Hands-on AI based 3D Vision
Summer Semester 26

Lecture 7 – Neural Radiance Fields

UniveProf. Dr.-Ing Gerard Pons-Moll
rsity of Tübingen / MPI-Informatics

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

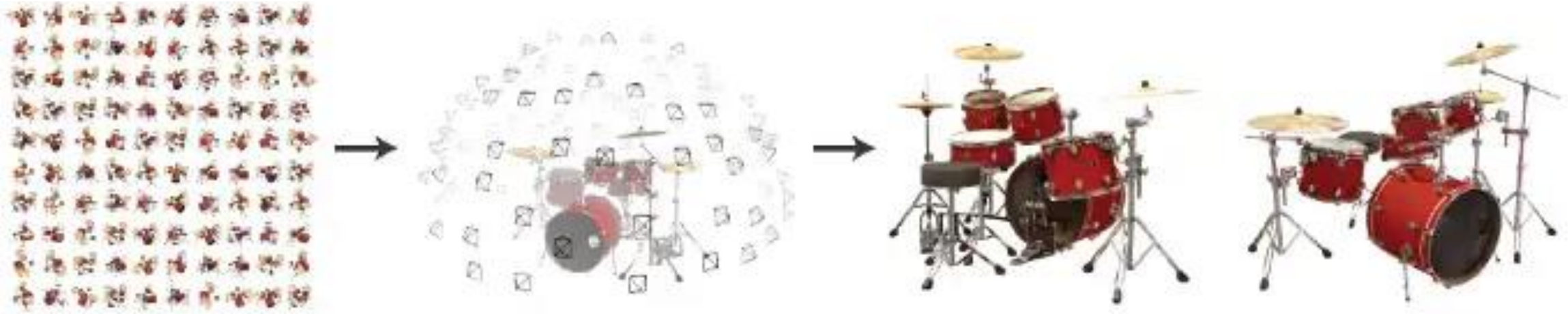
Neural Radiance Fields: The Evolution of 3D Reconstruction

Neural Radiance Fields: The Evolution of 3D Reconstruction

Overview of concepts:

- **NeRF (2020) - The Foundational Breakthrough**
- DNerf (2021)- Dynamic Nerf
- HyperNeRF (2021) - Topological Dynamics
- NeRF-W (2021) - Unconstrained Capture
- Control-NeRF (2023) - Scene Manipulation
- Instant-NGP (2022) – Training Speed

NeRF (2020) - The Foundational Breakthrough



Input:
Sparsley sampled
images of the scene

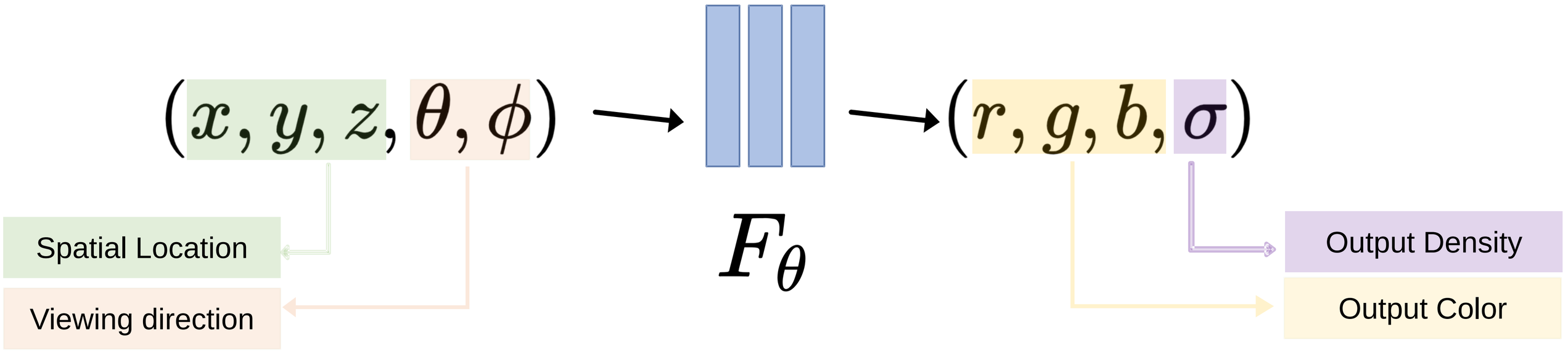
Learn scene
representation

Novel view synthesis
of the scene

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

Novel View Synthesis

Learning radiance field representation of scene:



Volume Rendering

Given color and density (r, g, b, σ) , we calculate the color of every camera ray using:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t))dt$$

Camera ray: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$

Near and far bounds

Color (r,g,b) at r(t)

Volume density: Probability of a ray terminating at an infinitesimal particle at location r(t)

Accumulated transmittance along ray :

$$T(t) = \exp(-\int_{t_n}^{t_f} \sigma(\mathbf{r}(s))ds)$$

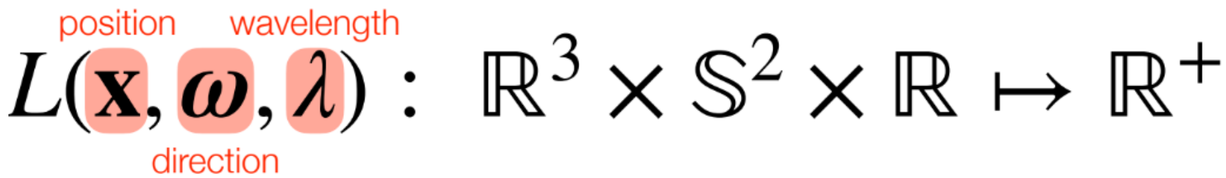
Radiance

Radiance is (differential) energy

- per unit area, solid angle, and wavelength

Radiance

- density of photons at a point
- traveling in the same direction
- at given wavelength



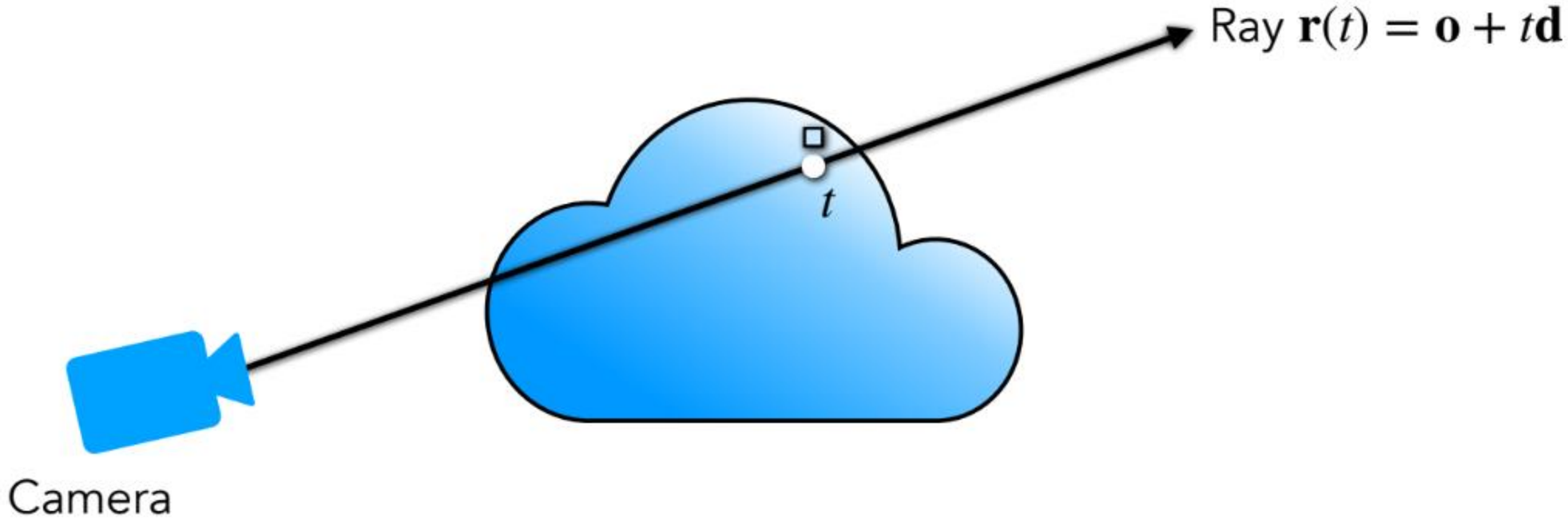
Radiance along an unblocked ray is constant (energy conservation)

The “Light Field” is the radiance for every possible ray

Volumetric Radiance

Scene is a cloud of tiny colored particles

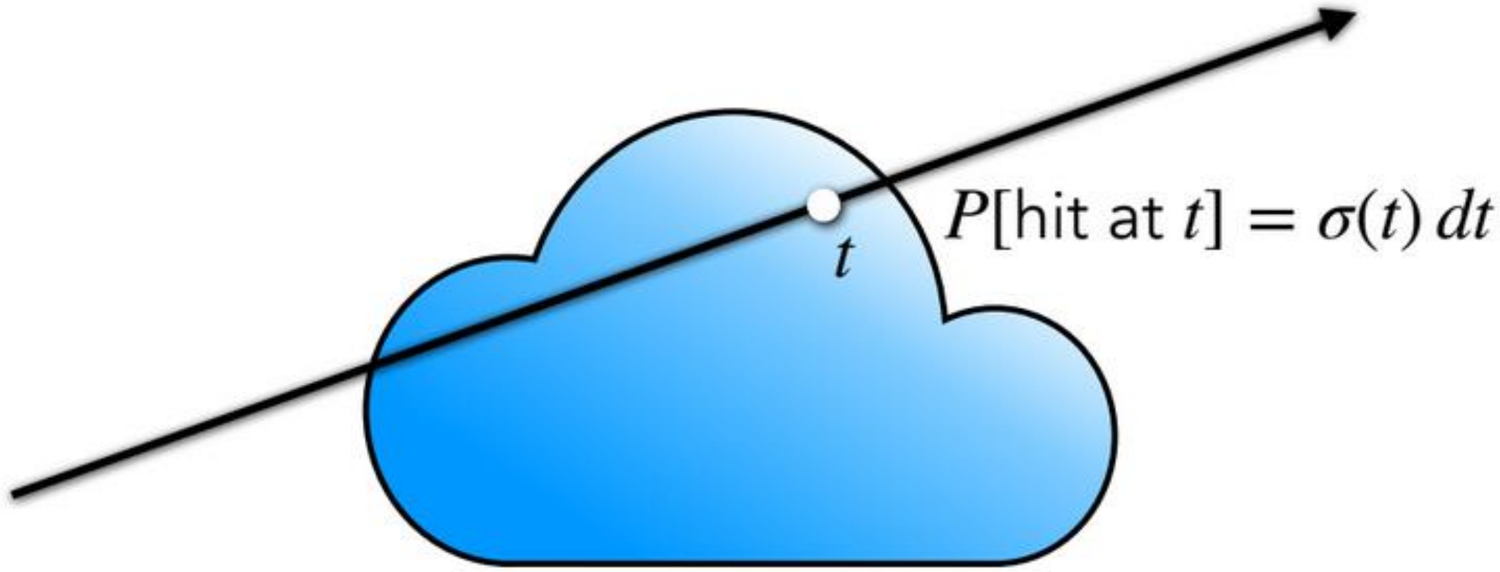
- Their color changes according to viewpoint
- If a ray traveling through the scene hits a particle at , t
- we return its radiance/color $\mathbf{c}(t)$



Volumetric Density

Probability that ray stops in a small interval around t is $\sigma(t)\,dt$

- Also called **Volume**[tric] **Density**



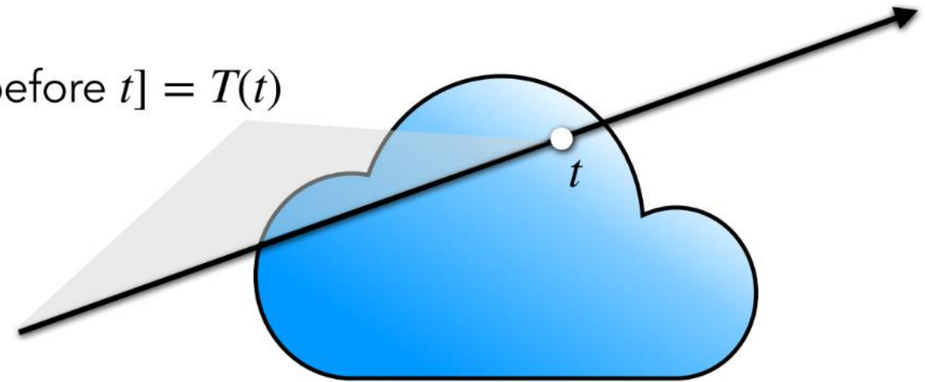
Scene Representation

Our scene representation is therefore a **field** (neural? maybe):

$$\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \times \mathbb{R}^+; \quad \Phi(\mathbf{x}) = (\sigma, \mathbf{c})$$

While evaluating the filed along a ray, we retrieve the color or $\mathbf{c}(t)$ f
is visibl $\mathbf{r}(t)$

Transmittance $T(t)$ s the probability of **no** particles hit in $[0, t)$ e



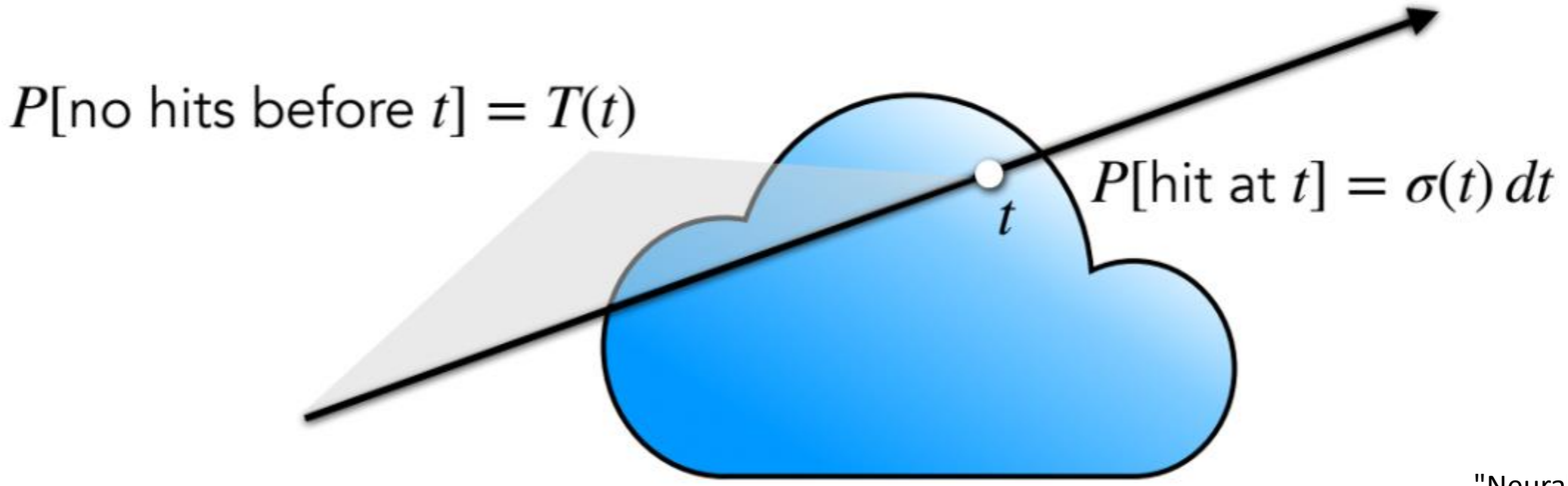
"Neural Rendering", Andrea Tagliasacchi

Relating $\sigma(t)$, $T(t)$

Hit probabilities are statistically independent along ray (P[A,B]=P[A]P[B])

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

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



Transmittance

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

$$\frac{T(t + dt) - T(t)}{dt} = - T(t)\sigma(t)$$

$$T'(t) = - T(t)\sigma(t)$$

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

$$\int_a^b \frac{T'(t)}{T(t)} dt = - \int_a^b \sigma(t) dt$$

$$\log T(t) \,|_a^b = - \int_a^b \sigma(t) dt$$

$$T(a \to b) := \frac{T(b)}{T(a)} = \exp\left(- \int_a^b \sigma(t) dt \right)$$

$$T(0 \to t) := T(t) = \exp\left(- \int_0^t \sigma(u)\, du \right)$$

No hits before t is equal to integral over density up until t.

"Neural Rendering", Andrea Tagliasacchi

Expected Ray Termination

$1 - T(t)$ can be seen as cumulative distribution function (CDF)

- probability that ray hits something before reaching t
- $1 - T(t)$ is non-decreasing
- $T(t)$ is (right) continuous

$$\lim_{t \to -\inf} (1 - T(t)) = 0$$

$$\lim_{t \to +\inf} (1 - T(t)) = 1$$

$$T(0 \to t) := T(t) = \exp\left(- \int_0^t \sigma(u)\, du \right)$$

- $T'(t) = T(t) \cdot \sigma(t)$ is a probability density function (PDF) that represents the probability that a ray stops at t

Volume Rendering

$T'(t) = T(t) \cdot \sigma(t)$ is the probability that a ray stops at t

Expected color along a ray is a convex combination of colors

$$C = \int_0^\infty T(t) \cdot \sigma(t) \cdot \mathbf{c}(t)\, dt \quad \text{where} \quad T(t) = \exp\left(- \int_0^t \sigma(u)\, du \right)$$

How do we solve this?

- Discretize the nested integral

"Neural Rendering", Andrea Tagliasacchi

Approximating the integral

Split the ray up into N segments with endpoints $\{t_1, t_2, \ldots, t_{N+1}\}$

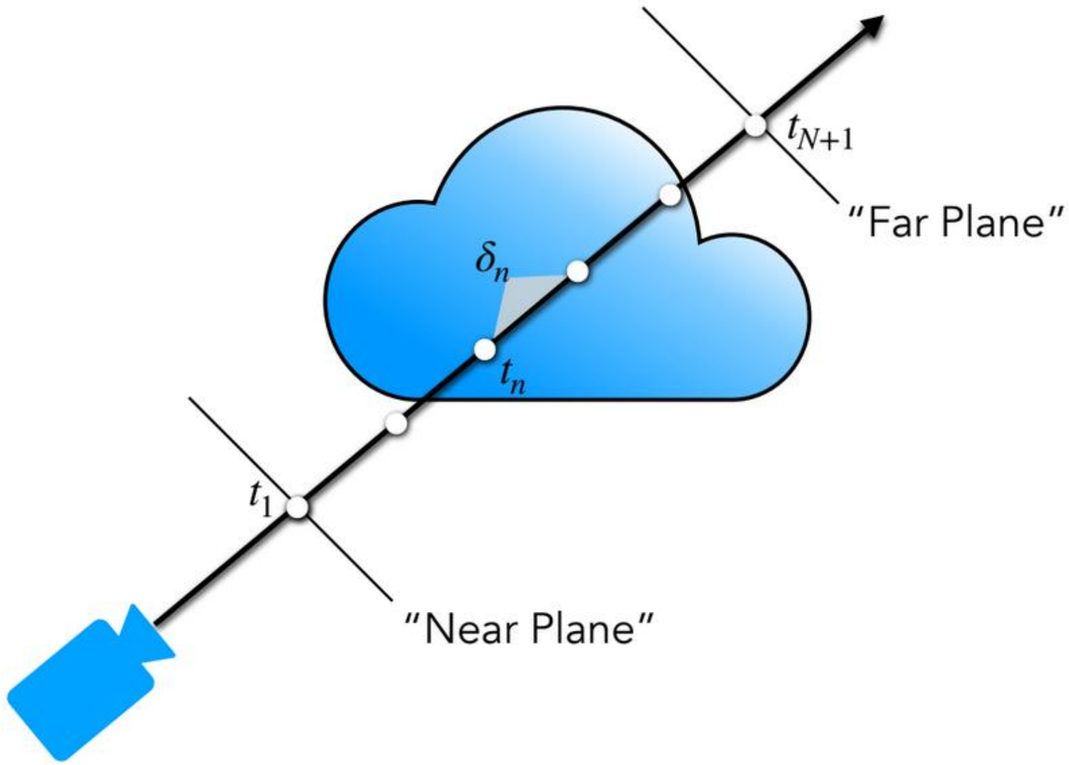
Segment length is $\delta_i = t_{i+1} - t_i$

- warning: non-necessarily uniform!

piecewise constant density

≠

piecewise constant transmittance

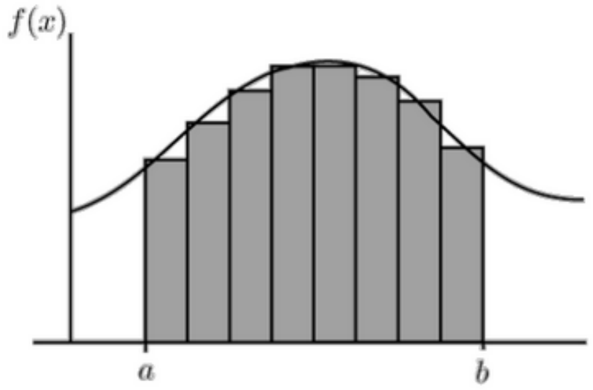


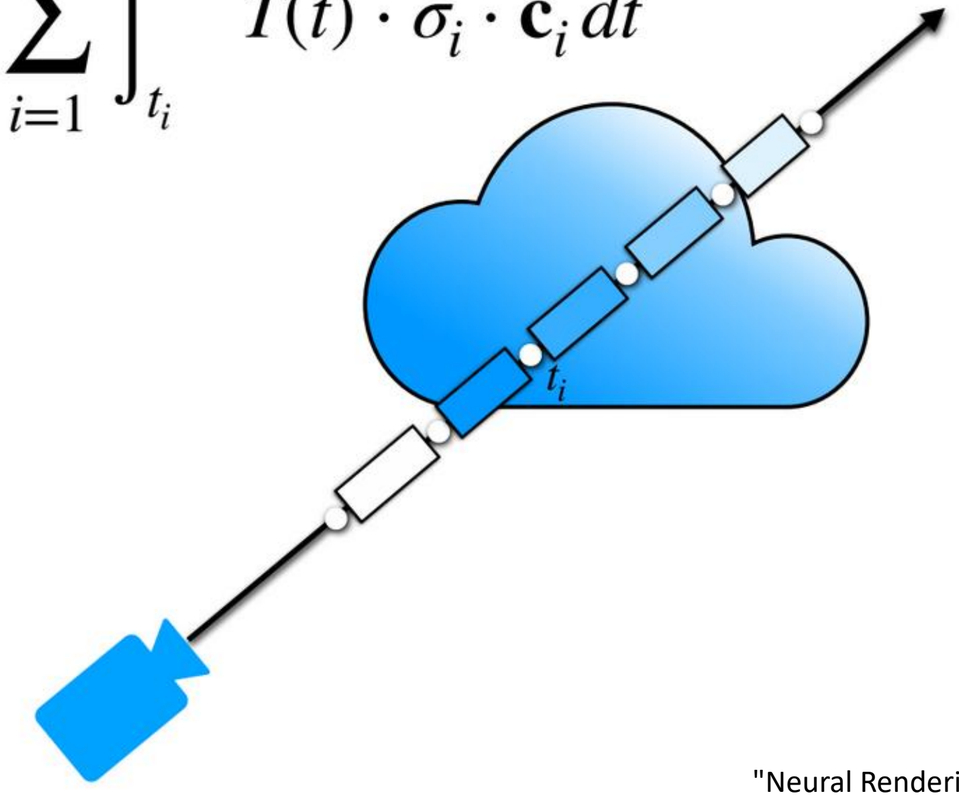
"Neural Rendering", Andrea Tagliasacchi

Approximating the nested integral

Assume volume density/color are constant within interval (i.e. Reimann)

$$C = \int_0^{\infty} T(t) \cdot \sigma(t) \cdot \mathbf{c}(t)\, dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t) \cdot \sigma_i \cdot \mathbf{c}_i\, dt$$





"Neural Rendering", Andrea Tagliasacchi

Forward Model - NeRF

$$\sum_{n=1}^{N} \int_{t_n}^{t_{n+1}} T(t) \cdot \sigma_n \cdot \mathbf{c}_n \, dt$$

$$\sum_{n=1}^{N} \int_{t_n}^{t_{n+1}} T(0 \to t_n) \cdot T(t_n \to t) \cdot \sigma_n \cdot \mathbf{c}_n \, dt \qquad \text{(i.i.d. process)}$$

$$\sum_{n=1}^{N} T(0 \to t_n) \int_{t_n}^{t_{n+1}} T(t_n \to t) \cdot \sigma_n \cdot \mathbf{c}_n \, dt \qquad \text{(constant)}$$

$$\sum_{n=1}^{N} T_n \cdot (1 - \exp(-\sigma_n \delta_n)) \cdot \mathbf{c}_n \quad \text{where} \quad T_n = \exp\left(\sum_{k=1}^{n-1} - \sigma_k \delta_k \right) \qquad \text{(tedious)}$$

"Neural Rendering", Andrea Tagliasacchi

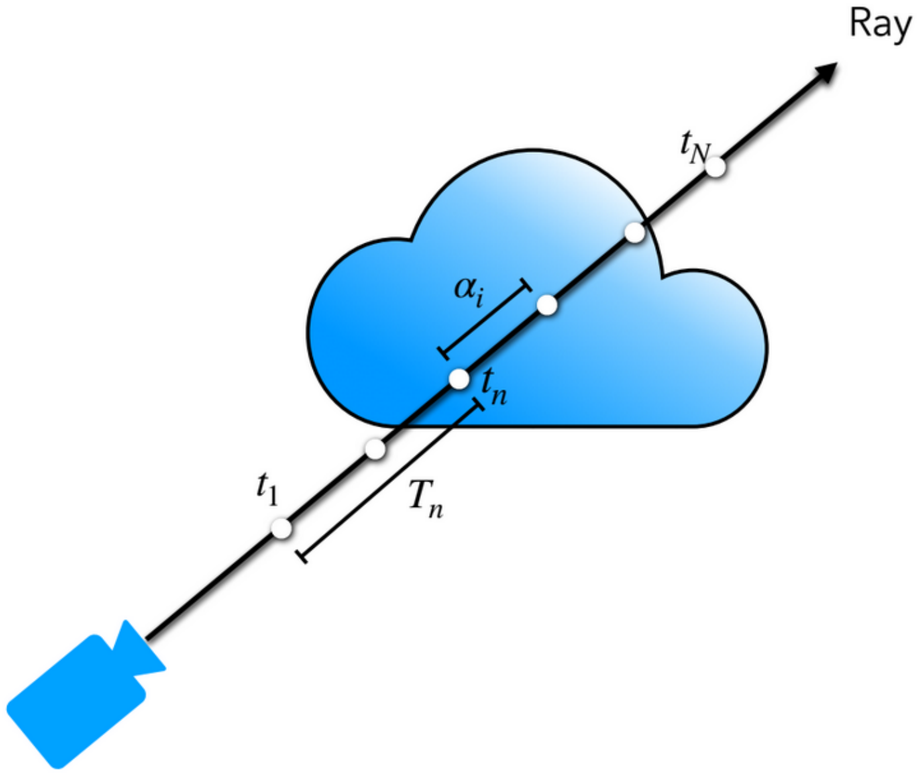
NeRF as Alpha Blending

$$C = \sum_{n=1}^{N} T_n \cdot (1 - \exp(-\sigma_n \delta_n)) \cdot \mathbf{c}_n \quad \text{where} \quad T_n = \exp\left(\sum_{k=1}^{n-1} - \sigma_k \delta_k \right)$$

$$C = \sum_{n=1}^{N} T_n \cdot \alpha_n \cdot \mathbf{c}_n, \text{ where}$$

$$\alpha_n = 1 - \exp(-\sigma_n \delta_n) \quad \ldots \text{ segment opacity}$$

$$T_n = \prod_{k=1}^{n-1} (1 - \alpha_k) \ldots \text{ segment occlusion}$$



"Neural Rendering", Andrea Tagliasacchi

Volume Rendering in NeRF

Given color and density (r, g, b, σ) , we calculate the color of every camera ray using:

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

Color (r,g,b) at t_i

Distance between adjacent samples

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

Uniform N samples

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right]$$

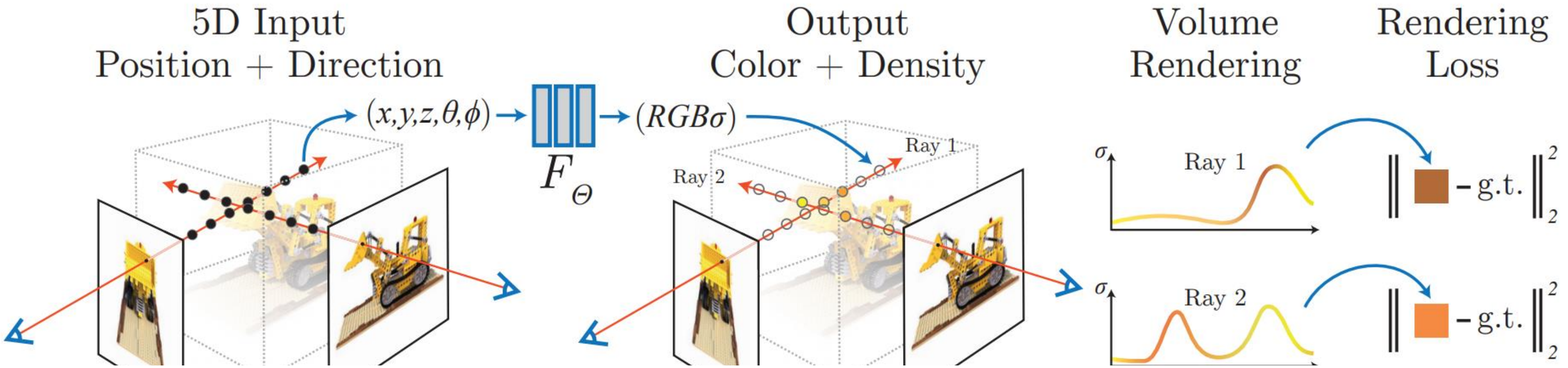
Alpha(in traditional alpha composting)

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

Accumulated transmittance along ray

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

Training Neural Radiance Fields



- **March camera rays** through the scene to generate a **sampled set of 3D points**
- Use those **points and** their corresponding 2D **viewing directions** as input to the neural network to produce an output **set of colors and densities**
- Use classical volume rendering techniques to **accumulate those colors and densities** into a 2D image
- **Minimize** error between **rendered color and GT color**

Neural Radiance Fields

View dependent illumination

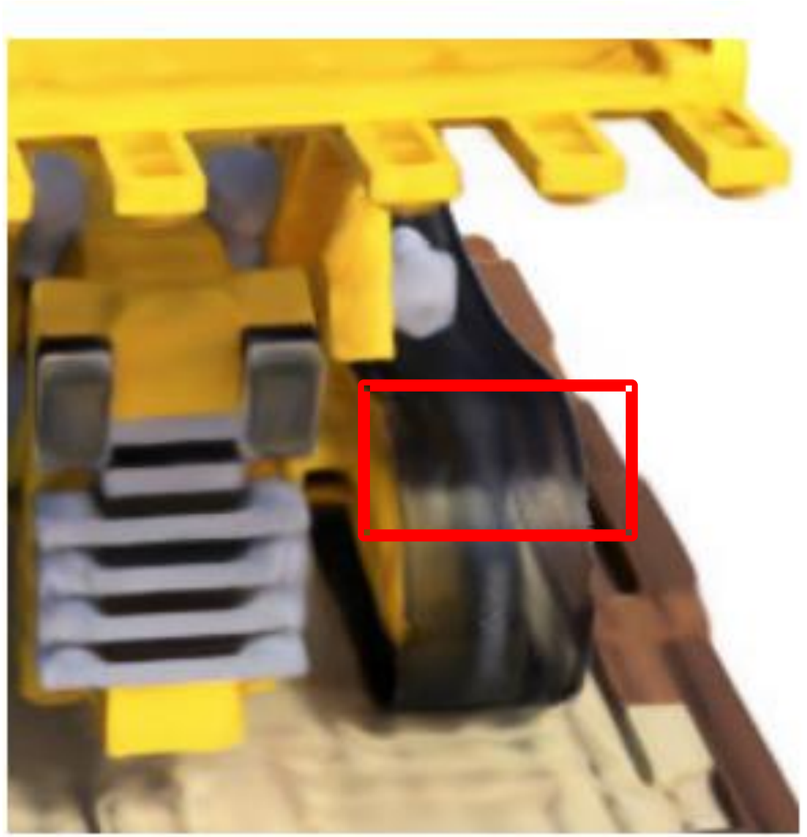
- Effect of Θ and ϕ on the output field

$$(x, y, z, \theta, \phi) \rightarrow \boxed{\quad} \rightarrow (RGB\sigma)$$

Novel view synthesis using NeRF

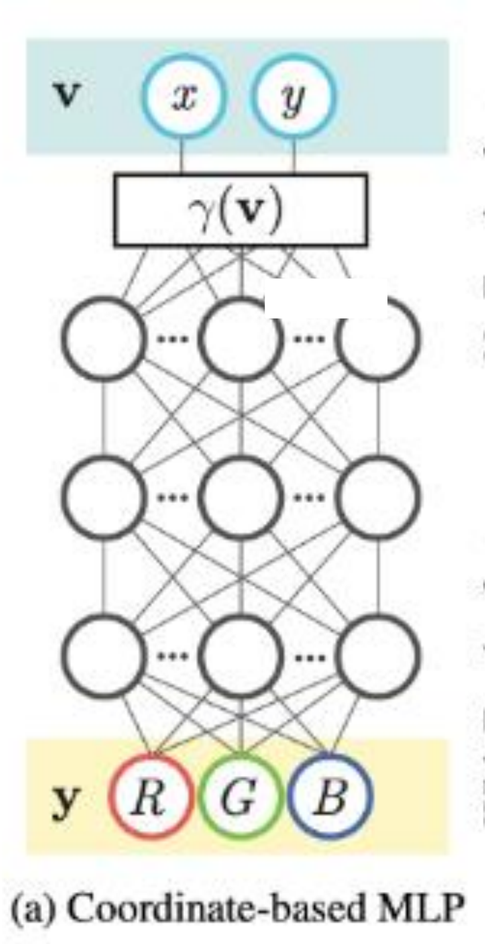
Generated results are blurry

Ground Truth



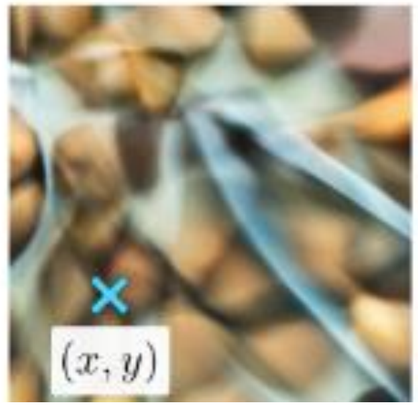
No Positional Encoding

Why blurry results?

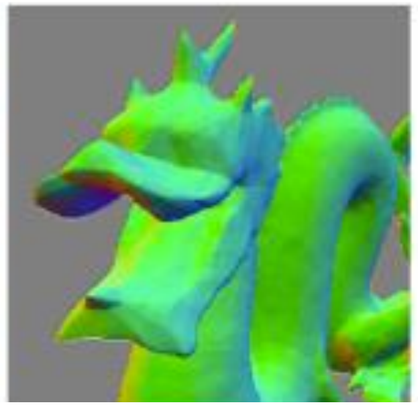


(a) Coordinate-based MLP

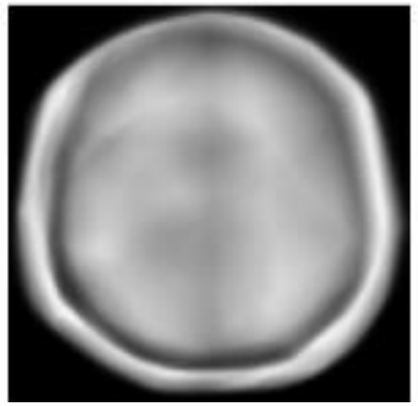
Coordinate based neural network fail to learn high frequency details for all kind of data including RGB image, 3D shape, density , etc



RGB



3D Shape



Density

Radiance

field

(density, color)

Tannick et al. NeurIPS 20

Solution

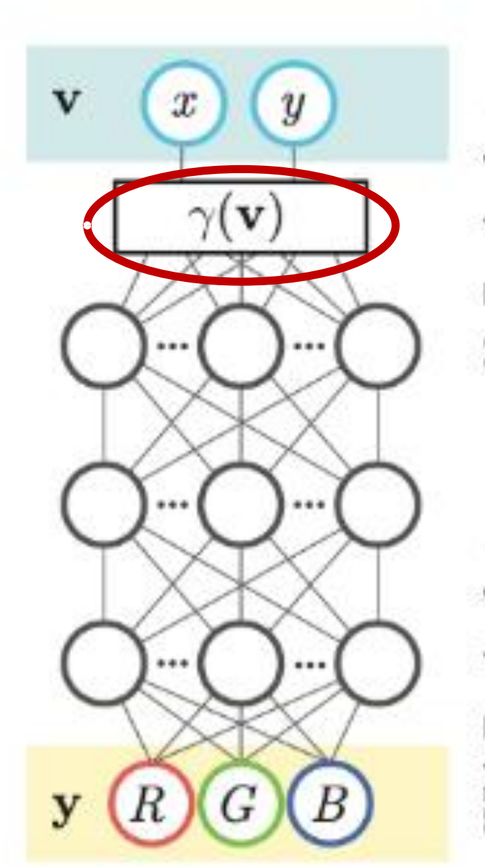
- In naive setting, the bandwidth of the Neural Tangent Kernel limits the spectrum of the recovered/learned function.
- Using a Fourier feature mapping transforms the neural kernel into a stationary kernel in our low-dimensional problem domains and increase the spectrum.

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

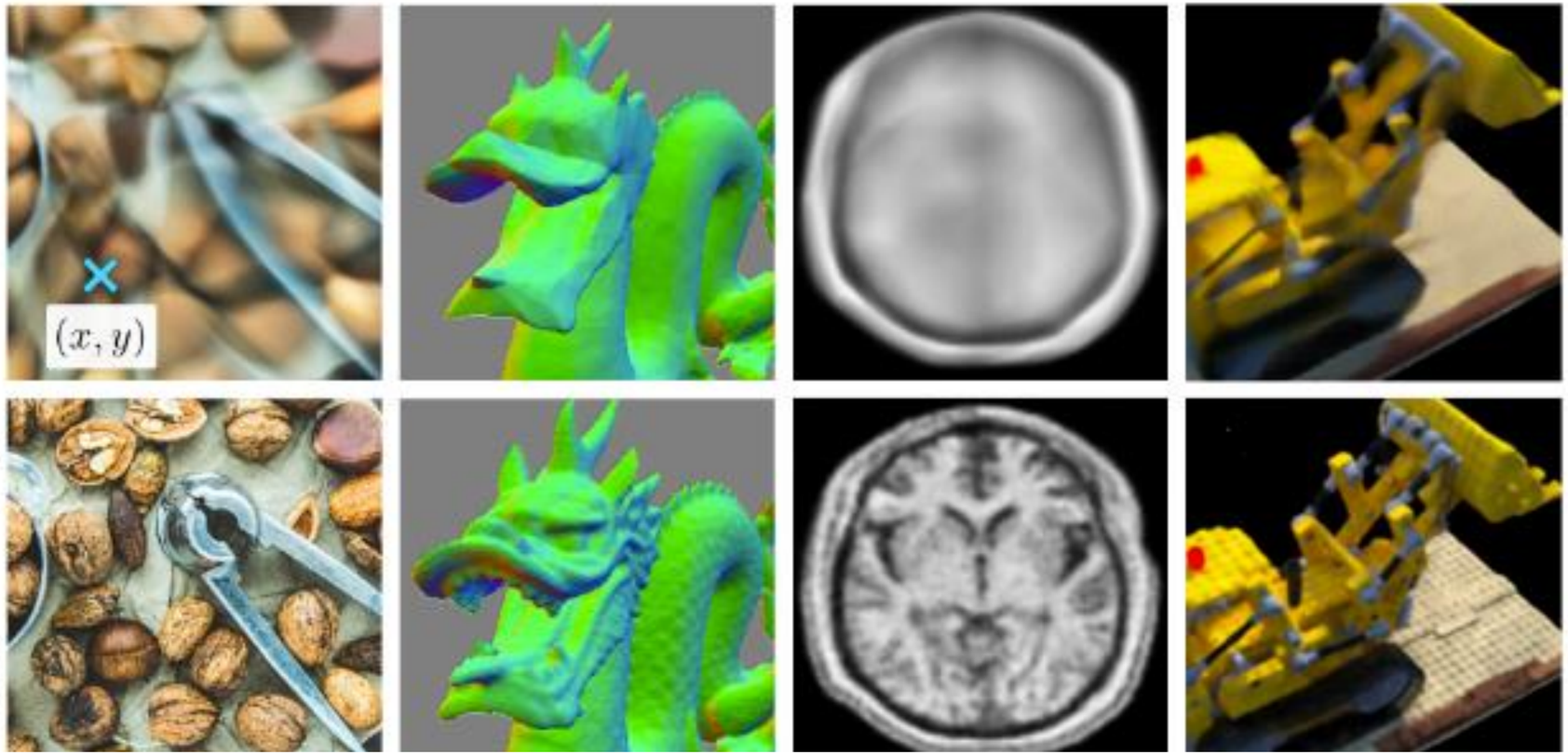
Coordinate input e.g. pixel location for images, 3D point for NeRF

Tannick et al. NeurIPS 20

Fourier Features in Coordinate MLPs



(a) Coordinate-based MLP



RGB Density

3D Shape Radiance field

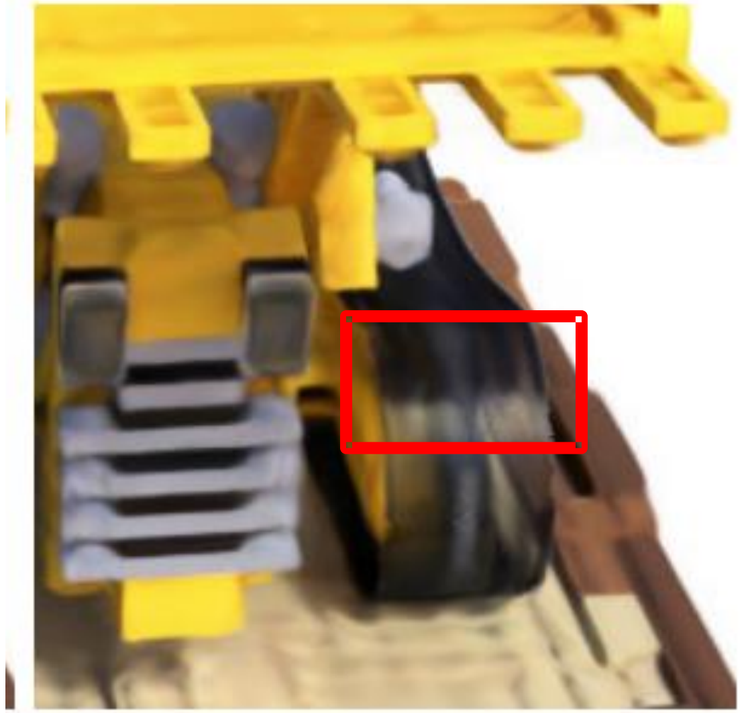
$(x,y) \rightarrow \text{RGB}$ $(x,y,z) \rightarrow \text{occupancy}$ $(x,y,z) \rightarrow \text{density}$ $(x,y,z) \rightarrow \text{RGB, density}$

Tannick et al. NeurIPS 20

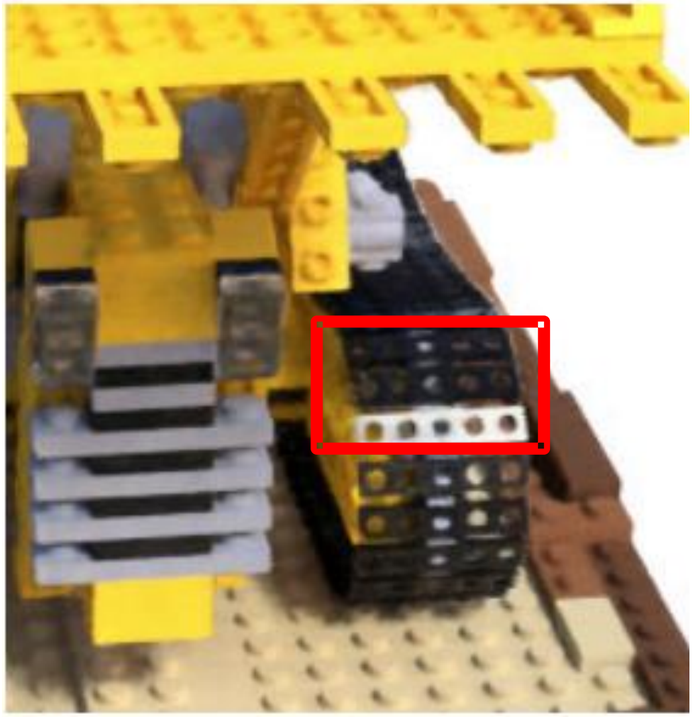
Positional Encoding in NeRF

With fourier features/positional encoding, NeRF learns high frequency details.

Ground Truth



No Positional Encoding



Complete Model

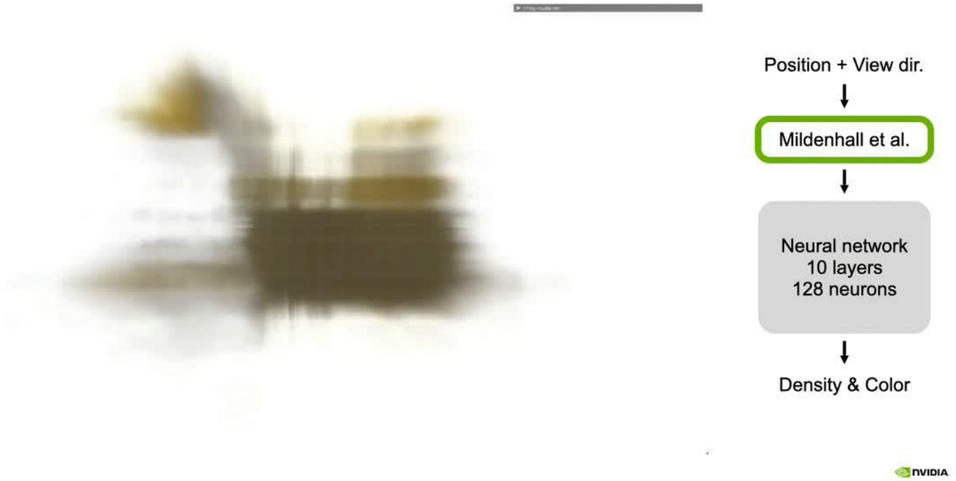
Tannick et al. NeurIPS 20

Positional Encoding in NeRF

With fourier features/positional encoding, NeRF learns high frequency details.

Without positional encoding

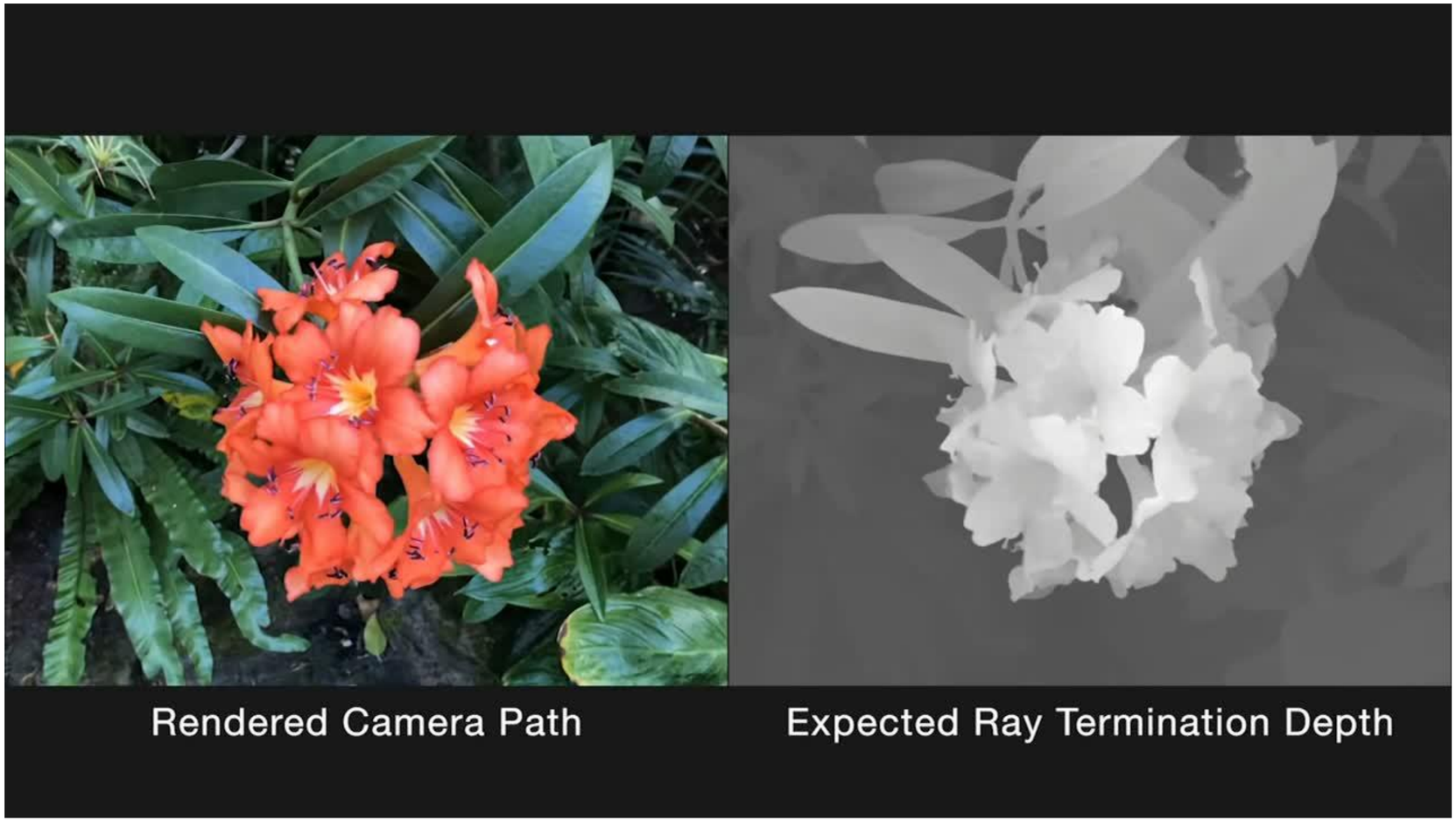
With positional encoding



Mueller et al. SIGGRAPH 22

Geometry in NeRF

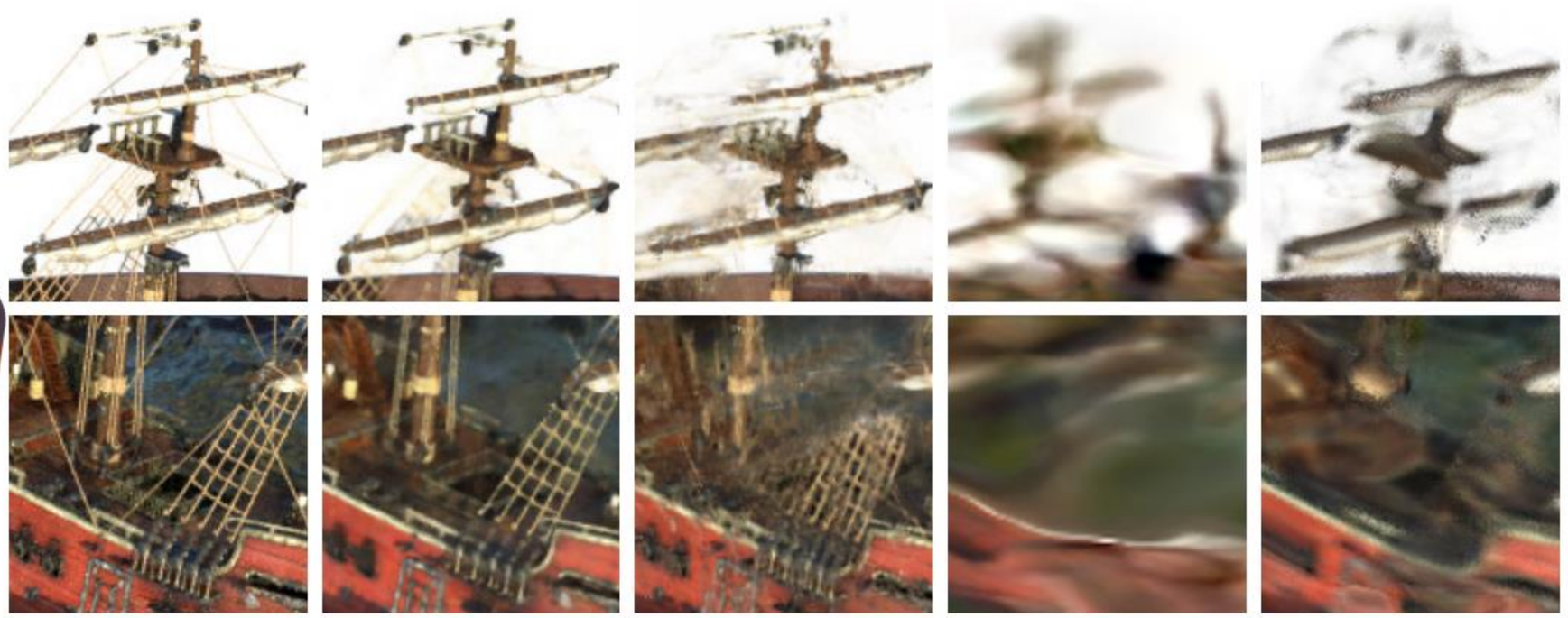
Scene geometry can be approximated using threshold



Tannick et al. ECCV 20

NeRF - Qualitative Comparison

Ship



Ground Truth | NeRF | LLFF | SRN | NV

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

Limitations of NeRF

1. Scene specific and only static scene can be modeled.

GT image of a dynamic scene

Image generated with NeRF

Limitations of NeRF

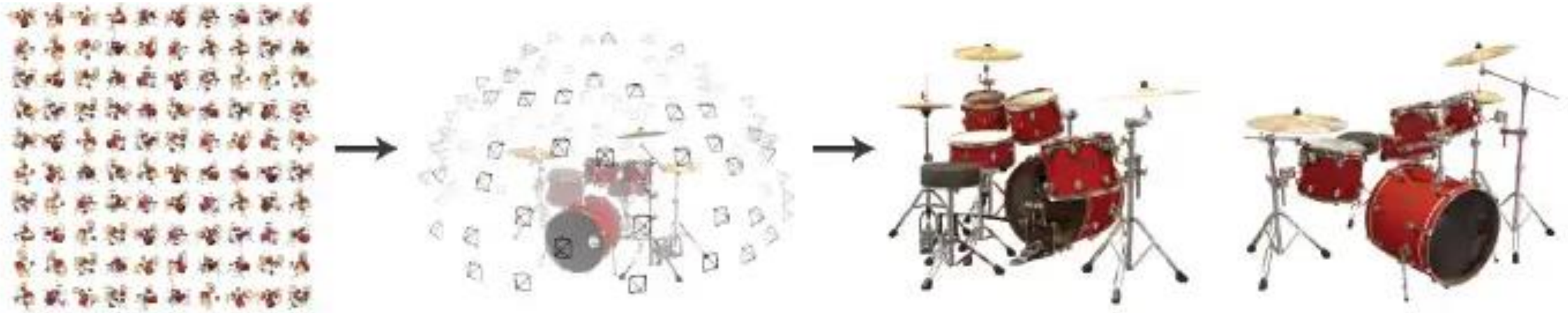
2. No editing and control

- Learned scene cannot be modified.
- Scene is memorized within the network

Limitations of NeRF

3. Generalization

- Scene specific models.
- Large number of images are needed



Limitations of NeRF

4. Expensive training:

- Training is slow(10 hours-up to few days)
- Inference is also not real time

Limitations of NeRF

5. Surface extracted is not accurate and depends on threshold.

GT image NeRF (density threshold $\sigma = 50$)

$\sigma = 1$ $\sigma = 10$ $\sigma = 50$ $\sigma = 100$ $\sigma = 500$

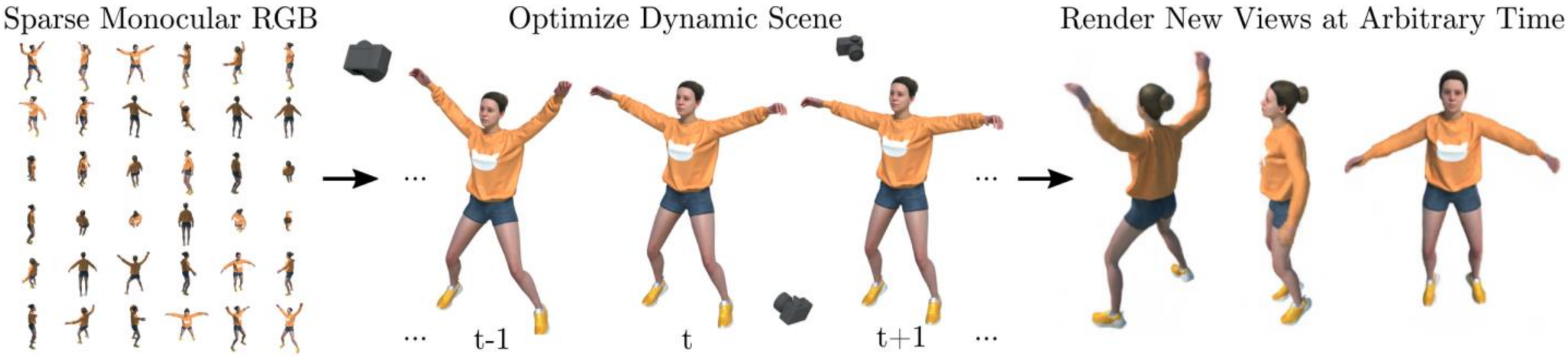
Limitations of NeRF

1. Scene specific and only static scene can be modeled.

GT image of a dynamic scene

Image generated with NeRF

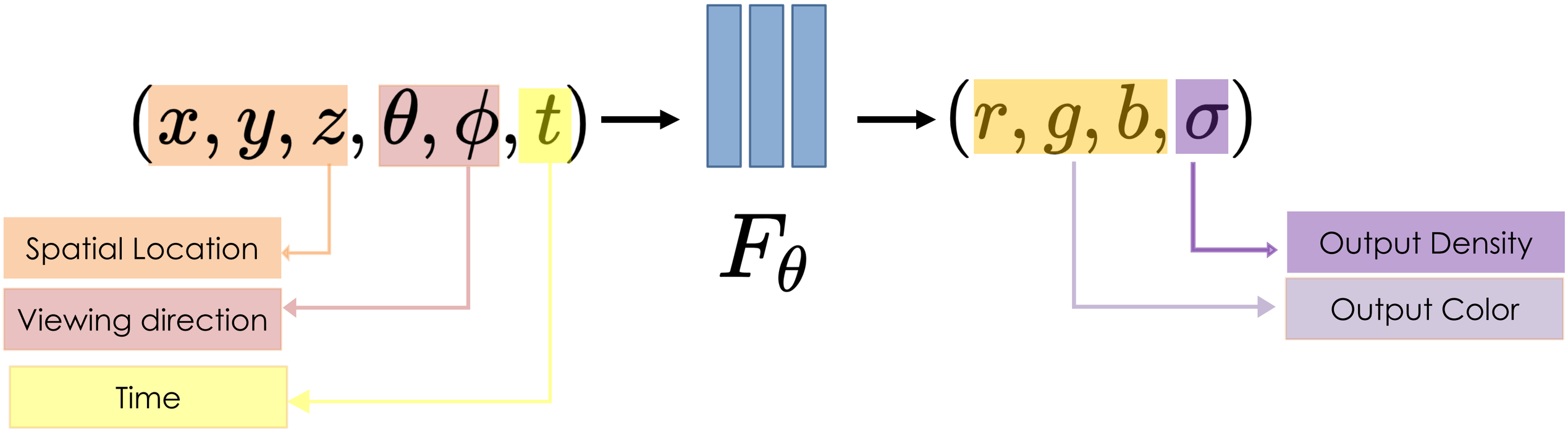
What about dynamic scenes?



Pumarola et al. CVPR'21

What about dynamic scenes?

Learn radiance field given 3d point, viewing direction and time



Can we learn this mapping directly using NeRF?

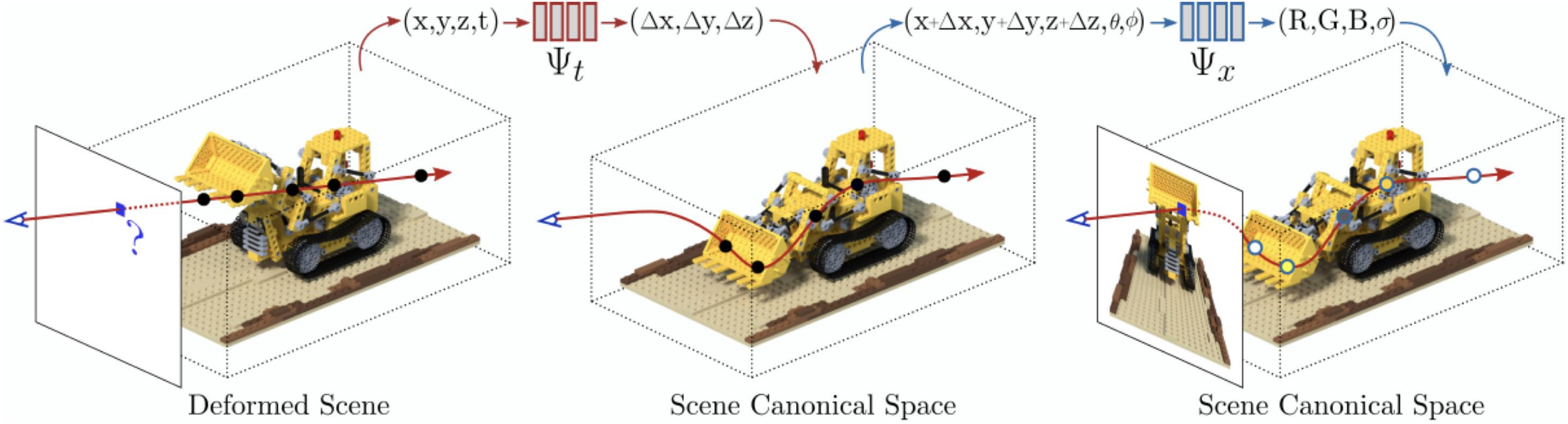
Learn radiance field given 3d point, viewing direction and time

Ground truth NeRF + time

Pumarola et al. CVPR'21

Proposed solution: D-NeRF

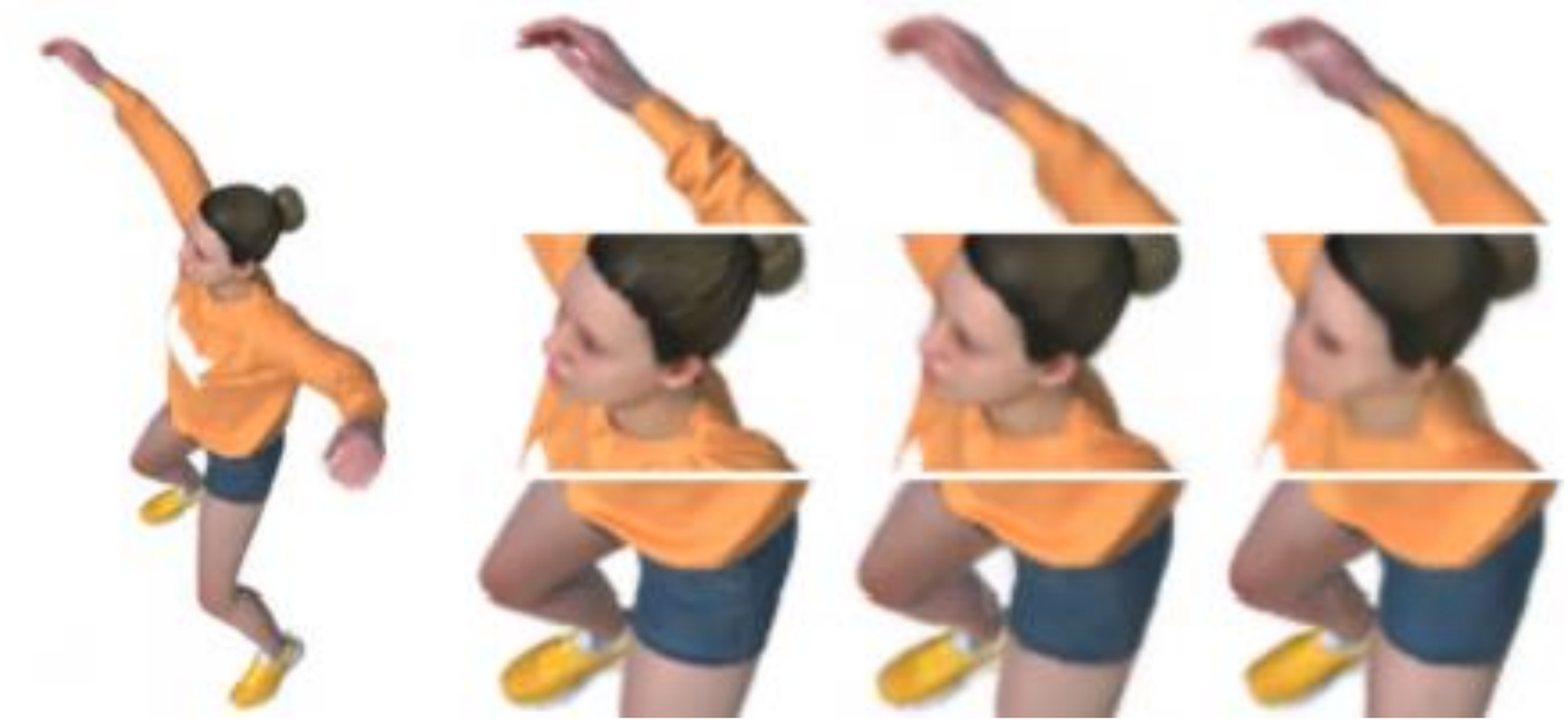
Learn canonical shape and radiance field in the canonical shape



Pumarola et al. CVPR'21

Proposed solution: D-NeRF

Learn canonical shape and radiance field in the canonical shape



Ground truth D-NeRF NeRF + time

Pumarola et al. CVPR‘21

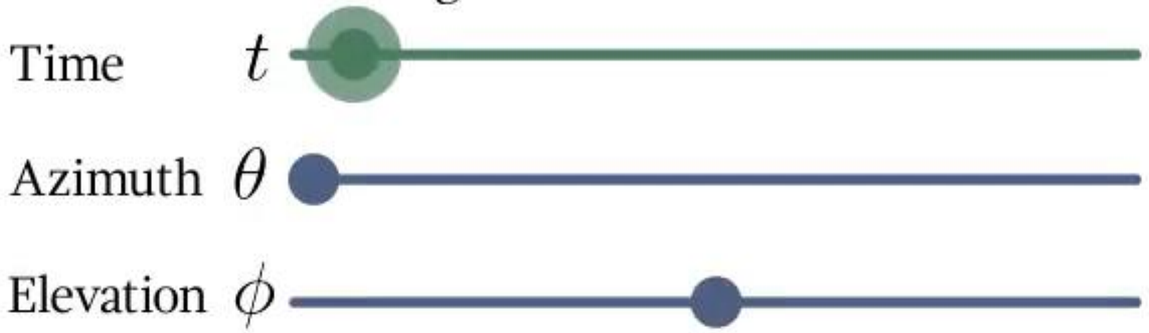
Synthesis Results

D-NeRF

Closest Input View

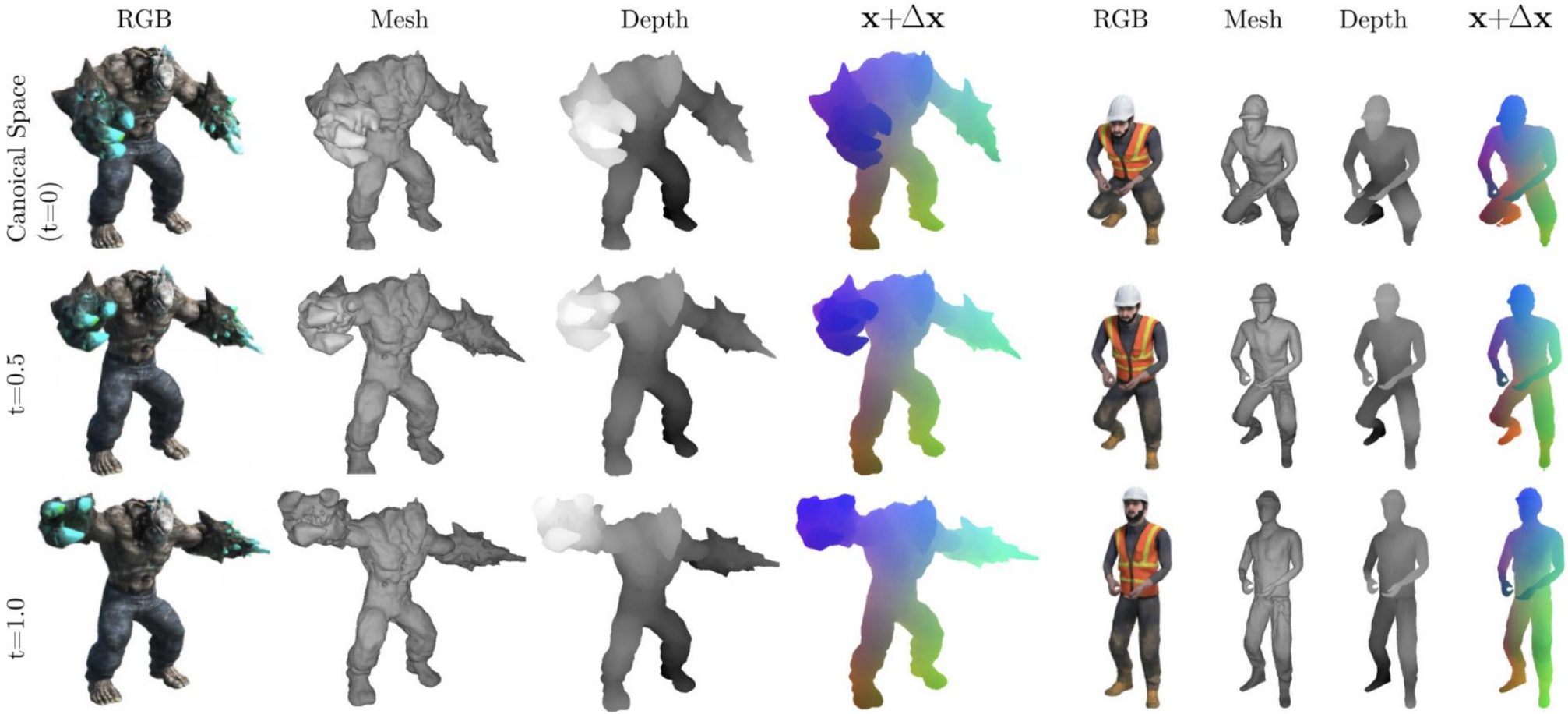
Closest Input Time

Time & View Conditioning:



Pumarola et al. CVPR'21

D-NeRF: Visualization of learned scene representation



Pumarola et al. CVPR'21

Conclusion:Dynamic Scenes with D-NeRF

- Disentangle time dependent deformation from neural rendering network.

Conclusion:Dynamic Scenes with D-NeRF

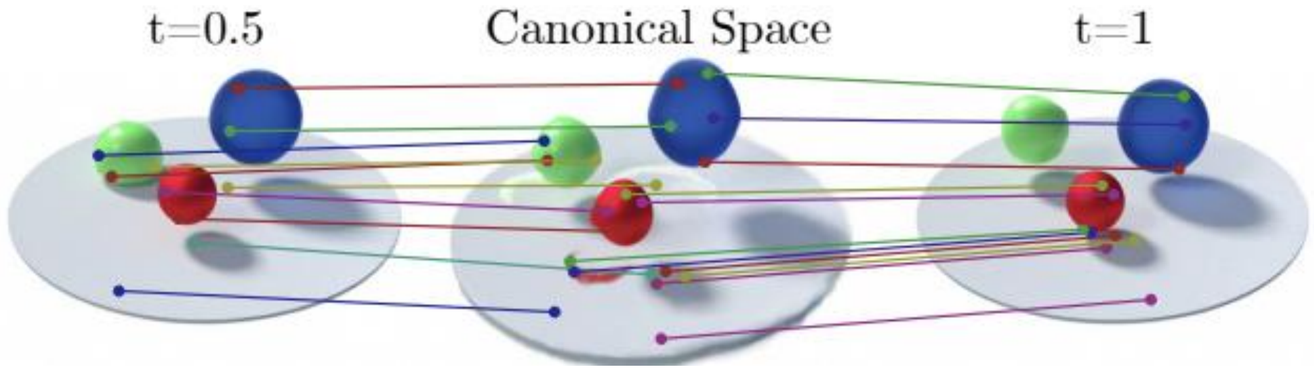
- Disentangle time dependent deformation from neural rendering network.
- Correspondence between canonical shape and deformed shape is defined by

D-NeRF Canonical Mapping
(color-coded as $\mathbf{x} + \Delta\mathbf{x}$)

Conclusion:Dynamic Scenes with D-NeRF

- Disentangle time dependent deformation from neural rendering network.
- Correspondence between canonical shape and deformed shape is defined by
- Time varying shading effects are modeled.

D-NeRF Canonical Mapping
(color-coded as $\mathbf{x} + \Delta\mathbf{x}$)



Limitations of NeRF

1. Scene specific and only static scene can be modeled.

GT image of a dynamic scene

Image generated with NeRF

NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections

NeRF-W extends Neural Radiance Fields (NeRF) to handle real-world internet photos, which often violate NeRF's assumptions due to:

- Photometric variations (lighting, exposure, white balance differences)
- Transient objects (people, cars, occlusions)

NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections
Ricardo Martin-Brualla and Noha Radwan and Mehdi S. M. Sajjadi and Jonathan T. Barron and Alexey Dosovitskiy and Daniel Duckworth

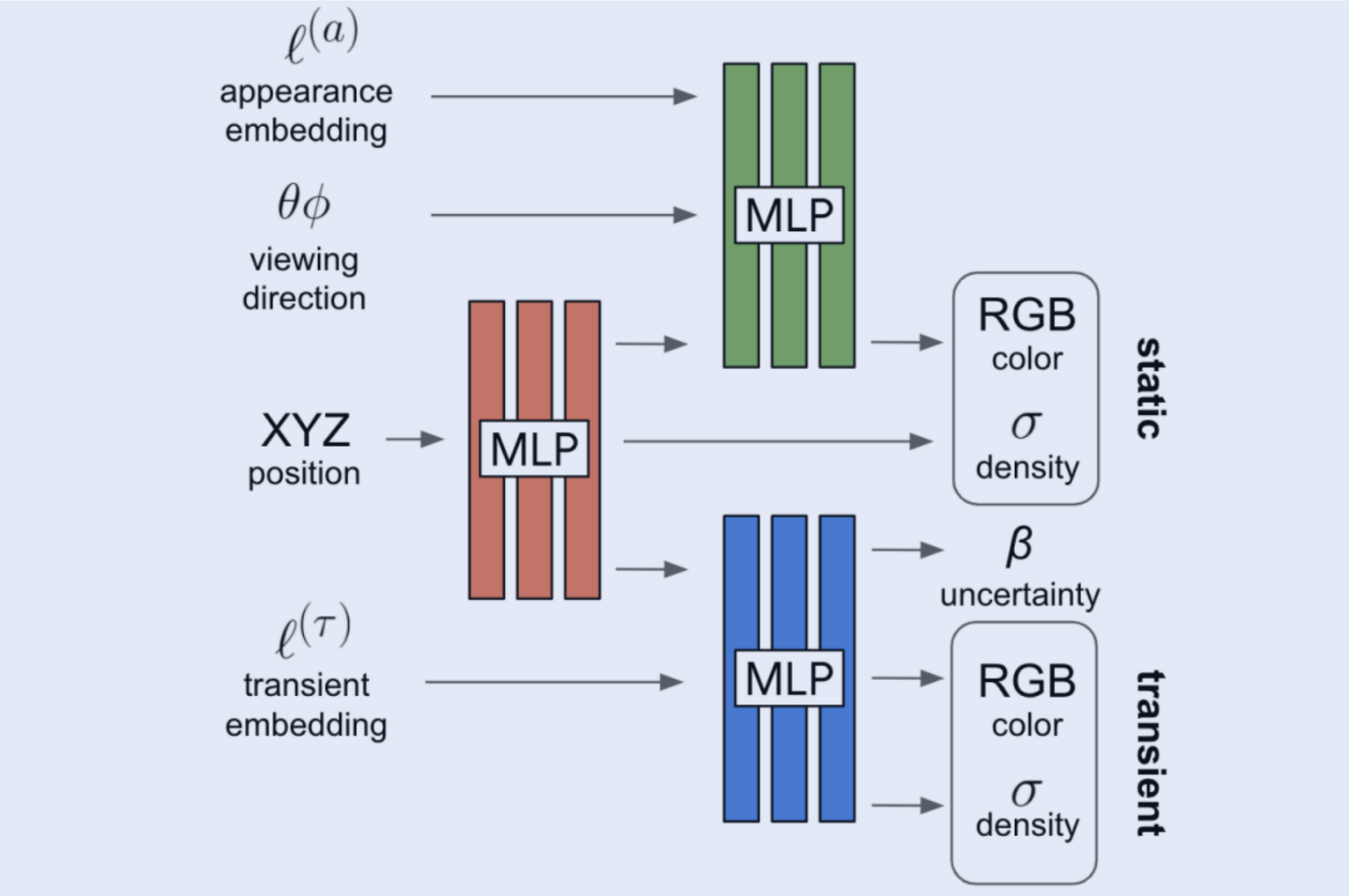
NeRF-W (2021) - In-the-Wild Robustness

- Introduced per-image appearance embeddings and transient uncertainty fields, handling lighting changes and occlusions.
- Achieved better performance on unconstrained PhotoTourism datasets through learned robustness.



NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections
Ricardo Martin-Brualla and Noha Radwan and Mehdi S. M. Sajjadi and Jonathan T. Barron and Alexey Dosovitskiy and Daniel Duckworth

NeRF-W - Architecture



NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections
Ricardo Martin-Brualla and Noha Radwan and Mehdi S. M. Sajjadi and Jonathan T. Barron and Alexey Dosovitskiy and Daniel Duckworth

NeRF-W - Components

Latent Appearance Modeling

- Each image gets an appearance embedding vector (trained alongside the model).
- Modifies NeRF's radiance output to depend on per-image lighting and post-processing variations while keeping the geometry static
- Enables smooth interpolation between lighting conditions

NeRF-W replace the image-independent radiance with an image-dependent radiance, , which also introduces a dependency on image index i to the approximated pixel color $c_i(t)$ $\hat{C}_i$:

$$\hat{C}_i(r) = R(r, c_i, \sigma) = \sum_{k=1}^{K} T(t_k)\alpha(\sigma(t_k)\delta_k)c_i(t_k) \quad where\, T(t_k) = \exp\left(-\sum_{k'=1}^{k-1}\sigma(t_{k'})\delta_{k'}\right)$$

NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections
Ricardo Martin-Brualla and Noha Radwan and Mehdi S. M. Sajjadi and Jonathan T. Barron and Alexey Dosovitskiy and Daniel Duckworth

NeRF-W - Components

Latent Appearance Modeling

- Each image gets an appearance embedding vector (trained alongside the model).
- Modifies NeRF's radiance output to depend on per-image lighting and post-processing variations while keeping the geometry static
- Enables smooth interpolation between lighting conditions

NeRF-W replace the image-independent radiance with an image-dependent radiance, , which also introduces a dependency on image index i to the approximated pixel color $c_i(t)$ $\hat{C}_i$:

$$\hat{C}_i(r) = R(r, c_i, \sigma) = \sum_{k=1}^{K} T(t_k)\alpha(\sigma(t_k)\delta_k)c_i(t_k) \quad where\ c_i(t) = \mathrm{MLP}_{\theta_2}\left(z(t), \gamma_d(d), \ell_i^{(a)}\right)$$

NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections
Ricardo Martin-Brualla and Noha Radwan and Mehdi S. M. Sajjadi and Jonathan T. Barron and Alexey Dosovitskiy and Daniel Duckworth

NeRF-W (2021) - In-the-Wild Robustness

NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections
Ricardo Martin-Brualla and Noha Radwan and Mehdi S. M. Sajjadi and Jonathan T. Barron and Alexey Dosovitskiy and Daniel Duckworth

NeRF-W - Components

Modelling Transient Objects & Uncertainty

- Adds a secondary "transient" MLP head to model moving and occluding objects.
- Renders both **static and transient** components (density + color) but discards transient parts at test time.
- Predicts per-ray uncertainty (β) to downweight unreliable pixels

The expected color of r(t) then becomes the alpha composite of both the static component $\sigma(t), c_i(t)$ and the transient component $\sigma_i^{(\tau)}(t), c_i^{(\tau)}(t)$

$$\hat{C}_i(r) = \sum_{k=1}^{K} T_i(t_k) \left[\alpha \left(\sigma(t_k)\delta_k \right) c_i(t_k) + \alpha \left(\sigma_i^{(\tau)}(t_k)\delta_k \right) c_i^{(\tau)}(t_k) \right]$$

NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections
Ricardo Martin-Brualla and Noha Radwan and Mehdi S. M. Sajjadi and Jonathan T. Barron and Alexey Dosovitskiy and Daniel Duckworth

NeRF-W - Components

Modelling Transient Objects & Uncertainty

- Adds a secondary "transient" MLP head to model moving and occluding objects.
- Renders both static and transient components (density + color) but discards transient parts at test time.
- Predicts per-ray uncertainty (β) to downweight unreliable pixels

The expected color of r(t) then becomes the alpha composite of both the static component $\sigma(t), c_i(t)$ and the transient component $\sigma_i^{(\tau)}(t), c_i^{(\tau)}(t)$

$$where\ T_i(t_k) = \exp\left(-\sum_{k'=1}^{k-1}\left[\sigma(t_{k'}) + \sigma_i^{(\tau)}(t_{k'})\right]\delta_{k'}\right)$$

NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections
Ricardo Martin-Brualla and Noha Radwan and Mehdi S. M. Sajjadi and Jonathan T. Barron and Alexey Dosovitskiy and Daniel Duckworth

NeRF-W - Summary

(a) Photos (b) Renderings

NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections
Ricardo Martin-Brualla and Noha Radwan and Mehdi S. M. Sajjadi and Jonathan T. Barron and Alexey Dosovitskiy and Daniel Duckworth

NeRF-W - Disadvantages

Sparse View Problems

- Degraded quality in rarely observed scene areas (e.g., ground surfaces)
- Poor reconstruction for oblique viewing angles

Camera Calibration Sensitivity

- Blurry artifacts from incorrect camera pose estimations
- Inherits NeRF's dependency on accurate calibration

Inherited NeRF Weaknesses

- Struggles with specular/reflective surfaces
- High computational cost during training
- Limited generalization to unseen viewpoints

Transient Object Handling

- May leave residual artifacts when removing occlusions
- Uncertainty estimation can mask static scene errors

NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections
Ricardo Martin-Brualla and Noha Radwan and Mehdi S. M. Sajjadi and Jonathan T. Barron and Alexey Dosovitskiy and Daniel Duckworth

Limitations of NeRF

2. No editing and control

- Learned scene cannot be modified.
- Scene is memorized within the network

Control-NeRF (2023) - Scene Manipulation

Hybrid 3D Representation:

- Decouples scene-specific 3D feature volumes from a shared neural rendering network, enabling both high-quality novel view synthesis and scene editing.

Scene-Agnostic Rendering:

- A single rendering network generalizes across scenes, allowing new scenes to be optimized without retraining the full model.

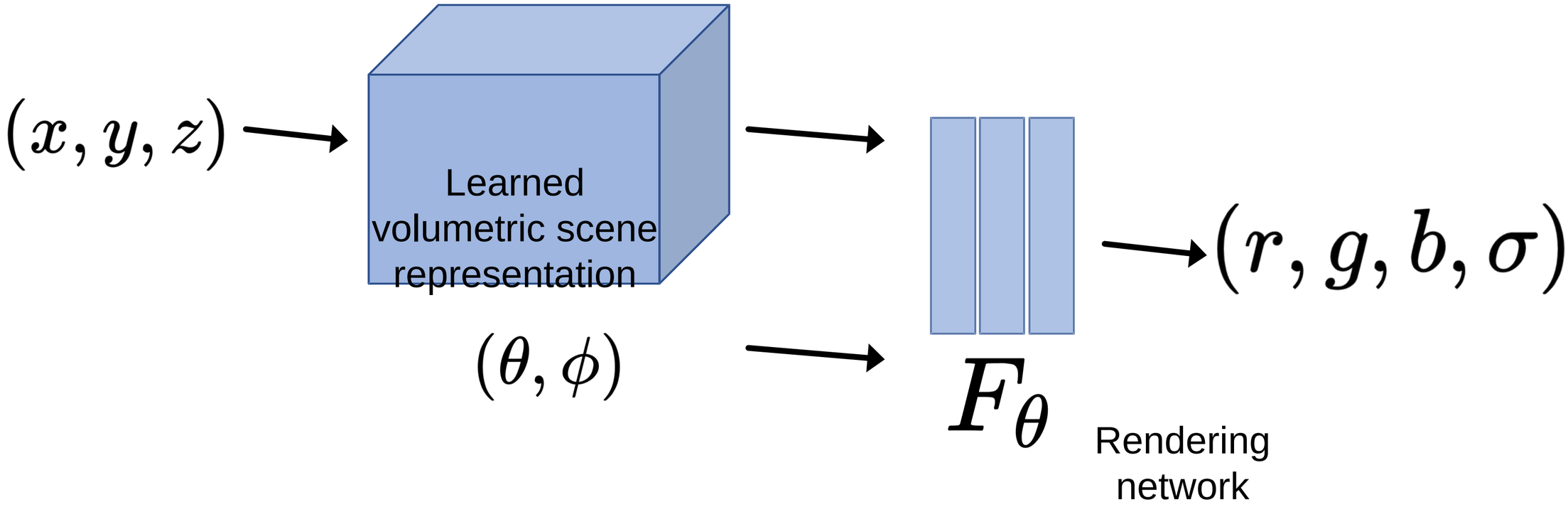
Post-Hoc Scene Manipulation:

- Enables intuitive 3D edits (object insertion, deformation, and scene mixing) by modifying feature volumes without retraining.

Control-NeRF: Editable Feature Volumes for Scene Rendering and Manipulation
Verica Lazova and Vladimir Guzov and Kyle Olszewski and Sergey Tulyakov and Gerard Pons-Moll

Control-NeRF

- **Prior Work**: Scene is memorized within the neural network, which makes compositing of scenes and editing hard.
- **Key Idea**: Decouple scene representation from neural rendering network.



Control-NeRF: Editable Feature Volumes for Scene Rendering and Manipulation
Verica Lazova and Vladimir Guzov and Kyle Olszewski and Sergey Tulyakov and Gerard Pons-Moll

Control-NeRF

1. Scene representation:

- Given a set of input images $\mathcal{I} = \{I_s^i\}_{i=1}^{N_s}$ from M training scenes
- Where $s \in \mathcal{S}$ is set of training scenes and $M = |\mathcal{S}|$
- Scene representation network learns a volumetric feature
- where $W \times H \times D$ is the spatial resolution of grid which feature vector of length F.

$V_s \in \mathbf{R}^{WHDF}$

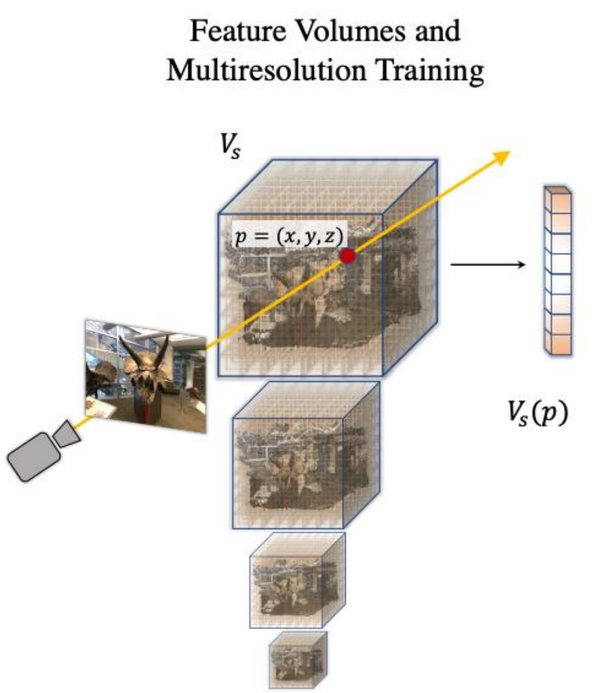
Learned volumetric feature for scene s

$W \times H \times D$ is the spatial resolution

F is length of feature vector

Control-NeRF

2. Neural rendering network with feature volumes

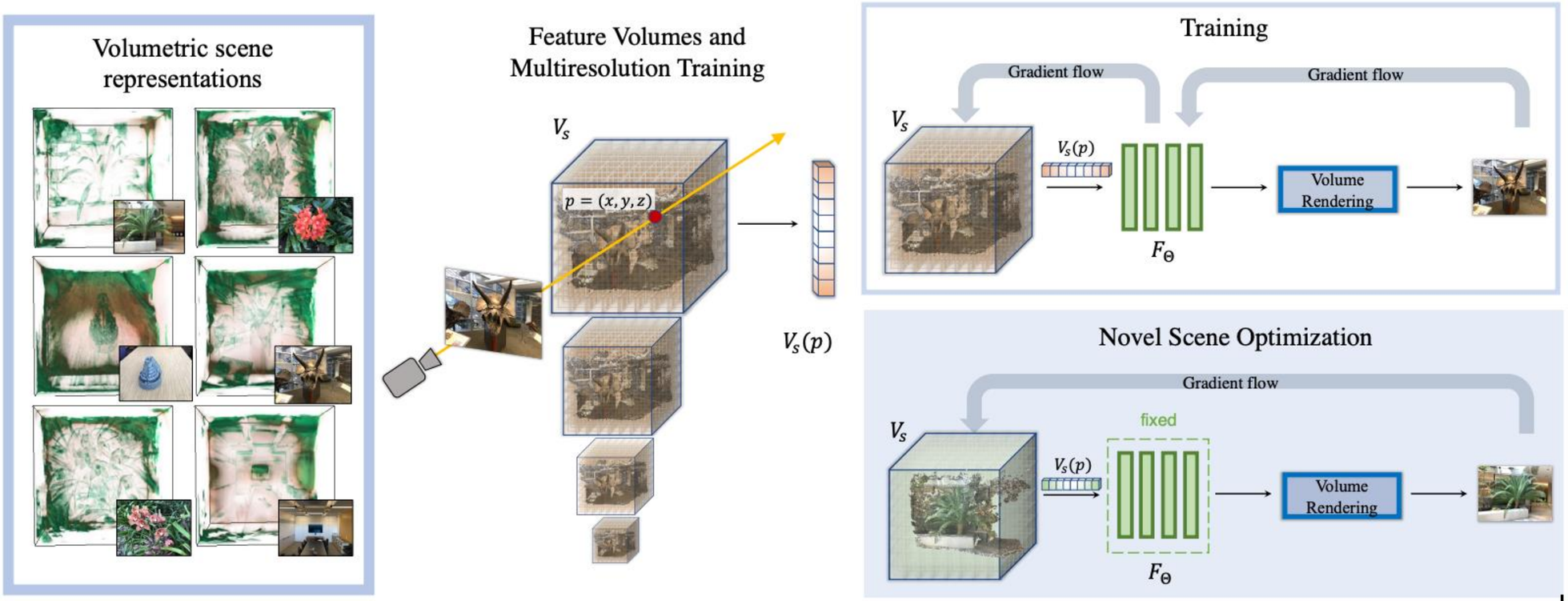


$V_s(p)$:Volumetric feature at query point p

Lazova et al. WACV‘23

Control-NeRF

Training and Inference



Lazova et al. WACV‘23

Control-NeRF: Training

1. **Multi-resolution Volume training**
   - Hierarchical training process is used to compute the volumes in a coarse-to-fine manner.
   - Train low resolution(16^3) volume till convergence.
   - Upsample the learnt feature volume and train till convergence

- Improved training time.
- High-quality image synthesis and manipulation.

Lazova et al. WACV‘23

Control-NeRF: Training

1. Multi Scene training

- Efficient training strategy: Sample one training scene and train for N iterations, before saving the volume grid

Lazova et al. WACV‘23

Control-NeRF: Training

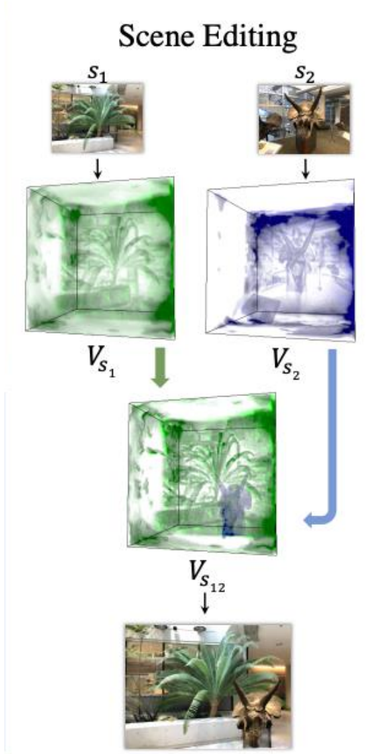
1. Generalization to Novel Scenes

- Fix neural rendering network and learn feature volume for novel scene.
- Given sufficient training scenes, the learnt radiance function can be applied to optimize for novel scenes more efficiently.

Lazova et al. WACV‘23

Control NeRF: Scene editing and manipulation

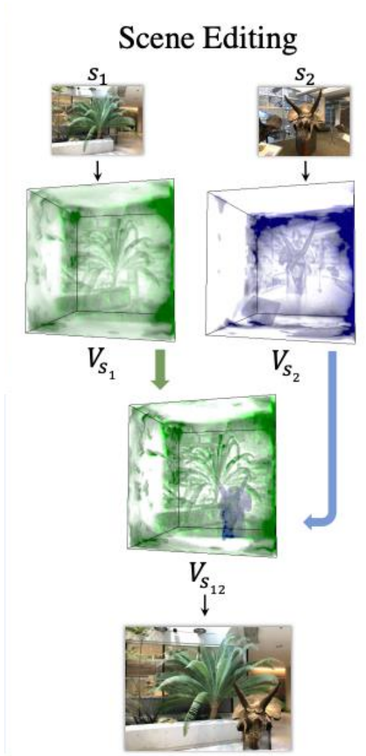
- Scene editing and composting with Control-NeRF:

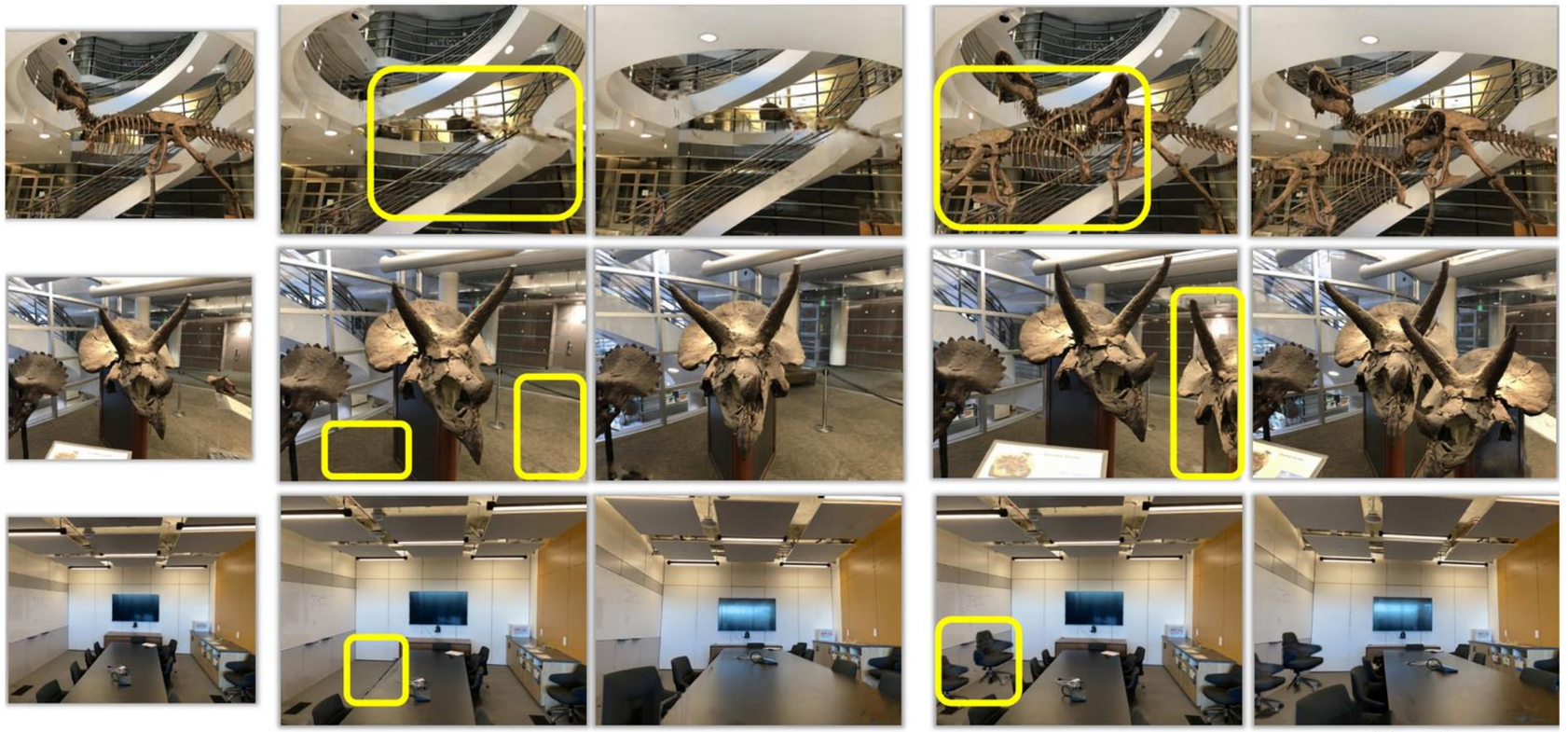


Lazova et al. WACV‘23

Control NeRF: Scene editing and manipulation

- Scene editing and composting with Control-NeRF:



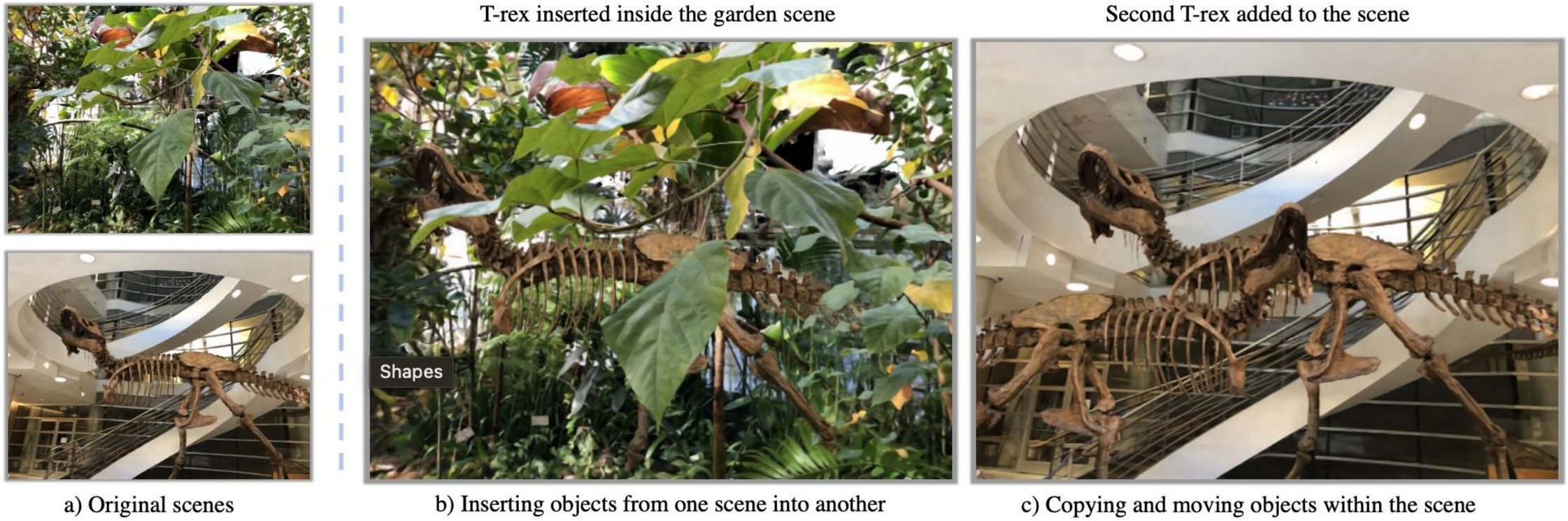


Lazova et al. WACV‘23

Goal: Scene Editing

Control-NeRF (2023) - Scene Manipulation

Control NeRF for scene manipulation and rendering



Control-NeRF: Editable Feature Volumes for Scene Rendering and Manipulation
Verica Lazova and Vladimir Guzov and Kyle Olszewski and Sergey Tulyakov and Gerard Pons-Moll

Limitations of NeRF

4. Expensive training:

- Training is slow(10 hours-up to few days)
- Inference is also not real time

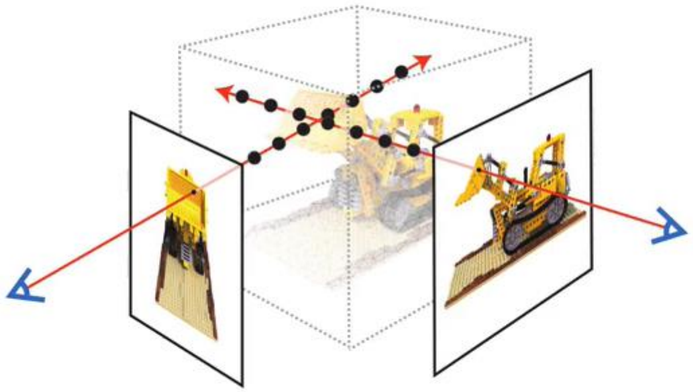
Instant Neural Graphics Primitives with a Hash Encoding (2022)

- **Train in minutes** (instead of days).
- **Encode high-frequency details** more compactly.
- **Render interactively** (tens of frames per second

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding
Thomas Mueller and Alex Evans and Christoph Schied and Alexander Keller

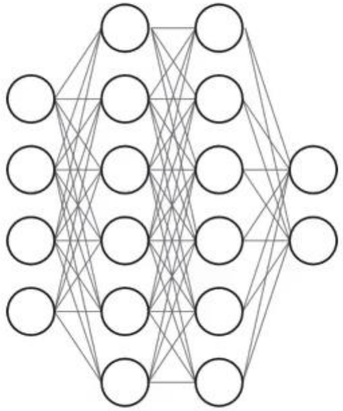
3 Pillars of Instant Neural Graphics Primitives

Rendering/training algorithm



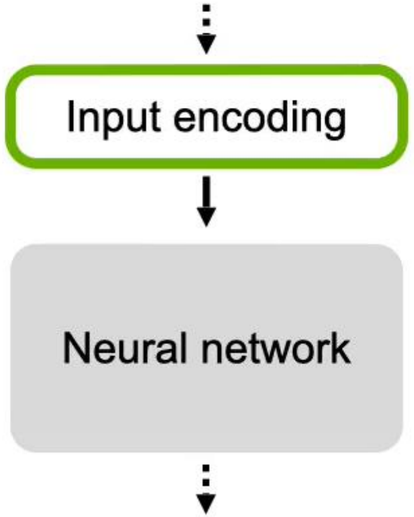
- Task-specific GPU implementation
- **10-100x fewer steps than naïve tensor-based approach**

Small neural network



- Fully fused implementation
- **5-10x faster than TensorFlow**

“Good” input encoding

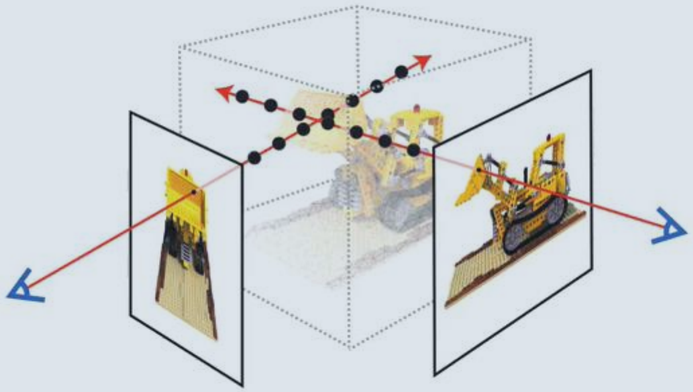


- Multiresolution hash encoding
- **Better speed-vs-quality tradeoff than prior work**
- **Task agnostic**

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding
Thomas Mueller and Alex Evans and Christoph Schied and Alexander Keller

Making Neural Graphics Primitives Instant

Rendering/training algorithm



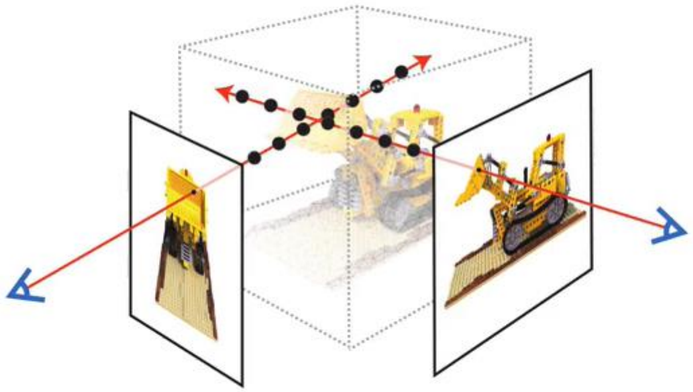
- Task-specific GPU implementation
- 10-100x fewer steps than naïve tensor-based approach

- Implements a dedicated rendering and training algorithm
- Skips all empty space until the object surface is hit
- Utilizes 10- 100x fewer steps than naive dense stepping
- A small number of steps leads to a smaller number of network queries

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding
Thomas Mueller and Alex Evans and Christoph Schied and Alexander Keller

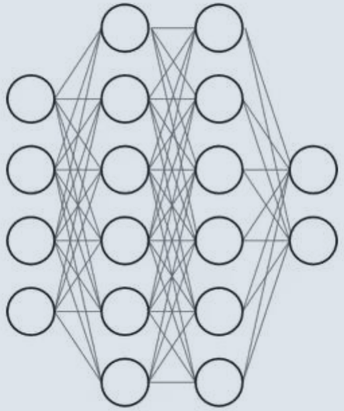
Making Neural Graphics Primitives Instant

Rendering/training algorithm



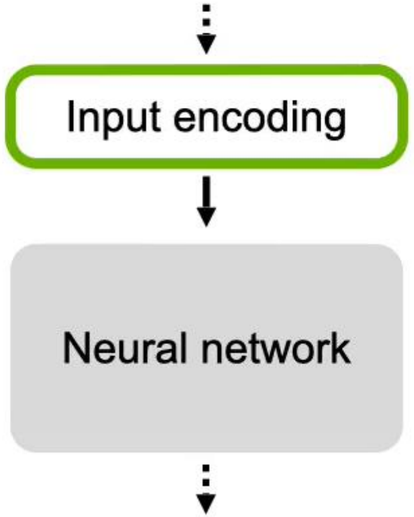
- Task-specific GPU implementation
- **10-100x fewer steps than naïve tensor-based approach**

Small neural network



- Fully fused implementation
- **5-10x faster than TensorFlow**

"Good" input encoding

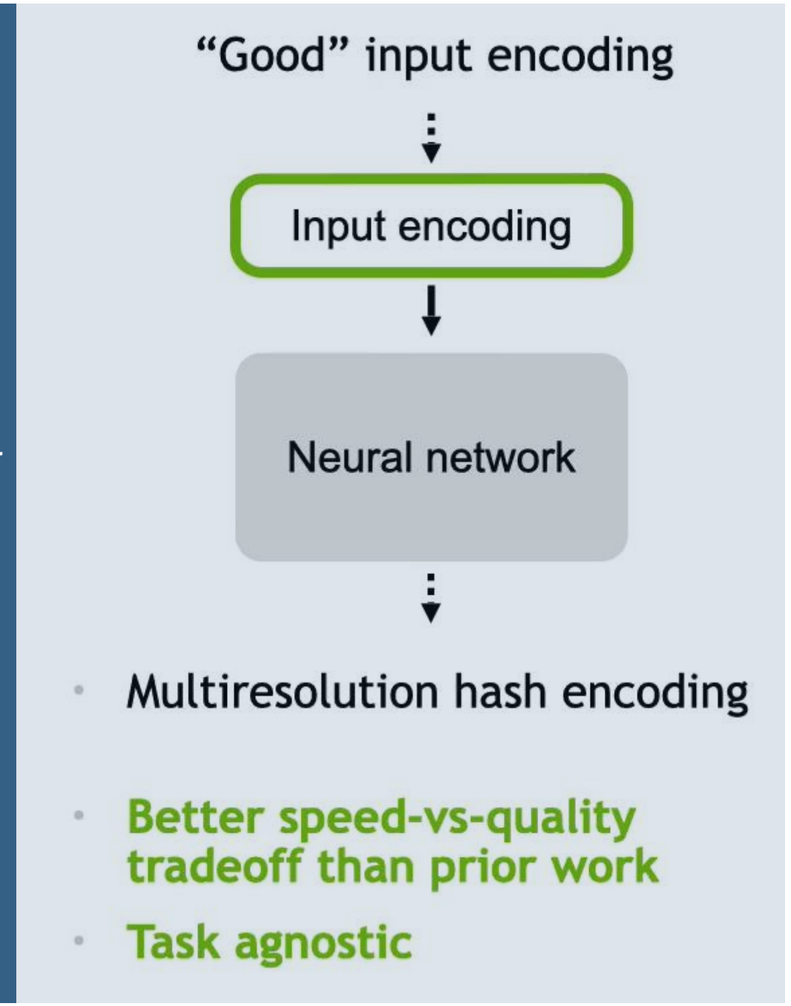


- Multiresolution hash encoding
- **Better speed-vs-quality tradeoff than prior work**
- **Task agnostic**

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding
Thomas Mueller and Alex Evans and Christoph Schied and Alexander Keller

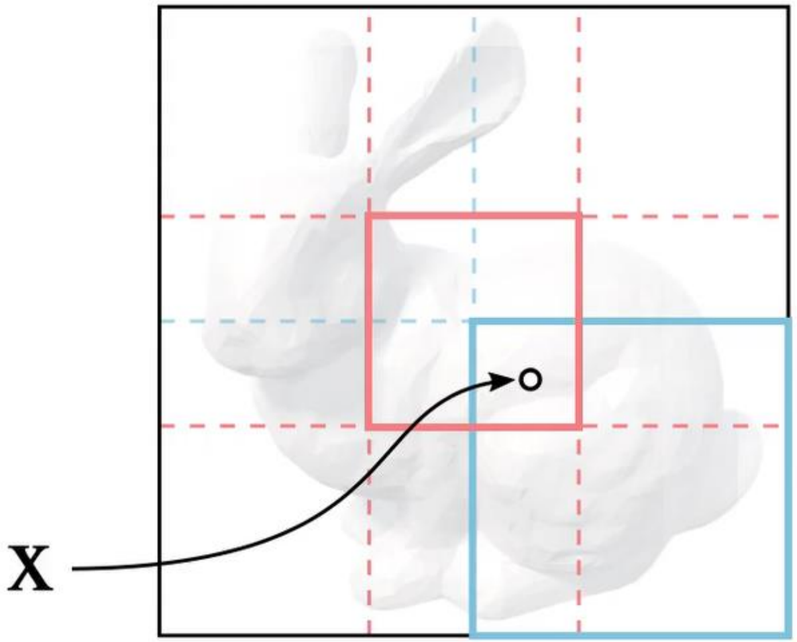
Making Neural Graphics Primitives Instant

- Builds on the prior work to make input encoding trainable, along with the weight of the network
- Builds a more general input encoding scheme using a multi-resolution hash grid
- Enables Neural Graphics Primitives to capture multi-resolution details while achieving speed-ups over dense grid structure



Instant Neural Graphics Primitives with a Multiresolution Hash Encoding
Thomas Mueller and Alex Evans and Christoph Schied and Alexander Keller

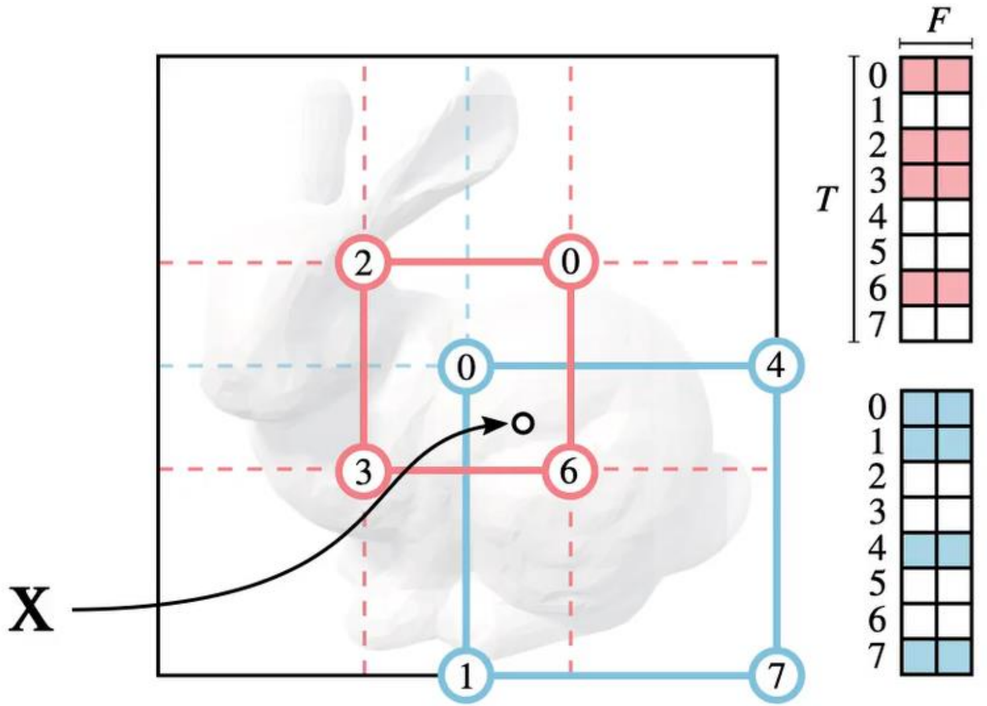
Making Neural Graphics Primitives Instant



- Why multi-resolution?
  - Automatic level of detail

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding
Thomas Mueller and Alex Evans and Christoph Schied and Alexander Keller

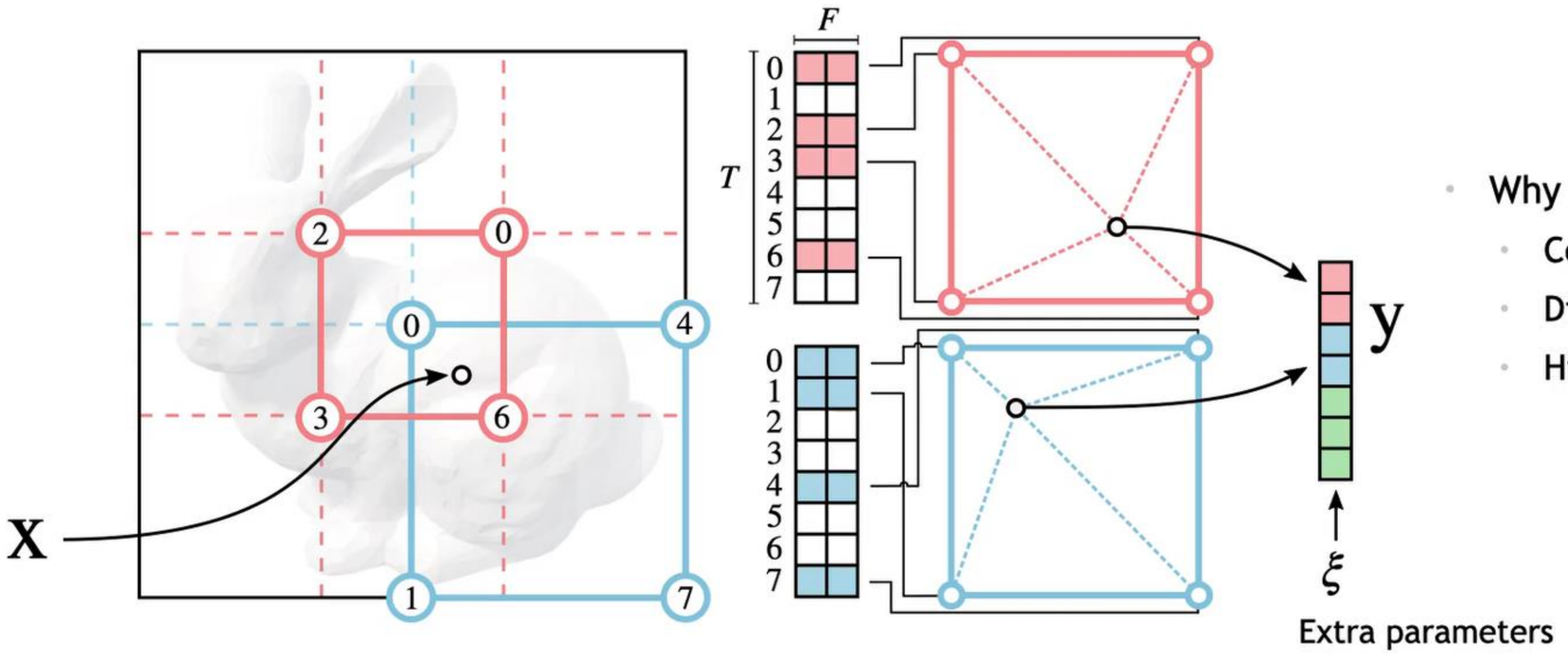
Making Neural Graphics Primitives Instant



Hashing: instead of allocating R^3 vectors, you allocate a much smaller table of size T and use a spatial hash function to map each grid cell coordinate to one of those T “buckets.”
Each bucket stores a learned feature vector

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding
Thomas Mueller and Alex Evans and Christoph Schied and Alexander Keller

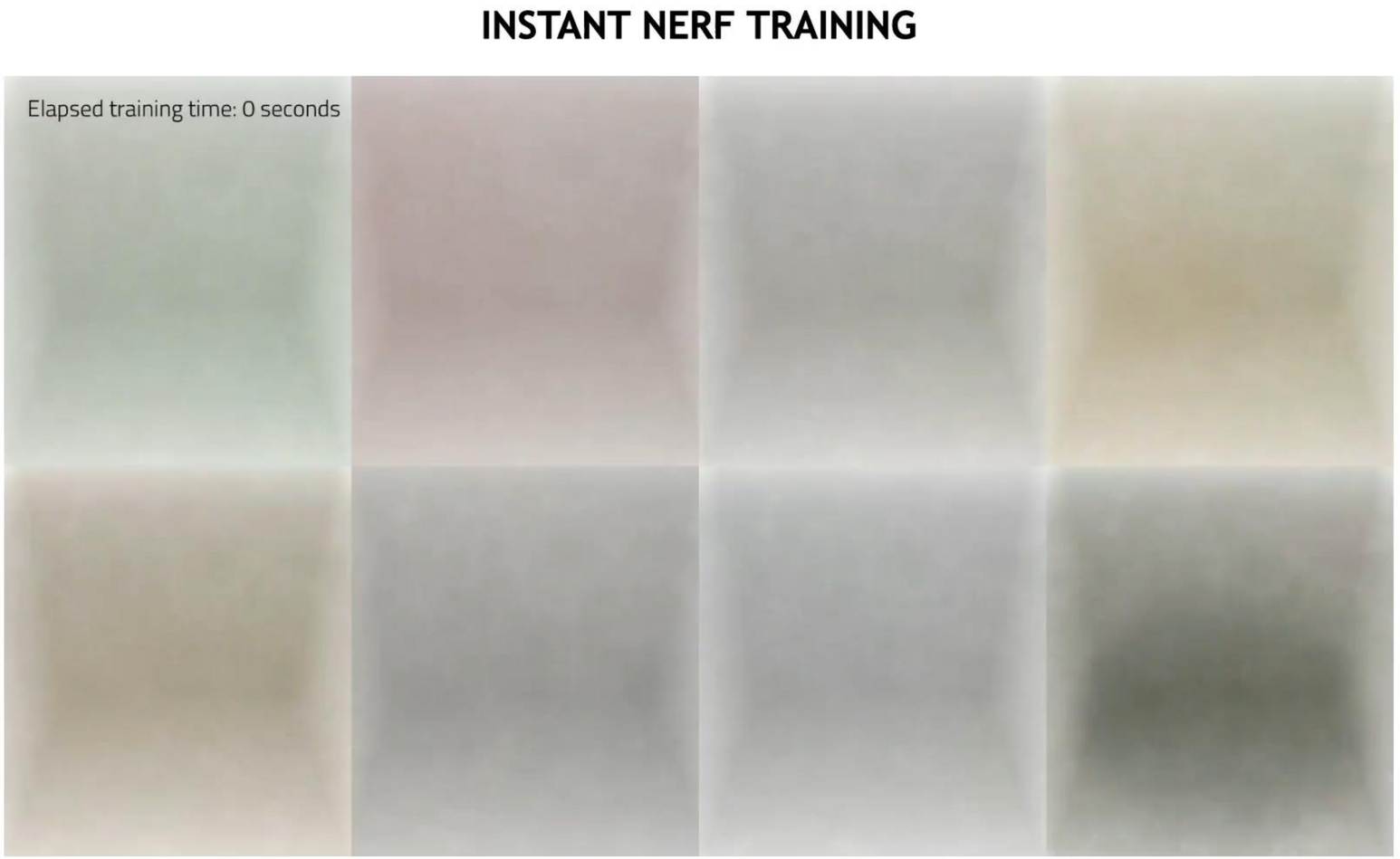
Making Neural Graphics Primitives Instant



- Why linear interpolation
  - Continuity
  - Differentiability
  - Higher order interpolation?

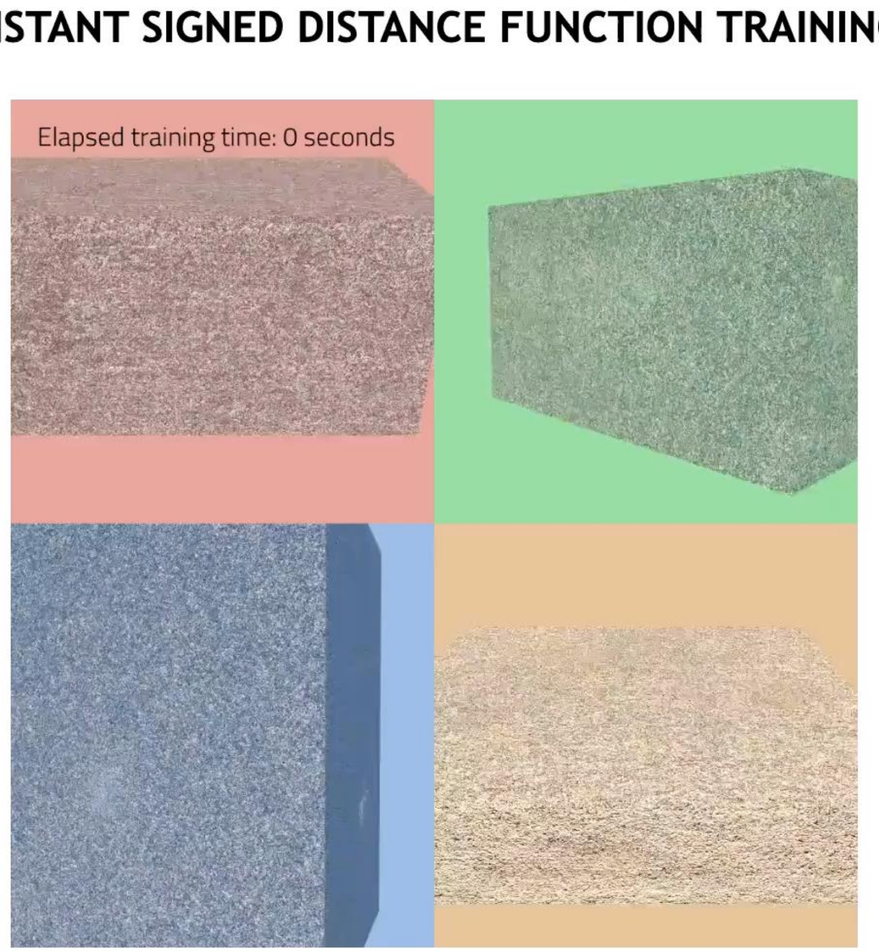
Instant Neural Graphics Primitives with a Multiresolution Hash Encoding
Thomas Mueller and Alex Evans and Christoph Schied and Alexander Keller

Instant NeRF training



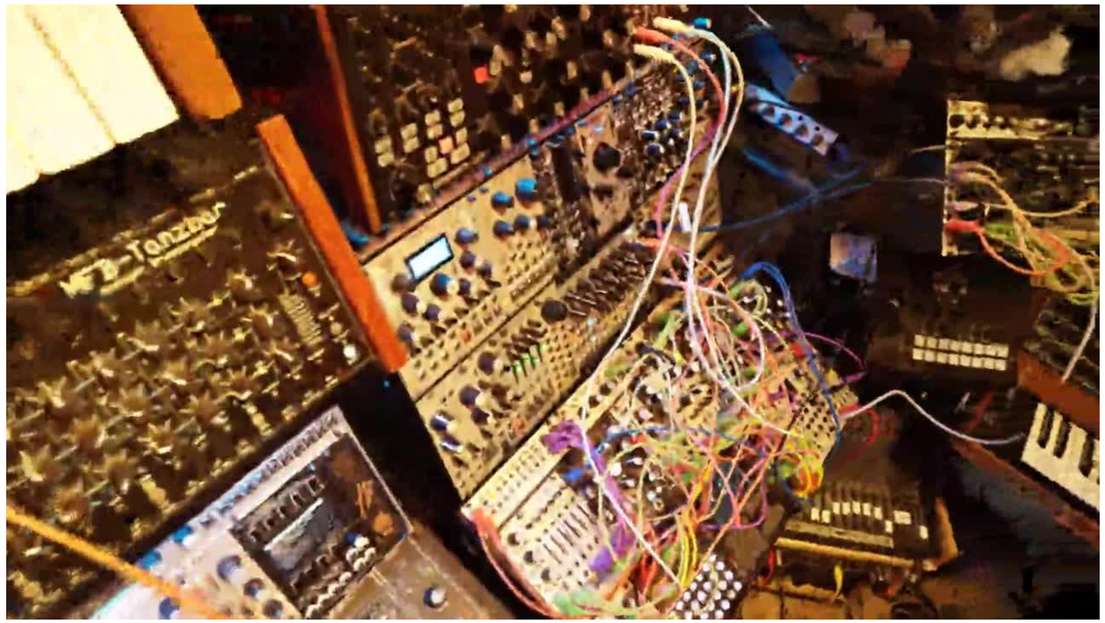
Mueller et al. SIGGRAPH‘22

Instant SDF training



Mueller et al. SIGGRAPH‘22

Instant Neural Graphics Primitives with a Hash Encoding (2022)



Instant Neural Graphics Primitives with a Multiresolution Hash Encoding
Thomas Mueller and Alex Evans and Christoph Schied and Alexander Keller

Instant Neural Graphics Primitives with a Hash Encoding - Disadvantages

Encoding Trade-offs:

- Concatenating multiresolution features enhances parallelism but increases memory and computational costs, while reducing the risk of losing detail.

Hash Function Limitations:

- Simple hashing is fast but lacks coherence, while advanced methods (e.g., PCG32) add overhead without clear quality improvements.

Microstructure Artifacts:

- Hash collisions cause grainy noise in outputs (e.g., SDFs), requiring filtering or smoothness priors for mitigation.

Future Optimization Needs:

- Differentiable hashing or evolutionary methods could improve hash functions, while sparse volumetric data (e.g., clouds) remains an open challenge.

Neural Radiance Fields: The Evolution of 3D Reconstruction

Overview of concepts:

- NeRF (2020) - The Foundational Breakthrough
- DNerf (2021)- Dynamic Nerf
- HyperNeRF (2021) - Topological Dynamics
- NeRF-W (2021) - Unconstrained Capture
- Control-NeRF (2023) - Scene Manipulation
- Instant-NGP (2022) – Training Speed

Extra slides

Upcoming ...

3D Gaussian Splatting for Real-Time Radiance Field Rendering

SIGGRAPH 2023

(ACM Transactions on Graphics)

Bernhard Kerbl* [1,2] Georgios Kopanas* [1,2] Thomas Leimkühler[3] George Drettakis[1,2]

* Denotes equal contribution

[1]Inria [2]Université Côte d'Azur [3]MPI Informatik

UniSurf (2021)- Unifying NeRFs with neural implicit surfaces

Surface extracted with Neural Radience Fields is not accurate and depends on the threshold.

GT image NeRF (density threshold $\sigma = 50$)

$\sigma = 1$ $\sigma = 10$ $\sigma = 50$ $\sigma = 100$ $\sigma = 500$

Neural Radiance Fields: The Evolution of 3D Reconstruction

Overview of concepts:

- **NeRF (2020) - The Foundational Breakthrough**
- DNerf (2021)- Dynamic Nerf
- HyperNeRF (2021) - Topological Dynamics
- NeRF-W (2021) - Unconstrained Capture
- Control-NeRF (2023) - Scene Manipulation
- Instant-NGP (2022) – Training Speed

Surface Rendering v/s Volume Rendering

Surface Rendering
(Implicit Surfaces)

\+ High quality geometry
\+ Clear surface definition

\- Mask supervision required
\- Texture mapping is blurry

Surface Rendering v/s Volume Rendering

Surface Rendering
(Implicit Surfaces)

- \+ High quality geometry
- \+ Clear surface definition
- − Mask supervision required
- − Texture mapping is blurry

Volume Rendering
(Radiance fields)

- − Surface is approximated
- \+ Without mask supervision
- \+ High quality novel views and sharp textures/colors

Input image Input mask IDR

Best of both worlds

Surface Rendering
(Implicit Surfaces)

+ High quality geometry
+ Clear surface definition

~~Mask supervision required~~
~~Texture mapping is blurry~~

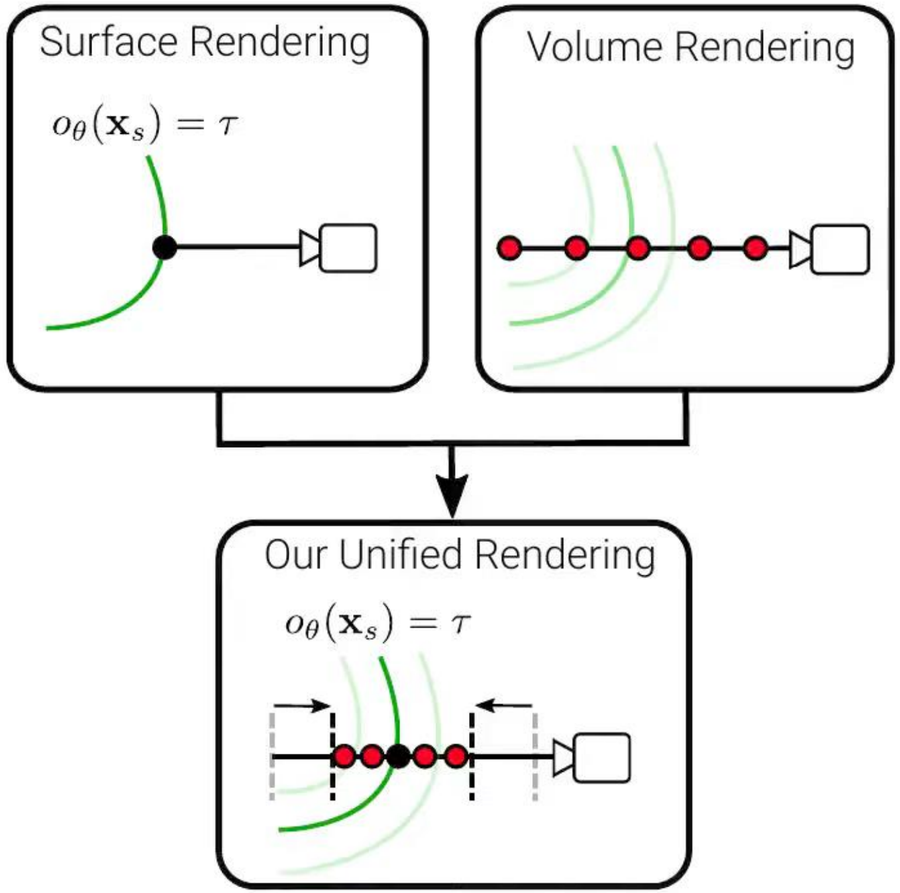
Input image Input mask IDR

Volume Rendering
(Radiance fields)

~~Surface is approximated~~

+ Without mask supervision
+ High quality novel views and sharp textures/colors

Unifying Implicit Surfaces and Radiance Fields



Oechsle et al. ICCV‘21

Unifying Implicit Surfaces and Radiance Fields

NeRF Volume rendering:

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

NeRF Volume rendering with density

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \alpha_i \Pi_{j=1}^{i} (1 - \alpha_j) \mathbf{c}_i$$

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

Oechsle et al. ICCV‘21

Unifying Implicit Surfaces and Radiance Fields

NeRF Volume rendering with density

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \alpha_i \Pi_{j=1}^{i} (1 - \alpha_j) \mathbf{c}_i$$

Oechsle et al. ICCV‘21

Unifying Implicit Surfaces and Radiance Fields

NeRF Volume rendering with density

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \alpha_i \Pi_{j=1}^{i} (1 - \alpha_j) \mathbf{c}_i$$

Key Idea: For solid objects $\alpha_i = 1 - \exp(-\sigma_i \delta_i))$, corresponds to an occupancy field O_i at ***i*** *th* sample.

$$\hat{C}_v(\mathbf{r}) = \sum_{i=1}^{N} o_i \Pi_{j=1}^{i} (1 - o_j) \mathbf{c}_i$$

Oechsle et al. ICCV‘21

Unifying Implicit Surfaces and Radiance Fields

NeRF Volume rendering with density

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \alpha_i \Pi_{j=1}^{i} (1 - \alpha_j) \mathbf{c}_i$$

Key Idea: For solid objects $\alpha_i = 1 - \exp(-\sigma_i \delta_i))$, corresponds to an occupancy field O_i at ***i*** *th* sample.

$$\hat{C}_v(\mathbf{r}) = \sum_{i=1}^{N} o_i \Pi_{j=1}^{i} (1 - o_j) \mathbf{c}_i$$

Given occupancy of surface, we can now render the same scene with surface rendering.

Oechsle et al. ICCV‘21

Unifying Implicit Surfaces and Radiance Fields

Key Idea:

- Volume rendering in early stage:
  - Optimization without mask

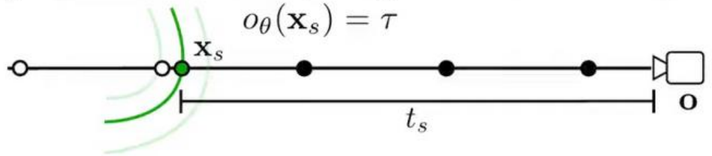
- Surface rendering in later stage:
  - Level-set surfaces

Oechsle et al. ICCV‘21

Unifying Implicit Surfaces and Radiance Fields

Rendering Procedure

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

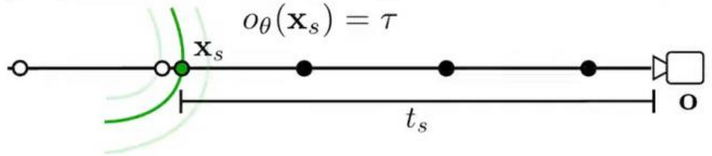


Oechsle et al. ICCV‘21

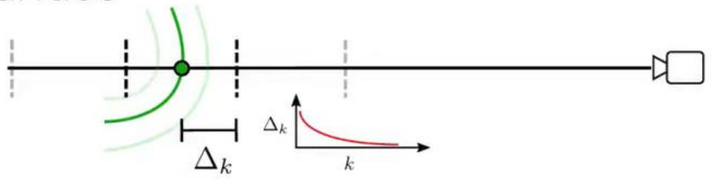
Unifying Implicit Surfaces and Radiance Fields

Rendering Procedure

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



- Define interval at the surface

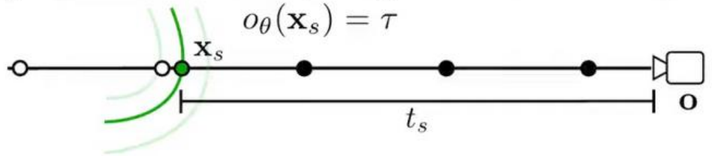


Oechsle et al. ICCV‘21

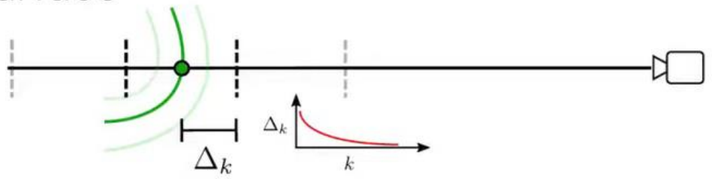
Unifying Implicit Surfaces and Radiance Fields

Rendering Procedure

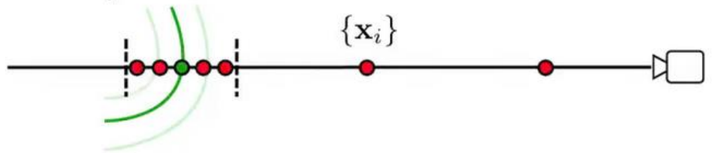
- Find surface along a ray: uniform sampling + iterative secant method



- Define interval at the surface



- Volume rendering with occupancies



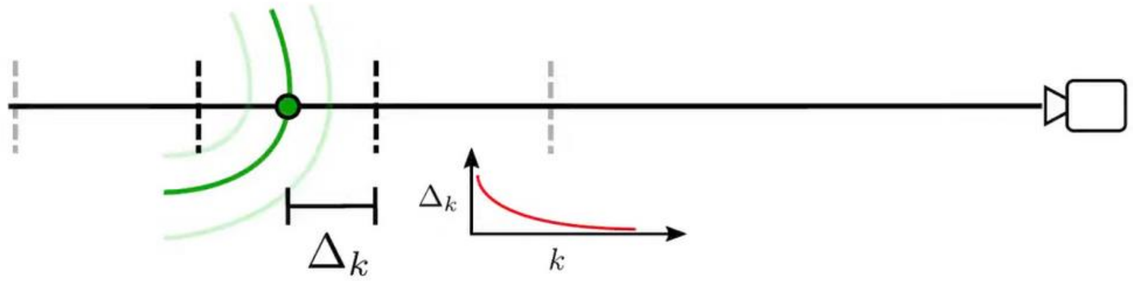
Oechsle et al. ICCV‘21

Unifying Implicit Surfaces and Radiance Fields

Rendering Procedure

Transition from Volume rendering to Surface rendering

- Exponential decay of interval Δ_k wrt. iterations k



Volume Rendering Rendering

Δk decreases

Surface

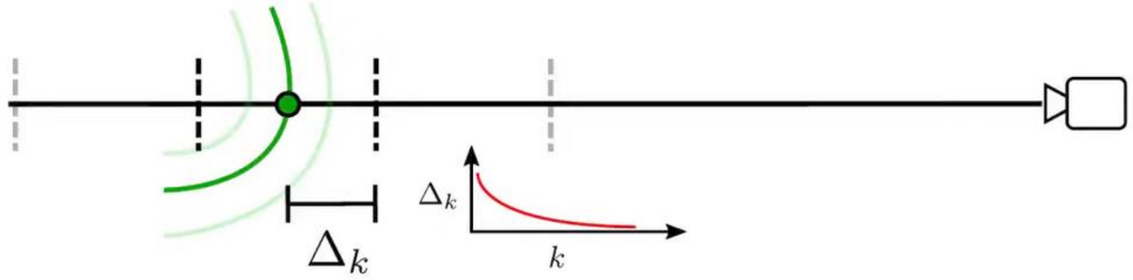
Oechsle et al. ICCV‘21

Unifying Implicit Surfaces and Radiance Fields

Rendering Procedure

Transition from Volume rendering to Surface rendering

- Exponential decay of interval Δ_k wrt. iterations k



Theorem

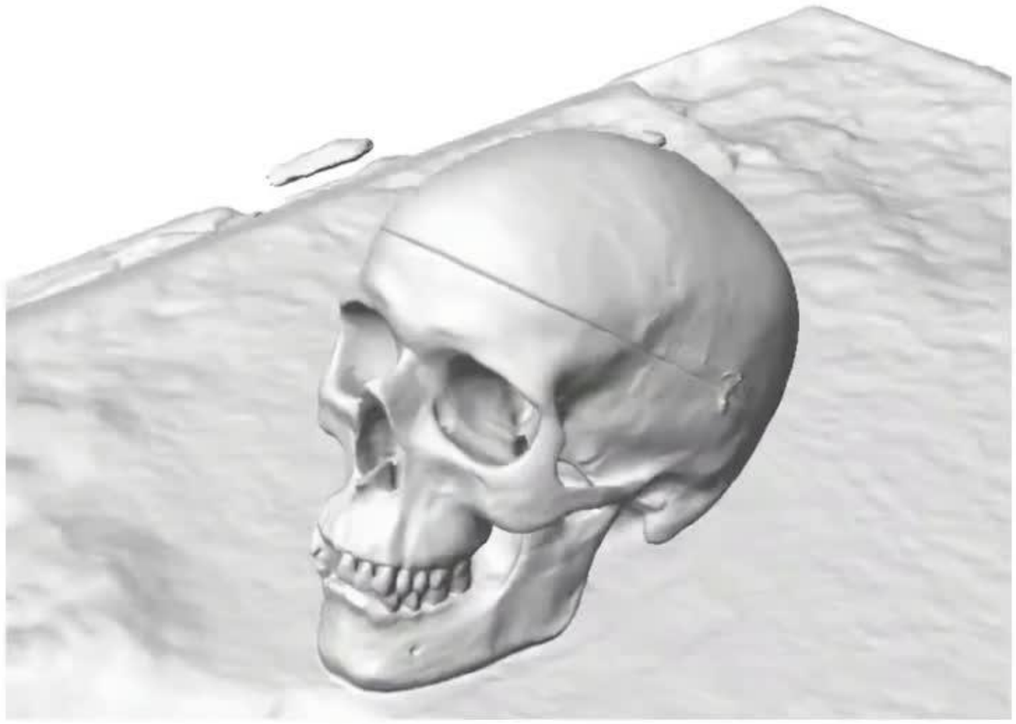
Volume and surface rendering become equivalent when reducing the interval and increasing the number of samples.

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

Oechsle et al. ICCV‘21

Unifying Implicit Surfaces and Radiance Fields

Comparison on the DTU MVS dataset



Ours

Oechsle et al. ICCV‘21