清華大學 Tsinghua University
交叉信息研究院 Institute for Interdisciplinary Information Sciences

Lecture 6: NeRF and Beyond

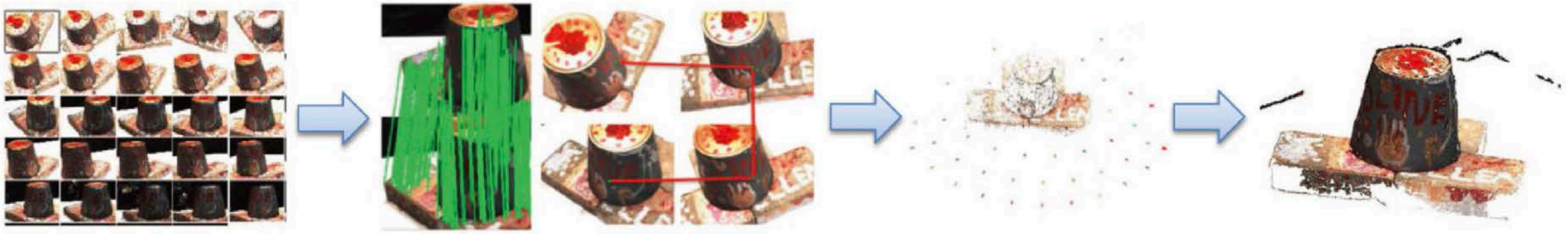
Li Yi

2025.03.27

Announcement

- Homework 2 released.
- Please use Wechat group to ask/answer questions and earn extra credits!

Recap



Input images

Find matched points across images

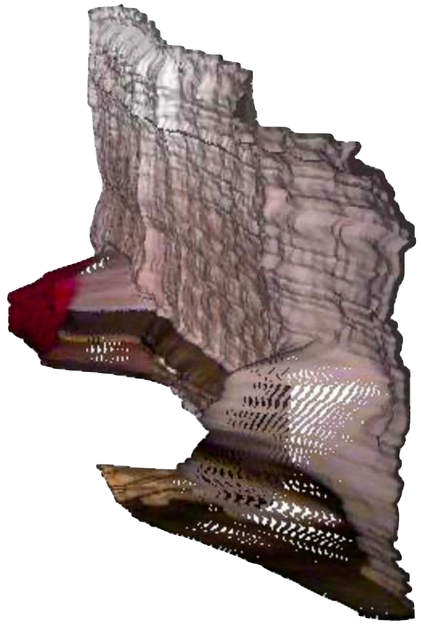
Structure from Motion (SFM)

dense reconstruction

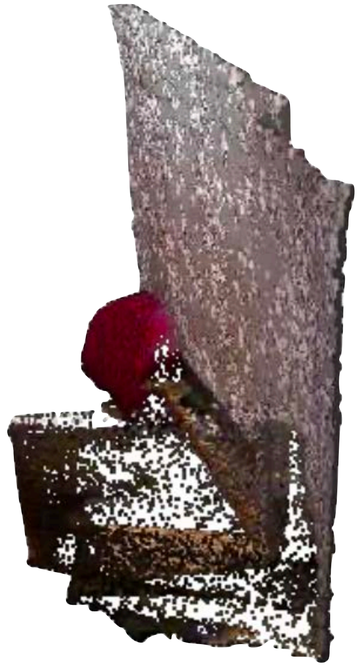
Recap

- Introduction to multi-view stereo (MVS)
- Classic MVS
- Learning-based MVS: a first pipeline
- Learning-based MVS: Improvements
  - Adaptive Space Sampling
  - ***Depth-Normal Consistency***

Depth Supervision Alone Does Not Give Smooth Surface



Prediction

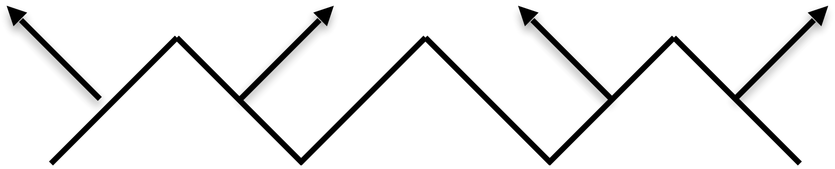


Ground truth

Slides credit: Hao Su

How to Improve Surface Smoothness?

- **Key observation**: Surface smoothness is reflected by surface normal.

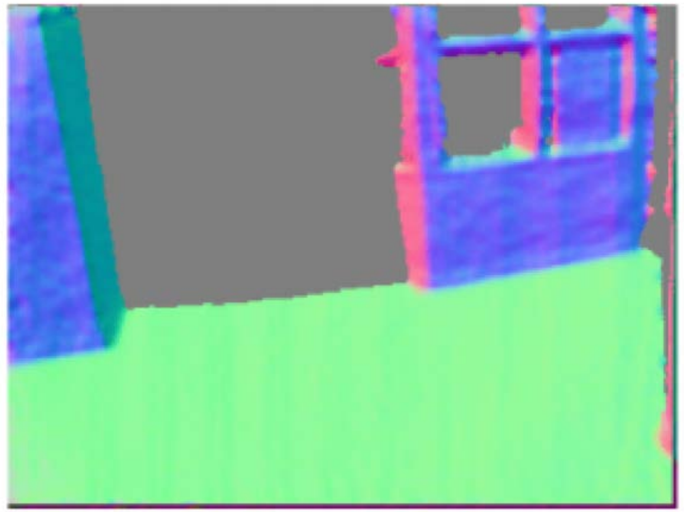


Rough surface

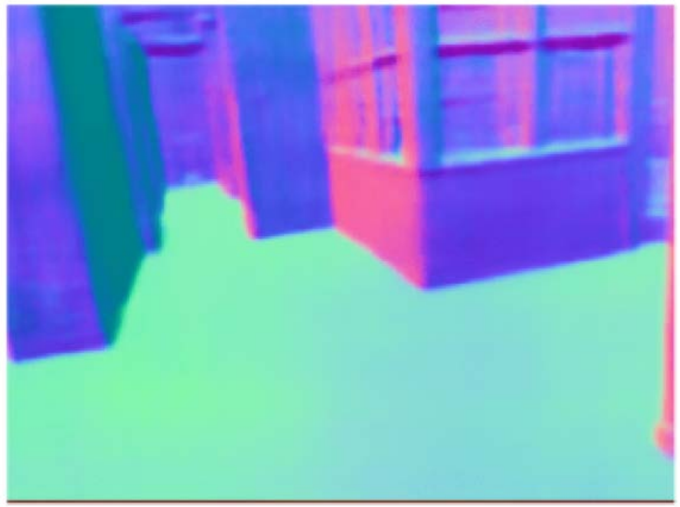
Plane surface

Slides credit: Hao Su

Observation: Normal Prediction is Easier



GT Normal

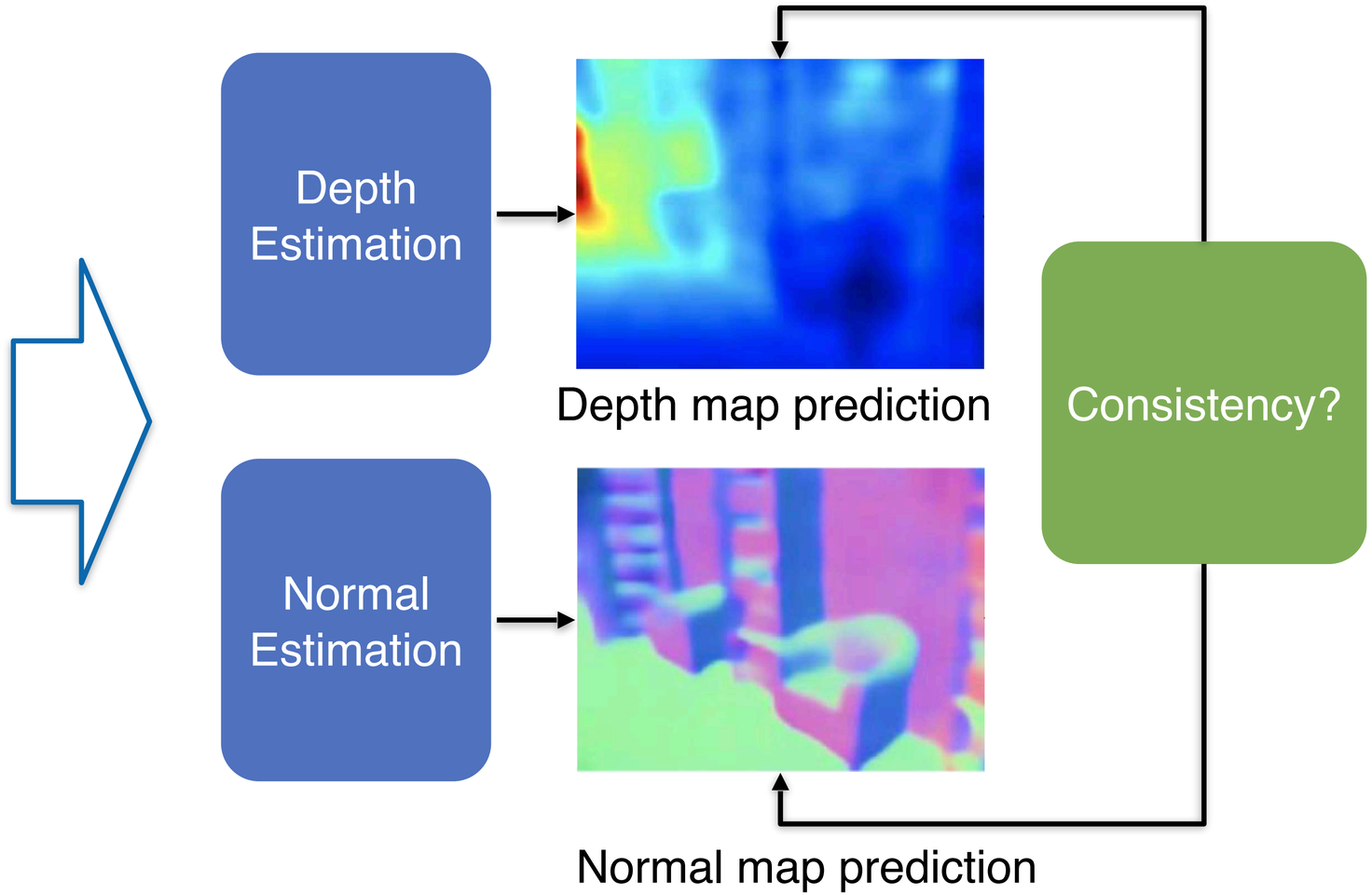


Predicted Normal

Slides credit: Hao Su

Depth Normal Consistency

- Estimate normal along with depth map.
- Regularize depth by normals.



Slides credit: Hao Su

Depth Normal Consistency

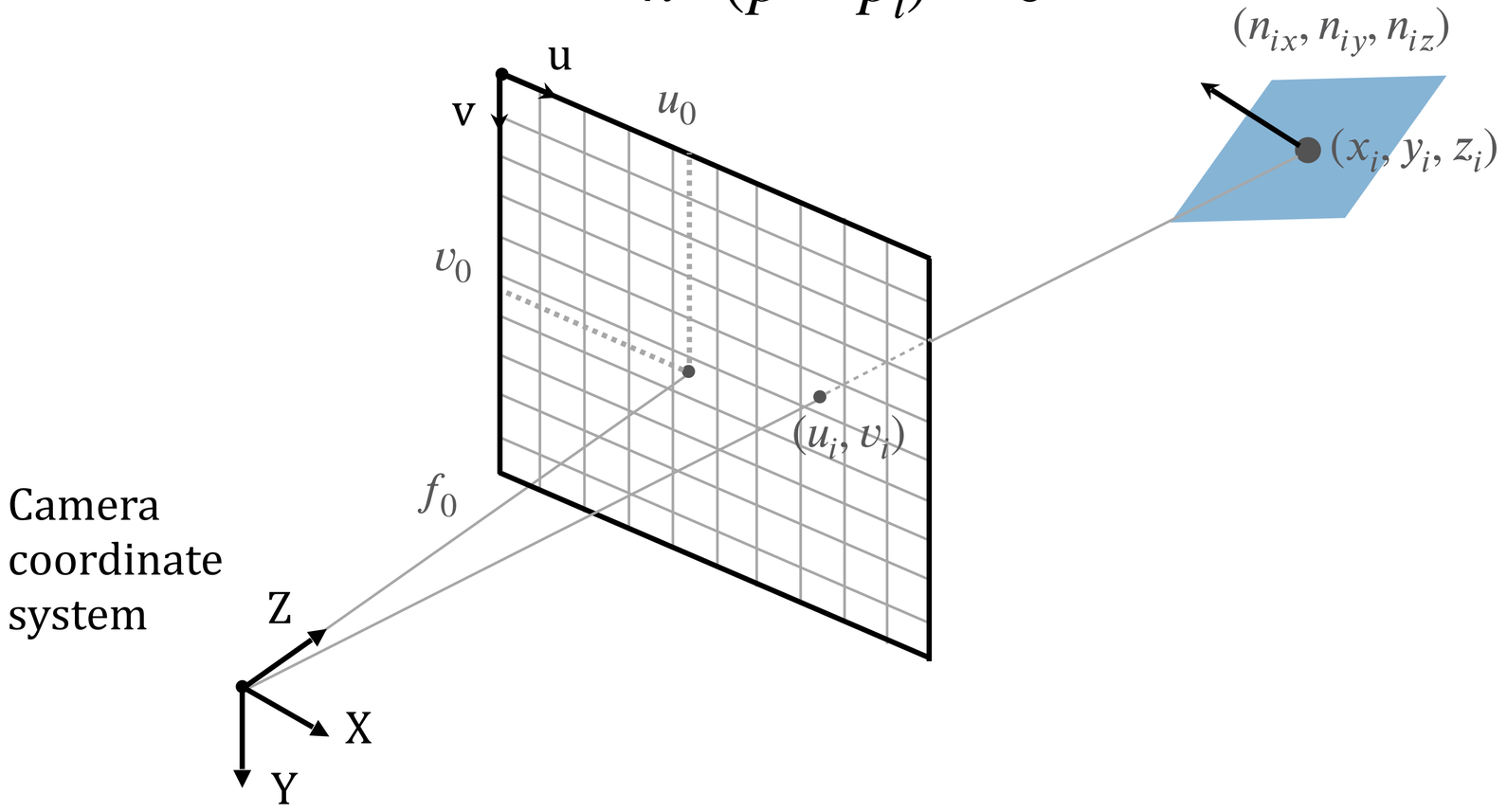
- Practice 1: Normal estimation as an auxiliary loss
  - Already quite effective
- Practice 2: Use normal estimation to correct depth

Slides credit: Hao Su

Refine Depth from Normal

- **Key assumption**: pixels within a local neighborhood lie on the same tangent plane.

$$\overrightarrow{n}^T(p - p_i) = 0$$

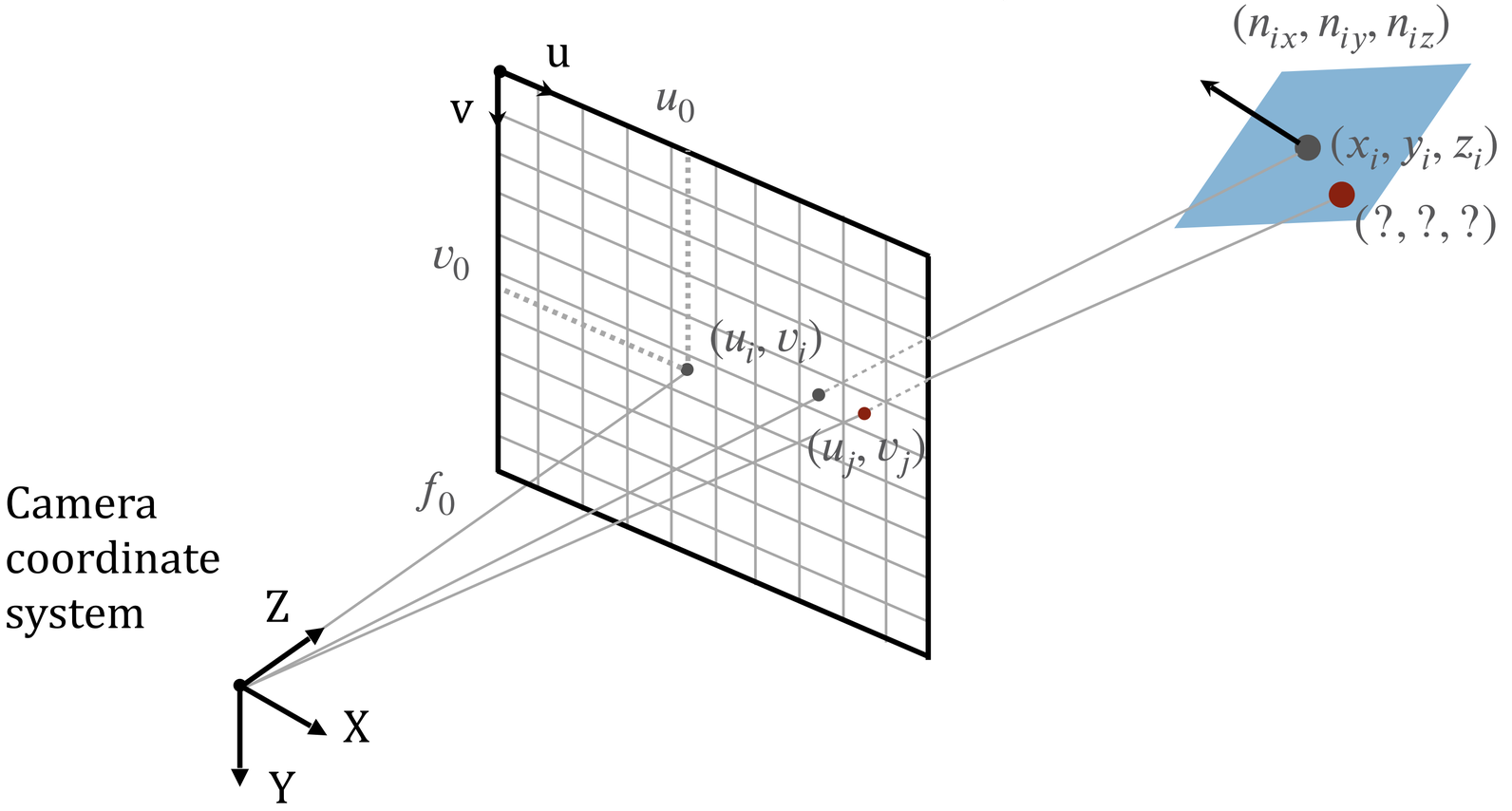


Slides credit: Hao Su

Refine Depth from Normal

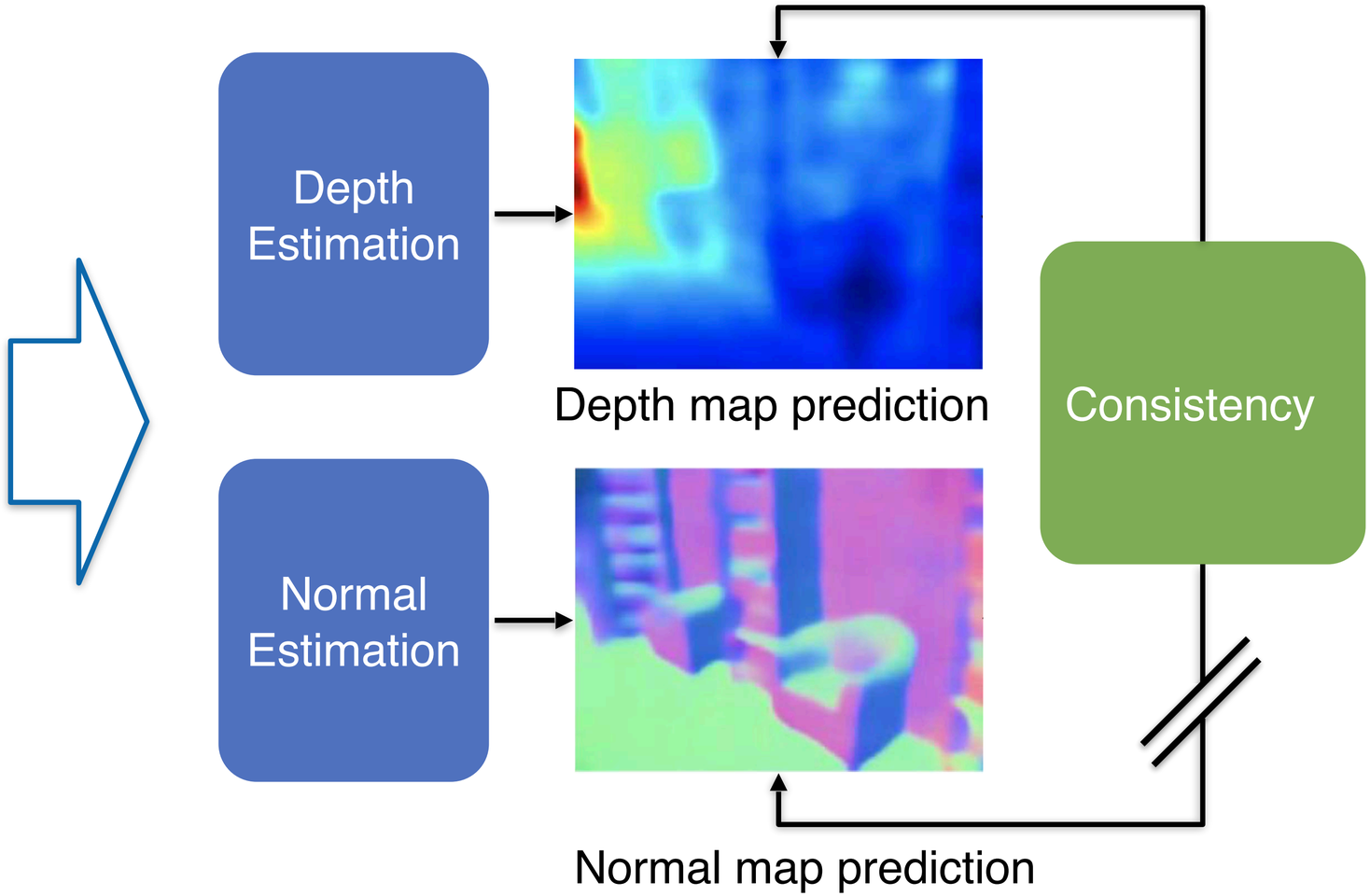
- Derive neighbor pixel depth from current pixel normal.

$$z'_{i\to j} = \frac{n_{ix}x_j + n_{iy}y_j + n_{iz}z_j}{(u_i - u_0)n_{ix}/f_0 + (v_i - v_0)n_{iy}/f_0 + n_{iz}}$$



Slides credit: Hao Su

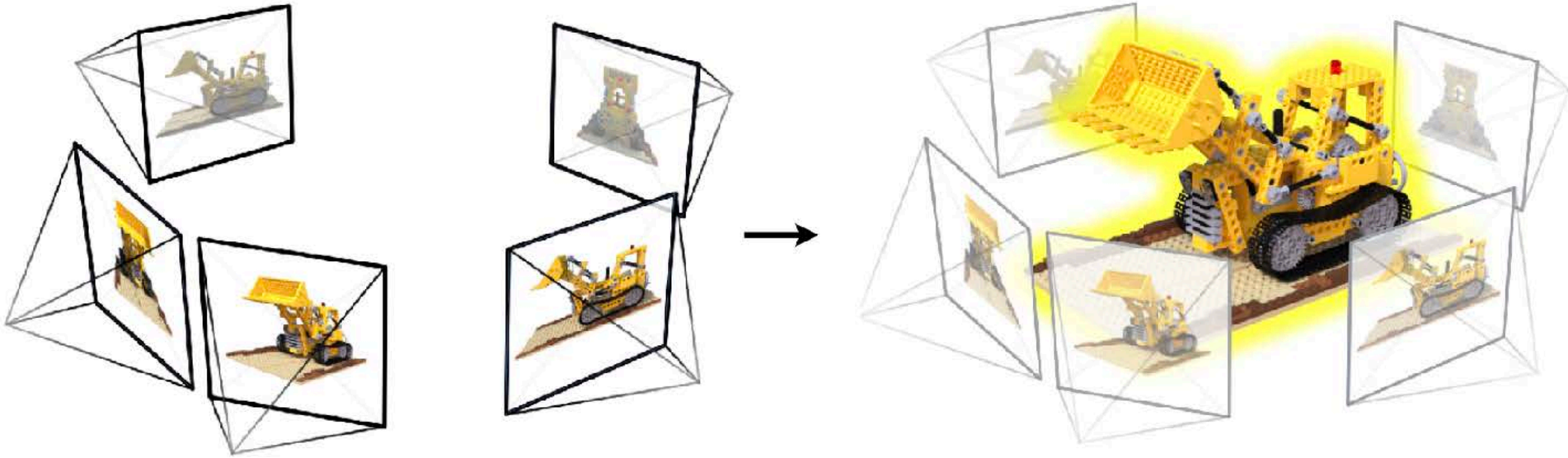
Depth Normal Consistency



Recap

- Deep volumetric stereo can lead to more robust matching and more complete reconstruction
- But volume-based methods are NOT computationally efficient, since the 3D target scene is sparse
- Adaptive sampling can improve computation efficiency and reconstruction quality
- Normal prediction is easier than depth, and can help improve depth accuracy and smoothness

Today's Focus



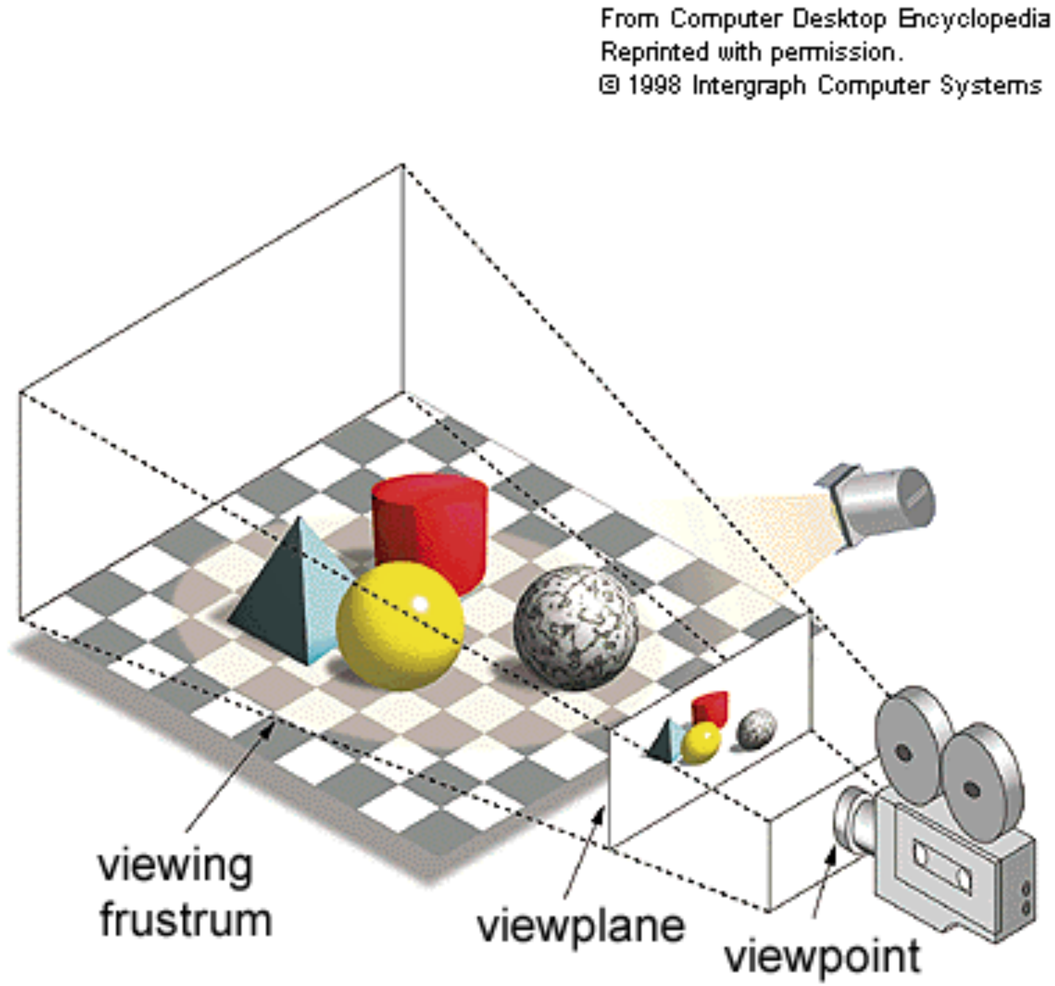
input images + camera poses

3D scene representation

NeRF

Rendering

3D Scene:
- Material
- Lighting
- Geometry
  (incl. animation)



Camera Def.
- Intrinsics
- Often:
  - focal length
  - principal point)

Camera View Point
- Extrinsics
- 6 DoF (3rot, 3trans)

Photo-realistic Image Synthesis

The Rendering Equation [Kajiya 86]

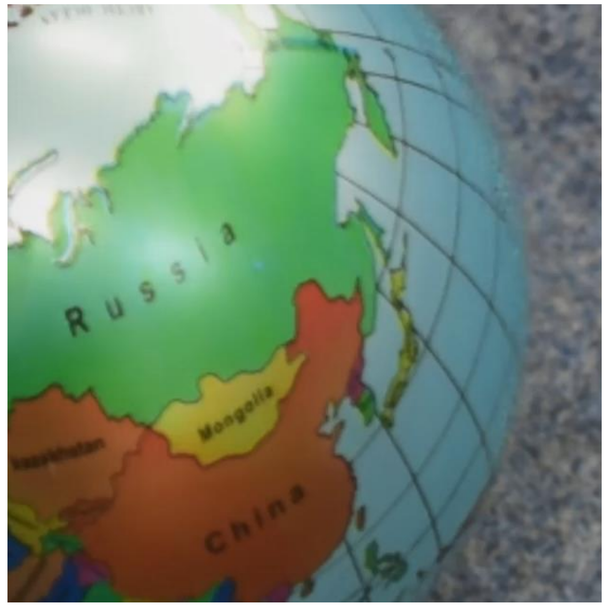
$$L_{\mathrm{o}}(\mathbf{x},\, \omega_{\mathrm{o}},\, \lambda,\, t) \;=\; L_{e}(\mathbf{x},\, \omega_{\mathrm{o}},\, \lambda,\, t) \;+\; \int_{\Omega} f_{r}(\mathbf{x},\, \omega_{\mathrm{i}},\, \omega_{\mathrm{o}},\, \lambda,\, t)\, L_{\mathrm{i}}(\mathbf{x},\, \omega_{\mathrm{i}},\, \lambda,\, t)\, (\omega_{\mathrm{i}} \cdot \mathbf{n})\, \mathrm{d}\,\omega_{\mathrm{i}}$$

Idea of Neural Rendering

Novel View point synthesis:

6 DoF Camera
Pose / View Point

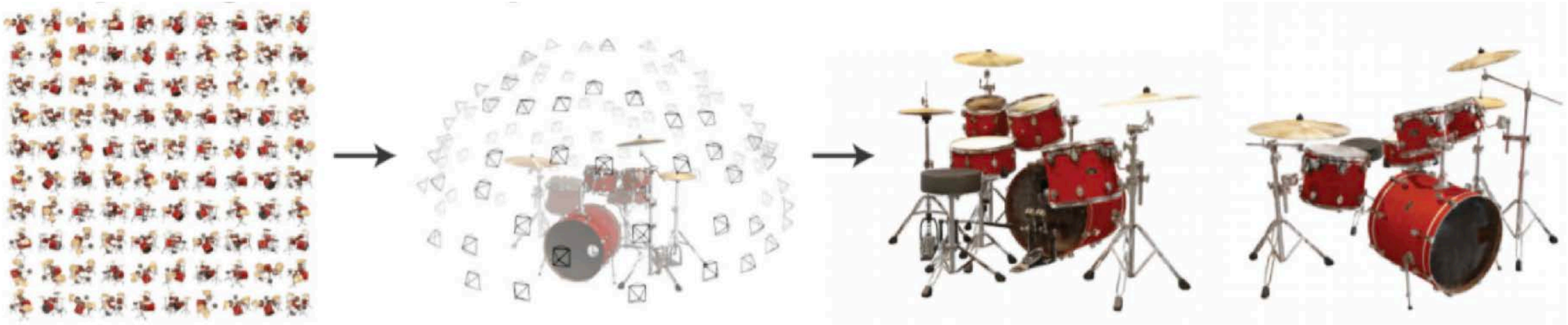
Neural Network
-> Encodes entire scene description, lighting, materials, etc.



Outline

- ***Neural Radiance Field (NeRF)***
  - Implicit Functions: an Illustration with 3D Surface Representation
  - Volume Rendering with Ray Marching
  - Learning NeRF
- NeRF Extensions

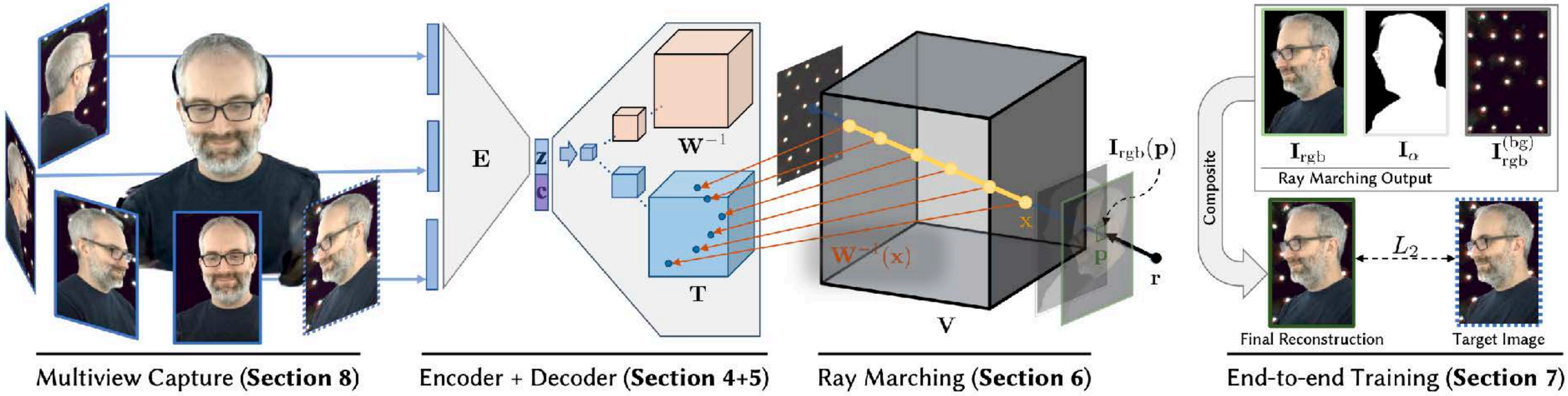
Problem Setting



NeRF was Originally Designed for View Synthesis

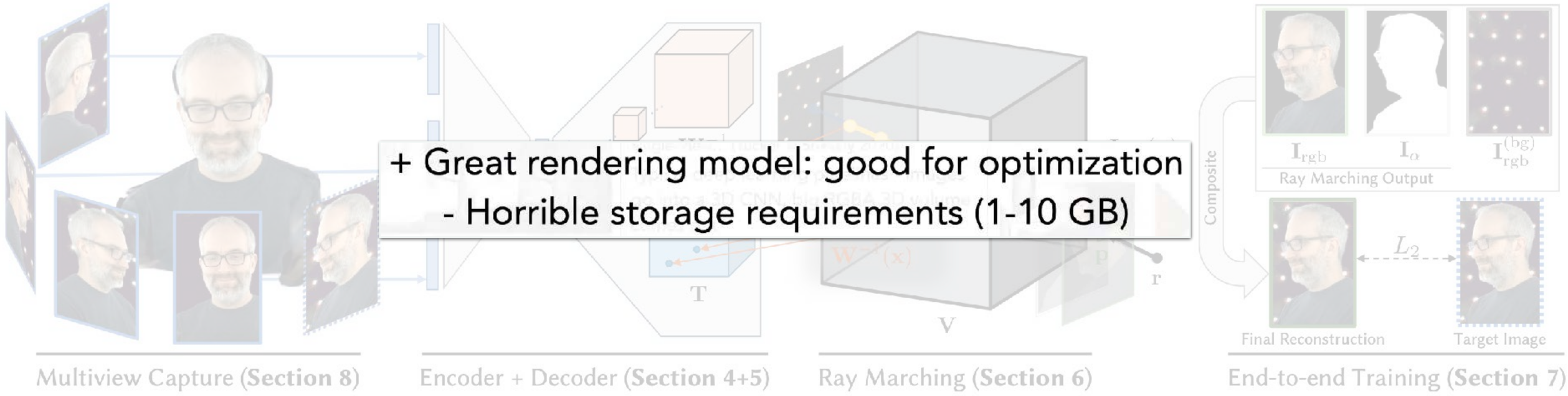
NeRF was Originally Designed for View Synthesis

RGB-Alpha Volume Rendering for View Synthesis



Neural Volumes: Learning Dynamic Renderable Volumes from Images, Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh, Siggraph 2019.

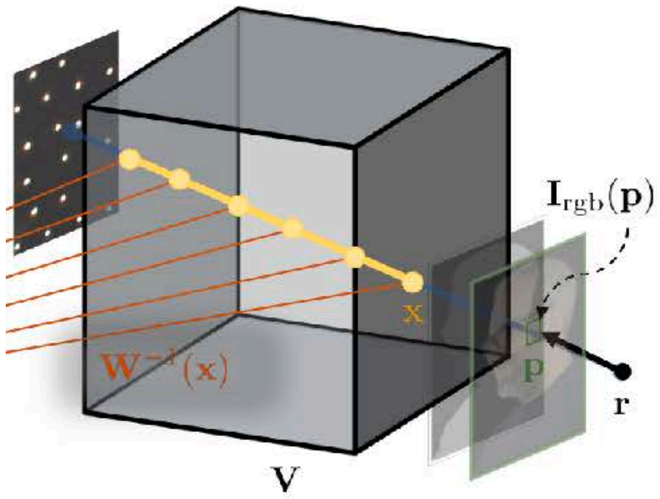
RGB-Alpha Volume Rendering for View Synthesis

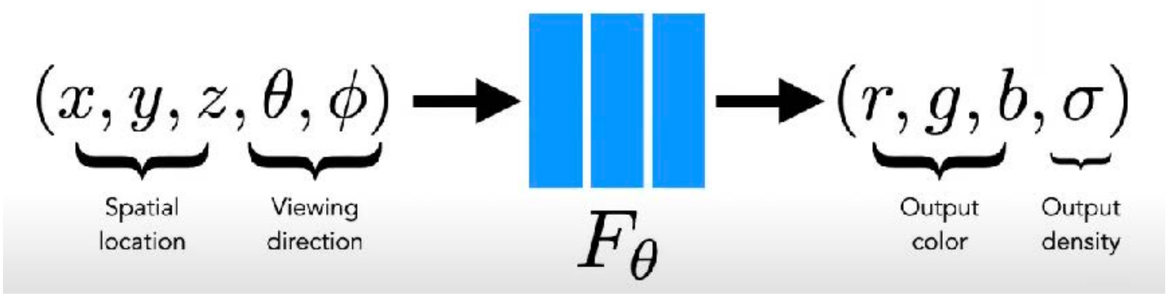


Neural Volumes: Learning Dynamic Renderable Volumes from Images, Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh, Siggraph 2019.

Key Idea of NeRF

- Use an implicit function to replace the volume representation
- Track light emission along different directions





Outline

- Neural Radiance Field (NeRF)
  - ***Implicit Functions: an Illustration with 3D Surface Representation***
  - Volume Rendering with Ray Marching
  - Learning NeRF
- NeRF Extensions

Explicit vs Implicit Representation (2D)

Explicit:

$$\mathbf{f}(\alpha) = (r\cos(\alpha), r\sin(\alpha))^T$$

Domain: [0,2π]

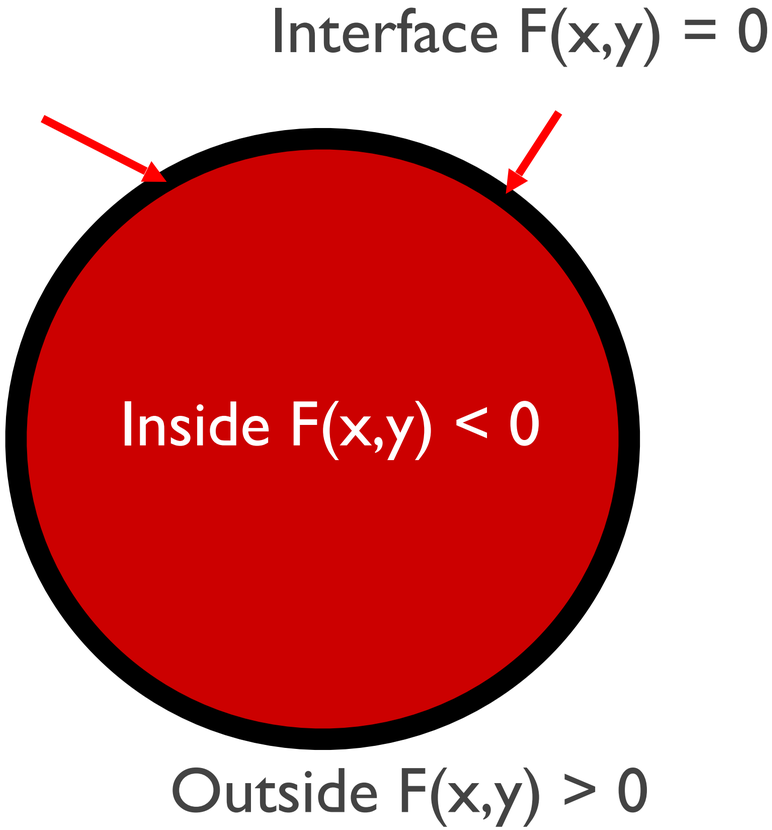
$\mathbf{f}([0, 2\pi])$

Implicit:

$$F(x, y) = \sqrt{x^2 + y^2} - r$$

Domain: (x,y) $\in \mathbb{R}^2$

$\Longrightarrow$ Circle is implicitly defined by $\{(x, y)|F(x, y) = 0\}$



$\mathbf{f}(\alpha)$ defines the interface

$F(x, y)$ defines the **Signed Distance Function** of the circle

Explicit vs Implicit Representation (3D)

Explicit:

$$\mathbf{f}(\alpha, \beta) = (r \sin(\alpha) \cos(\beta), -r \cos(\beta), r \sin(\alpha) \sin(\beta))$$

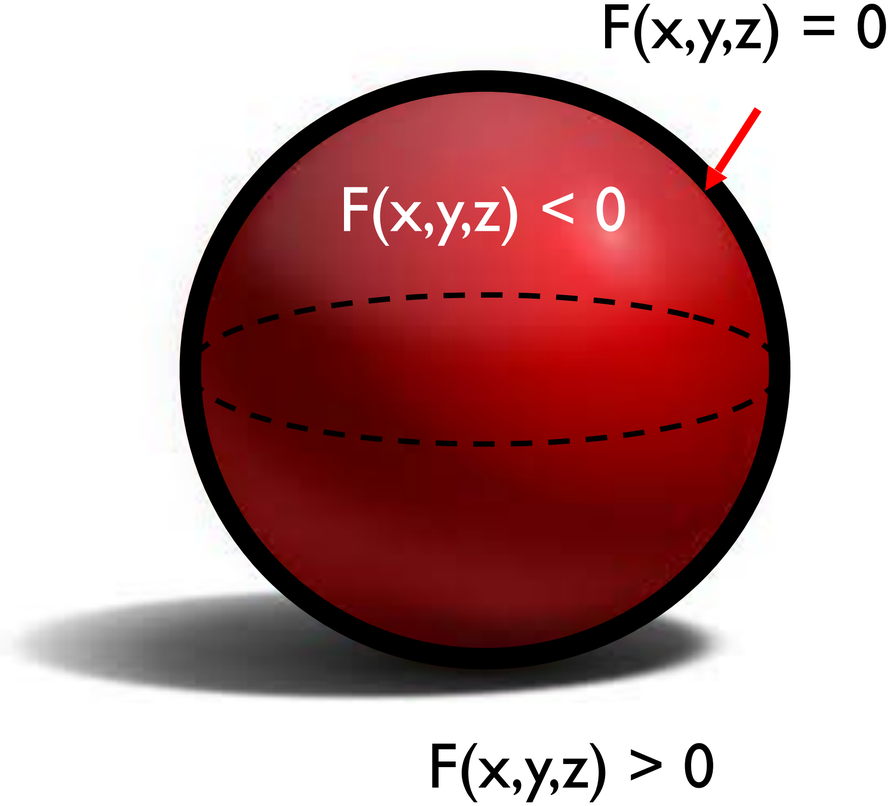
Domain: $\alpha \in [0; 2\pi], \beta \in [0; \pi]$

Implicit:

$$F(x, y, z) = \sqrt{x^2 + y^2 + z^2} - r$$

Domain: (x,y,z) $\in \mathbb{R}^3$

$\Longrightarrow$ Sphere is implicitly defined by $\{(x, y, z) | F(x, y, z) = 0\}$

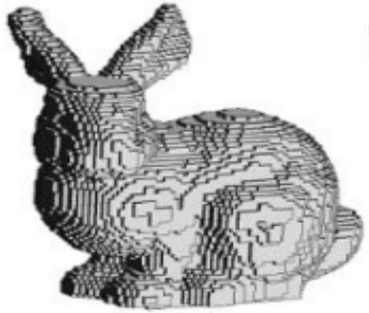


$\mathbf{f}(\alpha, \beta)$ defines the 3D surface

$F(x, y, z)$ defines the **Signed Distance Function** of the sphere

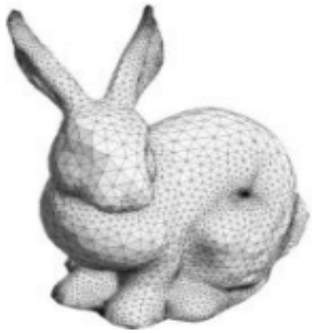
Representing 3D surfaces

Explicit:

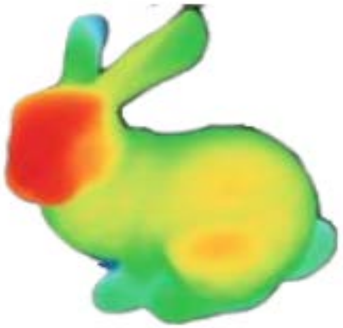


Voxels

Point clouds



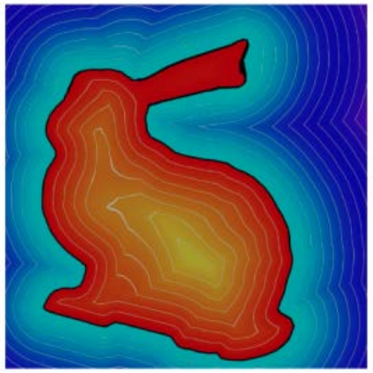
Mesh



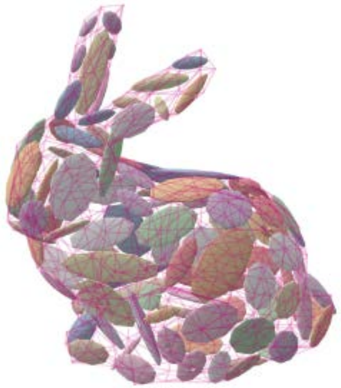
Depth

Surface Normals

Implicit:



Signed distance field



Mixture of primitives
(e.g gaussian mixtures)

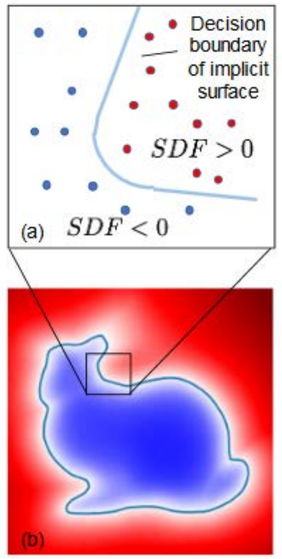
Thomas Funkhouser's talk at 3DGV seminar

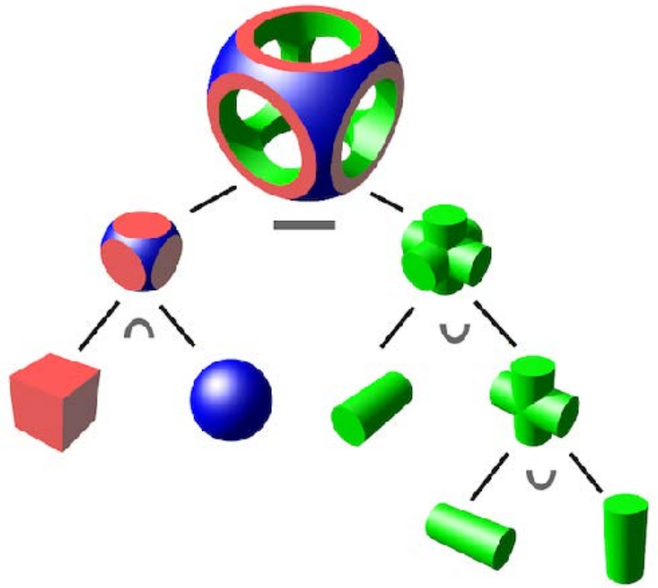
Signed Distance Field (SDF)

- Maps each 3D points ***p*** to it's signed distance to the object surface *S*. The sign is positive if the ***p*** is inside the object, and negative otherwise.

$$SDF(p) = \text{sign}(p) \cdot \min_{q \in \mathcal{S}} \|p - q\|$$

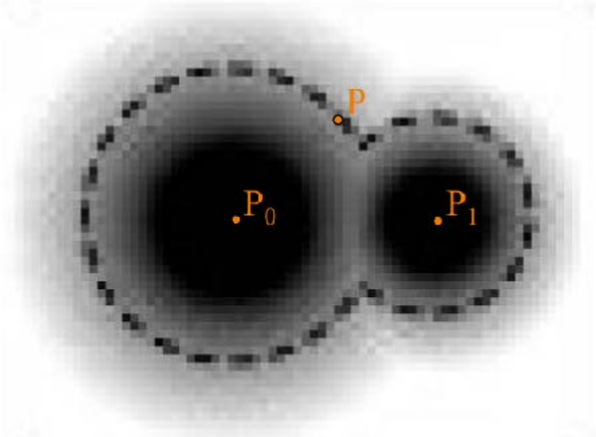
- Sign indicates whether the point ***p*** is inside (-) or outside (+) of the shape
- Shape's boundary as the zero-level-set of SDF
- Allows for Constructive Solid Geometry (CSG) through boolean operations





Mixture of Gaussians

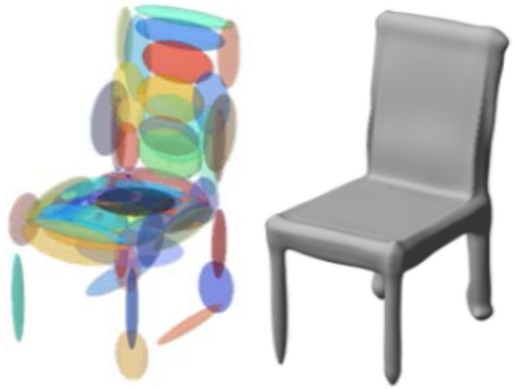
- Represents a shape as a mixture of local implicit functions (3D gaussians)



$$F(\mathbf{x}, \mathbf{\Theta}) = \sum_{i \in [N]} f_i(\mathbf{x}, \theta_i)$$

$$f_i(\mathbf{x}, \theta_i) = c_i \exp\left(\sum_{d \in \{x,y,z\}} \frac{-(\mathbf{p}_{i,d} - \mathbf{x}_d)^2}{2\mathbf{r}_{i,d}^2} \right)$$

- Shape's boundary is defined as an iso-level of the **global** implicit function



[1] Genova19
[2] Genova20

Representing 3D surfaces with Implicit Functions

Pros:

- Compared to **point clouds**: **clearly defines the (iso-)surface**
- Compared to **meshes**: can continuously **adapt to arbitrary topology**
- Compared to **voxels**: can be represented with **few parameters** (e.g. mixture of simple implicit functions)
- They are **continuous** in 3D
- Can give analytic normals, can be applied with boolean operations, etc

Representing 3D surfaces with Implicit Functions

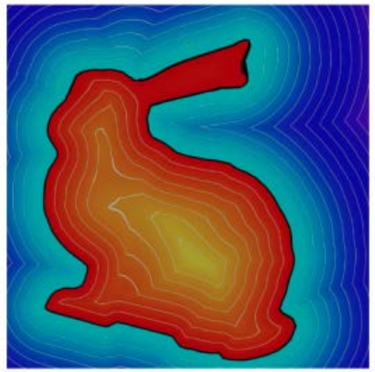
Pros:

- Compared to **point clouds**: **clearly defines the (iso-)surface**
- Compared to **meshes**: can continuously **adapt to arbitrary topology**
- Compared to **voxels**: can be represented with **few parameters** (e.g. mixture of simple implicit functions)
- They are **continuous** in 3D
- Can give analytic normals, can be applied with boolean operations, etc

Cons:

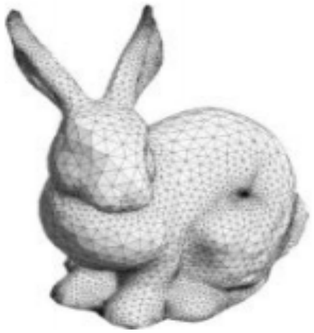
- SDF is well-defined for only watertight meshes (there is an interior and an exterior)
- Need extra steps to visualize

Converting Implicit Surfaces to meshes



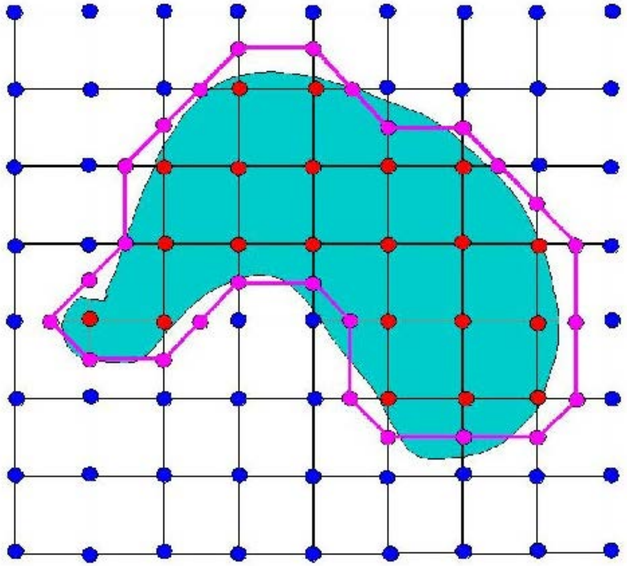
Implicit function

Extract (zero-level) iso-surface

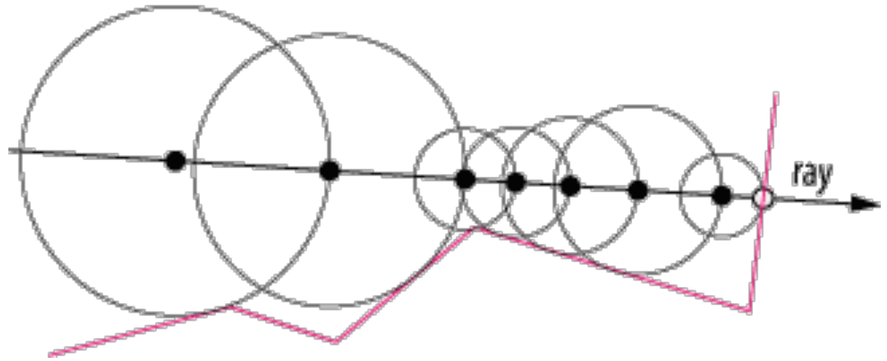


Mesh

Marching Cubes



Ray marching



Representing 3D surfaces with Implicit Functions

Pros:

- Compared to **point clouds**: **clearly defines the (iso-)surface**
- Compared to **meshes**: can continuously **adapt to arbitrary topology**
- Compared to **voxels**: can be represented with **few parameters** (e.g. mixture of simple implicit functions)
- They are **continuous** in 3D
- Can give analytic normals, can be applied with boolean operations, etc

Cons:

- SDF is well-defined for only watertight meshes (there is an interior and an exterior)
- Need extra steps to visualize
- Not all complex shapes can be efficiently / accurately represented with simple primitives

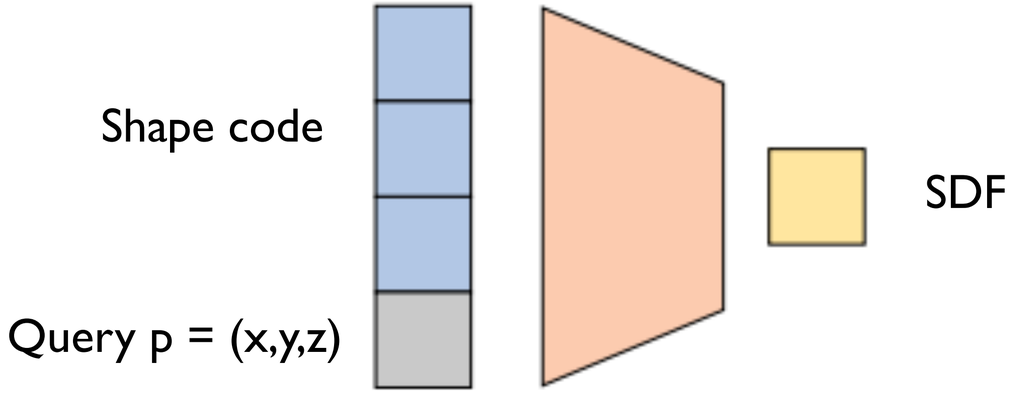
Representing 3D surfaces

DeepSDF: **Efficiently** representing complex shapes by learning their SDF

Idea: Learn a **continuous** representation of 3D implicit surfaces

Query p = (x,y,z), Shape latent code **z**

$$F(p; \mathbf{Z}) = SDF(p, \mathcal{M})$$

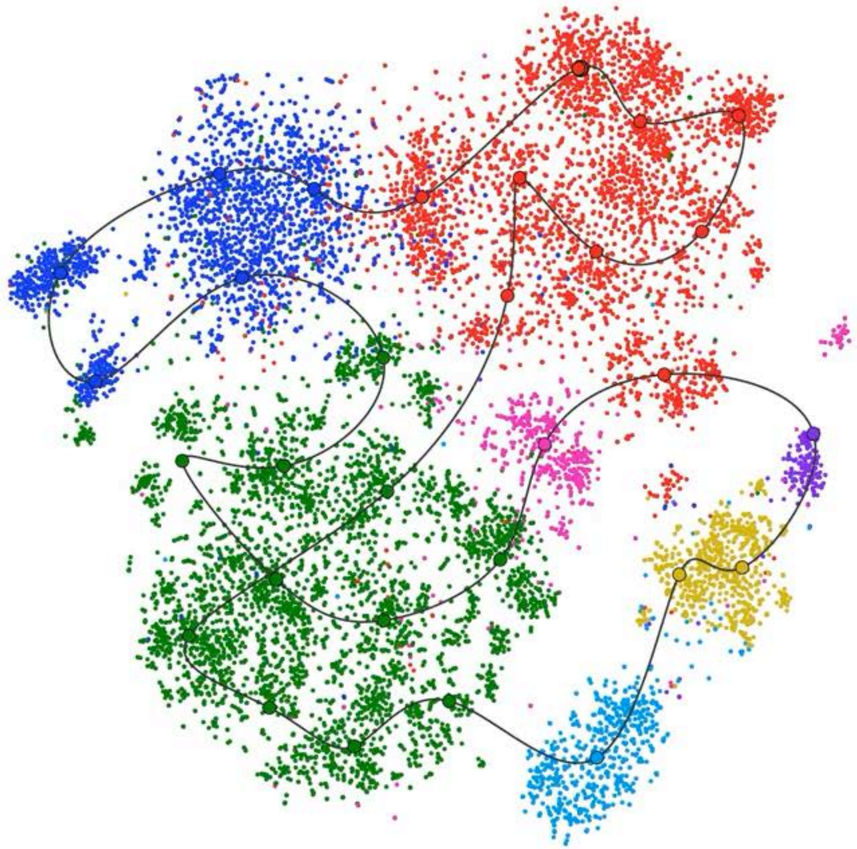


=> **Continuity** in 3D space AND shapes space

[3] Park19

Representing 3D surfaces

DeepSDF: Representing complex shapes by learning their SDF



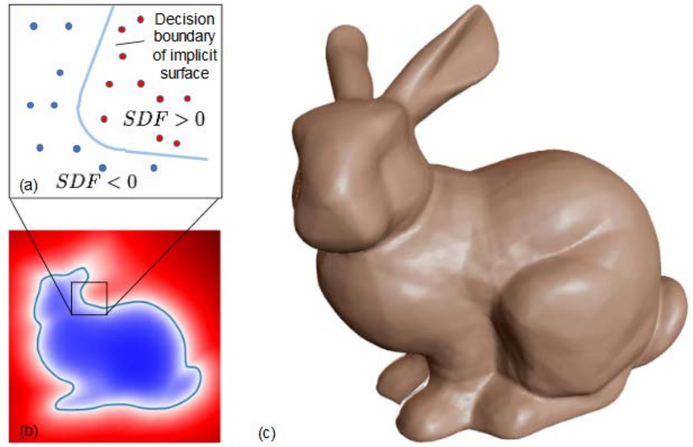
[3] Park19

Take home message on Implicit Functions

Representation of a continuous field

Learned implicit functions:

- Can represent complex shapes
- Are **continuous mappings** because they use **MLPs**
- Are applicable to N-D data: 2D images, 3D shapes, radiance fields

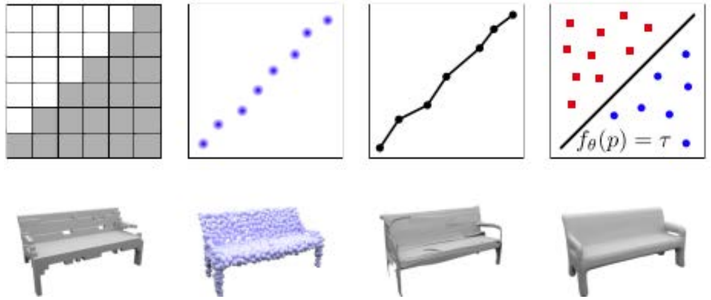


Visualization of implicit functions is done by extracting iso-surfaces:

1. Running inference for multiple queries in input space
2. Rendering the result by combining the queries

More works on Implicit Functions for 3D shape

- Occupancy Networks



[4] Mescheder19

- PiFu and PiFuHD

[5] Saito19
[6] Saito20

Outline

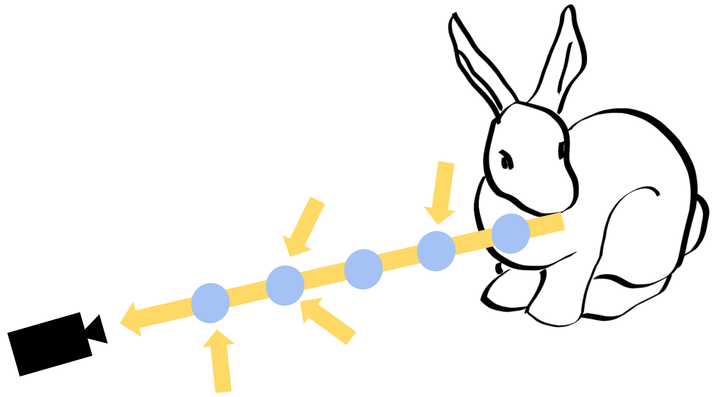
- Neural Radiance Field (NeRF)
  - Implicit Functions: an Illustration with 3D Surface Representation
  - ***Volume Rendering with Ray Marching***
  - Learning NeRF
- NeRF Extensions

General Idea

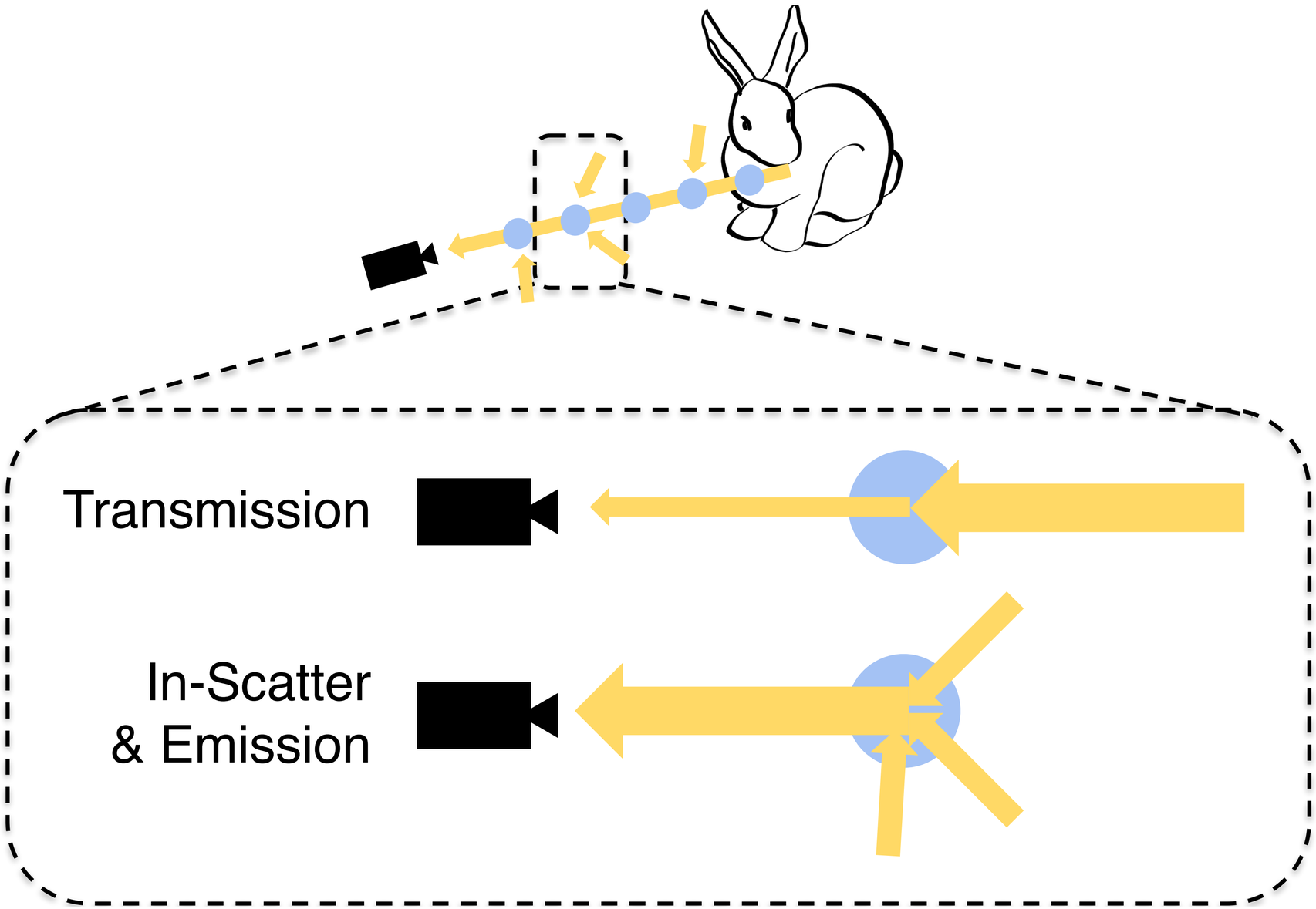
- The appearance of the surface will be observed at views along the camera ray

- If we have a **light transport model** from the surface along the ray to the pixels, we will know the pixel color

Volumetric Light Transport Model

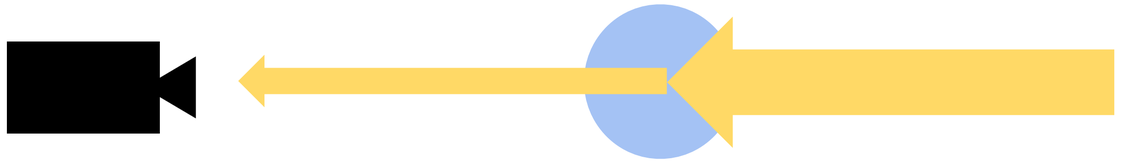


Volumetric Light Transport Model



Volumetric Light Transport Model

Transmission

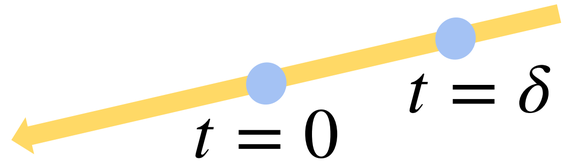


Attenuation coefficient σ (Transparency)

In-Scatter
& Emission

Emission Radiance c

Emission Radiance Passing a Ray Segment

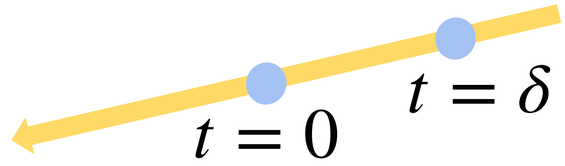


Beer-Lambert's Law: $\alpha(t) = 1 - \exp(-\sigma t)$

opacity

attenuation coefficient

Emission Radiance Passing a Ray Segment



Beer-Lambert's Law: $\alpha(t) = 1 - \exp(-\sigma t)$

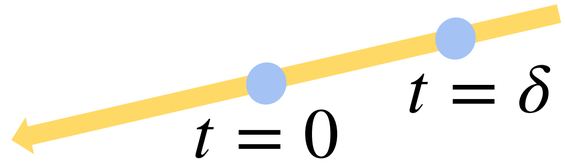
opacity

attenuation coefficient

emission radiance

Light emitted along a segment $= \int_0^{\delta} (1 - \alpha(t))c(t)\mathrm{d}t$

Emission Radiance Passing a Ray Segment



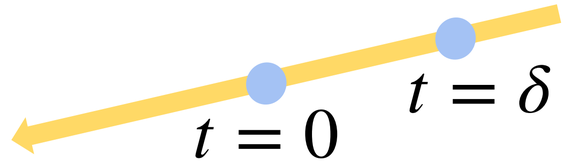
Beer-Lambert's Law: $\alpha(t) = 1 - \exp(-\sigma t)$

Light emitted along a segment

$$= \int_0^\delta (1 - \alpha(t))c(t)\mathrm{d}t \overset{c(t)=c}{\approx} c\int_0^\delta \exp(-\sigma t)\mathrm{d}t$$

$$= \frac{c}{\sigma}(1 - \exp(-\delta\sigma))$$

Emission Radiance Passing a Ray Segment



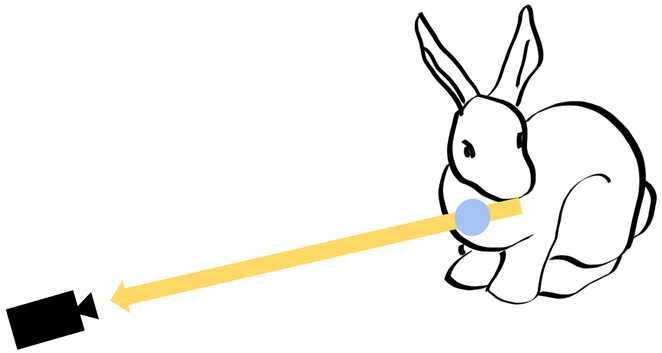
Beer-Lambert's Law: $\alpha(t) = 1 - \exp(-\sigma t)$

Light emitted along a segment

$$= \int_0^{\delta} (1 - \alpha(t)) c(t) \mathrm{d}t \overset{c(t)=c}{\approx} c \int_0^{\delta} \exp(-\sigma t) \mathrm{d}t$$

$$= \frac{c}{\sigma}(1 - \exp(-\delta\sigma)) = \alpha(\delta)\left(\frac{c}{\sigma}\right)$$

Discretized Radiance Integration (Ray Marching)

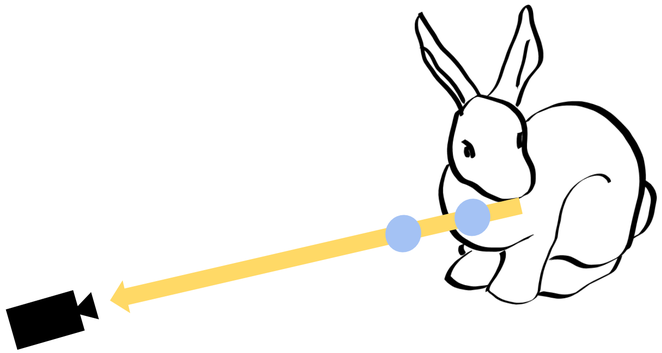


A single point

$$I_1 = \alpha_1 \left(\frac{c_1}{\sigma_1} \right)$$

I_1: light intensity after point 1
c_1: predicted emission radiance at point 1
α_1: opacity of point 1

Discretized Radiance Integration (Ray Marching)



2 points

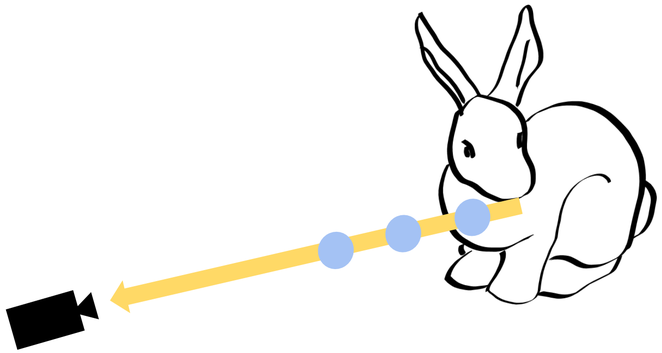
Point 1 acts like the previous case

$$I_1 = \alpha_1 \left(\frac{c_1}{\sigma_1} \right)$$

Point 2 additionally transmits I_2

$$I_2 = \alpha_2 \left(\frac{c_2}{\sigma_2} \right) + (1 - \alpha_2) I_1$$

Discretized Radiance Integration (Ray Marching)



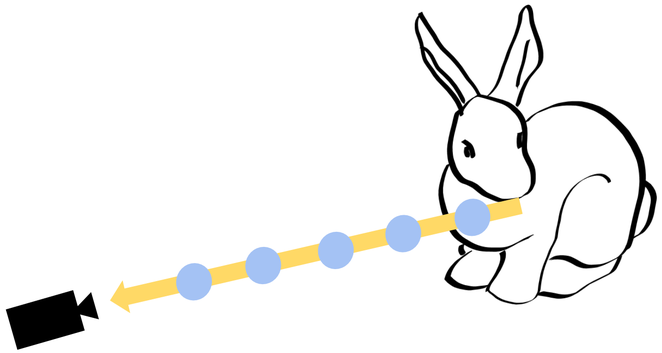
3 points

$$I_1 = \alpha_1 \left(\frac{c_1}{\sigma_1} \right)$$

$$I_2 = \alpha_2 \left(\frac{c_2}{\sigma_2} \right) + (1 - \alpha_2) I_1$$

$$I_3 = \alpha_3 \left(\frac{c_3}{\sigma_3} \right) + (1 - \alpha_3) I_2$$

Discretized Radiance Integration (Ray Marching)



In general n: the number of points

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

$$I = \sum_i T_i \alpha_i \left(\frac{c_i}{\sigma_i} \right)$$ = final radiance of the ray

Outline

- Neural Radiance Field (NeRF)
  - Implicit Functions: an Illustration with 3D Surface Representation
  - Volume Rendering with Ray Marching
  - ***Learning NeRF***
- NeRF Extensions

What NeRF learns

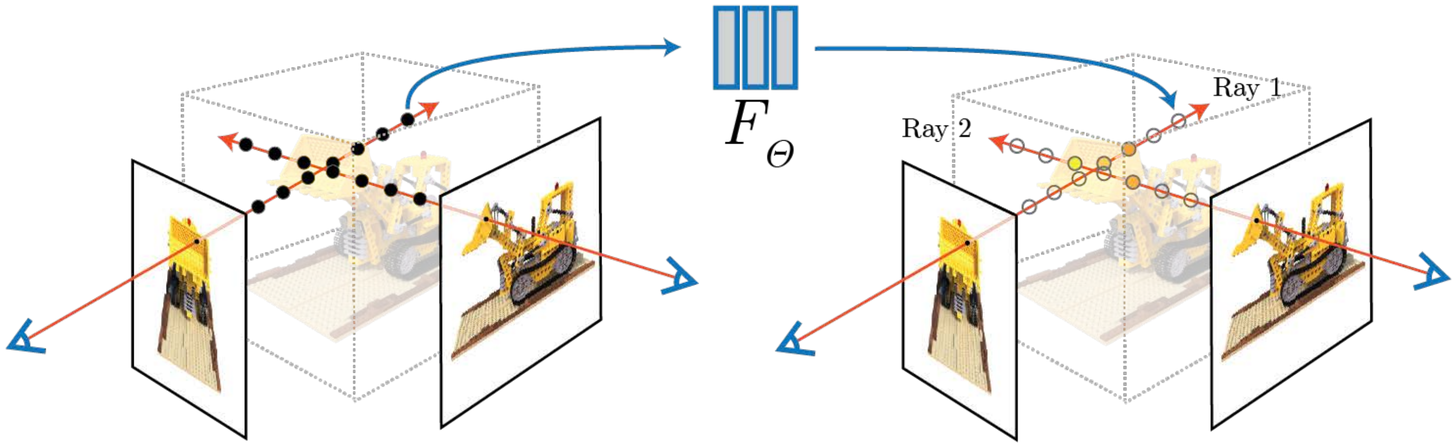
$$(\alpha_i, \frac{c_i}{\sigma_i}) = F_\Theta(x, y, z, \theta, \phi)$$

Note: It is quite common that σ_i and c_i are both close to zero, so we predict c_i/σ_i directly.

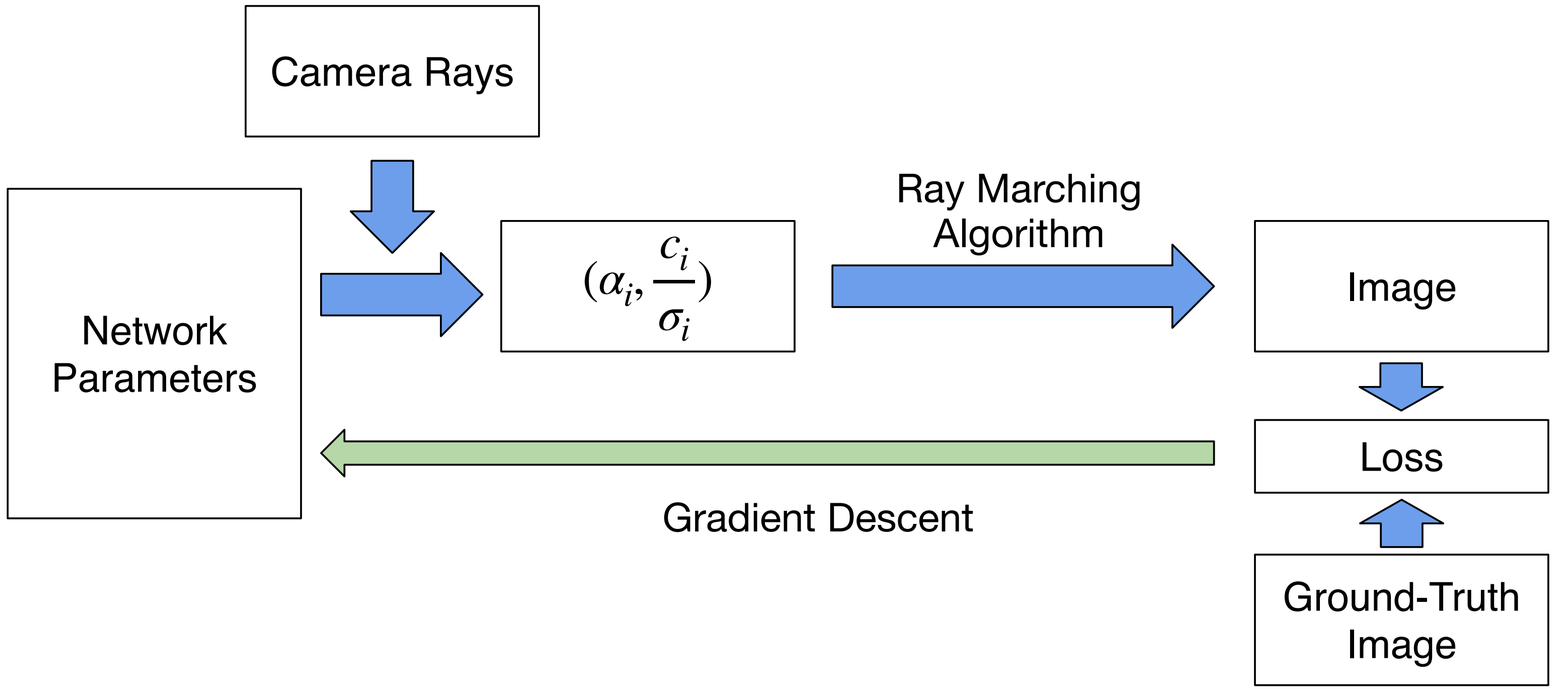
Pixel Loss

$$I = \sum_i T_i \alpha_i \left(\frac{c_i}{\sigma_i} \right)$$

Comparing I with ground-truth pixel value, we get a loss (e.g., L1, L2)

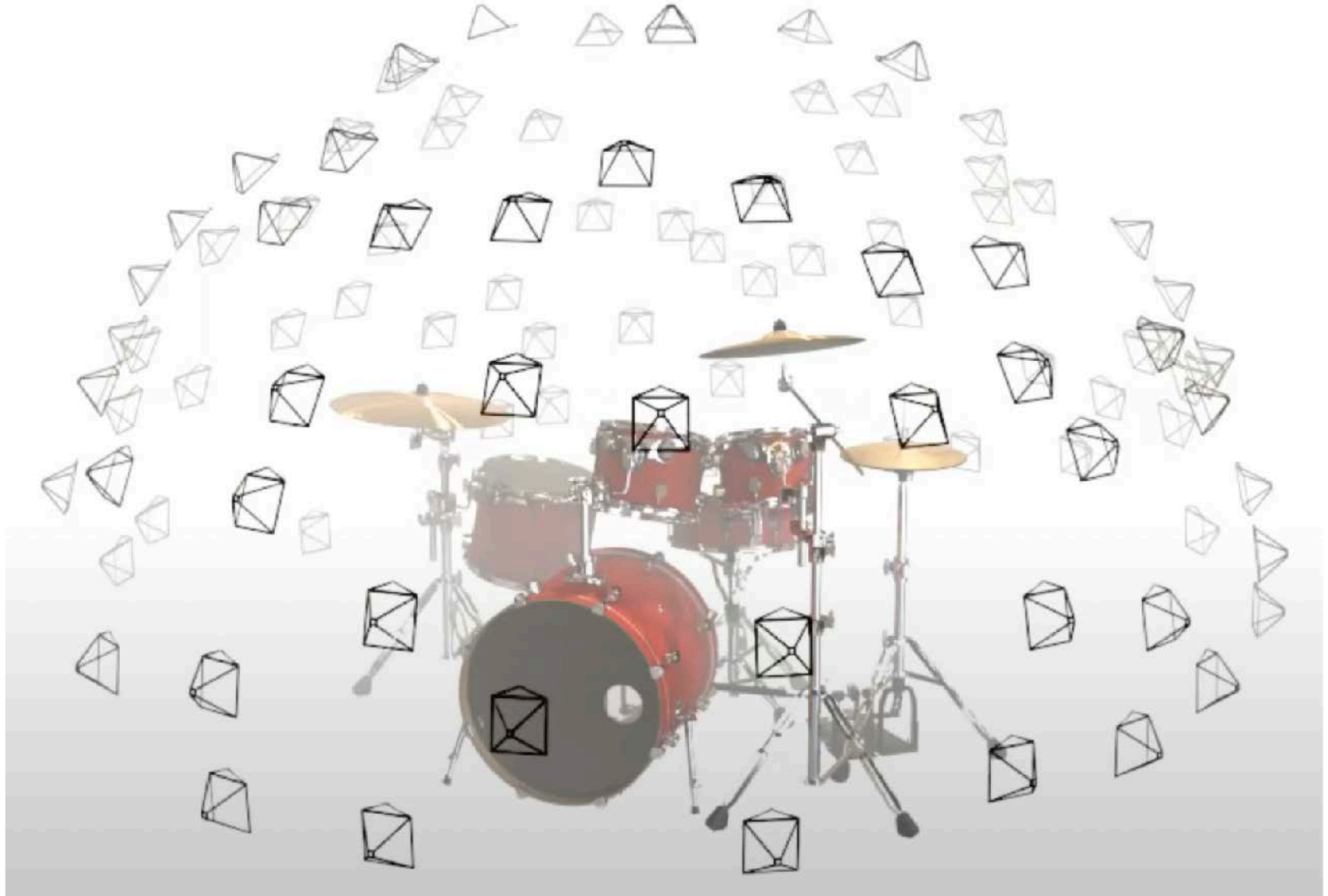


Train Pipeline in NeRF

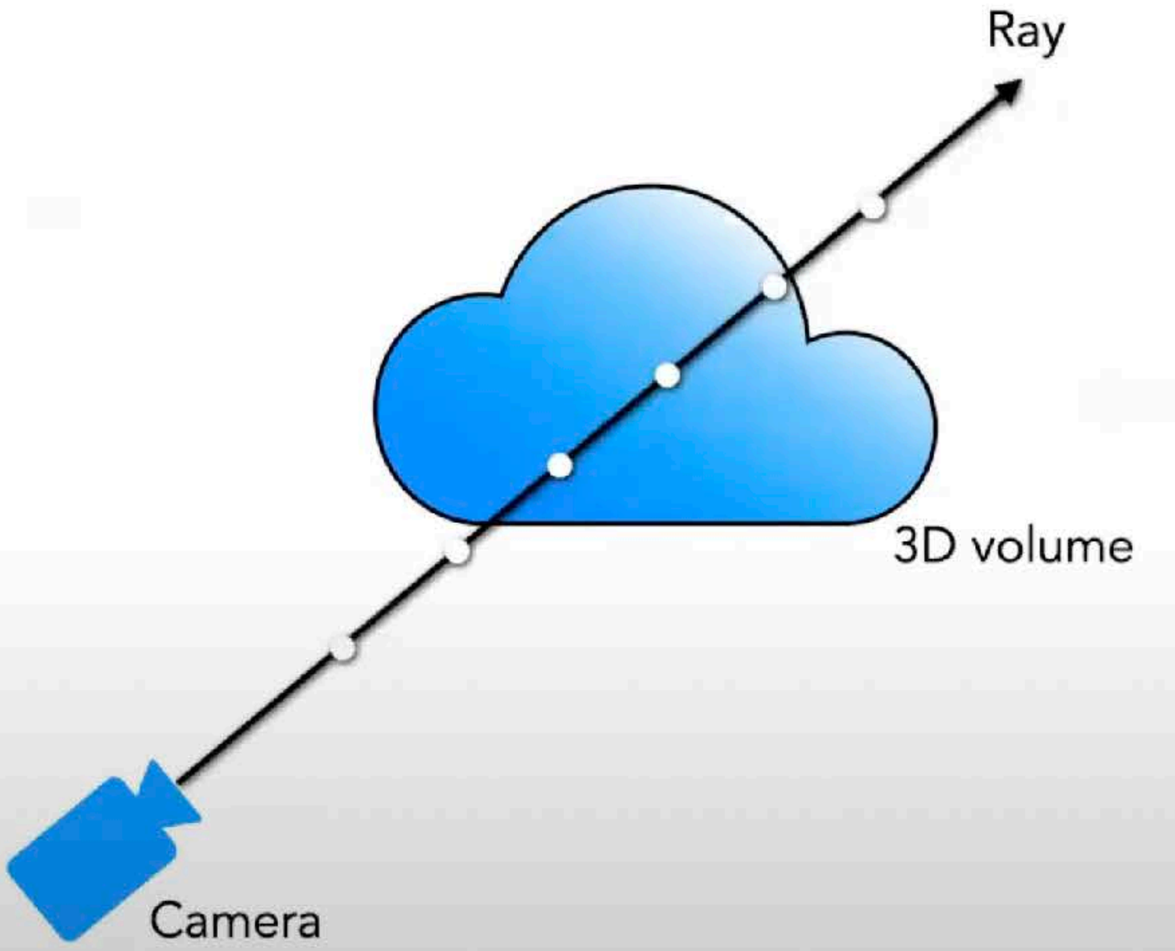


- Optimize on a single scene
  - store the scene in weights of the network
- Require ground-truth camera parameters

Train Networks to Reproduce All Input Views



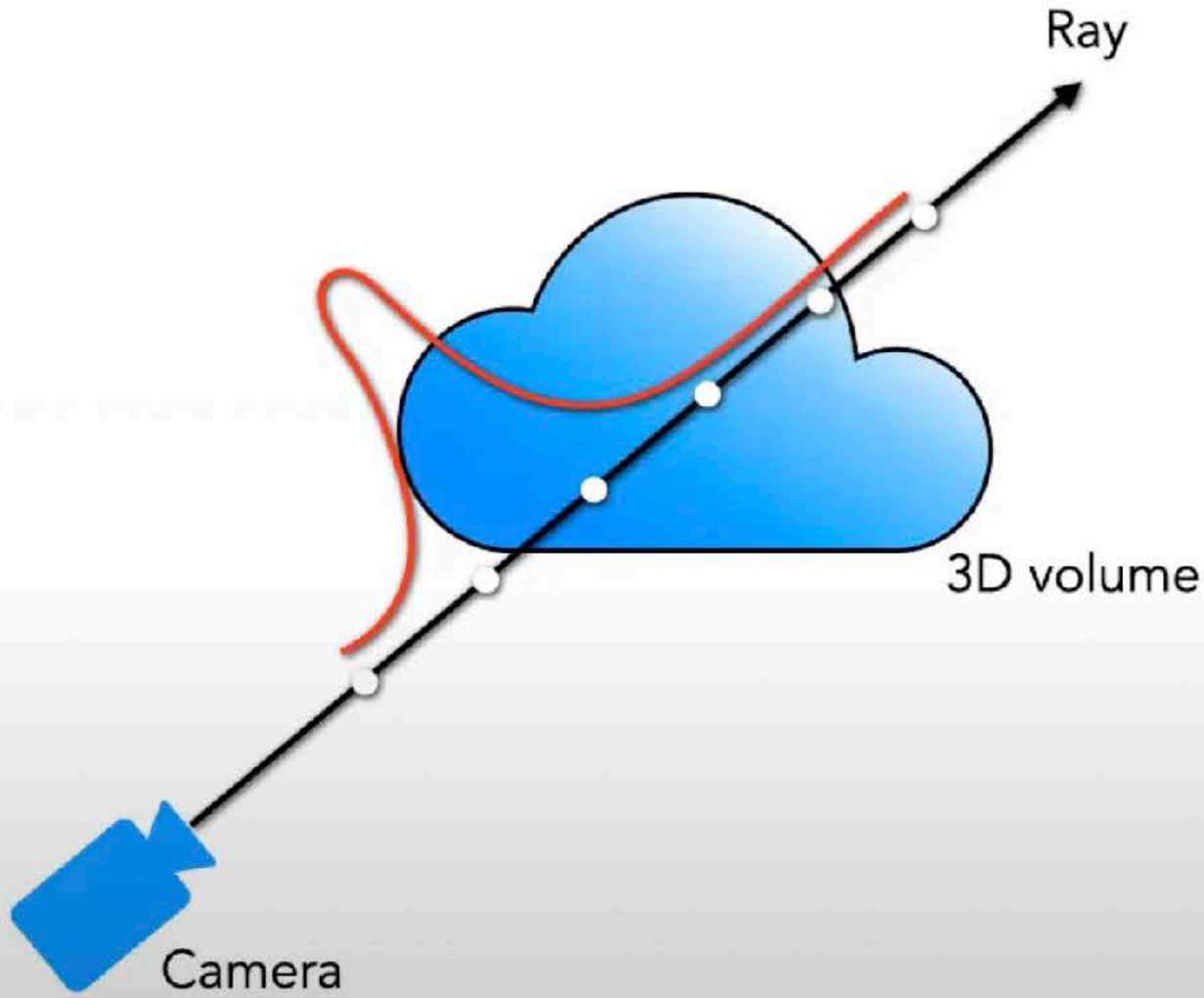
Tricks: Hierarchical Sampling



Tricks: Hierarchical Sampling - Coarse Sampling

$$C \approx \sum_{i=1}^{N} T_i \alpha_i c_i$$

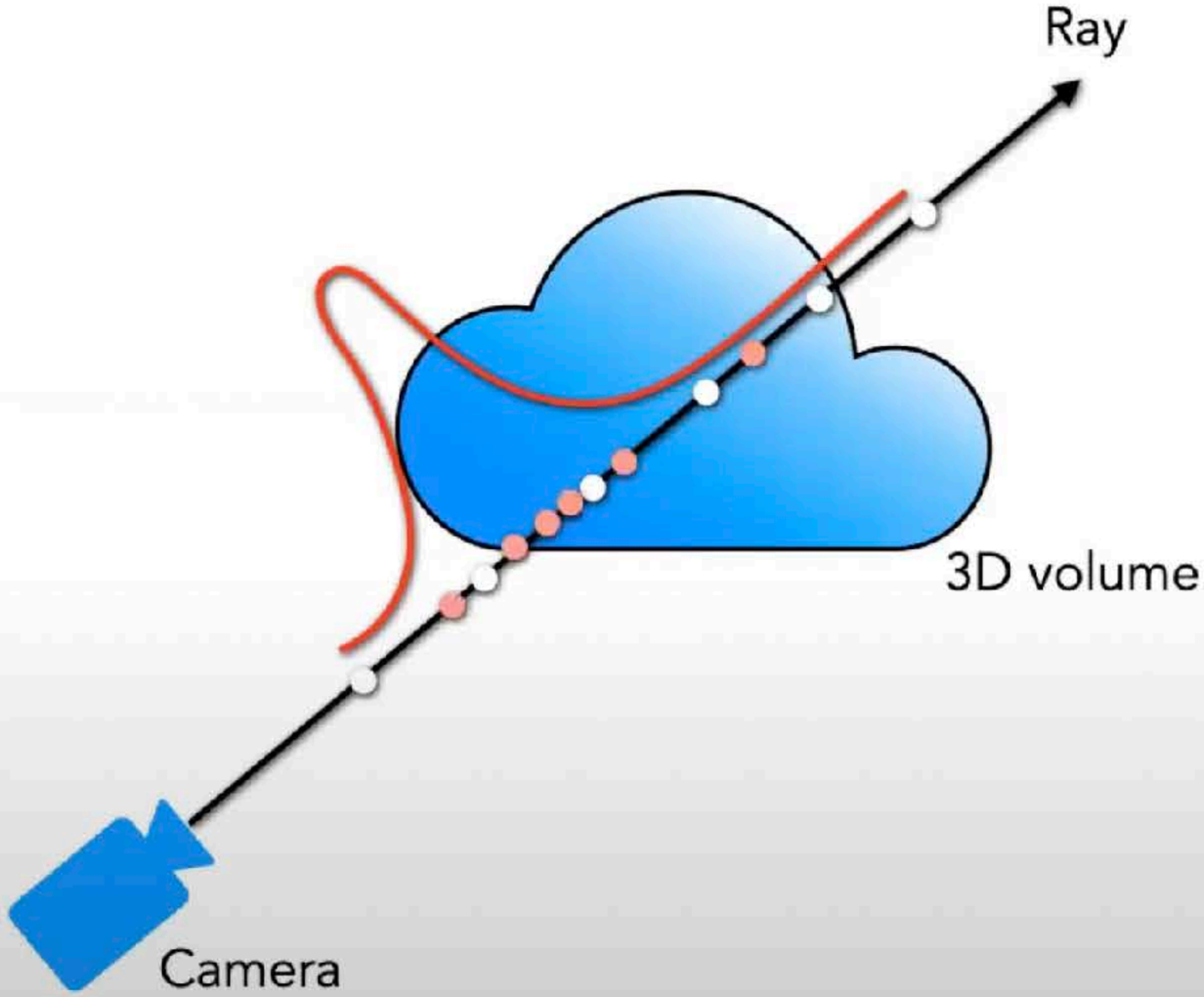
treat weights as probability distribution for new samples



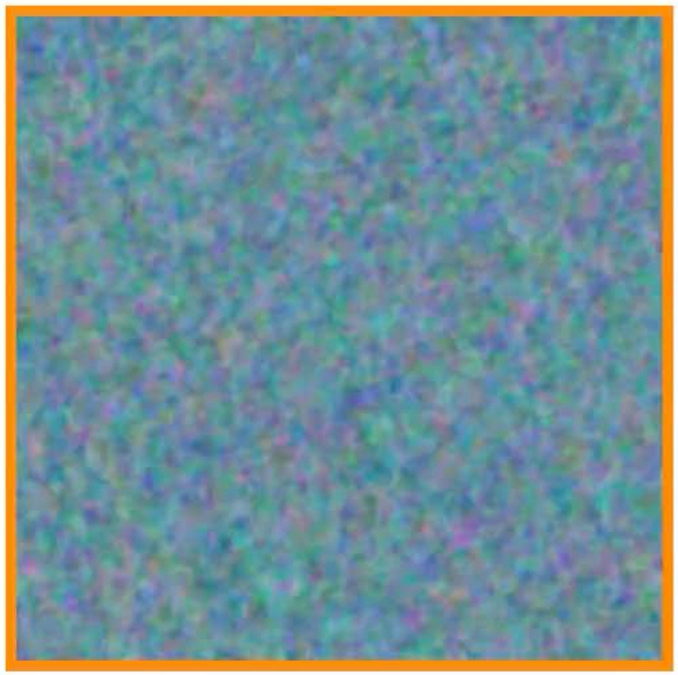
Tricks: Hierarchical Sampling - Fine Sampling

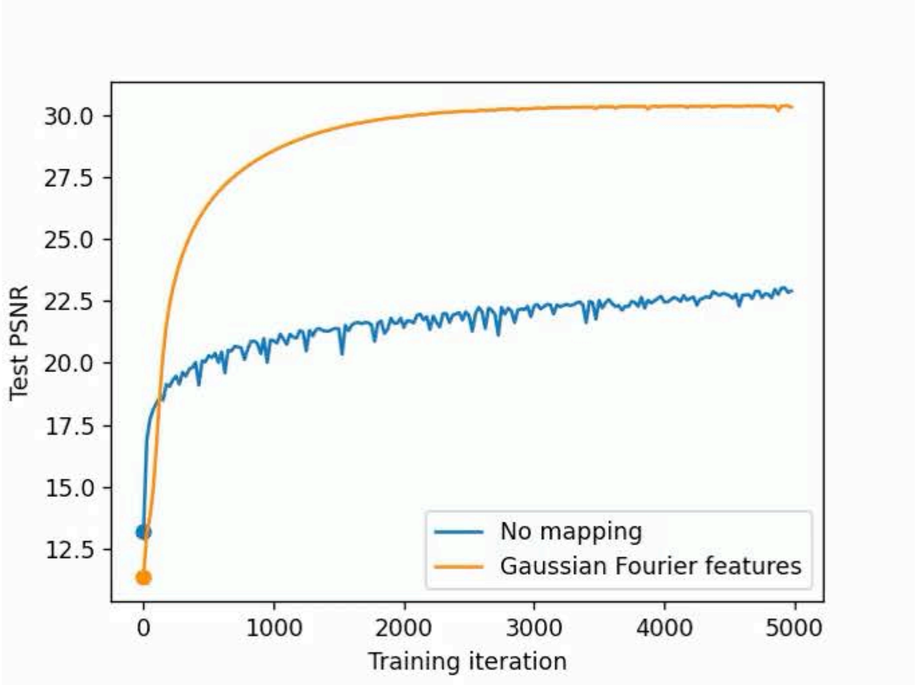
$$C \approx \sum_{i=1}^{N} T_i \alpha_i c_i$$

treat weights as probability distribution for new samples



Tricks: Positional Encoding





Tricks: Positional Encoding

- **<u>Positional encoding</u>** to map each input 5D coordinate into a higher dimensional space
  - Learning in high-frequency mappings is difficult to learn

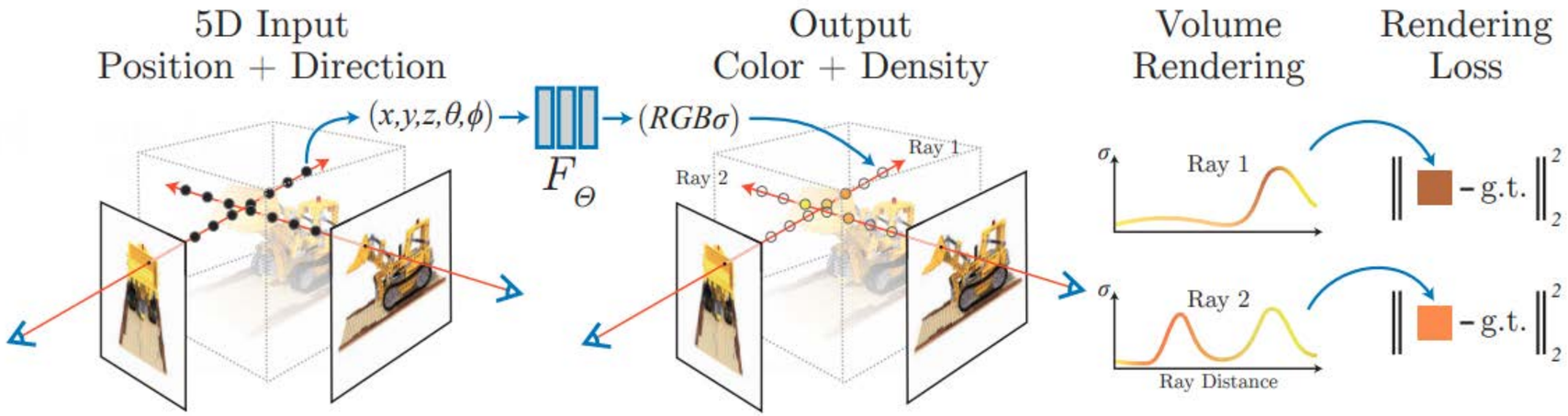
$$\gamma(p) = \left(\sin\left(2^0\pi p\right), \cos\left(2^0\pi p\right), \cdots, \sin\left(2^{L-1}\pi p\right), \cos\left(2^{L-1}\pi p\right)\right)$$

  - Fourier Basis feature mapping allocates neurons to different spatial frequency bands (frequency disentangling)

(a) Coordinate-based MLP (b) Image regression $(x,y) \rightarrow$ RGB (c) 3D shape regression $(x,y,z) \rightarrow$ occupancy (d) MRI reconstruction $(x,y,z) \rightarrow$ density (e) Inverse rendering $(x,y,z) \rightarrow$ RGB, density

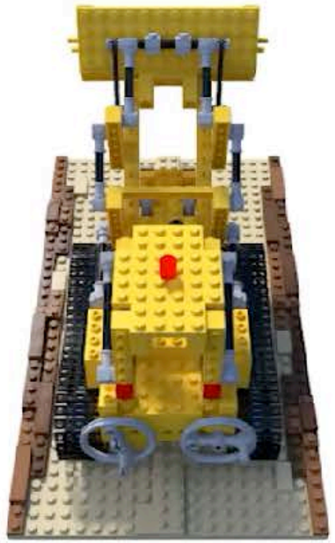
[7] Tancik20

NeRF in a Nutshell



- Learn the radiance field of a scene based on a collection of calibrated images
  - Use an MLP to learn continuous geometry and view-dependent appearance
- Use fully differentiable volume rendering with reconstruction loss
- Combines importance sampling and Fourier-basis encoding of 5D query to produce **high-fidelity novel view synthesis results**
- Allows efficient storage of scenes (x3000 gain over voxelized representations)

NeRF results





Remaining Challenges

- Handling dynamic scenes when acquiring calibrated views
- One network trained per scene - no generalization

Outline

- Neural Radiance Field (NeRF)
  - Implicit Functions: an Illustration with 3D Surface Representation
  - Volume Rendering with Ray Marching
  - Learning NeRF
- ***NeRF Extensions***

Remaining Challenges

- Handling dynamic scenes when acquiring calibrated views
  - ***D-NeRF: Neural Radiance Fields for Dynamic Scenes***
- One network trained per scene - no generalization
  - PixelNeRF

D-NeRF: Neural Radiance Fields for Dynamic Scenes

NeRF

- \- Only applicable to rigid scenes
- \- 5D continuous function
- \- Requiring multiple views of a rigid scene

D-NeRF

- \+ Applicable for rigid and non-rigid scenes
- \+ 6D continuous function by considering time-component as an additional input
- \+ Requiring a single view per time instant for non-rigid scenes.

[9] Pumarola20

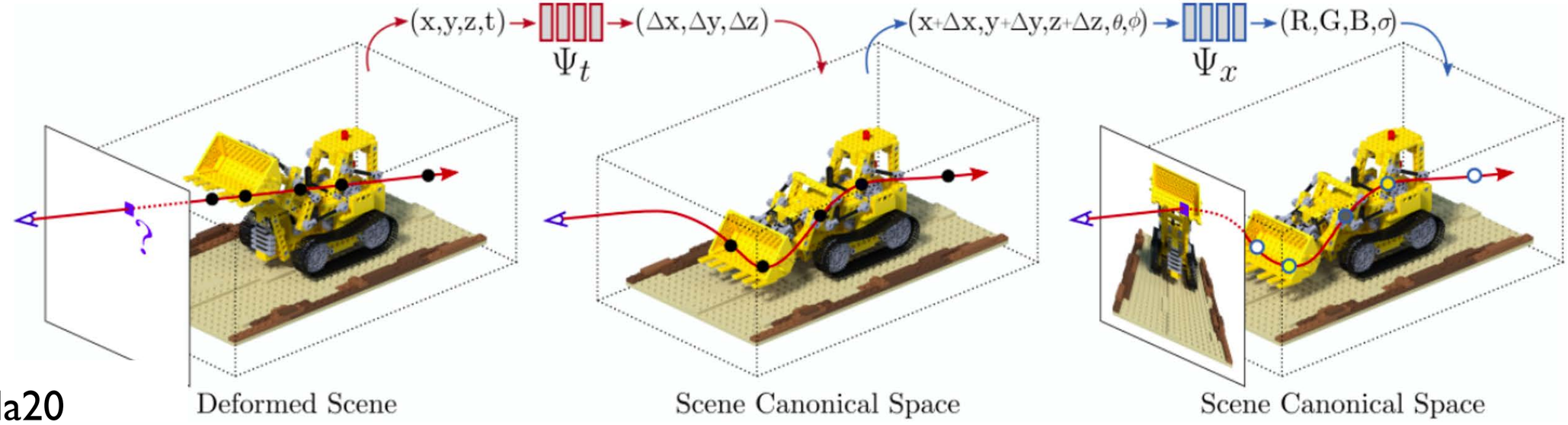
D-NeRF: Neural Radiance Fields for Dynamic Scenes

- **Deformation network** Ψ_t : to predict deformation field between the scene at time instant t and the scene in canonical space (t=0)

$$\Psi_t(\mathbf{x}, t) = \begin{cases} \Delta\mathbf{x}, & \text{if } t \neq 0 \\ 0, & \text{if } t = 0 \end{cases}$$

- **Canonical network** Ψ_x: to predict color and density in canonical configuration

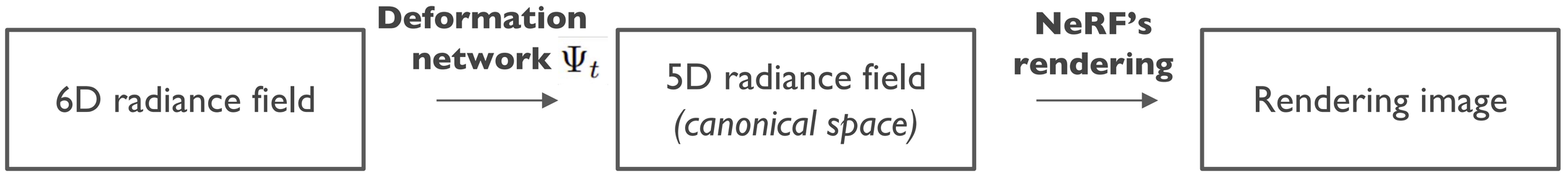
$$\Psi_x(\mathbf{x}, \mathbf{d}) \mapsto (\mathbf{c}, \sigma)$$



[9] Pumarola20

D-NeRF: Neural Radiance Fields for Dynamic Scenes

Volumetric rendering is the same as NeRF in **canonical space**:



Opacity

Predicted colors

$$C(p,t) = \int_{h_n}^{h_f} \mathcal{T}(h,t)\sigma(\mathbf{p}(h,t))\mathbf{c}(\mathbf{p}(h,t),\mathbf{d})dh$$

Volume density

where $\mathbf{p}(h,t) = \mathbf{x}(h) + \Psi_t(\mathbf{x}(h),t)$,

$[\mathbf{c}(\mathbf{p}(h,t),\mathbf{d}), \sigma(\mathbf{p}(h,t))] = \Psi_x(\mathbf{p}(h,t),\mathbf{d})$,

and $\mathcal{T}(h,t) = \exp\left(-\int_{h_n}^{h} \sigma(\mathbf{p}(s,t))ds\right)$.

[9] Pumarola20

D-NeRF: Neural Radiance Fields for Dynamic Scenes

D-NeRF
Neural Radiance Fields for Dynamic Scenes

A. Pumarola, E. Corona, G. Pons-Moll, F. Moreno-Noguer

Remaining Challenges

- Handling dynamic scenes when acquiring calibrated views
  - D-NeRF: Neural Radiance Fields for Dynamic Scenes
  - Deformable Neural Radiance Fields
- One network trained per scene - no generalization
  - ***PixelNeRF***

PixelNeRF: Neural Radiance Fields from One or Few Images

NeRF

- Optimizing NeRF of each scene independently
- Requiring many calibrated views
- Using canonical coordinate frame

PixelNeRF

[8] Yu20

+ Training across multiple scenes to learn a scene prior
+ Address few-shot view synthesis task with sparse set of views
+ Predicting a NeRF representation in the camera coordinate system
+ **Incorporate a variable number of posed input views**

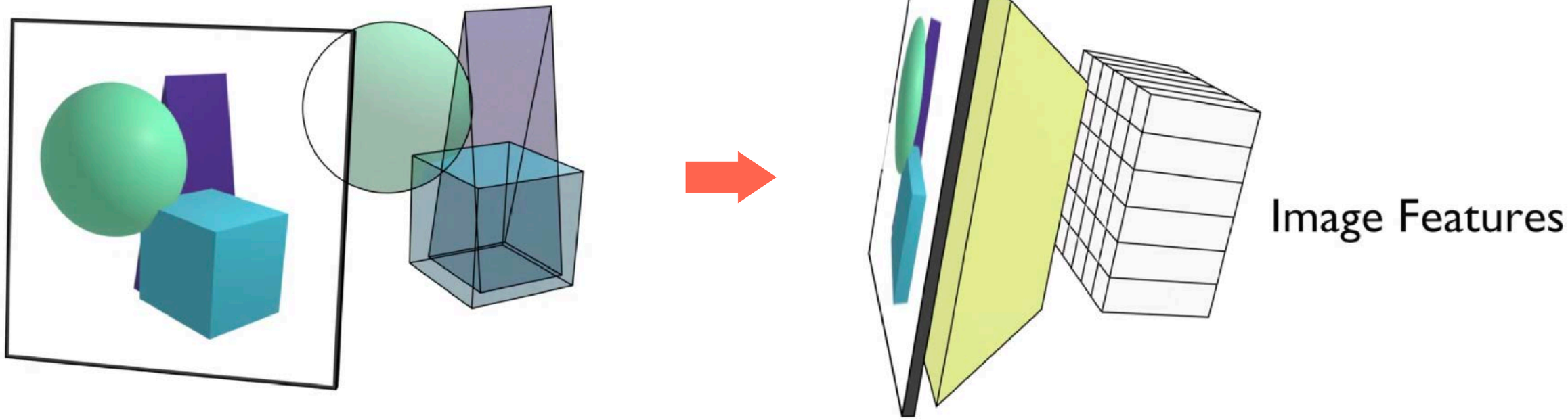
PixelNeRF

PixelNeRF

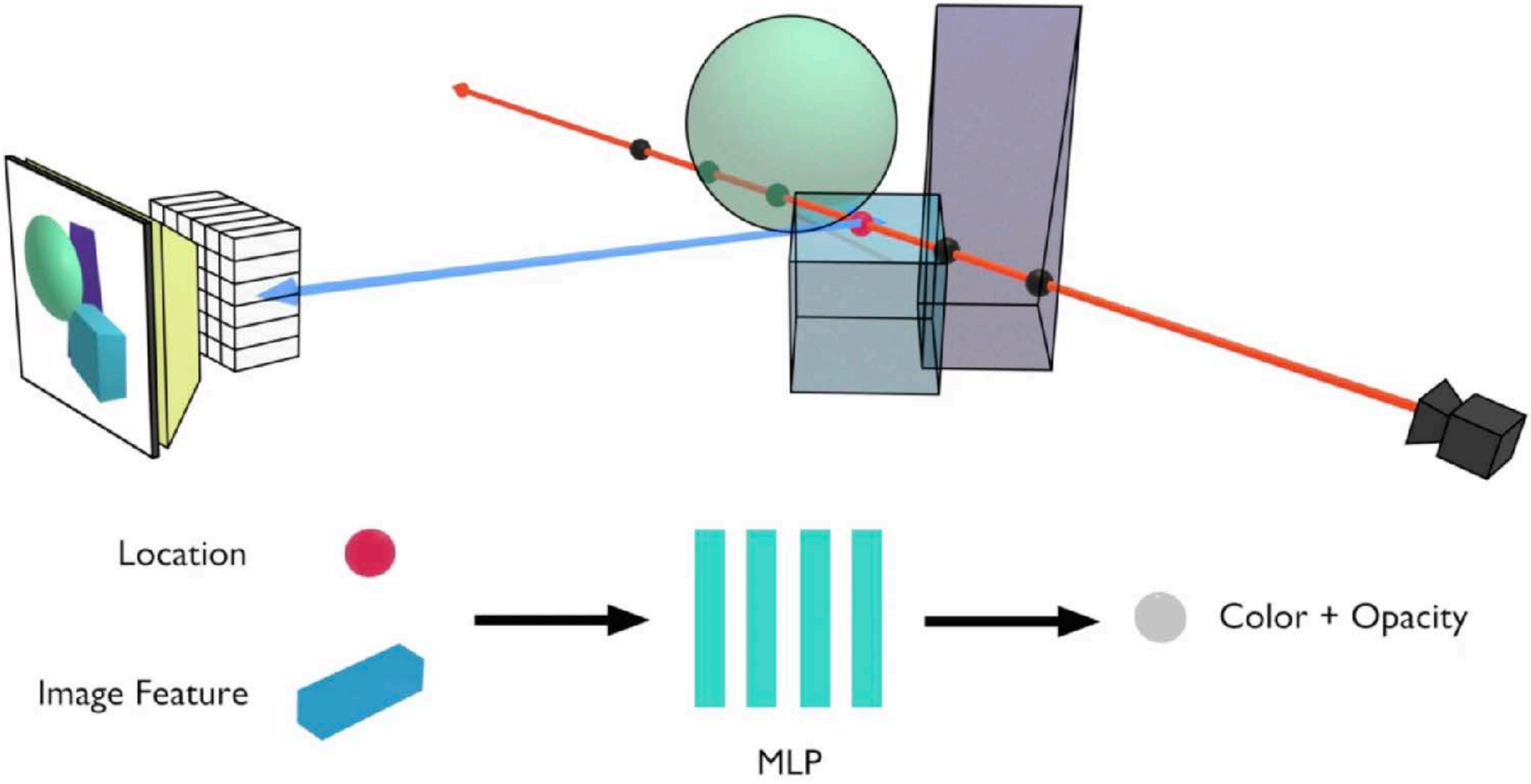
pixelNeRF

NeRF

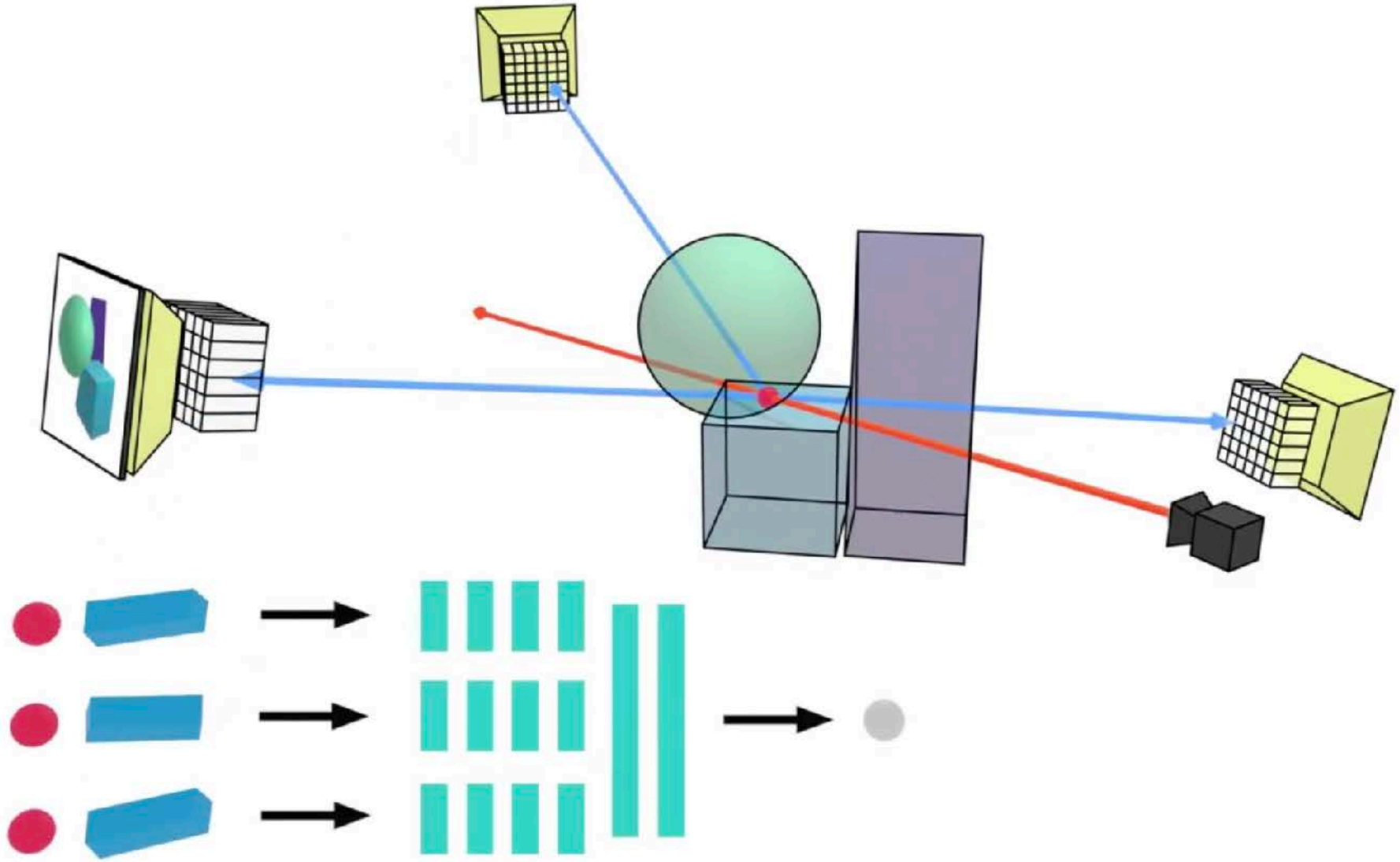
PixelNeRF



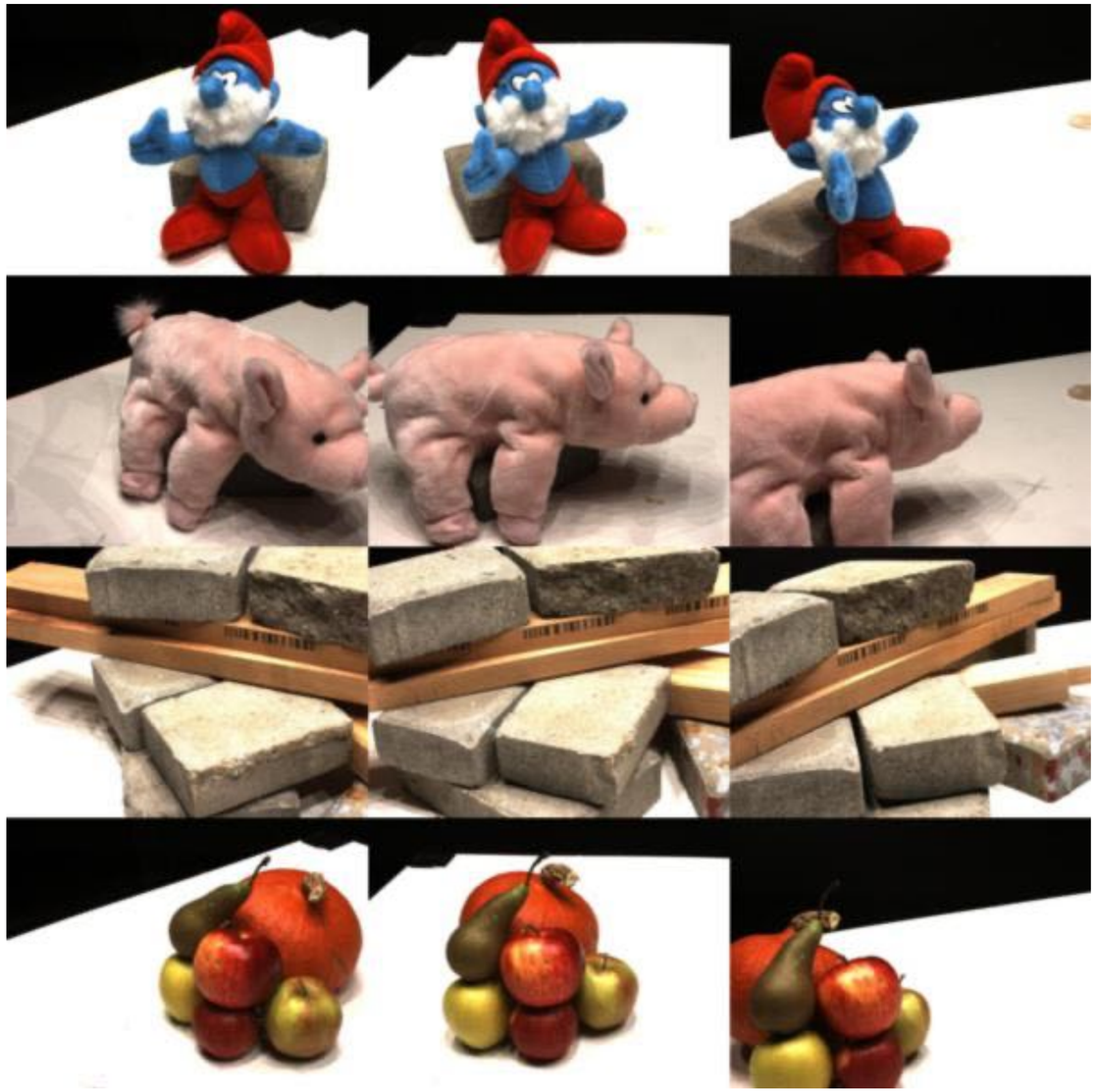
PixelNeRF

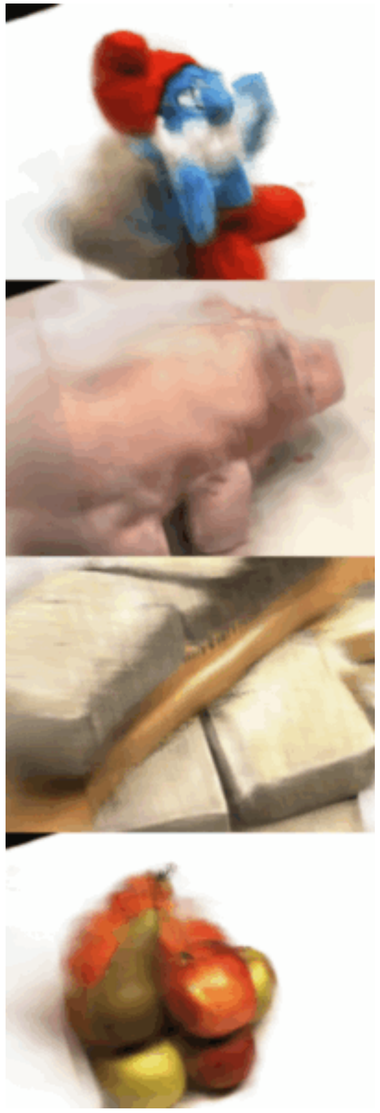


PixelNeRF



PixelNeRF





[8] Yu20

More Works on NeRF

Frank Dellaert

Professor, Robotics & Computer Vision

- Atlanta, GA
- Georgia Tech
- Email
- Twitter
- LinkedIn
- Github
- YouTube

NeRF Explosion 2020

21 minute read

Published: December 16, 2020

Frank Dellaert

Professor, Robotics & Computer Vision

- Atlanta, GA
- Georgia Tech
- Email
- Twitter
- LinkedIn
- Github

NeRF at ICCV 2021

12 minute read

Frank Dellaert

Professor, Robotics & Computer Vision

- Atlanta, GA
- Georgia Tech
- Email
- Twitter
- LinkedIn
- Github
- YouTube
- Google Scholar
- ORCID

NeRF at CVPR 2022

13 minute read

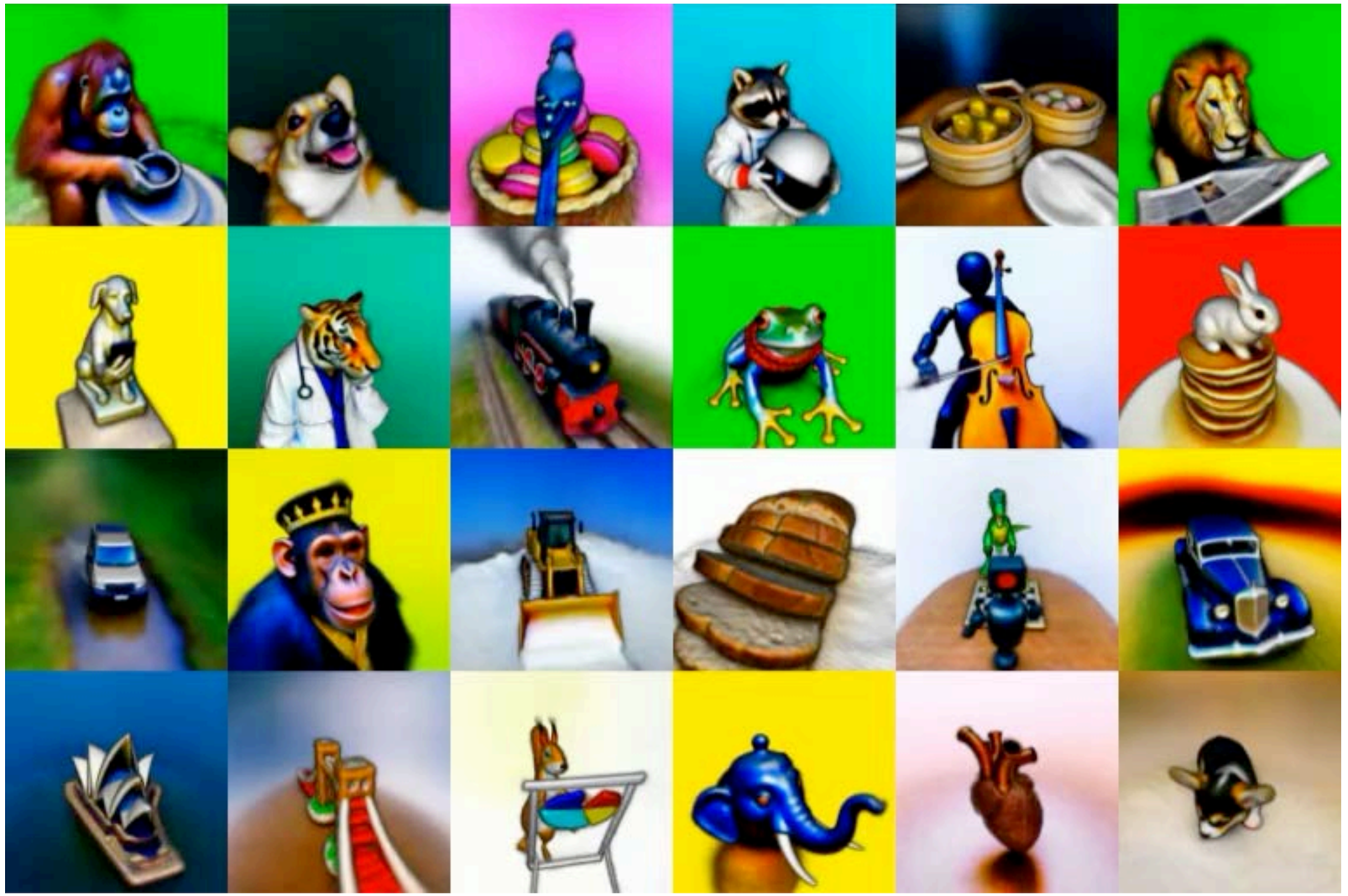
Published: June 21, 2022

There are more than 50 papers related to Neural Radiance Fields (NeRFs) at the CVPR 2022 conference. With my former student and now colleague at Google Research, Andrew Marmon, we rounded up all papers we could find and organized them here for our edification, and your reading pleasure.

Below are all the papers at CVPR'22 that we could find by scanning titles and reading the associated papers, sometimes rather superficially because of the sheer number. Please forgive any mis-characterizations and/or omissions, and feel free to flag them by DM to @fdellaert on twitter.

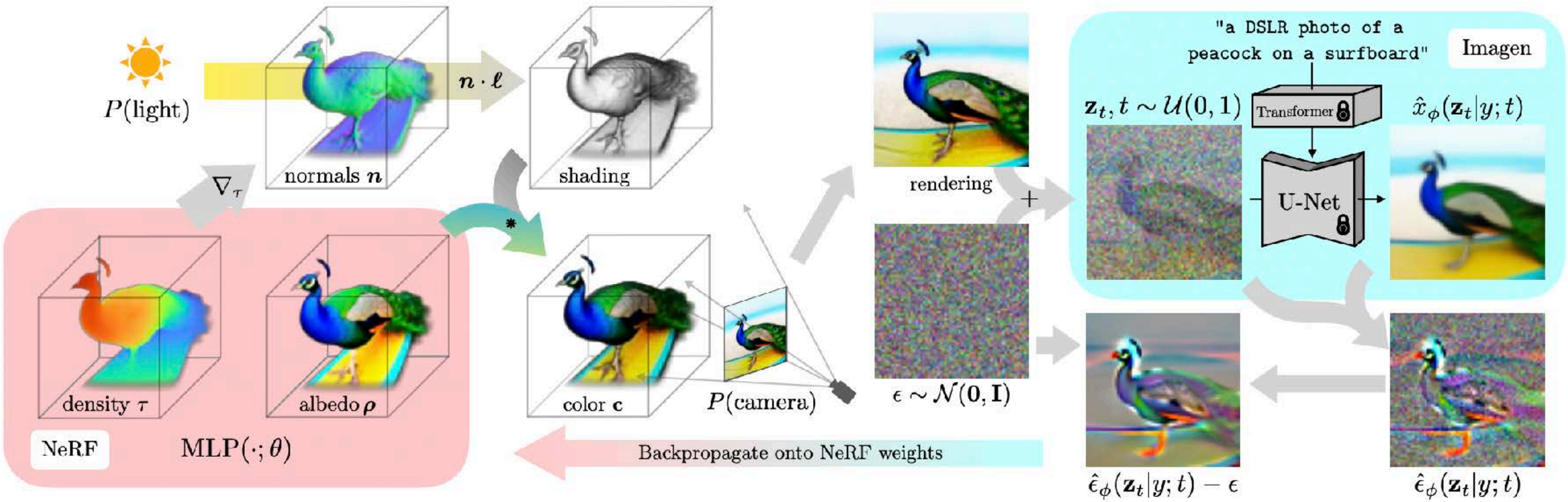
Important note: *all of the images below are reproduced from the cited papers, and the copyright belongs to the authors or the organization that published their papers, like IEEE. Below I reproduce a key figure or video for some papers under the fair use clause of copyright law.*

NeRF in AIGC



DREAMFUSION: TEXT-TO-3D USING 2D DIFFUSION
Poole et al. 2022

NeRF in AIGC

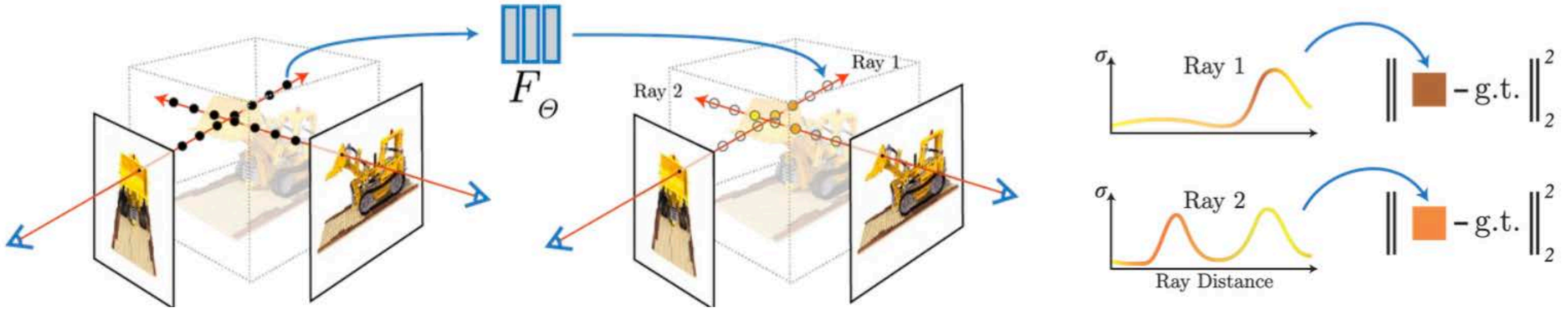


DREAMFUSION: TEXT-TO-3D USING 2D DIFFUSION
Poole et al. 2022

NeRF: Parameterize Radiance Field Densely

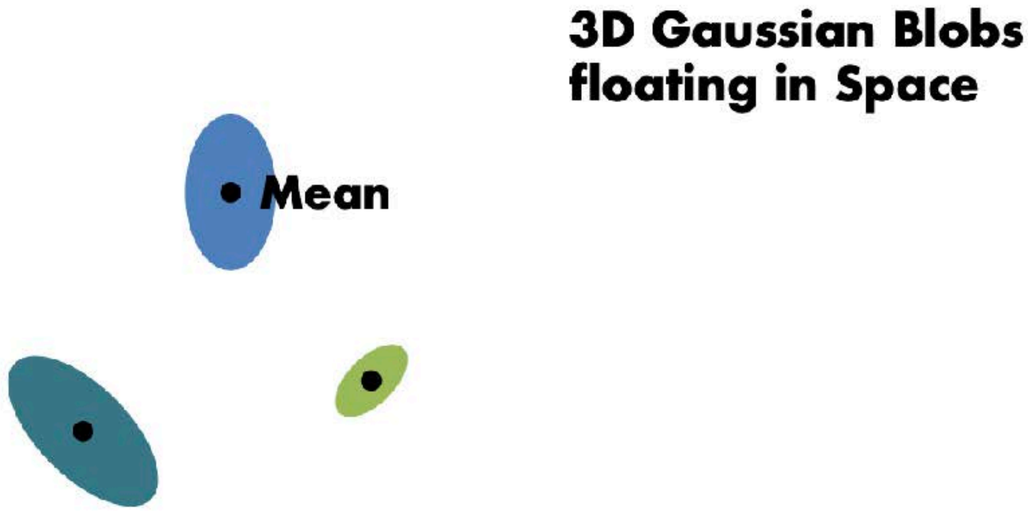
$$(x, y, z, \theta, \phi) \rightarrow \boxed{\quad} \rightarrow (RGB\sigma)$$

$$F_\Theta$$



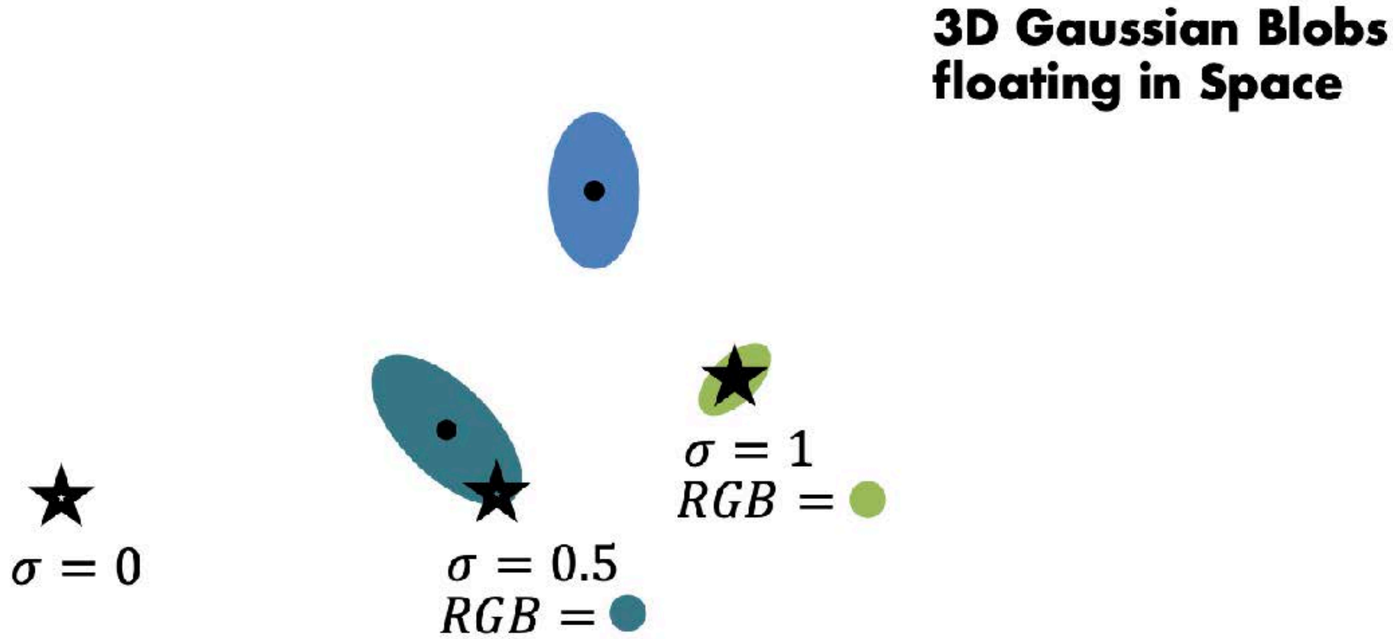
3D Gaussian Splatting (3DGS)

Key Idea: Parameterize Radiance Field *sparsely*,
only where density is nonzero

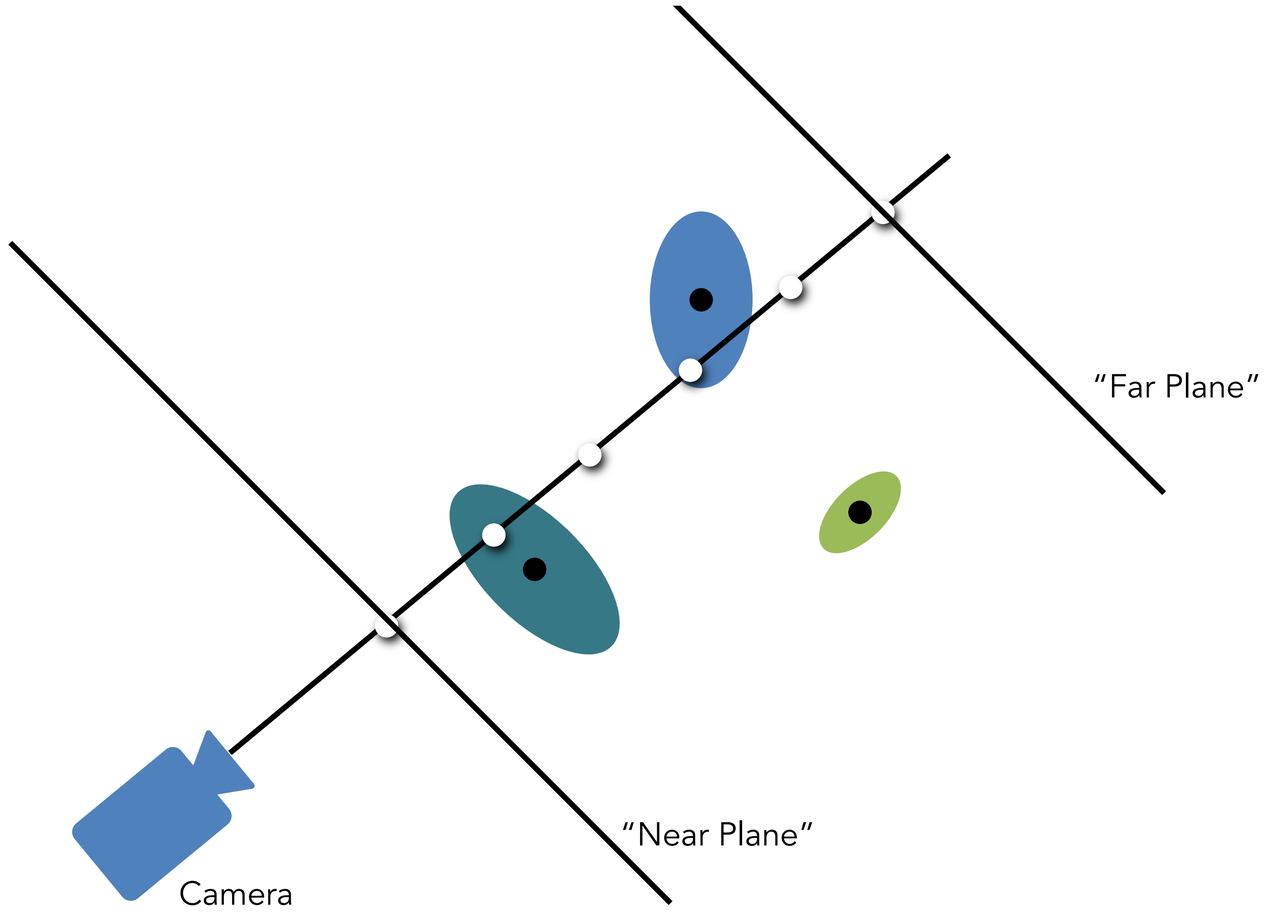


3D Gaussian Splatting (3DGS)

Key Idea: Parameterize Radiance Field *sparsely*,
only where density is nonzero

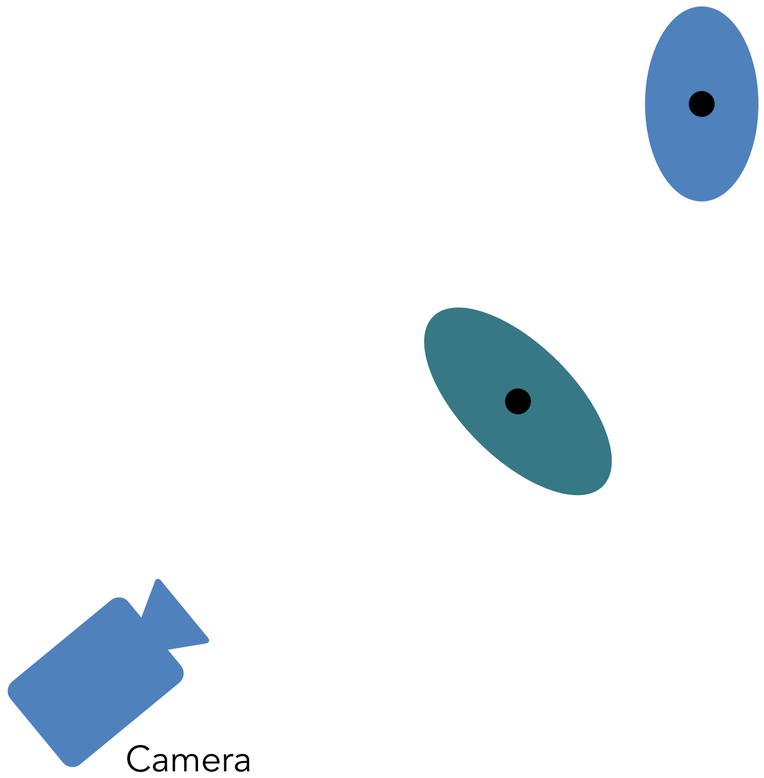


Still Volume Rendering?



Computation Properties of Gaussians

Gaussians are closed under affine transforms, integration



$$\mathcal{G}_{\mathbf{V}}(\mathbf{x}-\mathbf{p}) = \frac{1}{2\pi|\mathbf{V}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{p})^T \mathbf{V}^{-1}(\mathbf{x}-\mathbf{p})}$$

3D Covariance!

Affine mapping $\Phi = \mathbf{M}\mathbf{x} + \mathbf{p}$ of coordinates (such as cam2world matrix!):

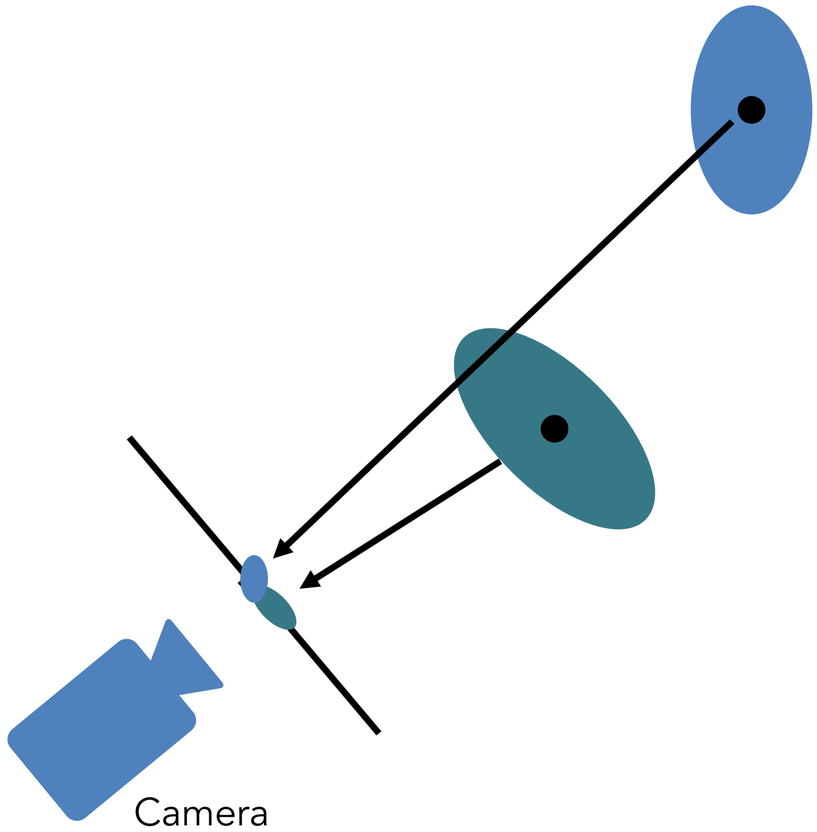
$$\mathcal{G}_{\mathbf{V}}(\Phi^{-1}(\mathbf{u}) - \mathbf{p}) = \frac{1}{|\mathbf{M}^{-1}|}\mathcal{G}_{\mathbf{M}\mathbf{V}\mathbf{M}^T}(\mathbf{u} - \Phi(\mathbf{p}))$$

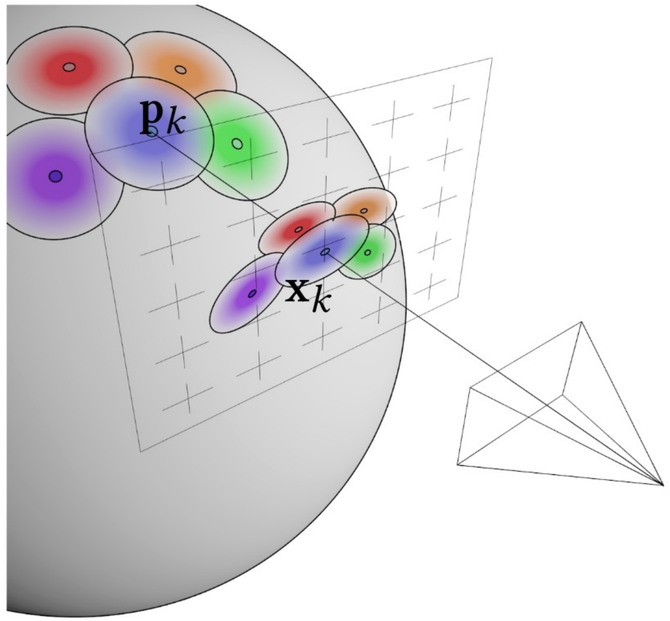
Integrate along axis:

$$\int_{\mathbb{R}} \mathcal{G}^3_{\mathbf{V}}(\mathbf{x}-\mathbf{p})\, dx_2 = \mathcal{G}^2_{\hat{\mathbf{V}}}(\hat{\mathbf{x}}-\hat{\mathbf{p}})$$

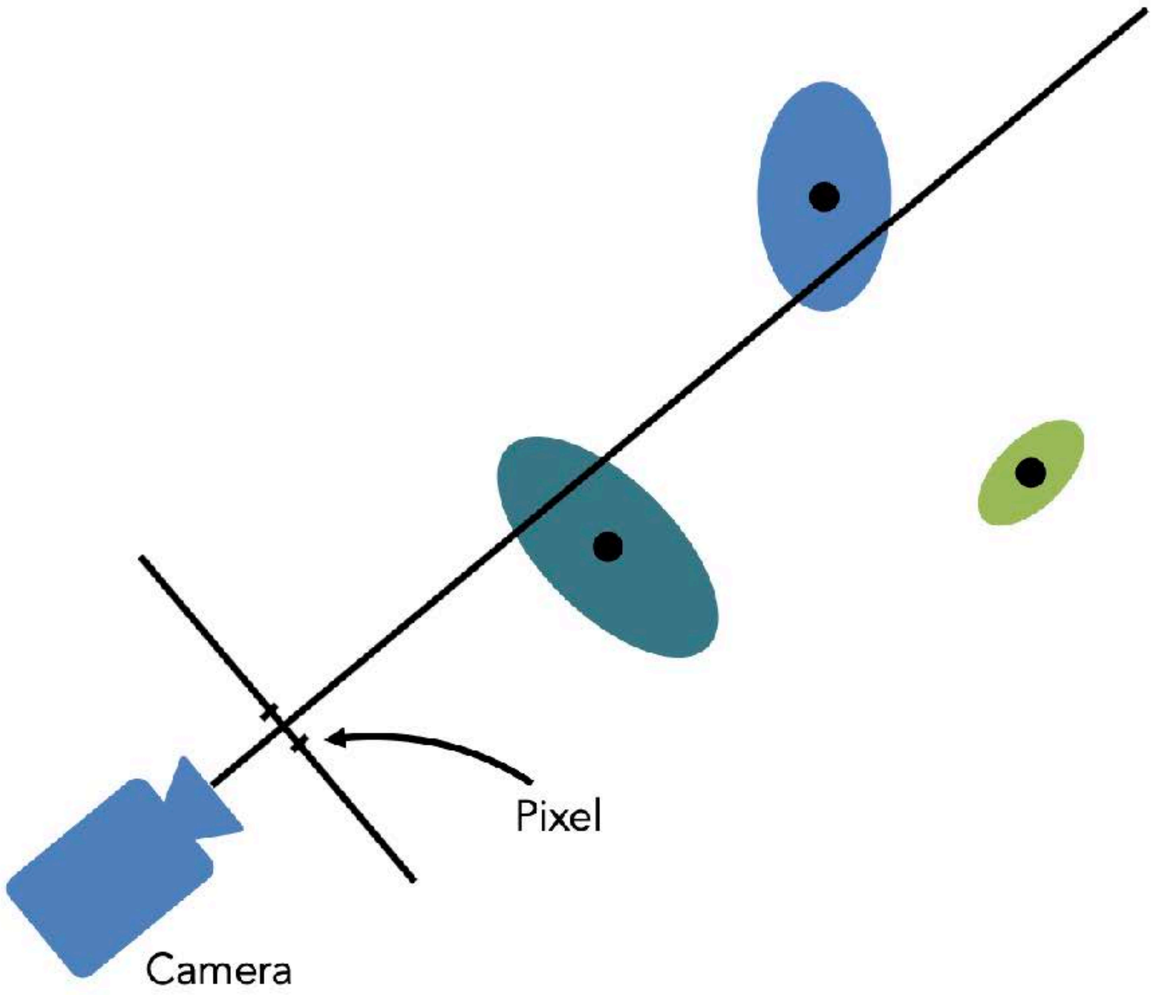
$$\mathbf{V} = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix} \Leftrightarrow \begin{pmatrix} a & b \\ b & d \end{pmatrix} = \hat{\mathbf{V}}$$

Projected 3D Gaussian makes 2D Gaussian

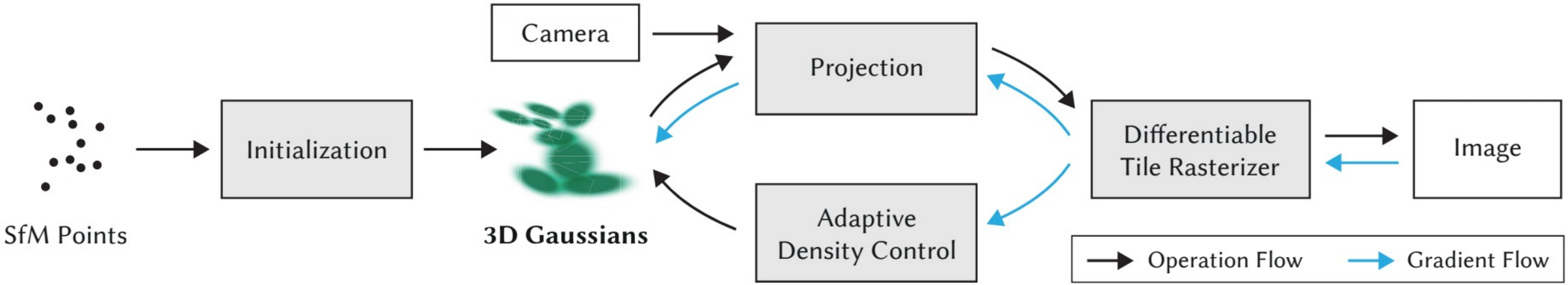




Using Rasterization Instead of Volume Rendering

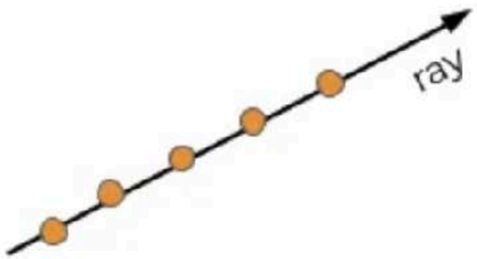


3DGS Framework

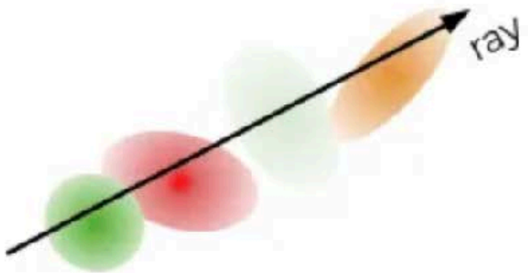


3D Gaussian Splatting

NeRF



Gaussian Splatting



Next Time

Single Image to 3D

References

[1] Genova et al., Learning Shape Templates with Structured Implicit Functions, ICCV 2019
[2] Genova et al., Local Deep Implicit Functions for 3D Shape, CVPR 2020
[3] Park et al., DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation, CVPR 2019
[4] Mescheder et al., Occupancy Networks: Learning 3D Reconstruction in Function Space, CVPR 2019
[5] Saito, Huang, Natsume et al., PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization, ICCV 2019
[6] Saito et al., PIFuHD: Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization, CVPR 2020
[7] Tancik, Srinivasan, Mildenhall et al., Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains, NeurIPS 2020
[8] Yu et al., PixelNeRF: Neural Radiance Fields from One or Few Images, Arxiv preprint 2020
[9] Pumarola et al., D-NeRF: Neural Radiance Fields for Dynamic Scenes, Arxiv preprint 2020
[10] Park et al., Deformable Neural Radiance Fields, Arxiv preprint 2020