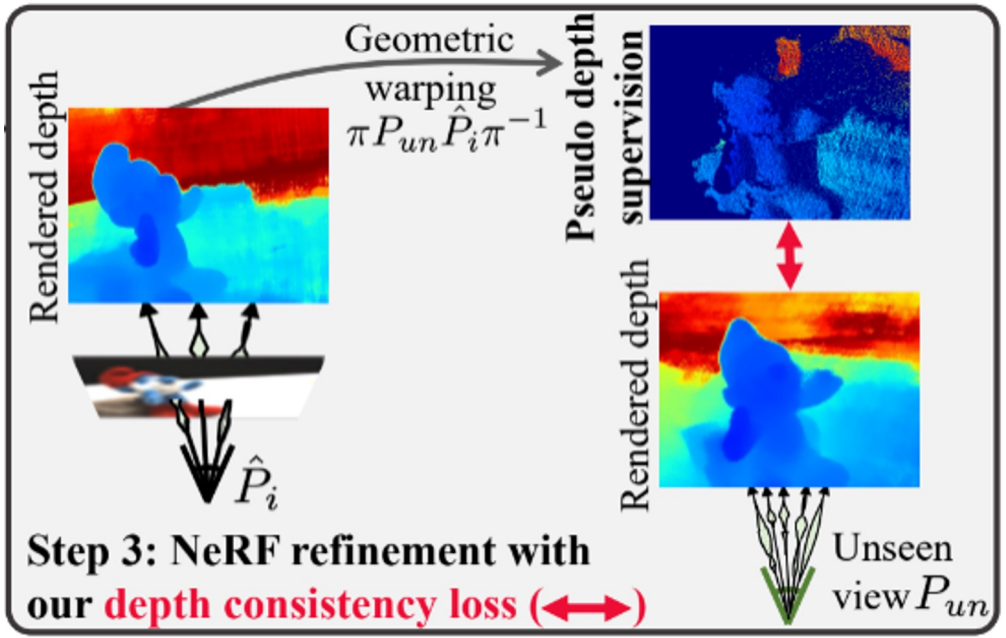
Goal:

- Geometrically connect the training images to convergence to a **globally consistency 3D solution** over poses and geometry.
- Direct supervision on **rendered depth ⇒ should be close to the real surface**

Parameters optimized
- Camera pose estimates
- Density prediction (F_θ)

- GT camera poses
- π Projection operator
- Matching pixels
- GT 3D point (corresponding to ✕)
- "Rendered" 3D point
- Pixel projection

Depth-consistency loss



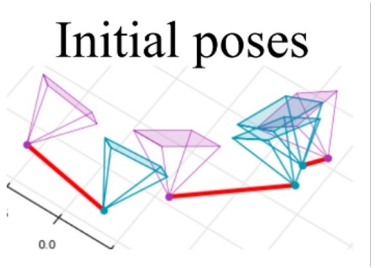
Goal: Ensure the reconstructed scene is **consistent from any viewing directions**, including the ones without RGB supervision.

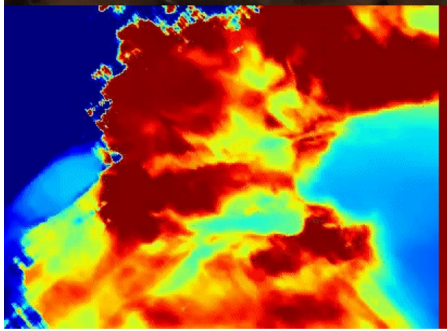
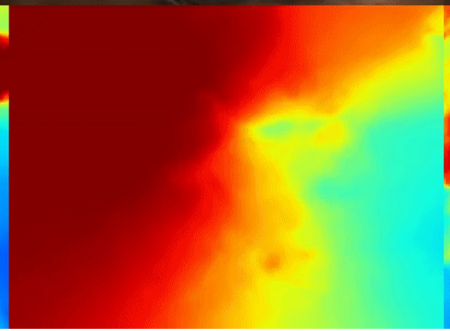
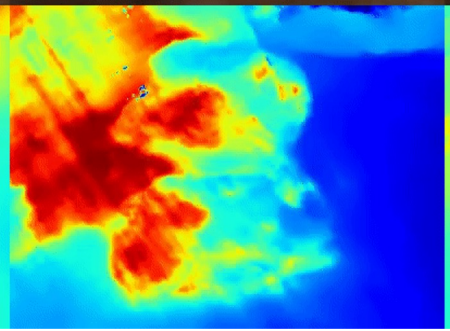
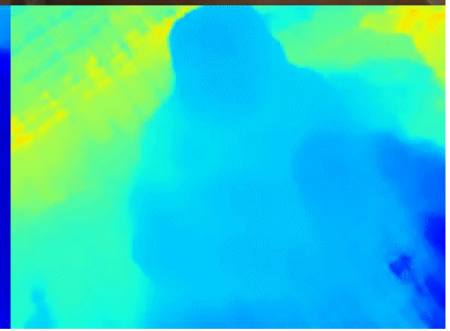
Main idea: Use the rendered depth from the training viewpoints to create pseudo-depth supervision for novel, unseen viewpoints.

We also include a visibility mask, to tackle occlusion.

This optimizes over the neural radiance field weights only. The poses are fixed here.

Inputs:



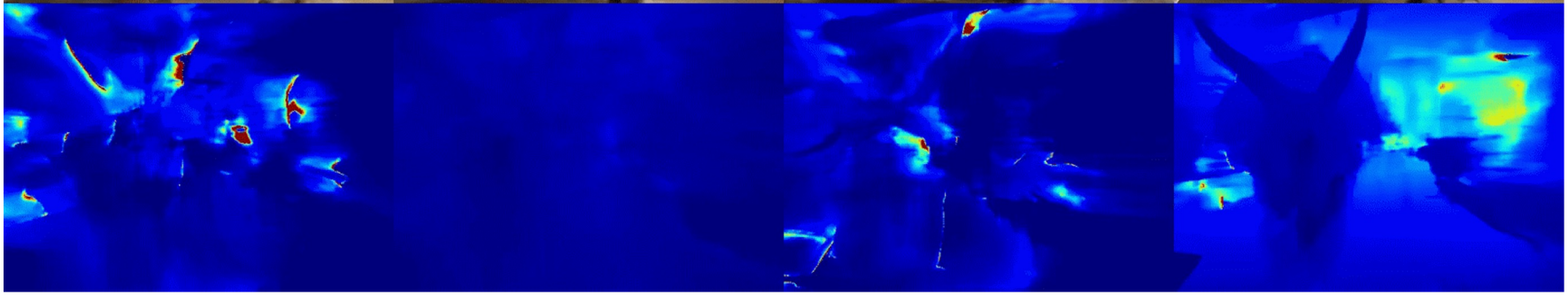
	BARF	RegBARF	SCNeRF	**SPARF (Ours)**
RGB				
Depth				

Inputs:

BARF | RegBARF | SCNeRF | **SPARF (Ours)**

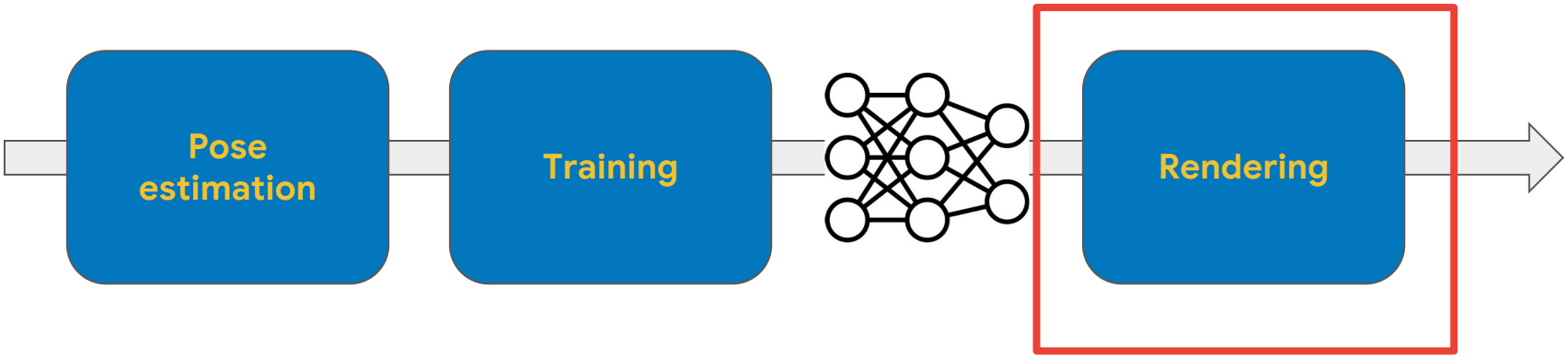
RGB

Depth



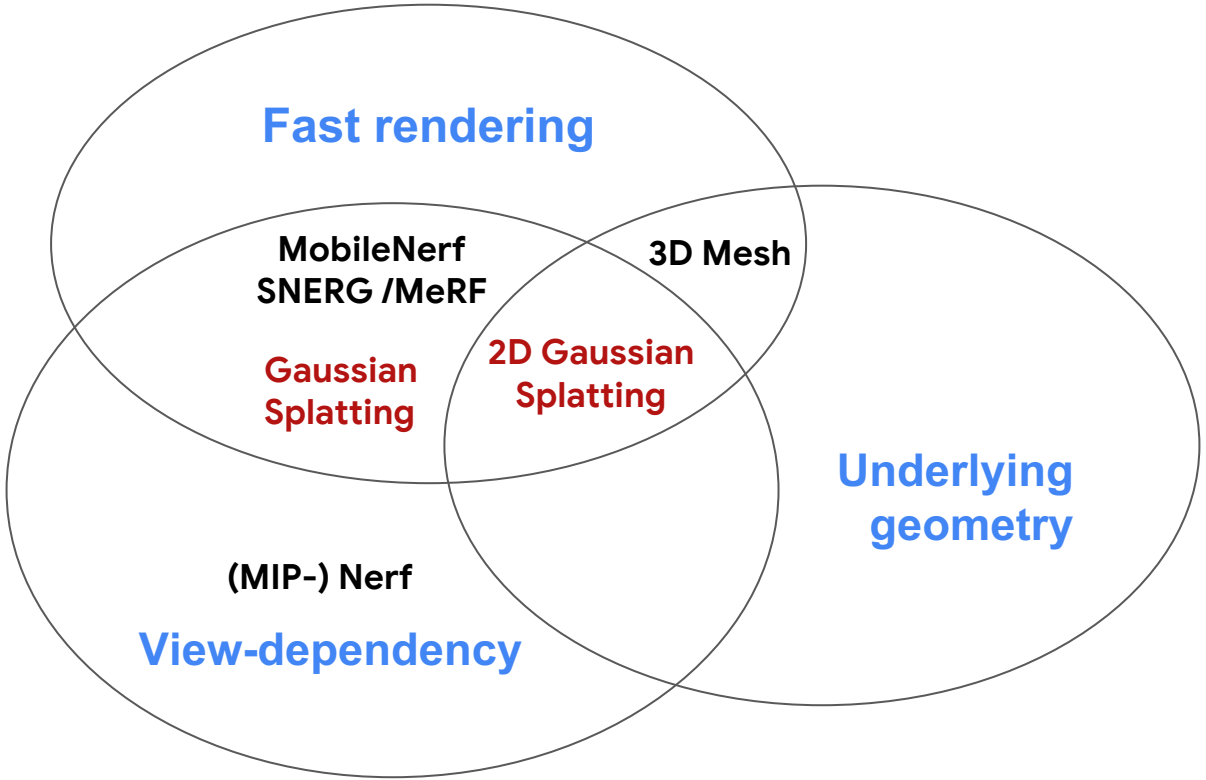
Limitations for real applications

- **Pose estimation** is a critical step. Nerf requires dense pose coverage with accurate pose estimation. **Noise and sparsity** highly affect quality.
- **Rendering in real time is still a problem**, especially on embedded/mobile settings



3DGS: 3D Gaussian Splatting

Nerf rendering dilemma



3D Gaussian Splatting

3DGS represents a scene with a set of 3D gaussians

$$C = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j),$$

α_i: blending weight given covariance Σ and opacity σ
c_i: color of the gaussians given its SH coefficients k and the viewing direction

Kerbl, Bernhard, et al. "3d gaussian splatting for real-time radiance field rendering." ACM Transactions on Graphics 42.4 (2023): 1-14.

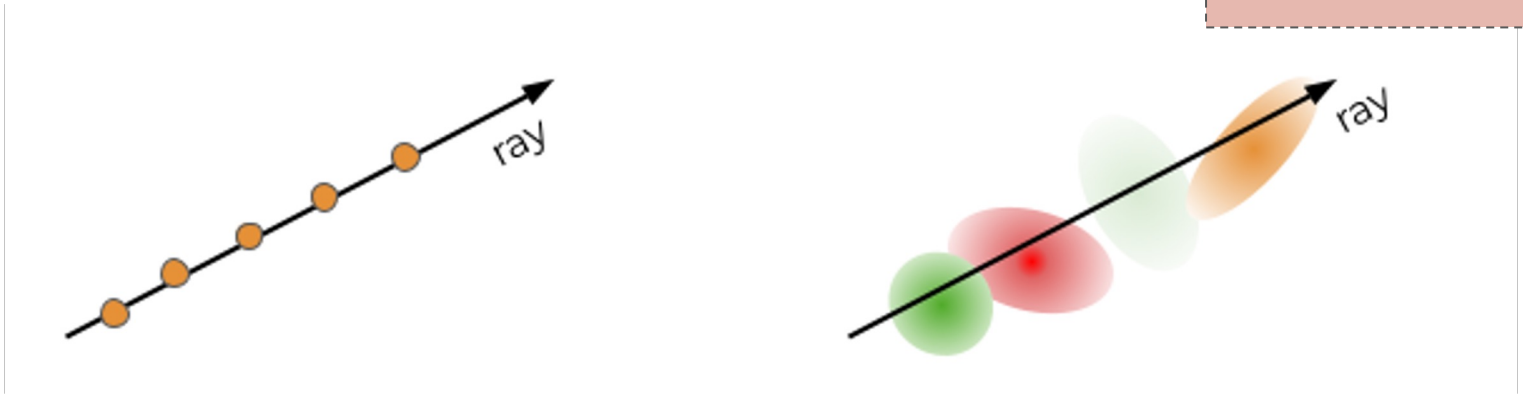
Each Gaussian being defined by a:

- 3D position $\mathbf{p} \in \mathbb{R}^3$,
- Covariance matrix $\mathbf{\Sigma} = \mathrm{RSS}^{\mathrm{T}}\mathrm{R}^{\mathrm{T}}$, defined by a
  - 3D rotation $\mathrm{R} \in \mathrm{SO}(3)$
  - 3D scale $\mathrm{s} \in \mathbb{R}^3$
- Opacity $\sigma \in [0, 1]$
- Third-degree **spherical harmonics (SH)** coefficients $\mathrm{k} \in \mathbb{R}^{16}$

3D Gaussian Splatting *vs.* Volumetric Renderin

Secret of NeRFs:

Differentiable Everywhere



$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i$$

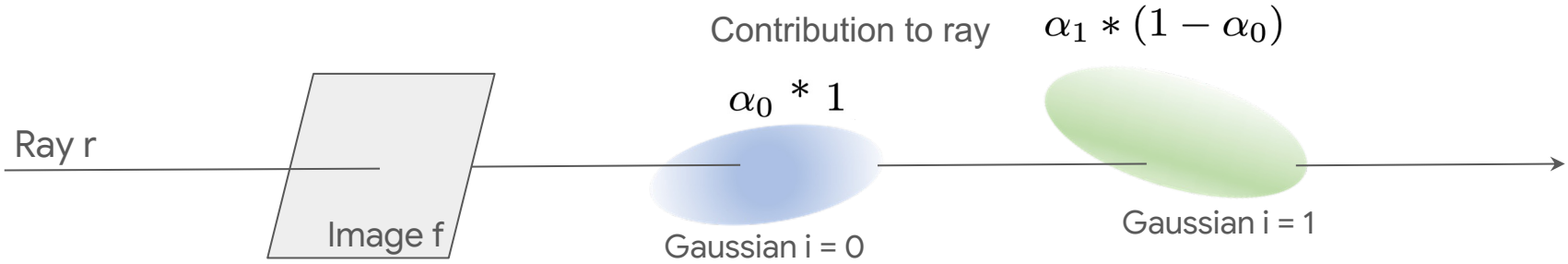
$$C = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1}(1 - \alpha_j),$$

T_i: transmittance of sampled point

α_i: blending weight given covariance and transparency

$\delta_i = t_{i+1} - t_i$: distance between sampled points

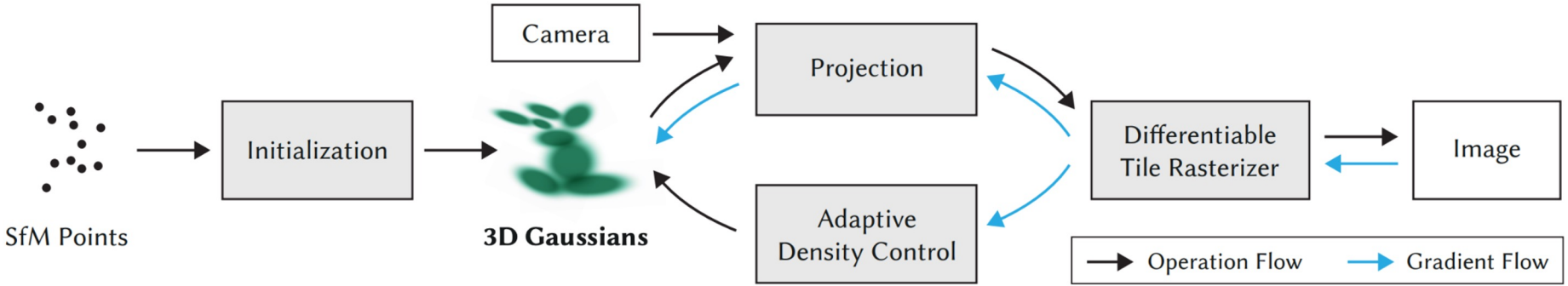
Blending of Gaussians



$$C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j),$$

α_i: blending weight given covariance and transparency

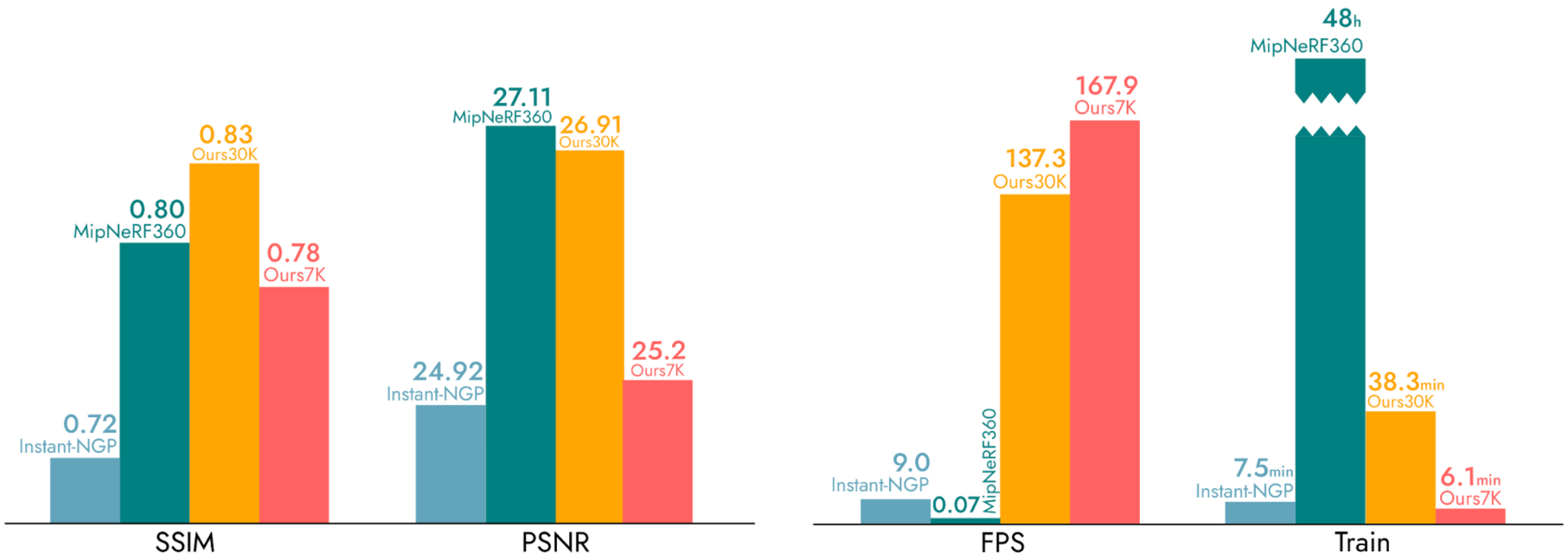
3D Gaussian Splatting



3DGS represents a scene with a set of 3D gaussians with a 3D mean, covariance matrix, size, and transparency together with a RGB color value and set of Spherical Harmonics for view dependency

- For rendering, 3DGS "splats" the 3D gaussians to 2D given the viewing direction following *Zwicker et al. SIGGRAPH 2021*
- Different to NeRFs, **3DGS does not conduct volumetric rendering** but **instead** follows **standard rasterization with alpha blending**
- To avoid over or under representing the scene an **adaptive density control** mechanism is employed, **"splitting" and "cloning" gaussians based on their view-space positional gradients** and **removing very transparent gaussian**

NeRF vs Gaussian Splatting

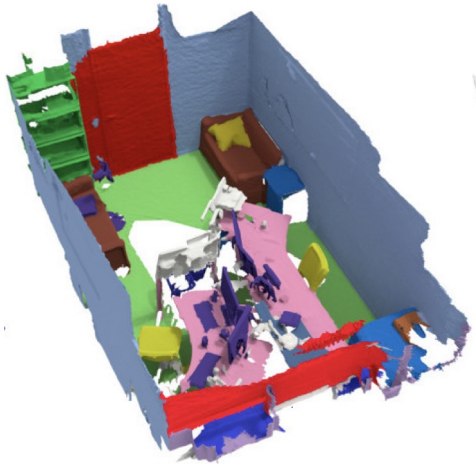


3DGS Interactive Demo

Implicit Semantic Understanding

Semantics for 3D Scene Understanding

Typical Tasks



3D Semantic Segmentation

Assign a semantic class to each point in a given 3D scene.

3D Instance Segmentation

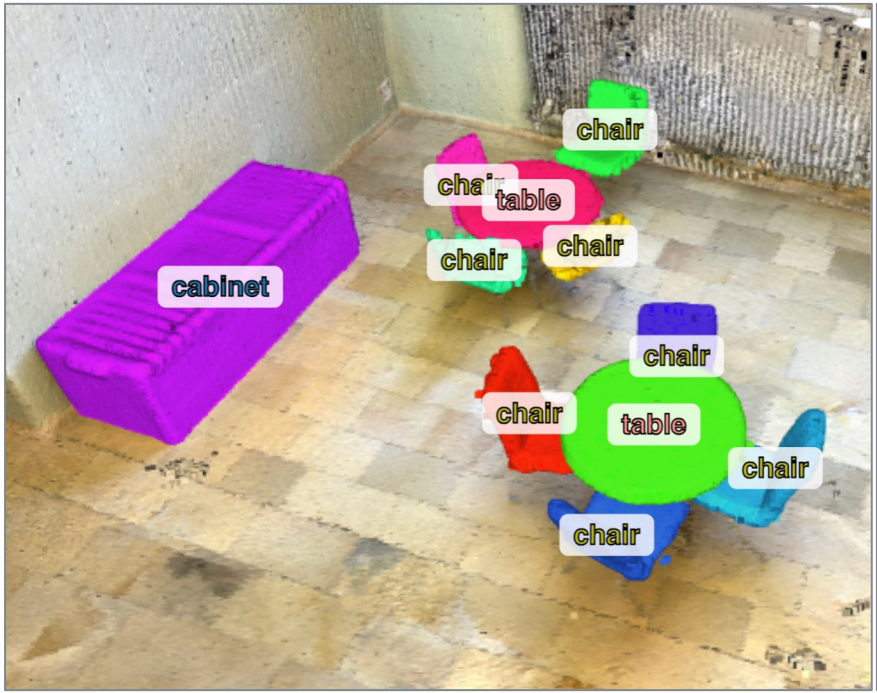
Predict instance masks and semantic labels for each object in a given 3D scene.

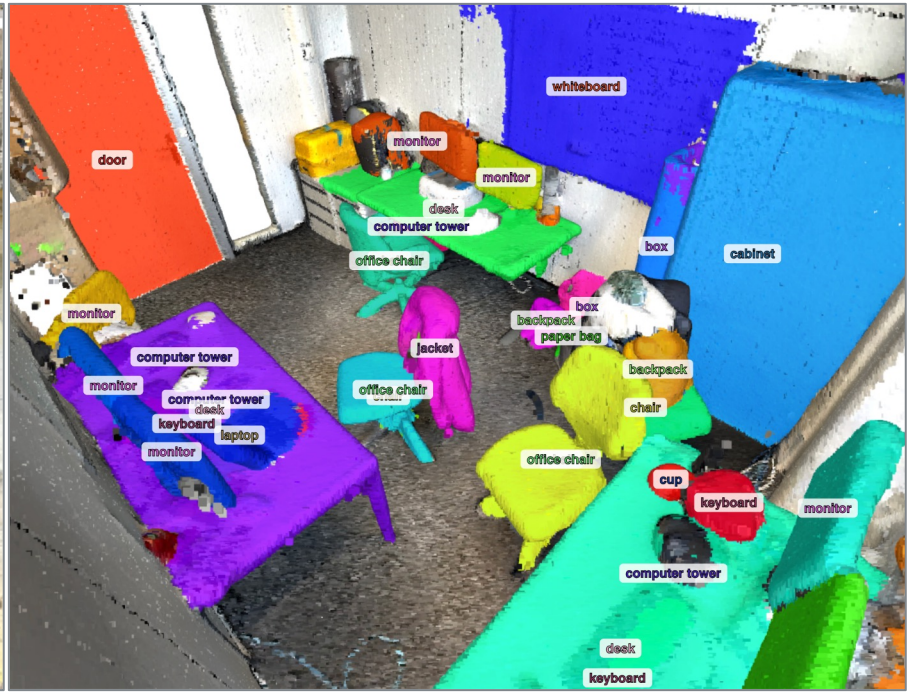
3D Object Detection

Detect the 3D bounding box of each object in a given 3D scene.

3D Scene Understanding

Example 3D Instance Segmentations from **Mask3D** [1]

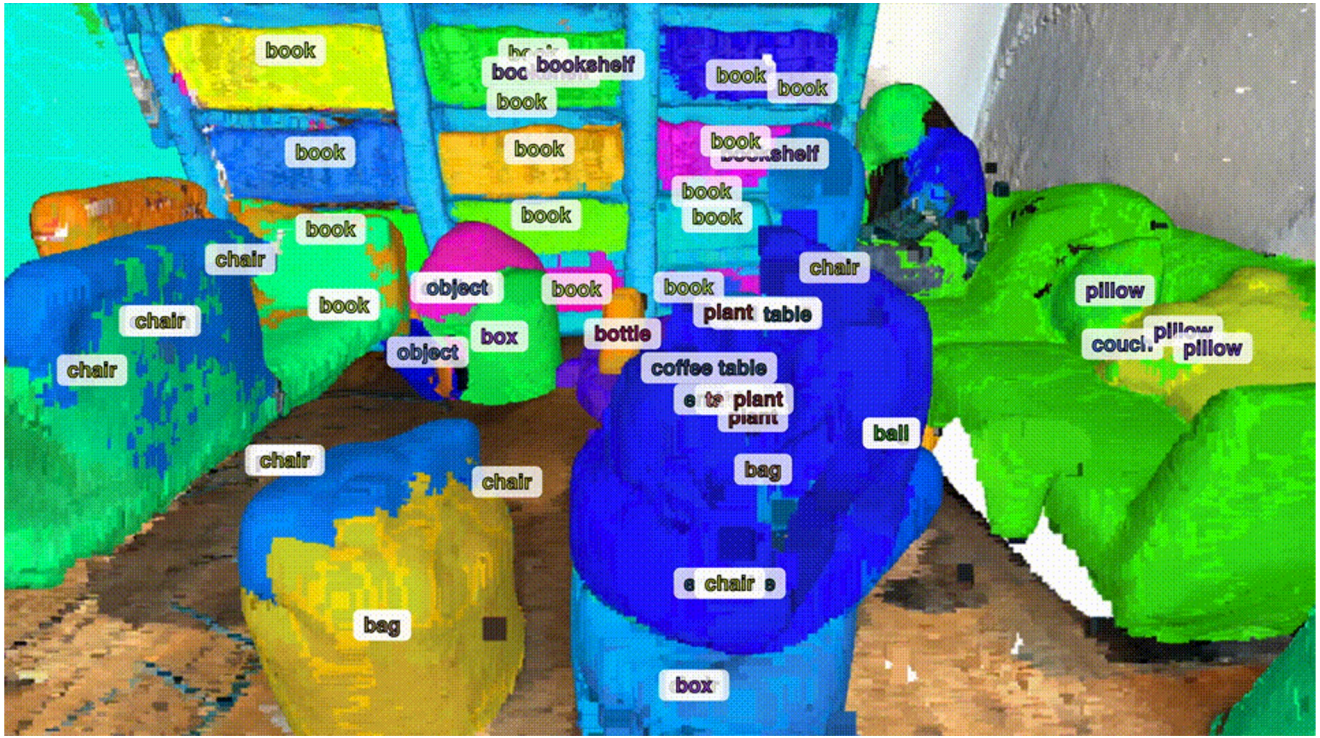




[1] Schult et al. "*Mask3D: Mask Transformer for 3D Instance Segmentation*" ICRA'23

3D Scene Understanding

Example 3D Instance Segmentations from **Mask3D** [1]



Input: 3D Point Cloud

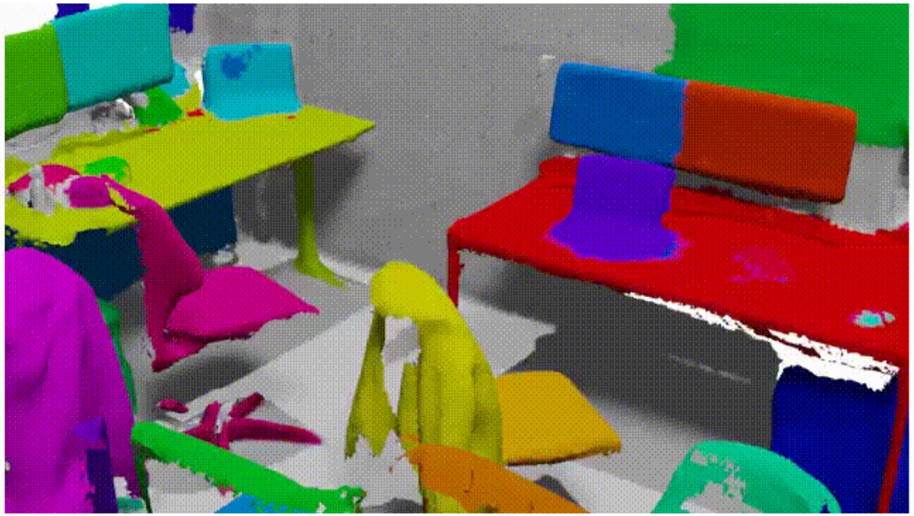
Output: 3D Semantic Instances

[1] Schult et al. "*Mask3D: Mask Transformer for 3D Instance Segmentation*" ICRA'23

3D Scene Understanding

Works well for semantic classes seen during training (closed-world setting)

Input 3D Scene

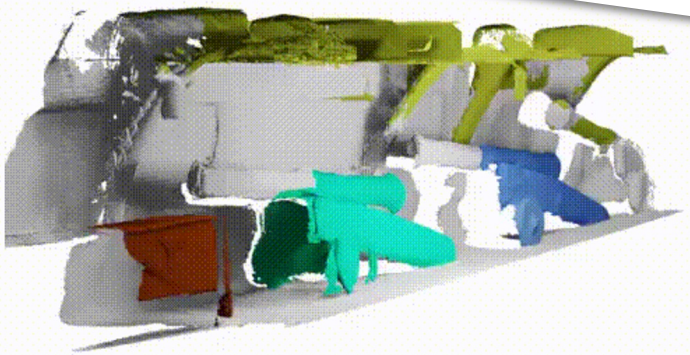


Predicted 3D Instance Masks

[1] Schult et al. "*Mask3D: Mask Transformer for 3D Instance Segmentation*" ICRA'23

3D Scene Understanding: *Limitations of **Closed-Set** Assump*

Limitations of **closed-world** assumption
Example "*in-the-wild*" scene (link)



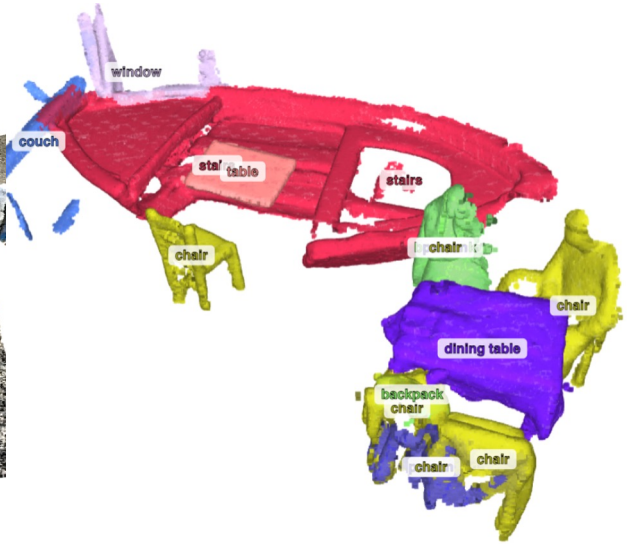
Input 3D Scene

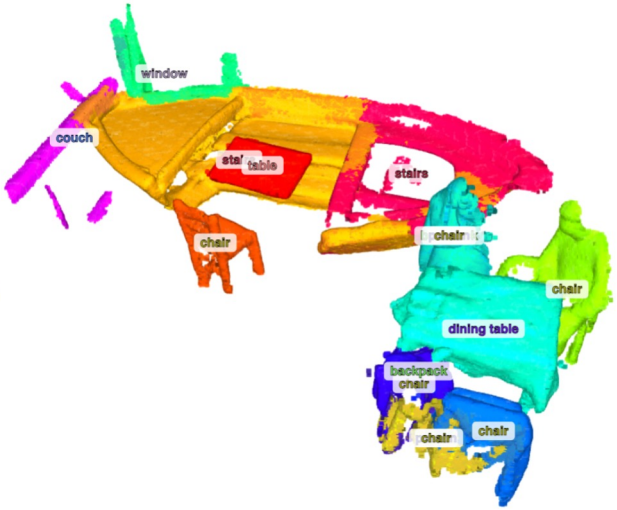
3D Semantics

Ceiling Nightstand Bench Couch

3D Scene Understanding: *Limitations of **Closed-Set** Assumption*

Limitations of **closed-world** assumption
Example "*in-the-wild*" scene (link)





Input 3D Scene **3D Semantics** **3D Instance Masks**

Couch Window Stairs Chair
Table Dining Table Backpack

Open-Vocabulary 3D Scene Understanding

Use Visual-Language-Model (VLM) to query scene.

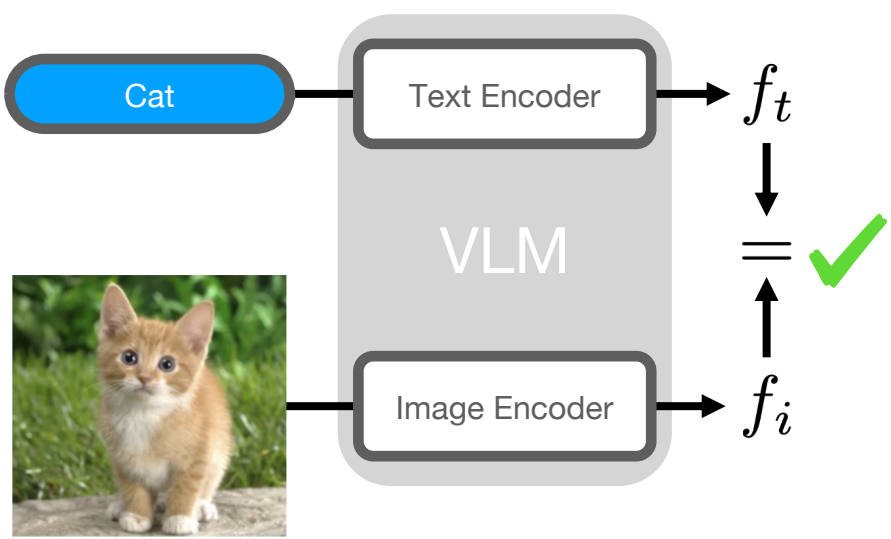
Mechanism for zero-short image segmentation:

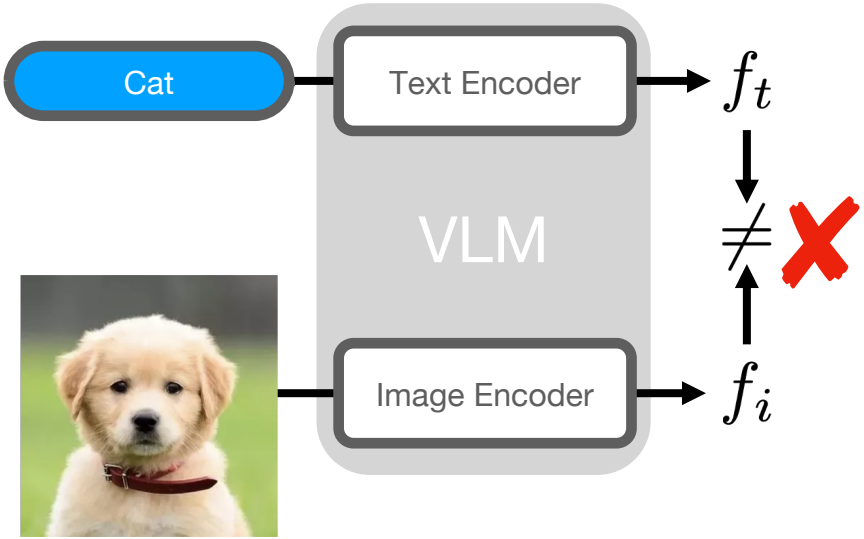
1. Compute CLIP [1] encoding of text query and per-pixel CLIP features via OpenSeg [2]
2. Get response from dot-product of normalized encodings

[1] Radford et al. "*Learning Transferable Visual Models From Natural Language Supervision*" ICML'21
[2] Ghiasi et al. "*Scaling open-vocabulary image segmentation with image-level labels*" ECCV'22
[3] Engelmann et al. "OpenNeRF" ICLR'24

How can we achieve Open-Vocabulary 3D Scene Understanding?

Large Visual Language Model (VLM) ***for example CLIP[1]***





[1] Radford et al. "*Learning Transferable Visual Models From Natural Language Supervision*" ICML'21

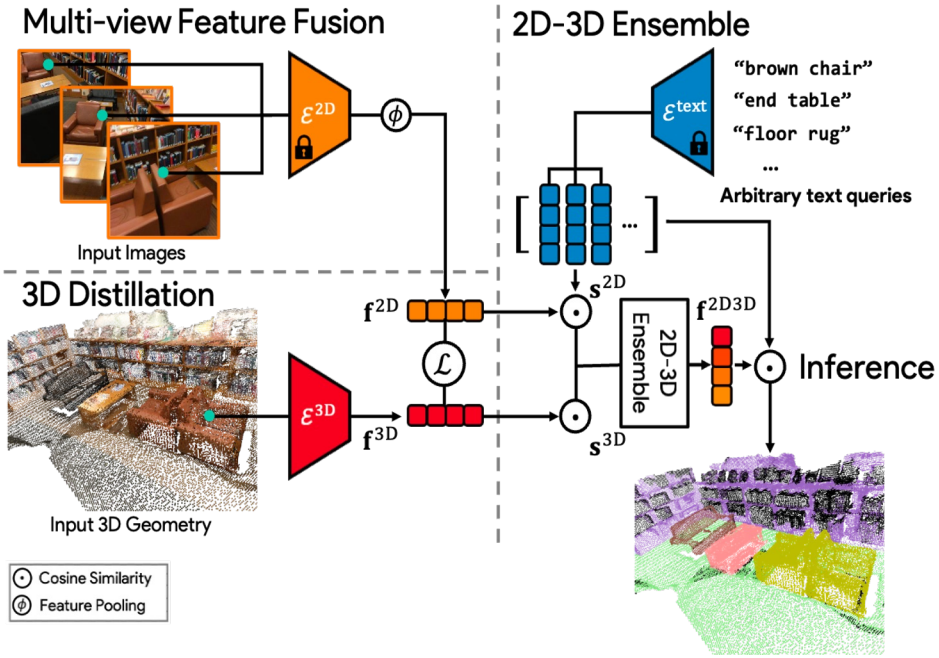
Open-Set 3D Semantic Segmentation

OpenScene: 3D Scene Understanding with Open Vocabularies

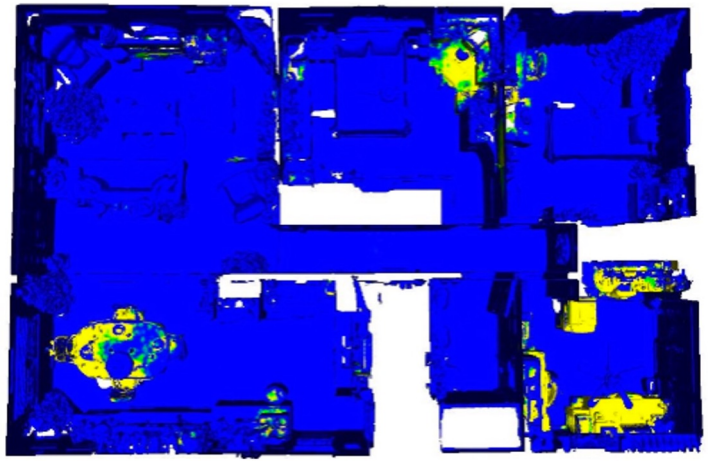
How do we transfer **open-set** scene understanding to **3D** scenes?

OpenScene [3] obtains **per-pixel** multi-view open-set features from LSeg [1] or OpenSeg [2] and projects them onto **3D points** of the scene point cloud.

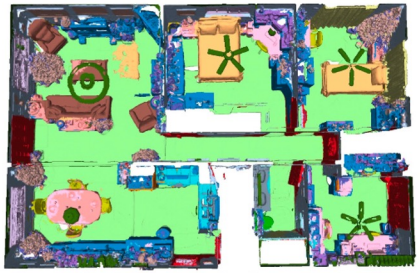
A sparse 3D CNN is then trained to predict **per-point** open-set features distilled from 2D ones. 2D and 3D features are ensembled via CLIP supervision, resulting in a scene with associated per-point open-set features.



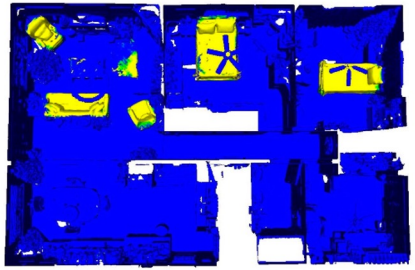
[1] Li et al. "*Language-driven Semantic Segmentation*" ICLR'22
[2] Ghiasi et al. "*Scaling open-vocabulary image segmentation with image-level labels*" ECCV'22
[3] Peng et al. "*OpenScene: 3D Scene Understanding with Open Vocabularies*" CVPR'23



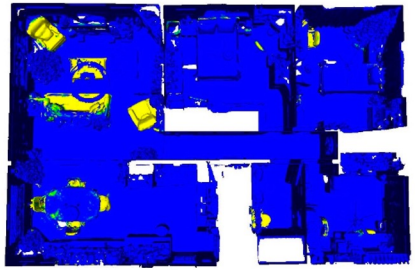
“work” - Activity



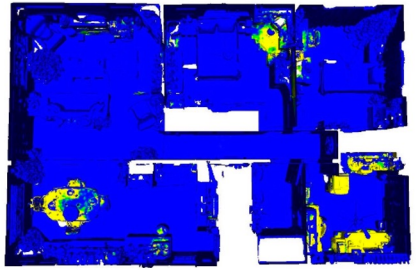
Zero-shot Semantic Segmentation



“anything soft” - Property

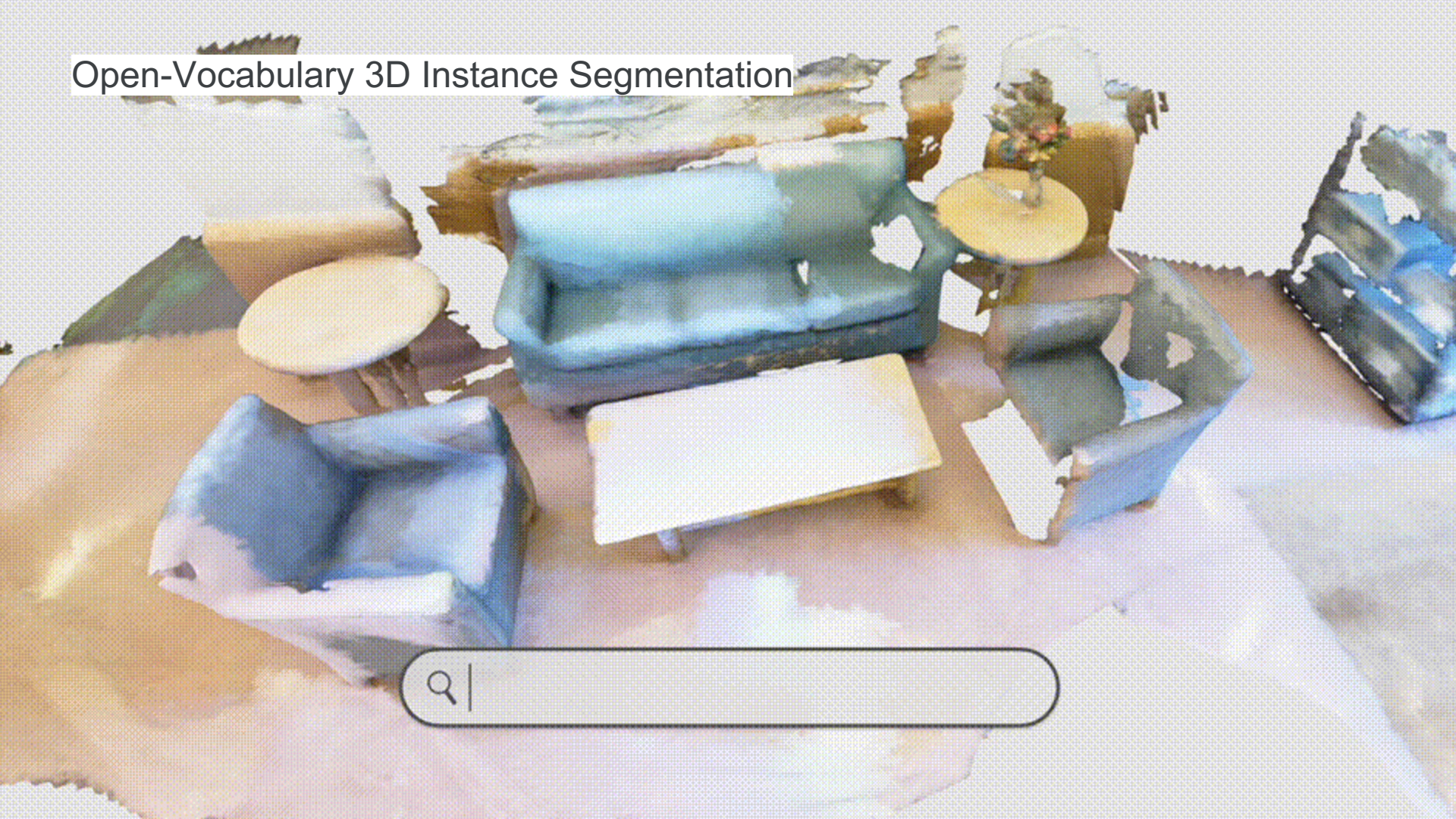


“where to sit” - Affordance



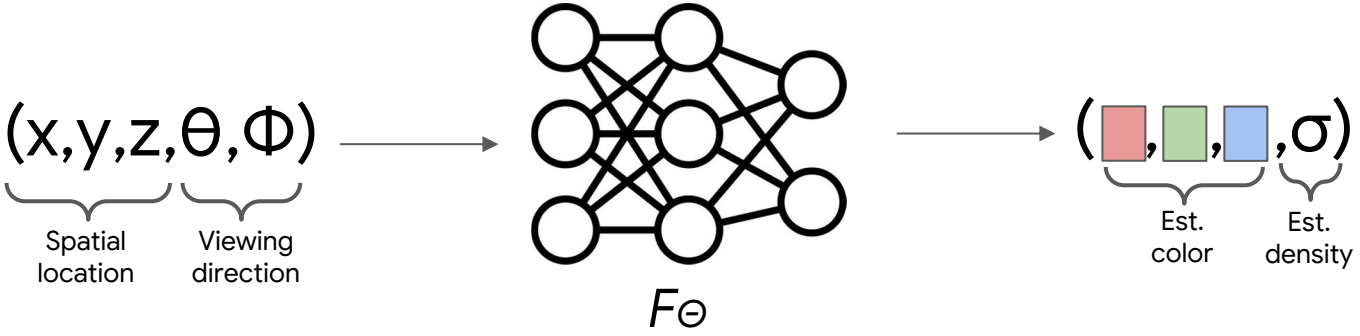
“work” - Activity

Open-Vocabulary 3D Instance Segmentation



What about open set with neural representations?

Nerf



- The network is a simple ReLU MLP that maps from location/view direction to color/density
- Density σ describes how solid/transparent a 3D point is (can model, e.g., fog)
- Conditioning on view direction allows for modeling **view-dependent effects**

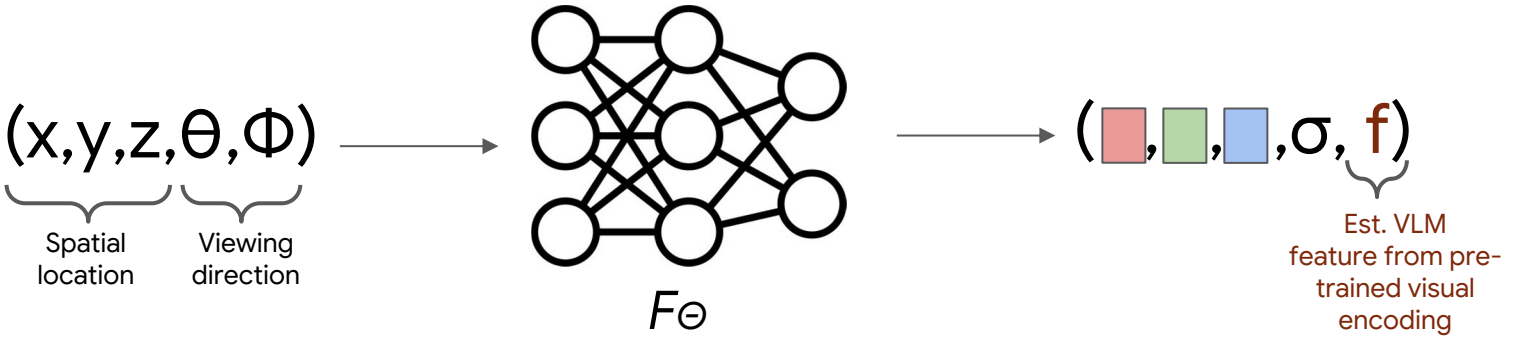
Input: posed images (no explicit 3D geometry or depth)
Task: Novel View Synthesis

Open-Set 3D Scene Understanding using **Implicit** Representations

Explicit v.s. Implicit Representations: Polygon Meshes and Point Clouds or NeRF Representations?

Can we use NeRF representations for Open-Set 3D Scene Understanding?

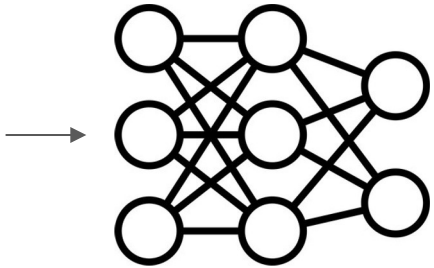
Idea: Ground CLIP features (or any other features from a pre-trained visual encoding aligned with language) volumetrically inside NeRFs (in addition to color and density).



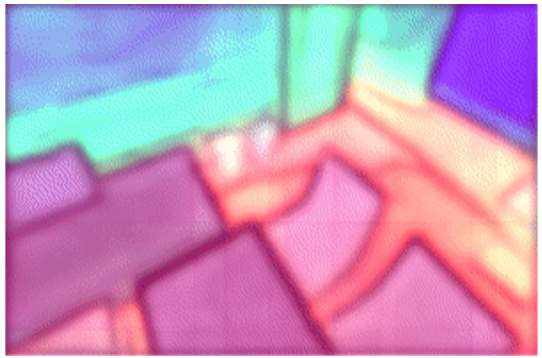
[1] Kerr et al. "LERF: Language Embedded Radiance Fields" ICCV'23
[2] Engelmann et al. "*Open-Set 3D Scene Segmentation with Rendered Novel Views*", arXiv'23

Open-Set 3D Scene Understanding using **Implicit** Representations

Posed RGB video



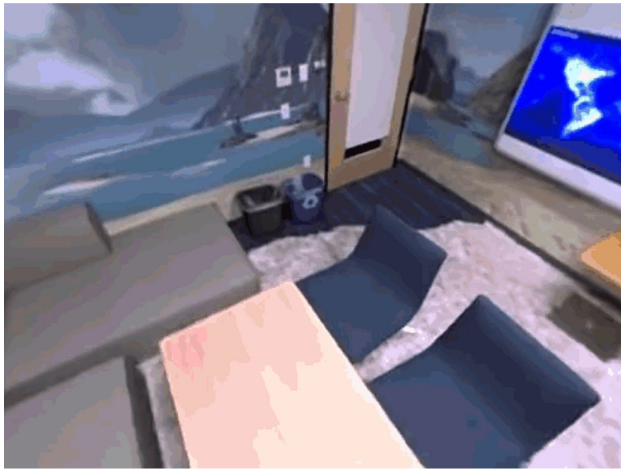
Trained LSeg / CLIP Model



LSeg / CLIP features

Training Data Generation

Novel view rendering



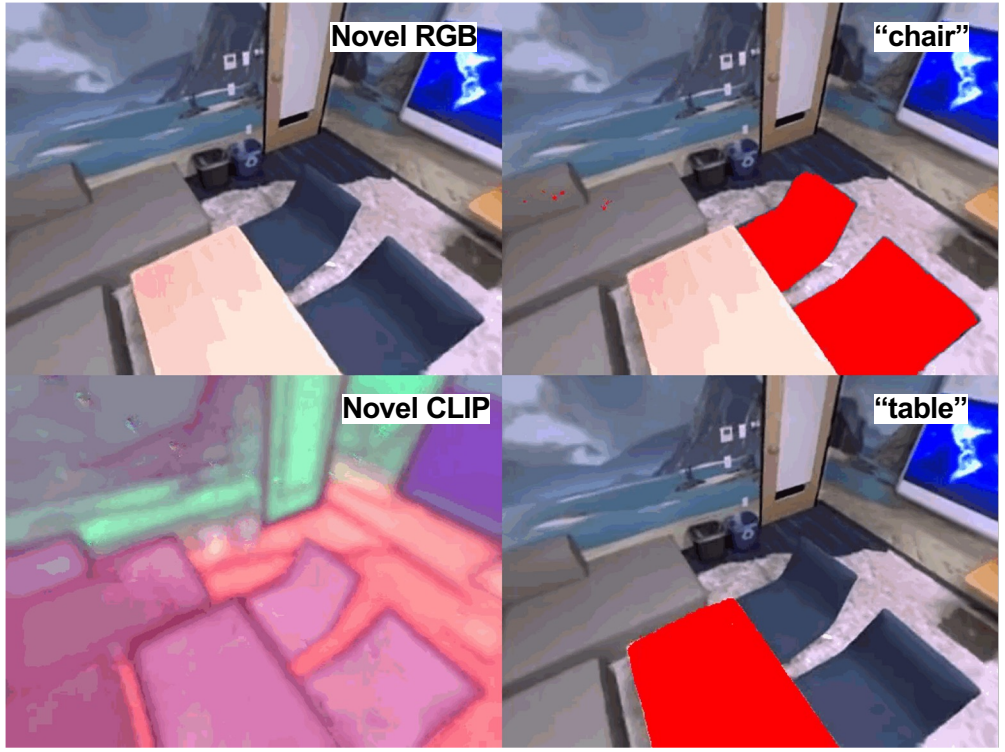
NeRF Rendered
Trajectory

NeRF for Open-Set 3D Scene Understanding

Posed RGB video

LSeg / CLIP features

LSeg is trained on ADE20K Dataset [(150 classes)]

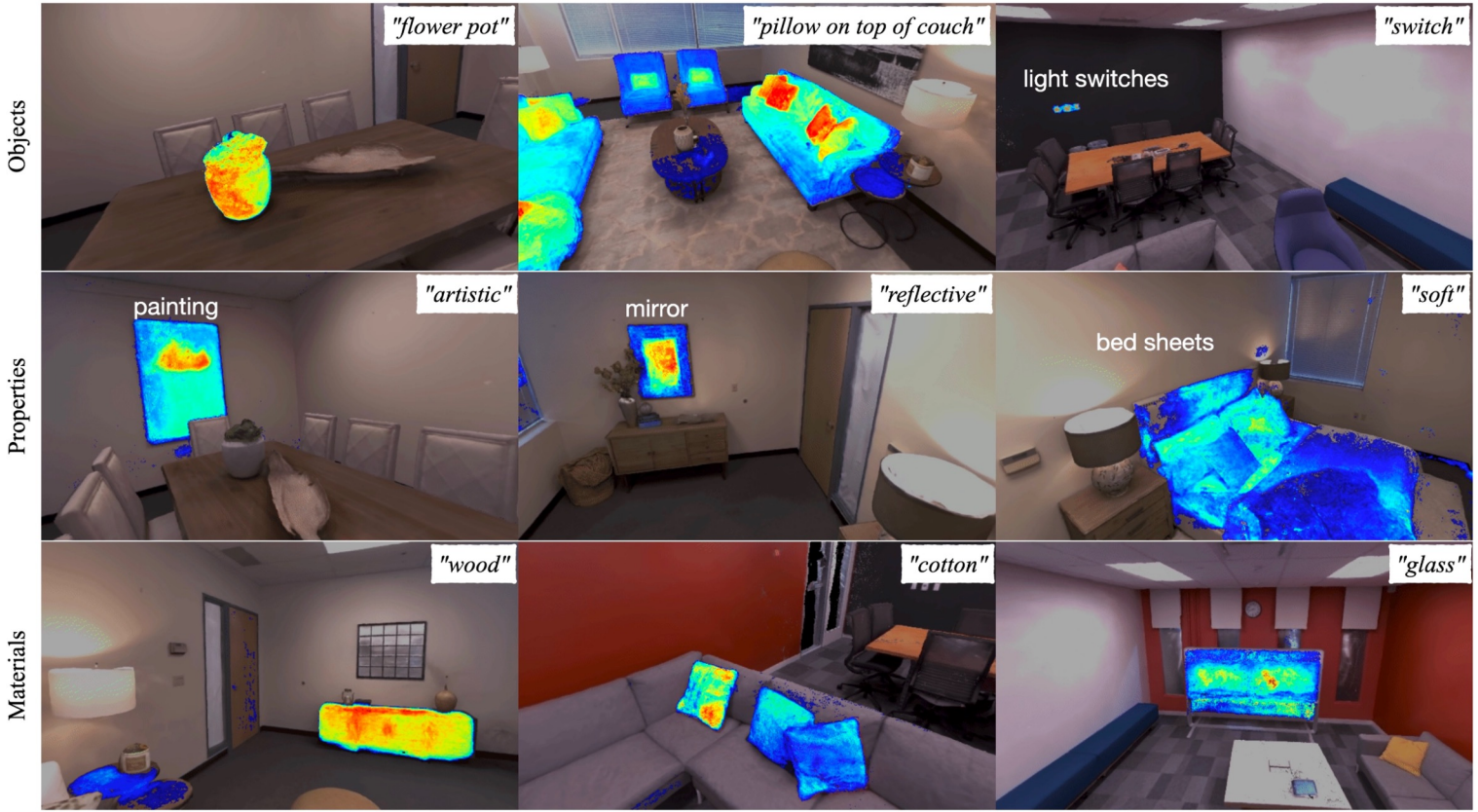


Novel views RGB (top), CLIP (bottom)

Queries: “chair” (top), “table” (bottom)

OpenSet 3D Scene Understanding: *Object, Properties, Materials*

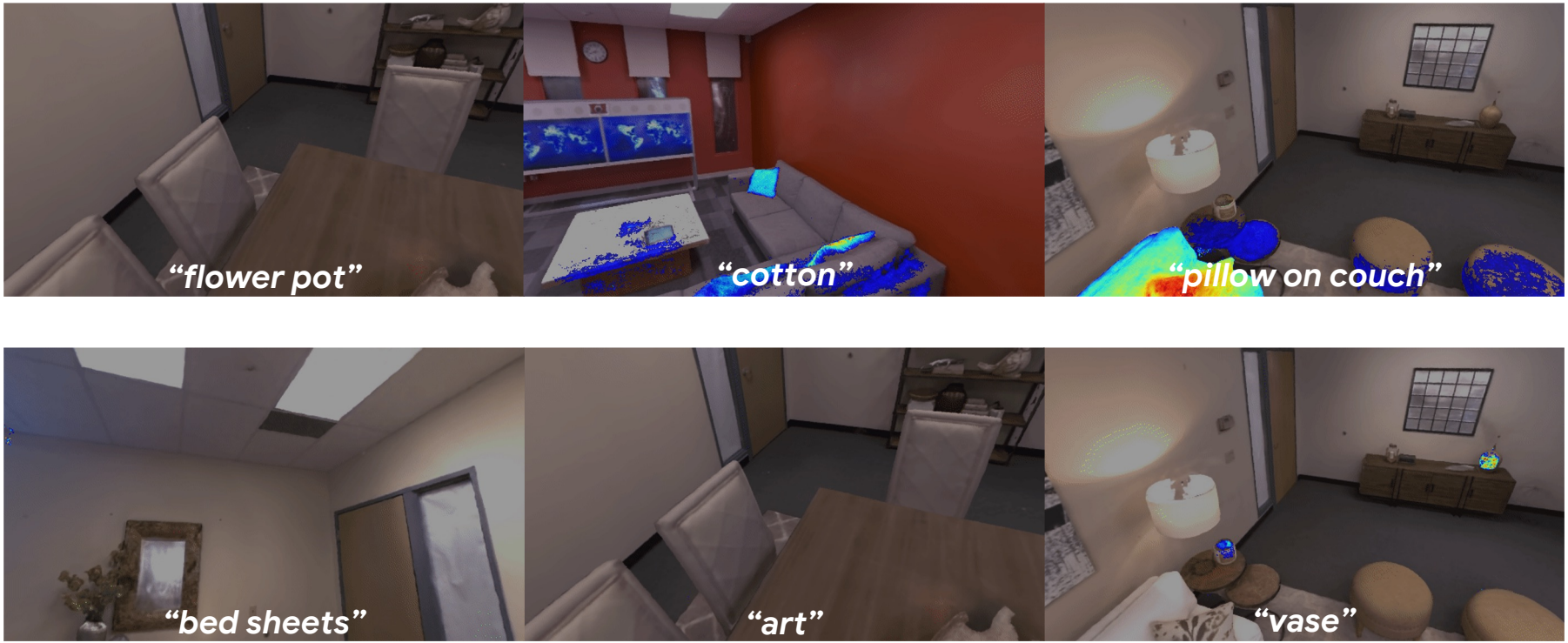
Localize arbitrary objects, properties or materials using open-vocabulary text queries.



Engelmann et al. "*Open-Set 3D Scene Segmentation with Rendered Novel Views*" arXiv'23

OpenSet 3D Scene Understanding: *Object, Properties, Materials*

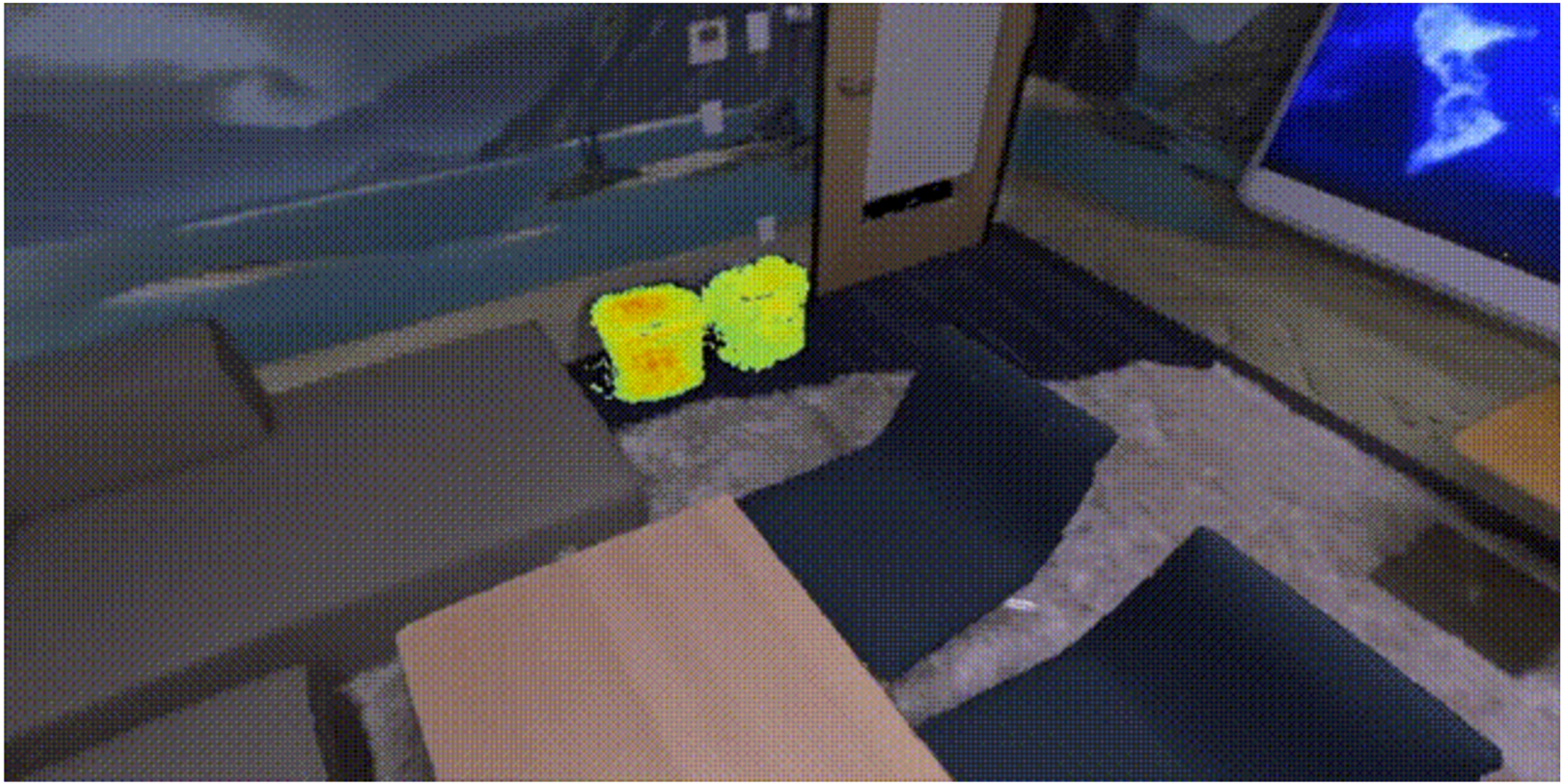
Localize arbitrary objects, properties or materials using open-vocabulary text queries.



Engelmann et al. “*Open-Set 3D Scene Segmentation with Rendered Novel Views*” arXiv’23

OpenSet 3D Scene Understanding: *Object, Properties, Materials*

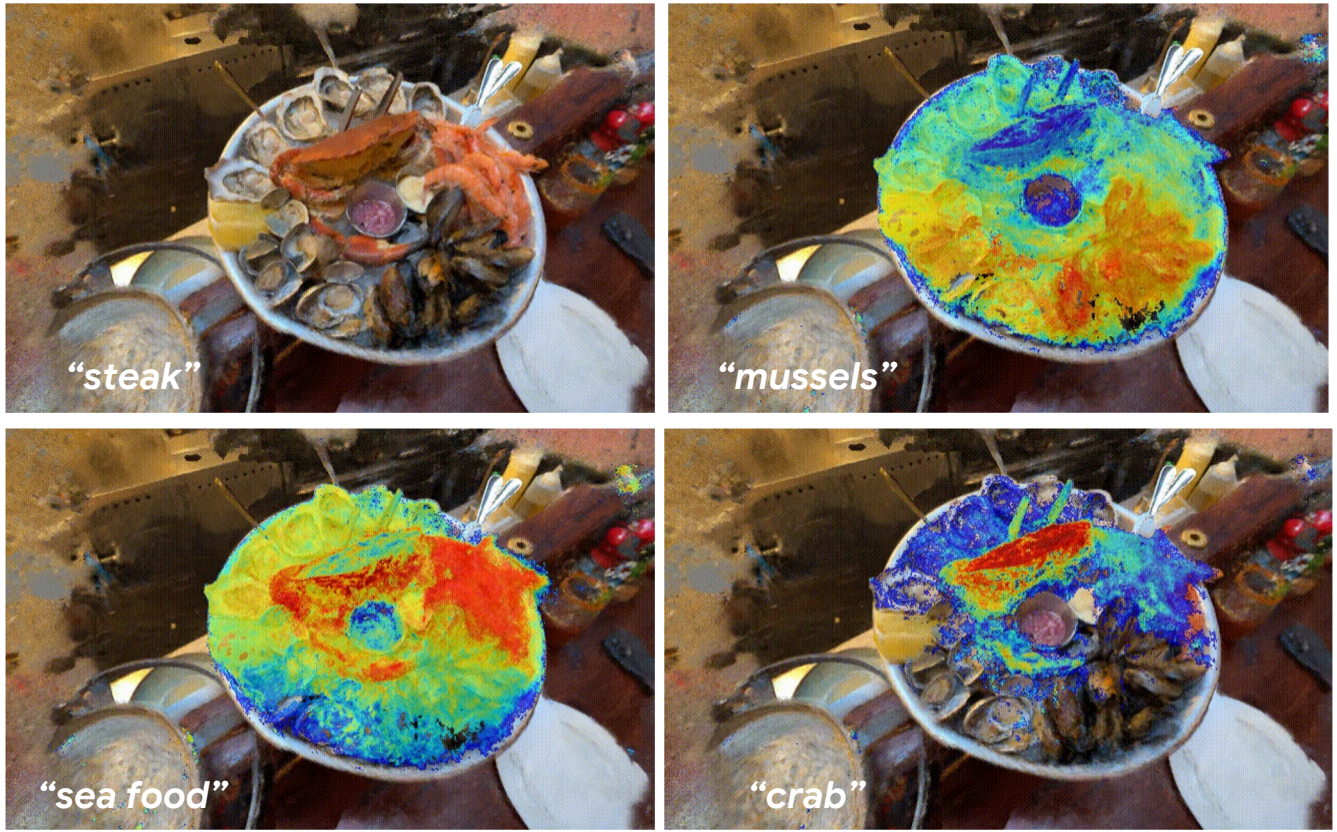
Localize arbitrary objects, properties or materials using open-vocabulary text queries.



Engelmann et al. "*Open-Set 3D Scene Segmentation with Rendered Novel Views*" arXiv'23

OpenSet 3D Scene Understanding: *Object, Properties, Materials*

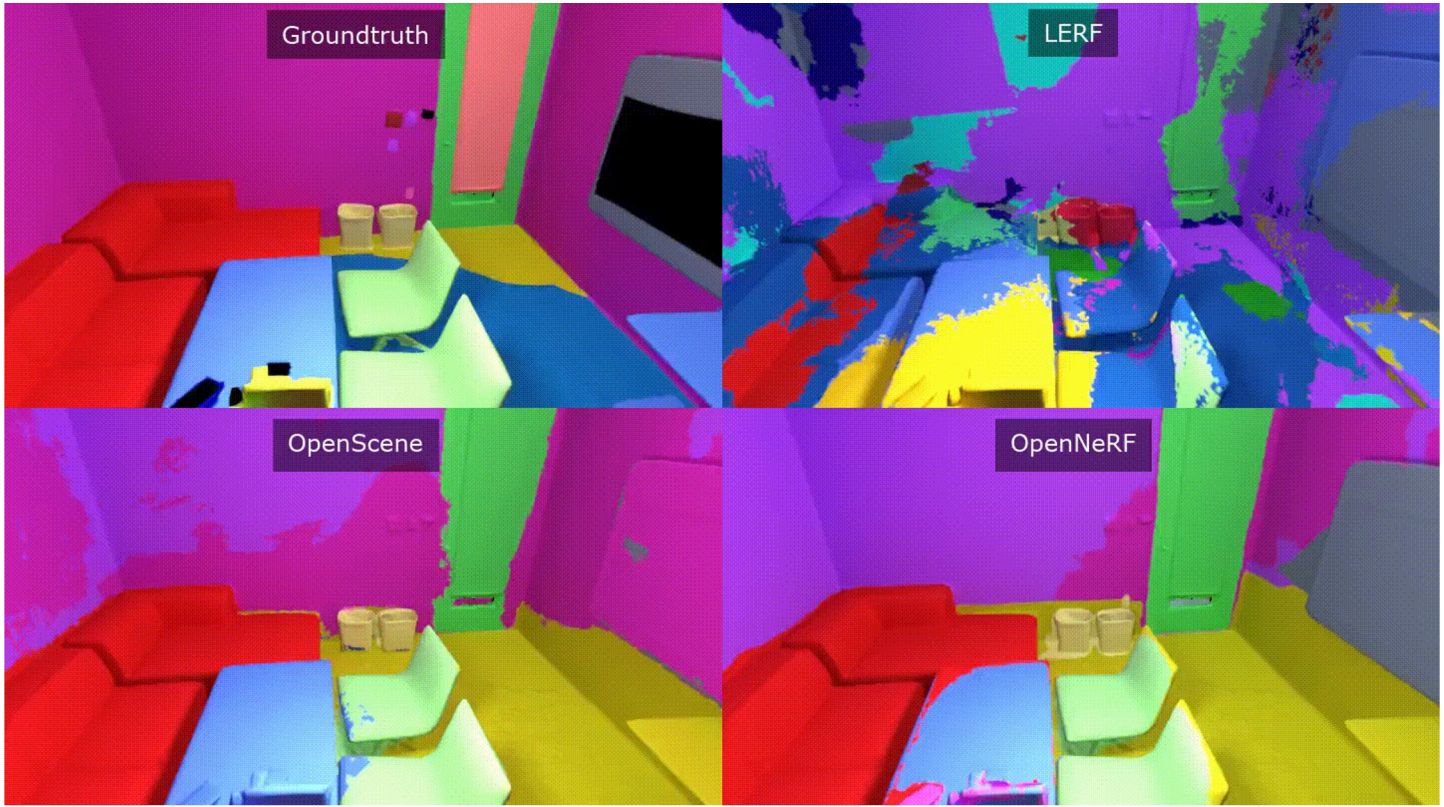
Localize arbitrary objects, properties or materials using open-vocabulary text queries.



Engelmann et al. “*Open-Set 3D Scene Segmentation with Rendered Novel Views*” arXiv’23

OpenNeRF: Open-Set 3D Neural Scene Segmentation

with Pixel-Wise Features and Rendered Novel Views

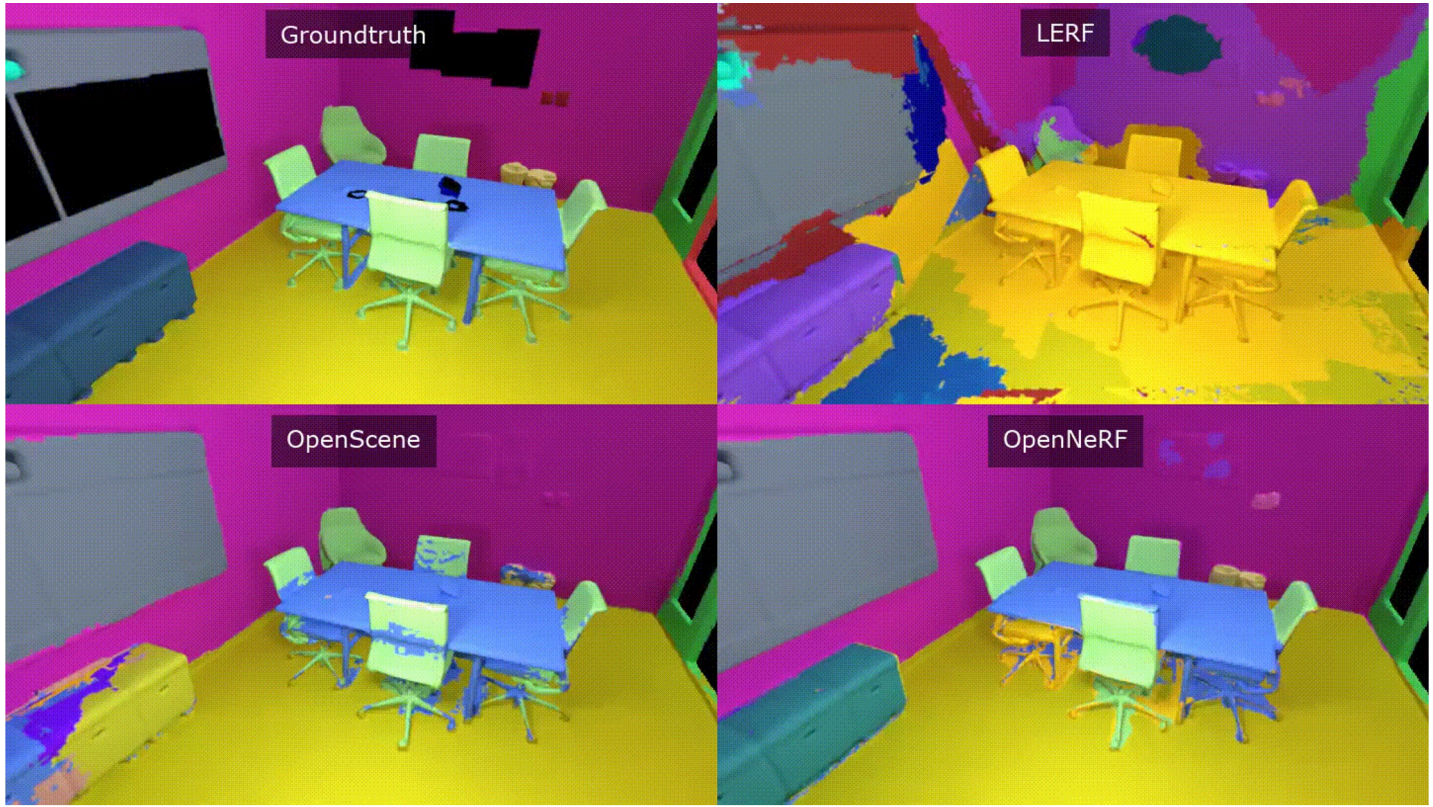


Wall Ceiling Floor Chair Blinds Sofa Table Rug Window Lamp Door Pillow Bench TV Screen Cabinet Pillar Blanket
TV Stand Cushion Bin Vent Bed Stool Picture IndoorPlant Desk Comforter Nightstand Shelf Vase PlantStand Basket Plate Monitor
Pipe Panel Desk Organizer Wall Plug Book Box Clock Sculpture Tissue Paper Camera Tablet Pot Bottle Candle Bowl Cloth Switch

[1] Kerr et al. "*LERF: Language Embedded Radiance Fields*" ICCV'23
[2] Peng et al. "*OpenScene: 3D Scene Understanding with Open Vocabularies*" CVPR'23
[3] Engelmann et al. "*OpenNeRF: OpenSet 3D Neural Scene Segmentation*" ICLR'24

OpenNeRF: Open-Set 3D Neural Scene Segmentation

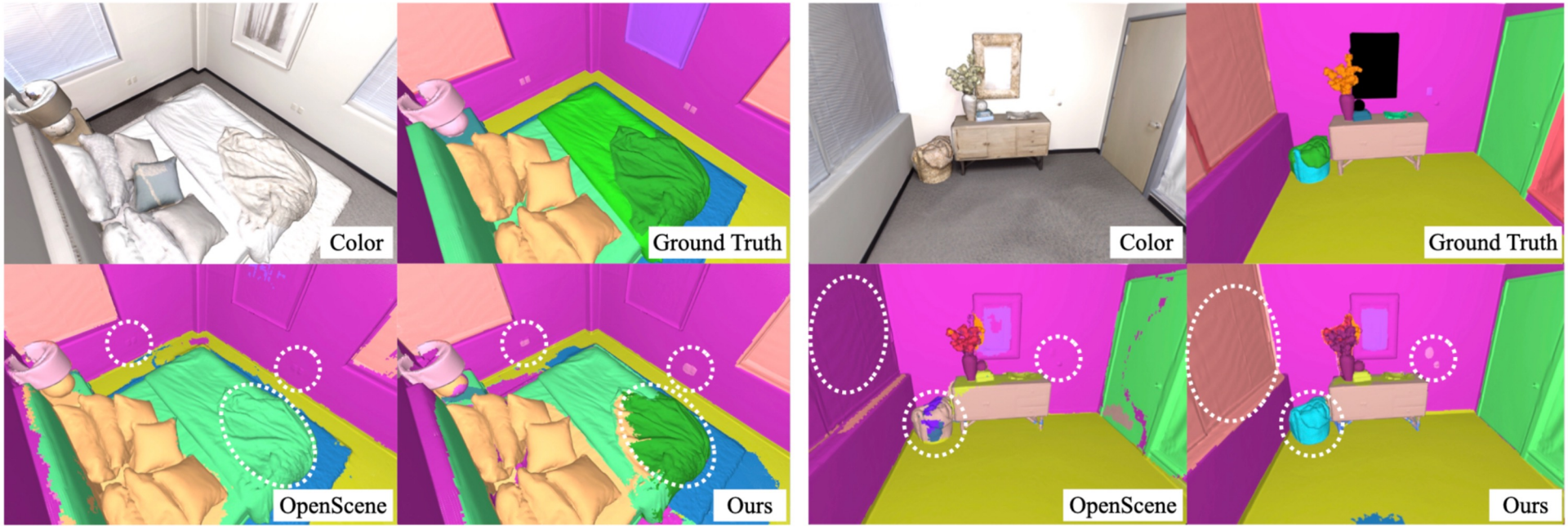
with Pixel-Wise Features and Rendered Novel Views



[1] Kerr et al. "*LERF: Language Embedded Radiance Fields*" ICCV'23
[2] Peng et al. "*OpenScene: 3D Scene Understanding with Open Vocabularies*" CVPR'23
[3] Engelmann et al. "*OpenNeRF: OpenSet 3D Neural Scene Segmentation*" ICLR'24

OpenSet 3D Scene Understanding: *Zero-Shot 3D Semantic Segmentation*

Evaluation on Replica



Wall Ceiling Floor Chair Blinds Sofa Table Rug Window Lamp Door Pillow Bench TV Screen Cabinet Pillar Blanket
TV Stand Cushion Bin Vent Bed Stool Picture IndoorPlant Desk Comforter Nightstand Shelf Vase PlantStand Basket Plate Monitor
Pipe Panel Desk Organizer Wall Plug Book Box Clock Sculpture Tissue Paper Camera Tablet Pot Bottle Candle Bowl Cloth Switch

[1] Peng et al. "*OpenScene: 3D Scene Understanding with Open Vocabularies*" CVPR'23
[2] Engelmann et al. "*Open-Set 3D Scene Segmentation with Rendered Novel Views*" arXiv'23

RadSplat: Radiance Field-Informed Gaussian Splatting for Robust Real-Time Rendering with 900+ FPS

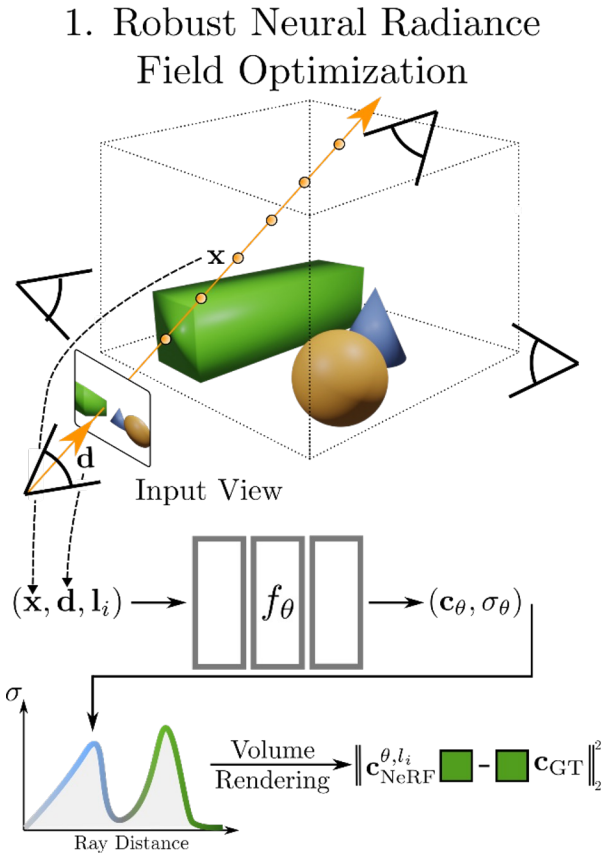
Key idea: Combine strengths of neural fields and point-based (3DGS) representations to achieve:

- Speed: **render fast**, ideally also on lower-compute devices
- Memory: lightweight representations with **low storage** footprint
- Robustness: achieve **high quality** even for **challenging real-world captures**

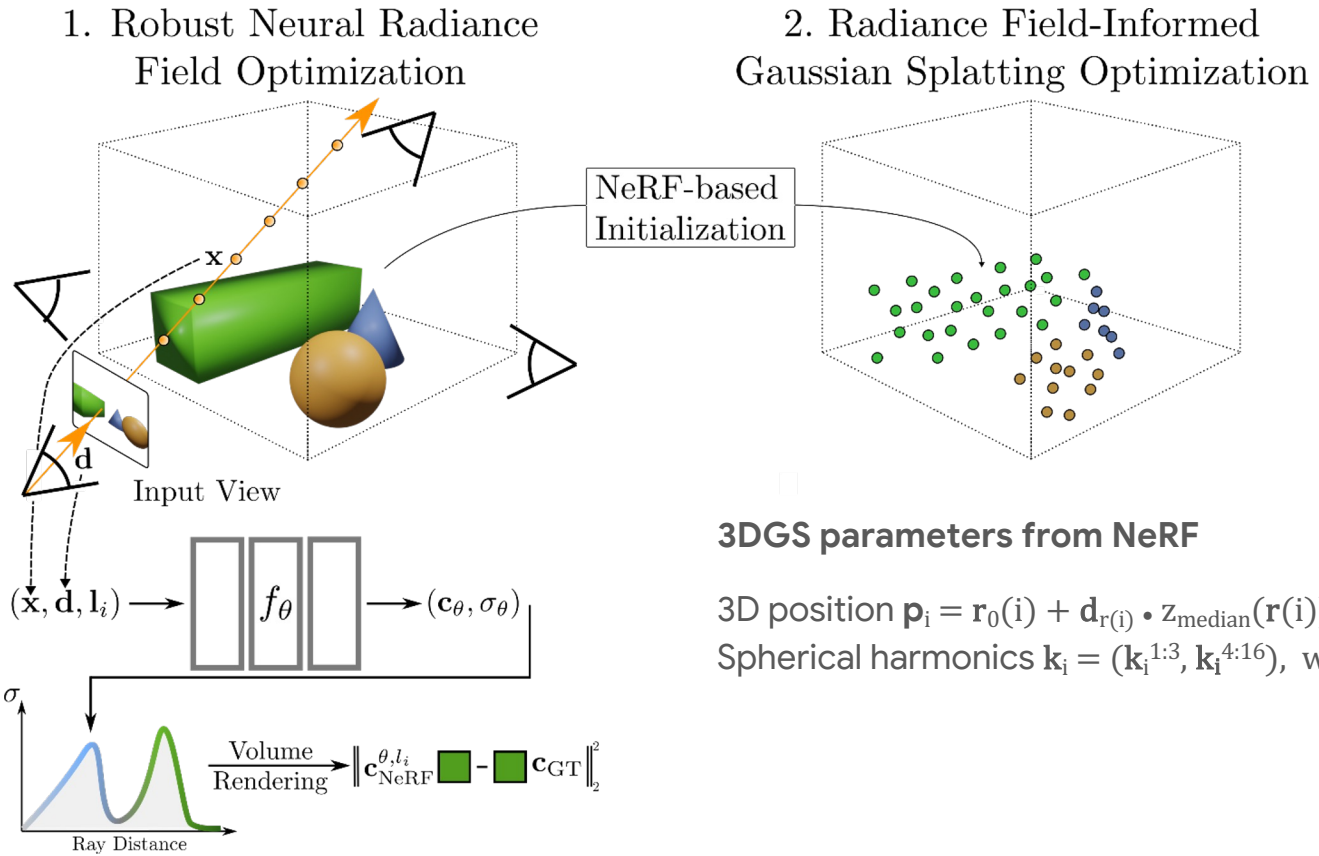
Main Contributions:

1. Use **NeRFs** as a **prior** and **as stable supervision** during 3DGS optimization
2. Use **pruning** to obtain **lightweight models** with as few Gaussians as possible
3. Perform visibility-based post processing to **further speed up rendering**

Train Robust NeRF



Initialize 3DGS representation from NeRF

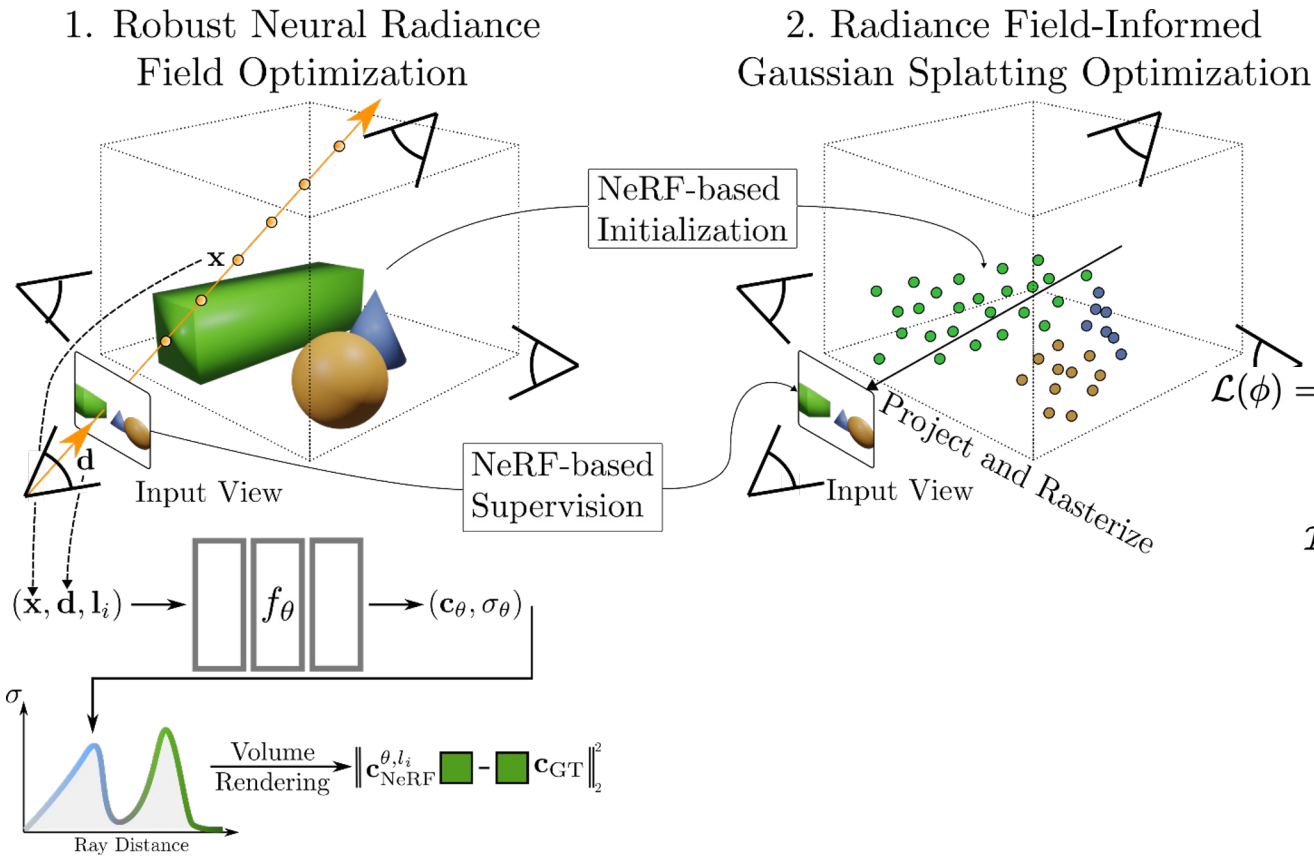


3DGS parameters from NeRF

3D position $\mathbf{p}_i = \mathbf{r}_0(i) + \mathbf{d}_{r(i)} \cdot z_{\text{median}}(\mathbf{r}(i))$

Spherical harmonics $\mathbf{k}_i = (\mathbf{k}_i^{1:3}, \mathbf{k}_i^{4:16})$, where $\mathbf{k}_i^{1:} = \mathbf{c}_{\text{NeRF}}(\mathbf{r}(i))$, and $\mathbf{k}_i^{4:16} = 0$

Use NeRF Renderings to Supervise Model



For each scene we optimize

$$\phi = \{(\mathbf{p}_i, \mathbf{k}_i, \mathbf{s}_i, o_i, \mathbf{q}_i)\}_{i=1}^{N_{\text{init}}}$$

with

$$\mathcal{L}(\phi) = (1-\lambda)\|I_f^i - I_\phi^i\|_2^2 + \lambda\,\text{SSIM}(I_f^i, I_\phi^i)$$

using our NeRF renderings

$$\mathcal{I}_f = \{I_f^j\}_{j=1}^{N} \qquad I_f^j = \{\mathbf{c}^{\theta,l_{\text{zero}}}_{\text{NeRF}}(\mathbf{r}_j(i))\}_{i=1}^{H\times W}$$

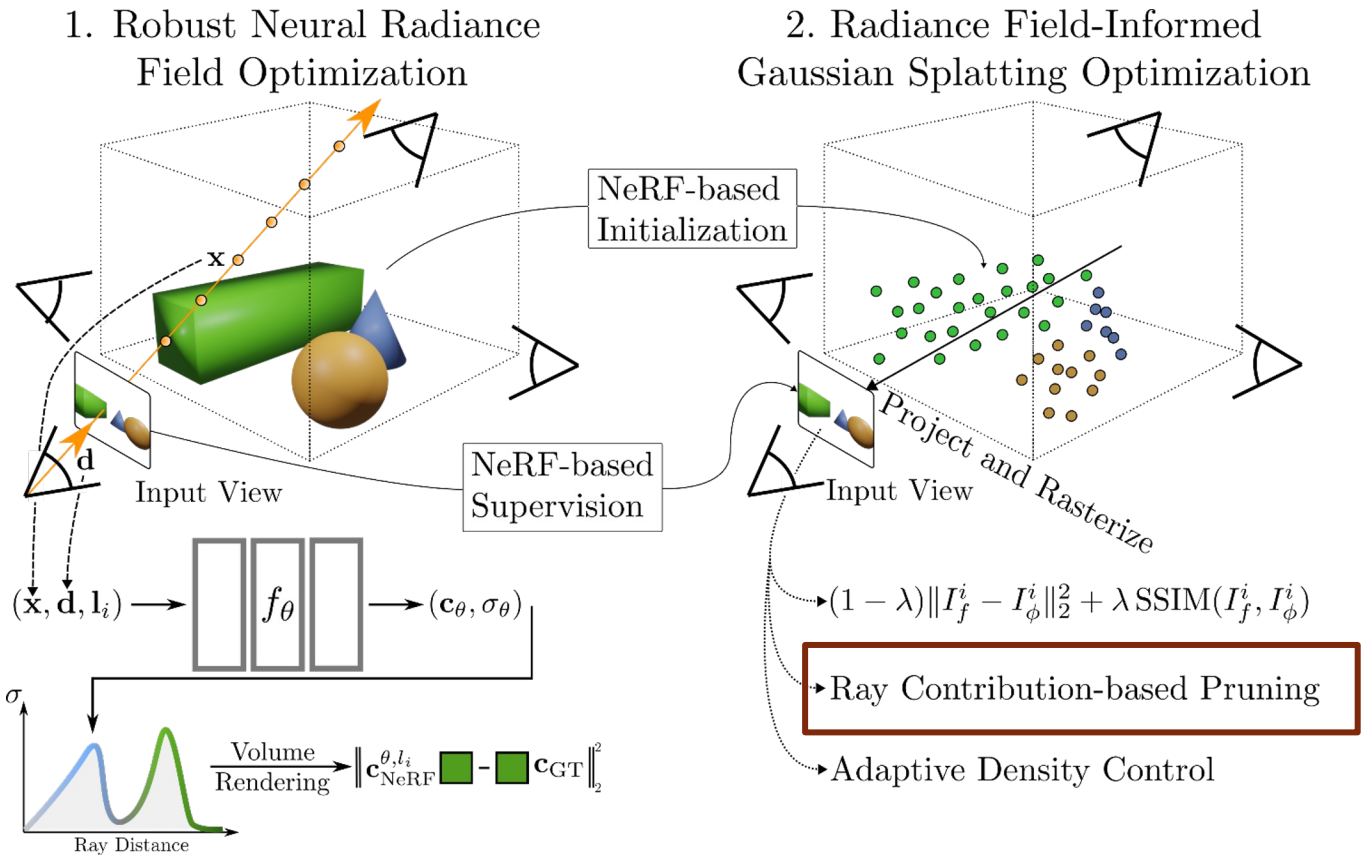
Original Images

NeRF Images

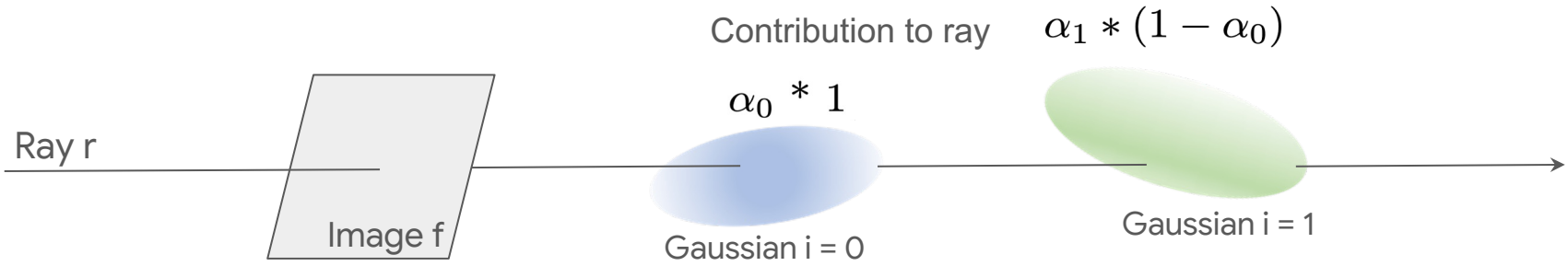
In the wild data often includes various **photometric distortions**

Utilizing GLO vectors in 3DGS is not trivial: we instead **render clean distortion free training images** using the learned NeRF prior setting the GLO vectors to 0

Optimize 3DGS Representation w/ Novel Pruning



Ray Contribution-based Pruning



Our importance score measure the **maximum contribution** of a **gaussian towards any pixel / ray r**

$$h(\mathbf{p}_i) = \max_{I_f \in \mathcal{I}_f, r \in I_f} \alpha_i^r \tau_i^r$$

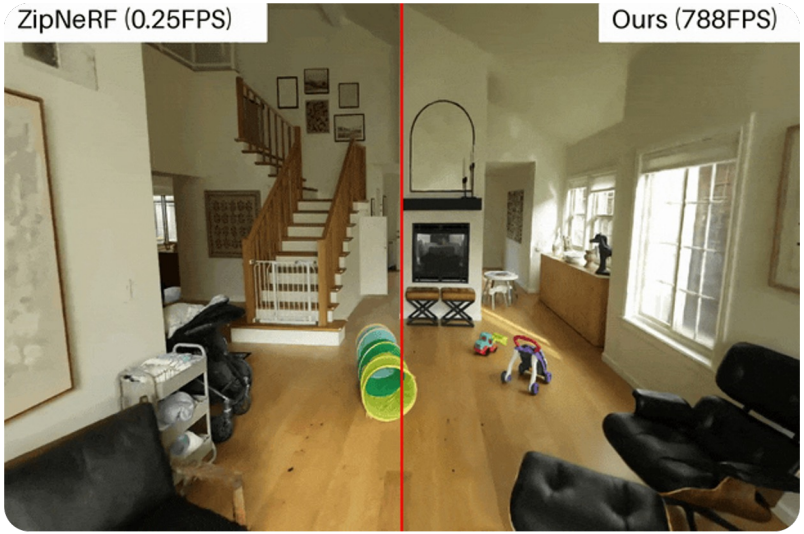
We **prune all gaussians that have no minimal contribution** to any training's image (performed twice)

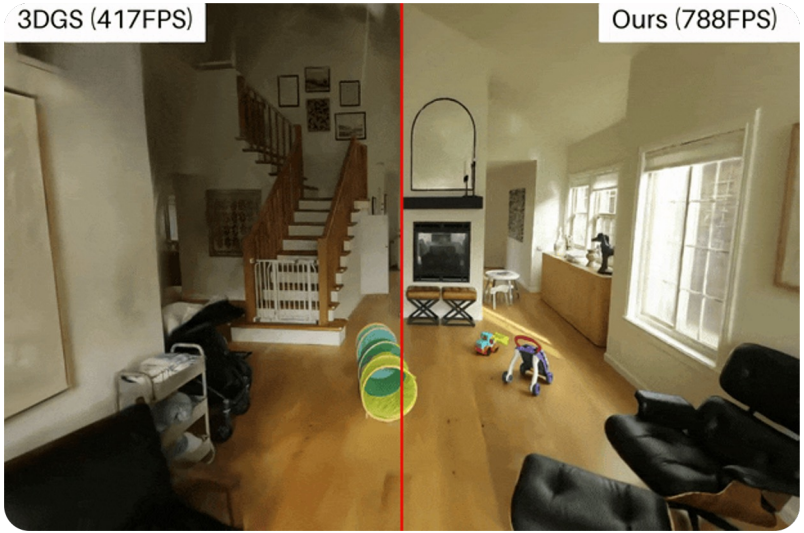
$$m_i = m(\mathbf{p}_i) = \mathbb{1}\left(h(\mathbf{p}_i) < t_{\text{prune}}\right) \quad \text{where} \quad t_{\text{prune}} \in [0, 1]$$

Oftentimes, only as little as **10% of all spawned gaussians have a significant contribution** to any ray

Complex Large-Scale Scenes

Compared to ZipNeRF, we achieve **similar quality** and **render much faster (3000x)**
Compared to 3DGS, we **achieve higher quality,** are more **robust**, and **render faster**





Evaluated on Nvidia 3090 Ti

Available at: m-niemeyer.github.io/radsplat

Thanks for the Attention!

Credits and collaborators (possibly incomplete):

Federico Tombari (TUM, Google)

Benjamin Busam (TUM)

Edo Collins (ETH)

Helisa Dhamo (Huawei)

Yan Di (TUM)

Elisabetta Fedele (ETH)

Nassir Navab (TUM)

Marie-Julie Rakotosaona (Google)

Christina Tsalicoglou (ETH)

Luc Van Gool (ETH)

Johanna Wald (Google)

Shun-Cheng Wu (TUM)

Guangyao Zhai (TUM)

Michael Niemeyer (Google)