Towards Efficient Neural Rendering

Sungjoo Yoo

Computing Memory Architecture Lab.

Computer Science and Engineering

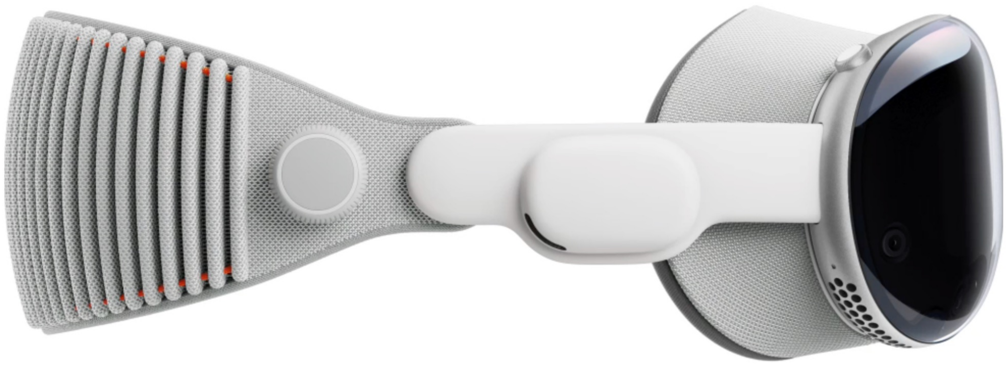
Seoul National University

Agenda

- Introduction to neural rendering
- Problem: high computation cost
- Efficient model #1: voxel grid
- Efficient model #2: mesh
- Summary and prospect

Photo-Realistic Experience on Mobile Devices

- Entertainment (games, sports, movies, music performance, …)
- Productivity (virtual office, meeting room, …)

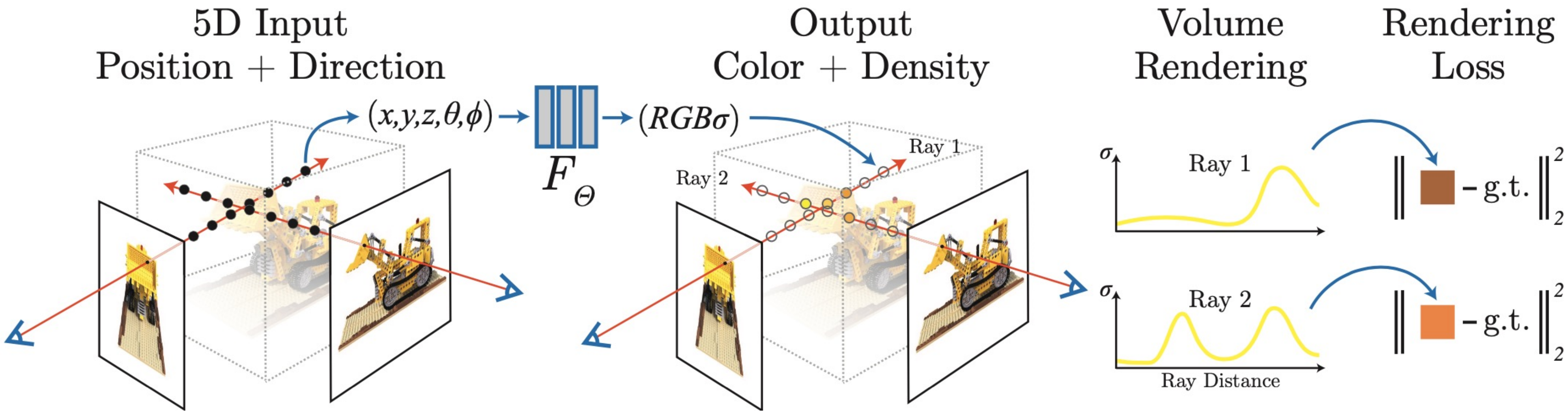


[https://www.apple.com/apple-vision-pro/]

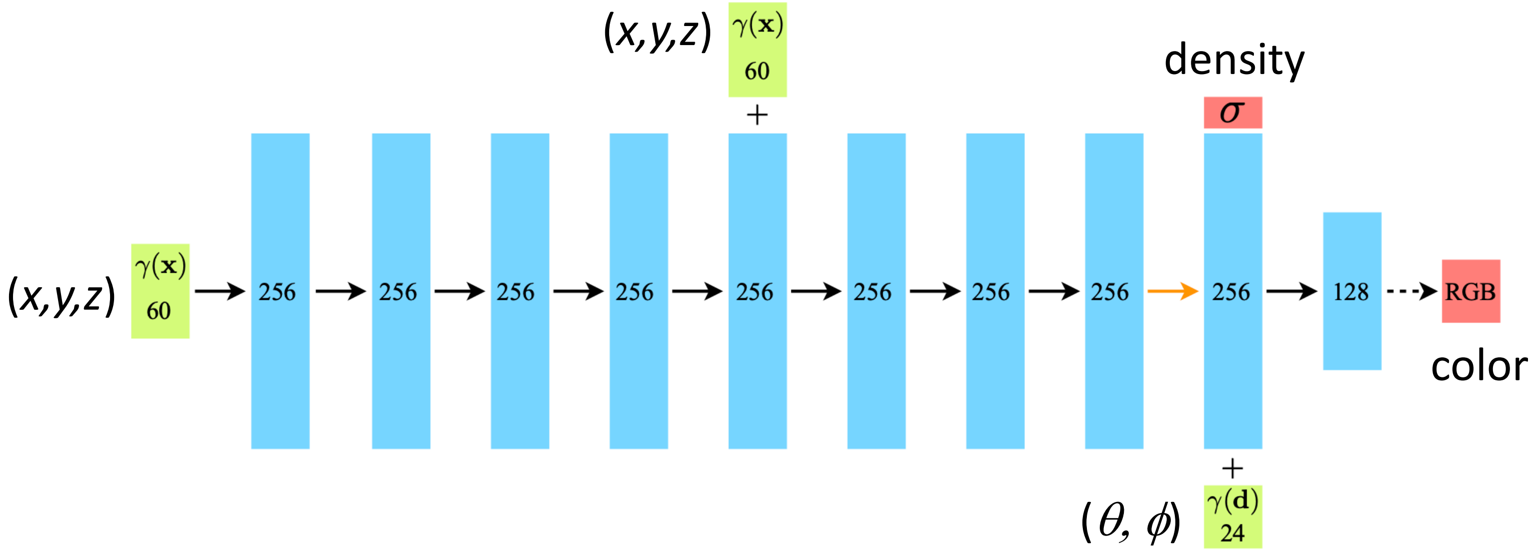
[Spatial VR meeting app]

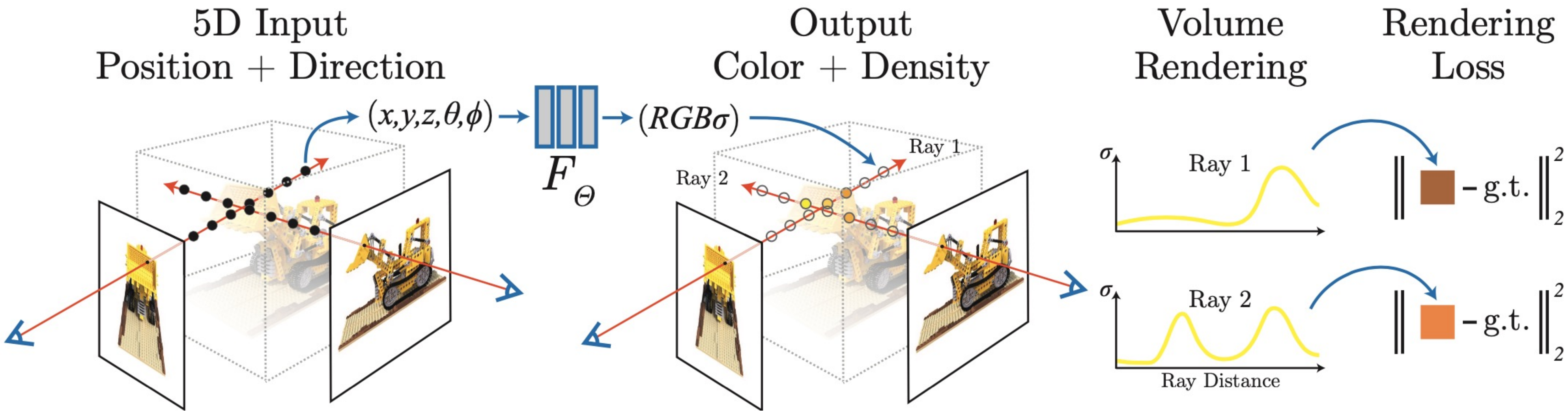
How to Obtain Virtual Objects/Scenes from Images?

- Recently (in 2020), neural rendering (aka neural radiance field, NeRF) was proposed
- It enables us to train a neural network (F_{Θ}), with captured images, to generate novel view images



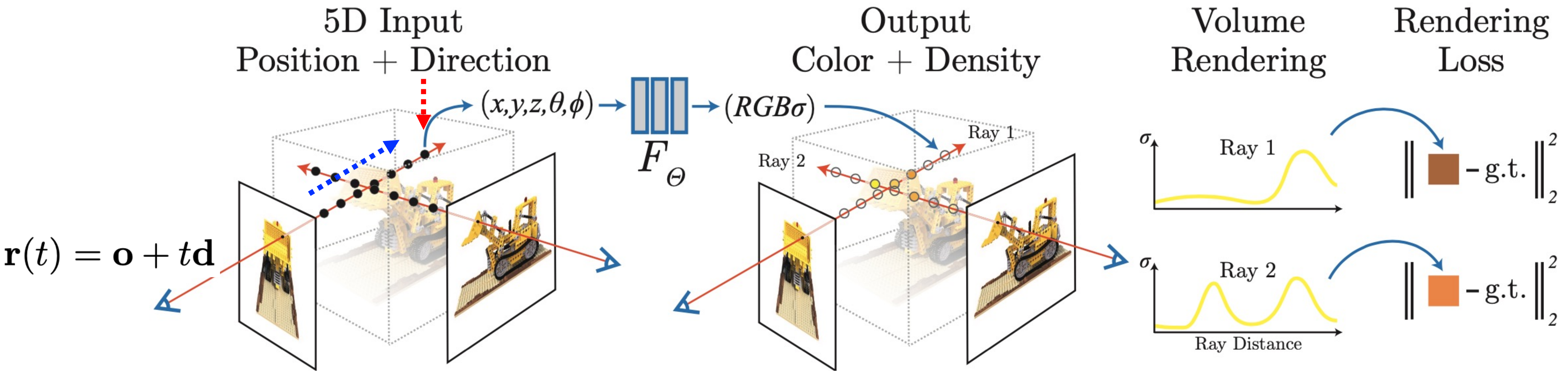
MLP for Radiance Field (Color and Density)





[NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis]

Volume Rendering: Alpha Composition



Transmittance = ray survival prob. on 1 … *i*-1

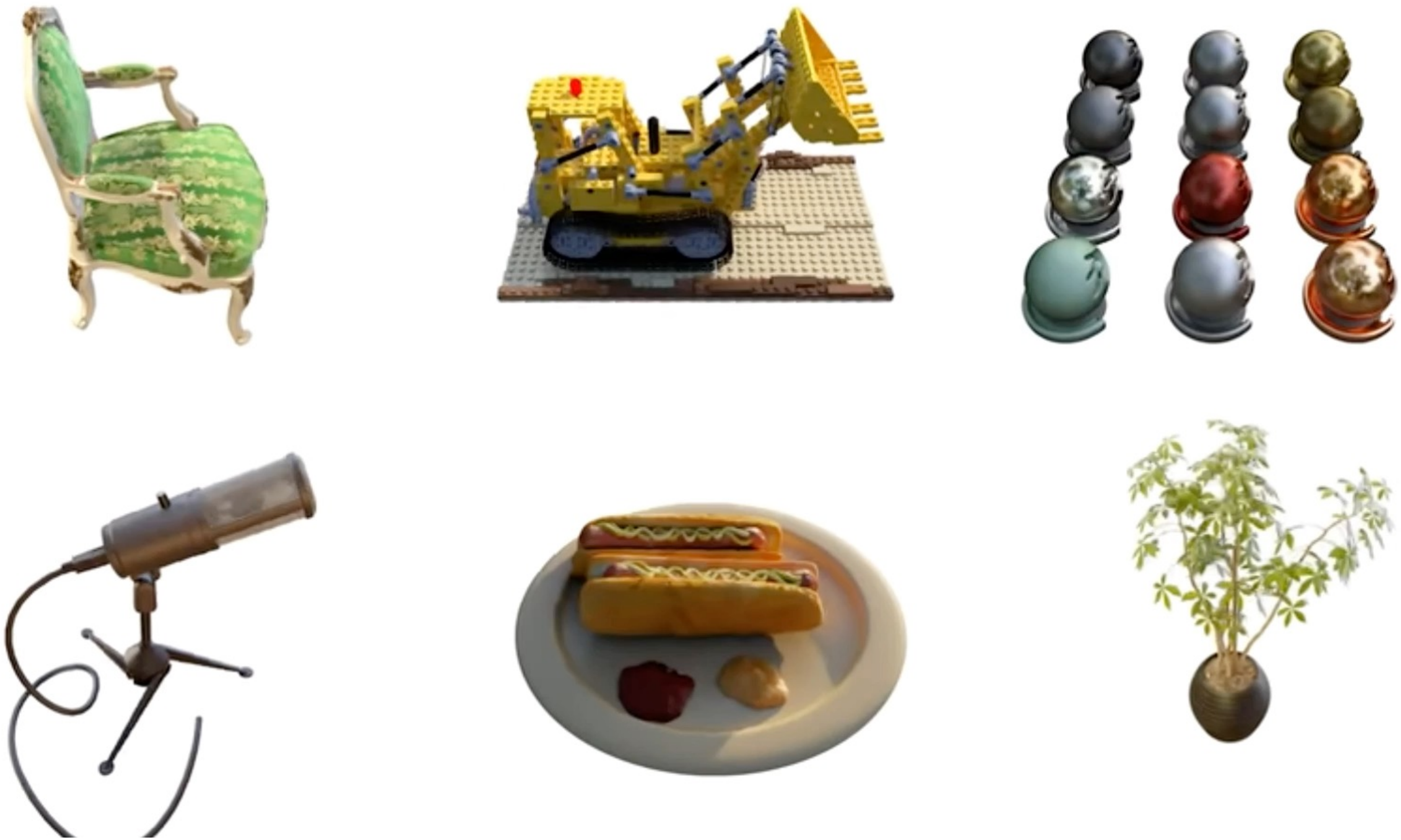
$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i \left(1 - \exp(-\sigma_i \delta_i)\right) \mathbf{c}_i \,, \quad \text{where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

probability of surface

w_i, weight of sample point *i*

[https://www.youtube.com/watch?v=JuH79E8rdKc]

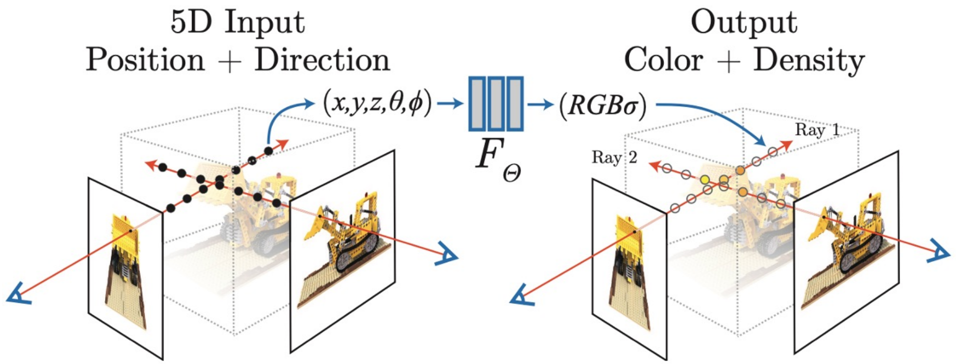
3D Models Trained from Images



Agenda

- Introduction to neural rendering
- **Problem: high computation cost**
- Efficient model #1: voxel grid
- Efficient model #2: mesh
- Summary and prospect

High Compute Cost of NeRF



- Assume a VR headset with 1440x1600 pixels, 90Hz
  - At each pixel, we run 8-layer MLP of 256 hidden dimension on 256 sample points
  - Total # multiplications = 1440x1600 x 90 x 256 x 256x256x7 = 2.4 x 10^{16} = 24 Peta multiplications/second
  - ~ 1000X more compute than 50 TOPS (of future mobile NPU) is needed for real time
- Methods for compute cost reduction
  - Voxel grid to exploit pre-computation
  - Mesh to exploit existing graphics pipeline
  - Light field network to exploit existing CNN acceleration

Agenda

- Introduction to neural rendering
- Problem: high computation cost
- **Efficient method #1: voxel grid**
- Efficient method #2: mesh
- Summary and prospect

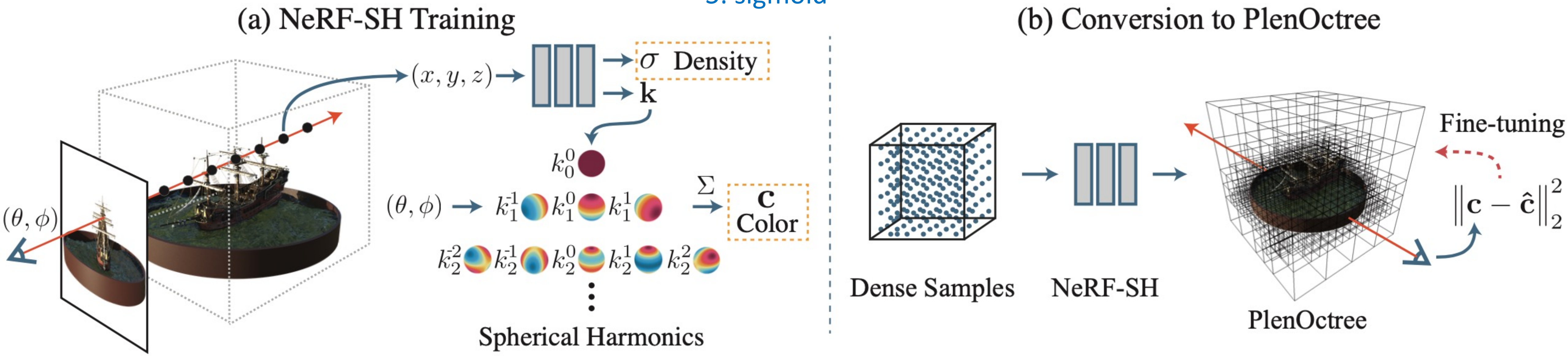
PlenOctree (ICCV 21)
Train NeRF, Build a Voxel Grid and Fine-tune It

- NeRF-SH, f(x)
- Color, c(d; k)

$$f(\mathbf{x}) = (\mathbf{k}, \sigma) \quad \text{where} \quad \mathbf{k} = (k_\ell^m)_{\ell:\, 0\leq\ell\leq\ell_{\max}}^{m:\, -\ell\leq m\leq\ell}$$

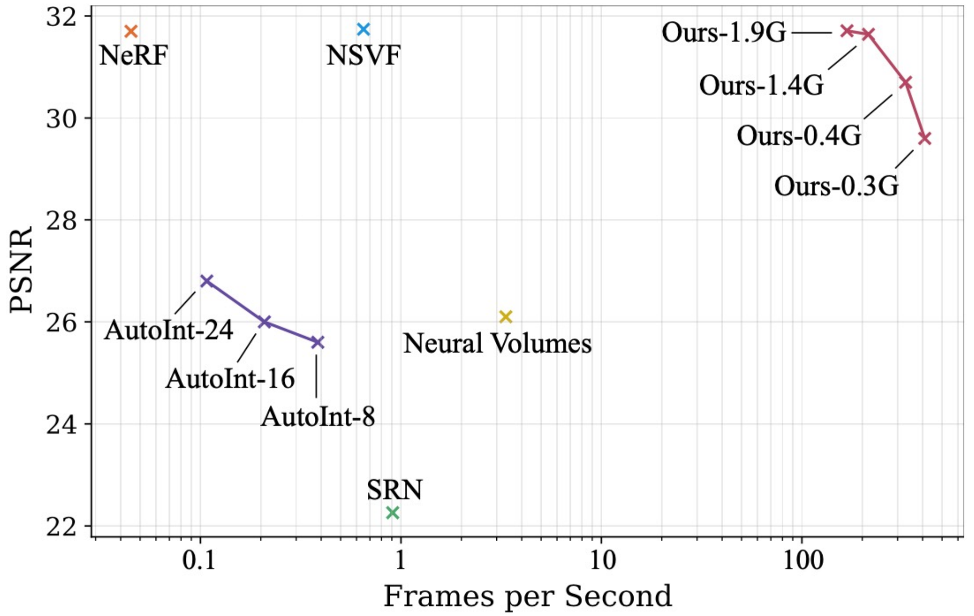
$$c(\mathbf{d}; \mathbf{k}) = S\left(\sum_{\ell=0}^{\ell_{\max}} \sum_{m=-\ell}^{\ell} k_\ell^m Y_\ell^m(\mathbf{d})\right)$$

S: sigmoid



>1000X Faster than NeRF

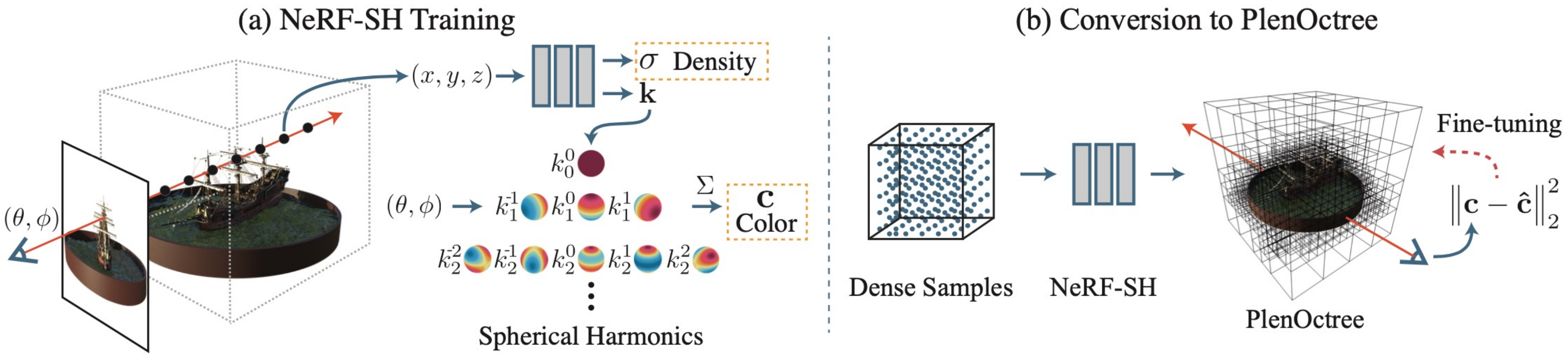
- Low compute cost/sample point
  - Voxel grid contains pre-computed values
  - Rendering requires simple computation on the **pre-computed values**
- Small # sample points/ray
  - Sparsity to skip compute on empty space
  - Visibility to skip compute on unseen space, e.g., interior of object

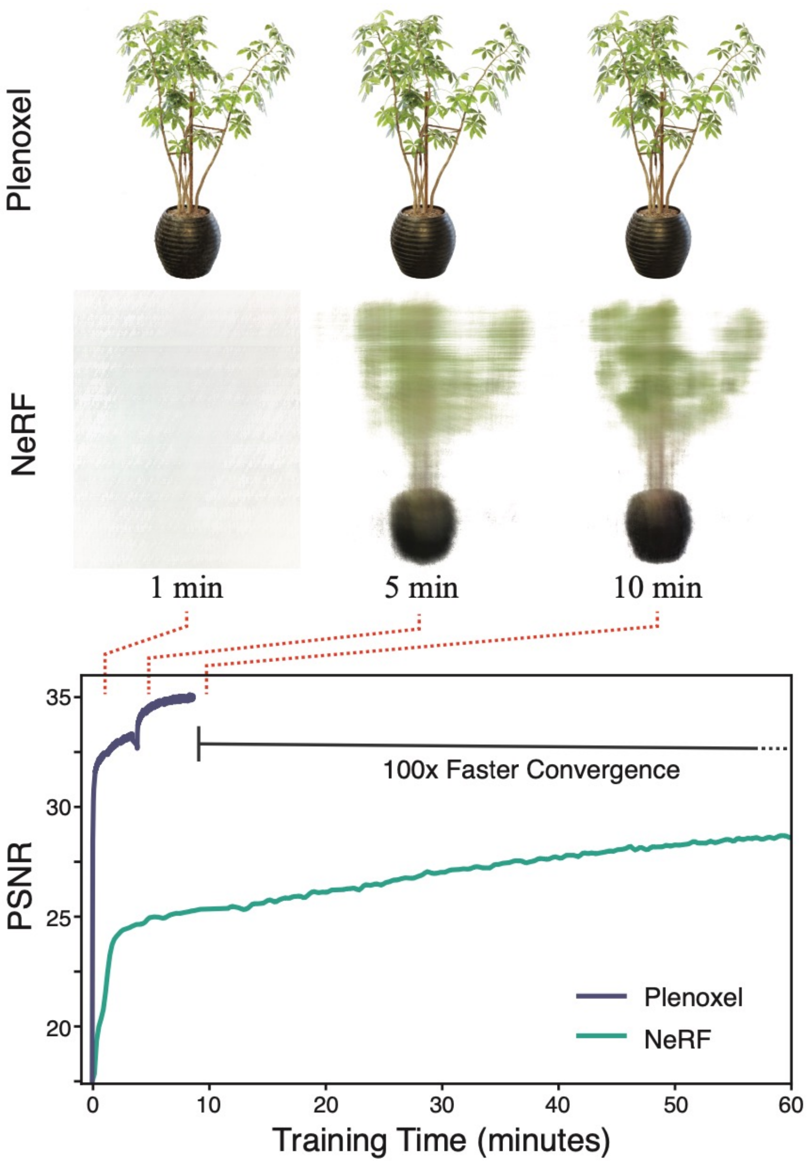


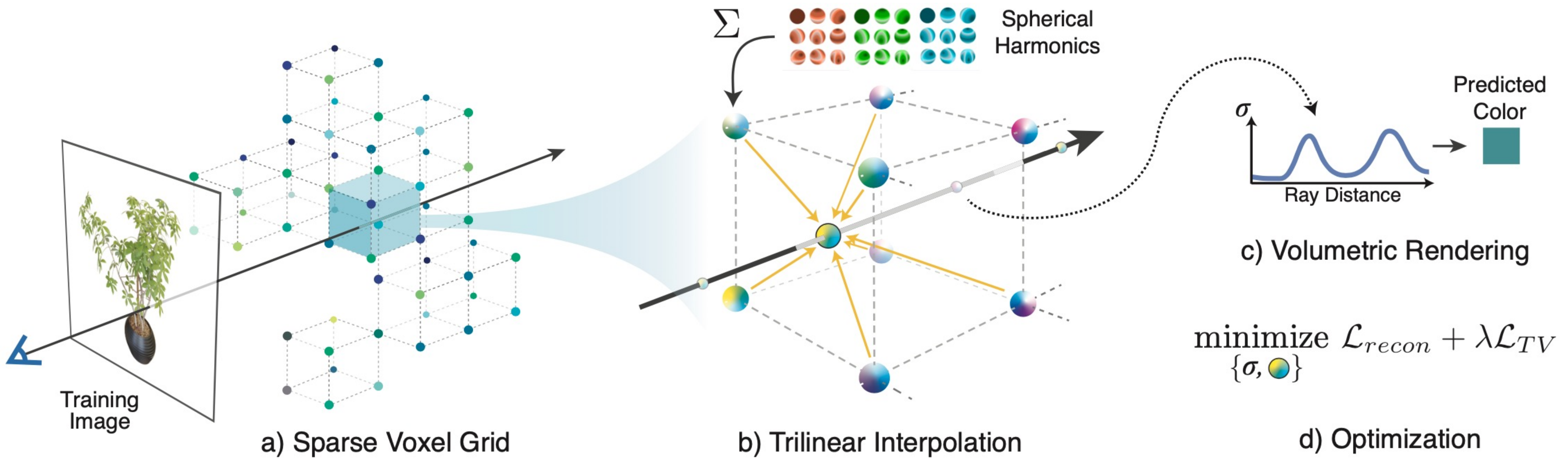
Tanks and Temples Dataset best second-best

	PSNR ↑	SSIM ↑	LPIPS ↓	FPS ↑
NeRF (original)	25.78	0.864	0.198	0.007
NeRF	27.94	0.904	0.168	0.013
SRN	24.10	0.847	0.251	0.250
Neural Volumes	23.70	0.834	0.260	1.000
NSVF	28.40	0.900	0.153	0.163
NeRF-SH	27.82	0.902	0.167	0.015
PlenOctree from NeRF-SH	27.34	0.897	0.170	42.22
PlenOctree after fine-tuning	27.99	0.917	0.131	42.22

Training Voxel Grid, From Scratch Without MLP, is Much Faster!







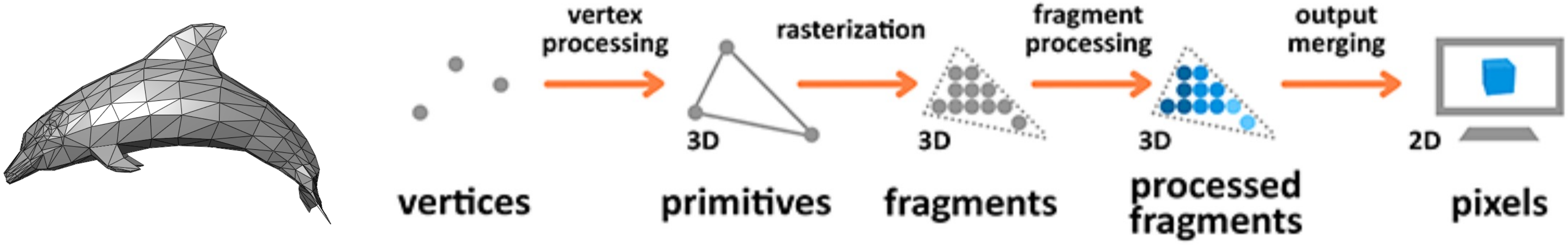
Agenda

- Introduction to neural rendering
- Problem: high computation cost
- Efficient method #1: voxel grid
- **Efficient method #2: mesh**
- Summary and prospect

[https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_on_the_web/Basic_theory]

How to Exploit Existing Fast Rendering Graphics Pipeline on GPU?

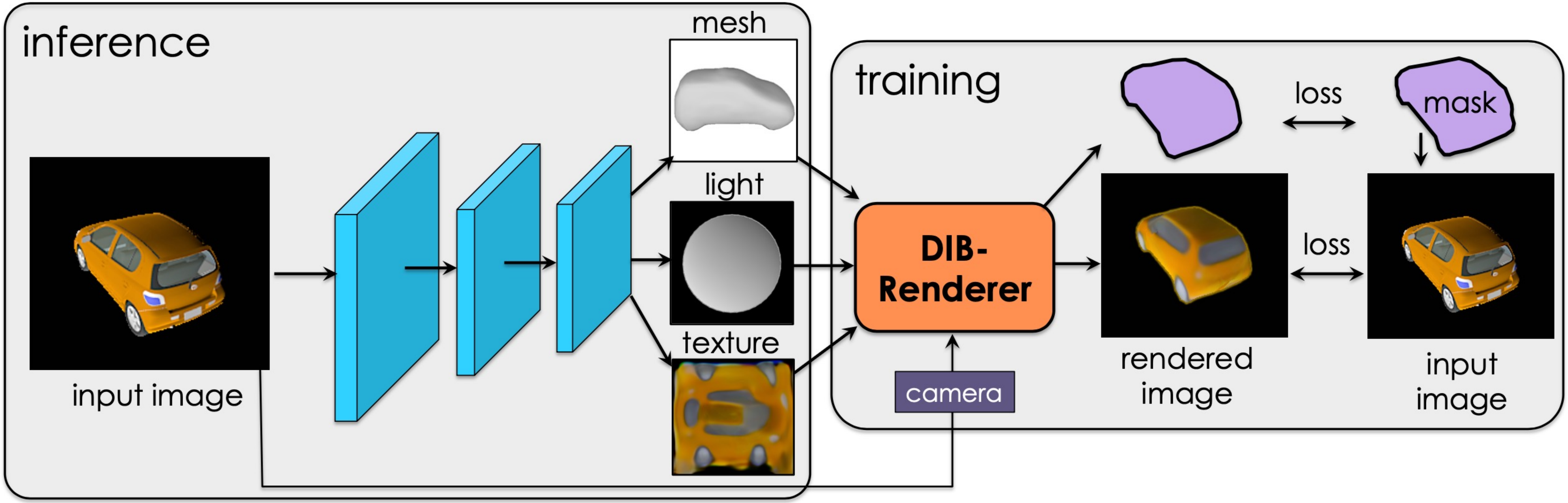
- Traditional rendering pipeline from mesh to pixel



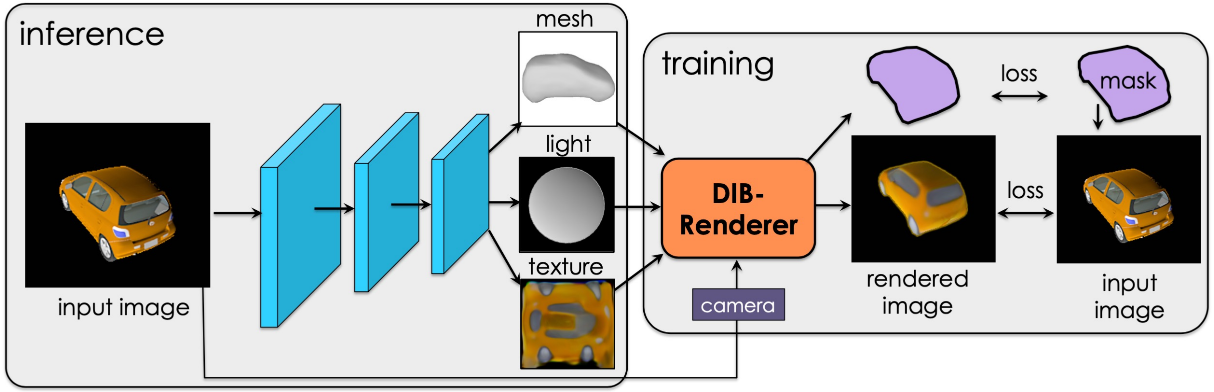
- Directly training a mesh model with captured images
  - **DIB-R++**, **MobileNeRF**, …
- Training NeRF and build a mesh from it (via marching cube)
  - Neural Duplex, **Re-Rend**, …

Differentiable Renderer (DIB-R)

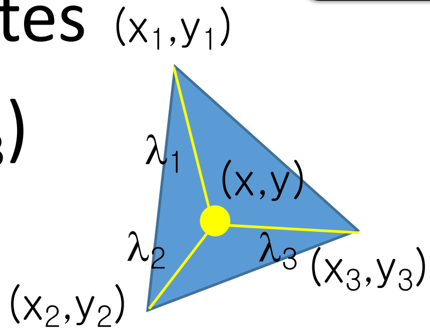
- CNN is trained to give mesh, texture (albedo) and light parameters (spherical harmonics approximation coefficients of environment map)



How to Learn Vertex Coordinates via SGD?



- Exploit the relationship btw pixel color and vertex coordinates
- Given (x_1,y_1), (x_2,y_2), (x_3,y_3)



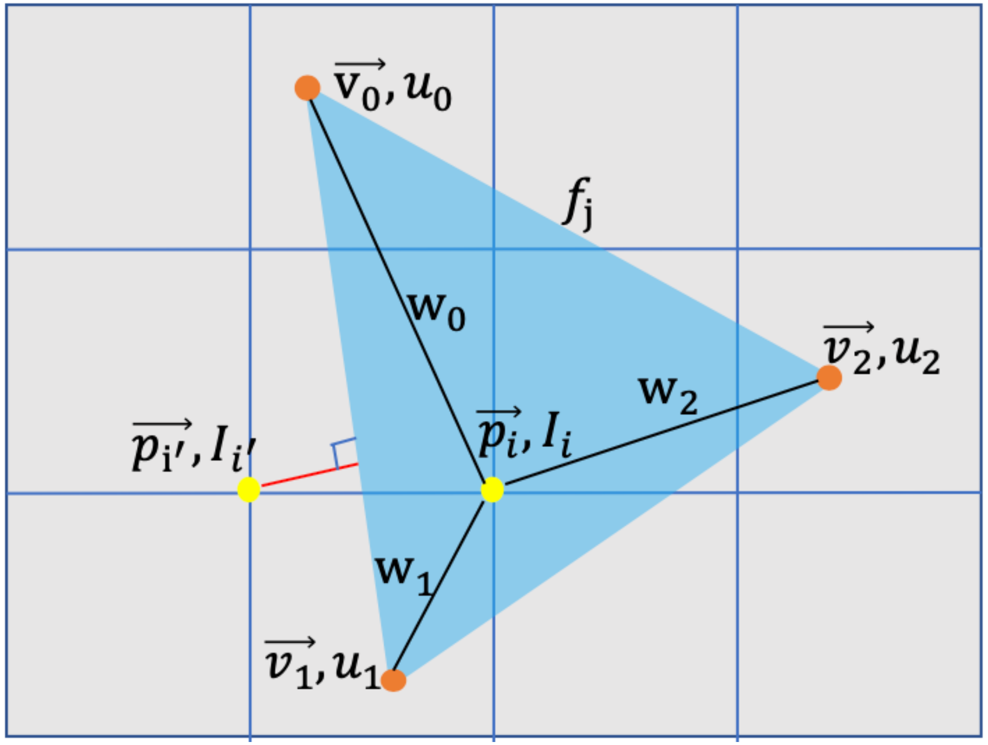
$$x = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3$$
$$y = \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3$$

$$\begin{pmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{pmatrix} = \frac{1}{2A}\begin{pmatrix} x_2y_3 - x_3y_2 & y_2 - y_3 & x_3 - x_2\\ x_3y_1 - x_1y_3 & y_3 - y_1 & x_1 - x_3\\ x_1y_2 - x_2y_1 & y_1 - y_2 & x_2 - x_1\end{pmatrix}\begin{pmatrix}1\\ x\\ y\end{pmatrix}$$

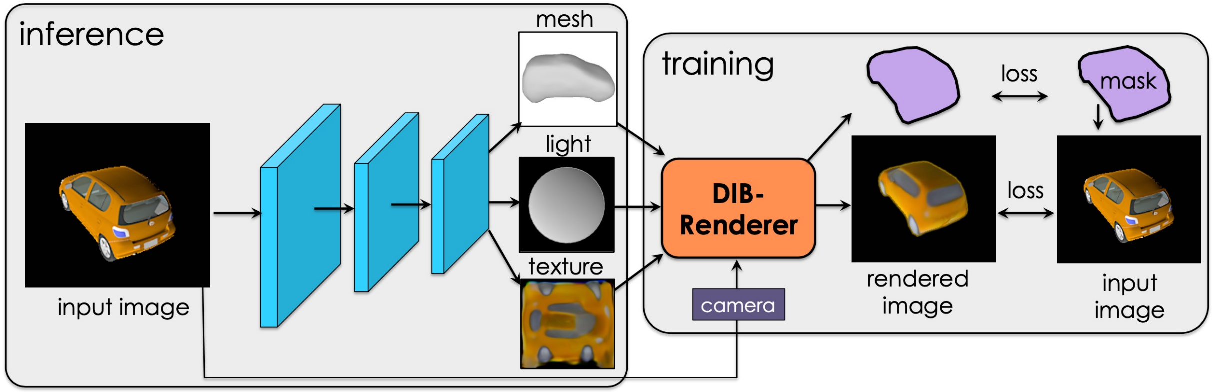
- Color I_i

Same equation

$$I_i = w_0 u_0 + w_1 u_1 + w_2 u_2$$
$$w_k = \Omega_k(\vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{p}_i), \quad k = 0, 1, 2.$$



Differentiable Rasterization



- Baycentric interpolation of vertex attributes, e.g., color

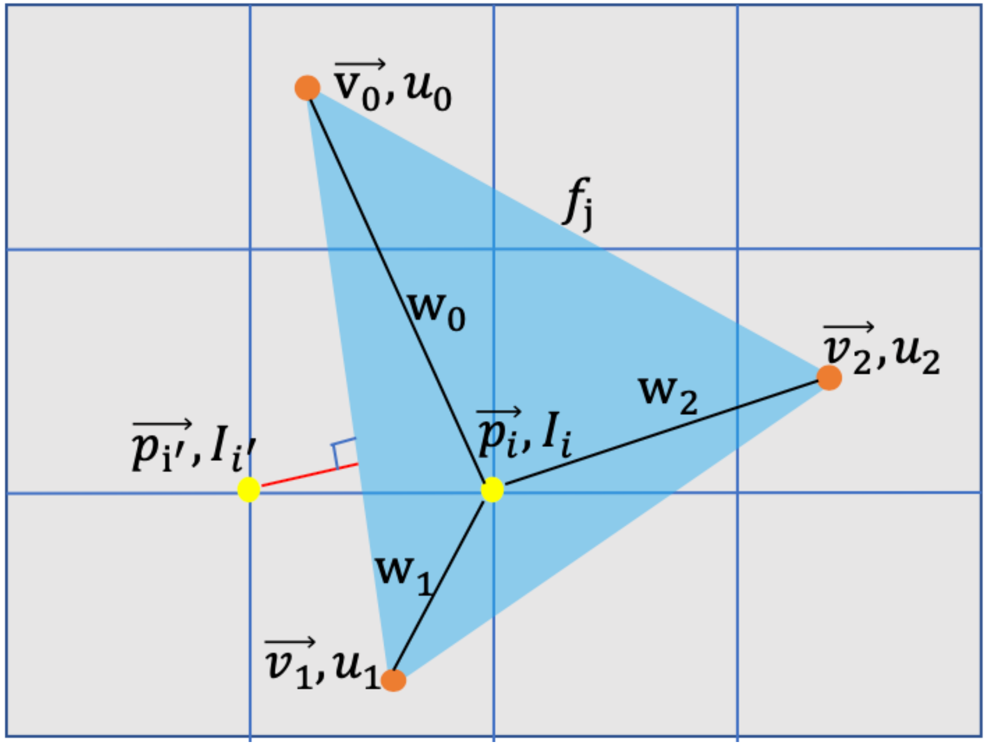
$$I_i = w_0 u_0 + w_1 u_1 + w_2 u_2$$

$$w_k = \Omega_k(\vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{p}_i), \quad k = 0, 1, 2.$$

- **Vertex coordinates, v's and attributes (color), u's can be learned via SGD**

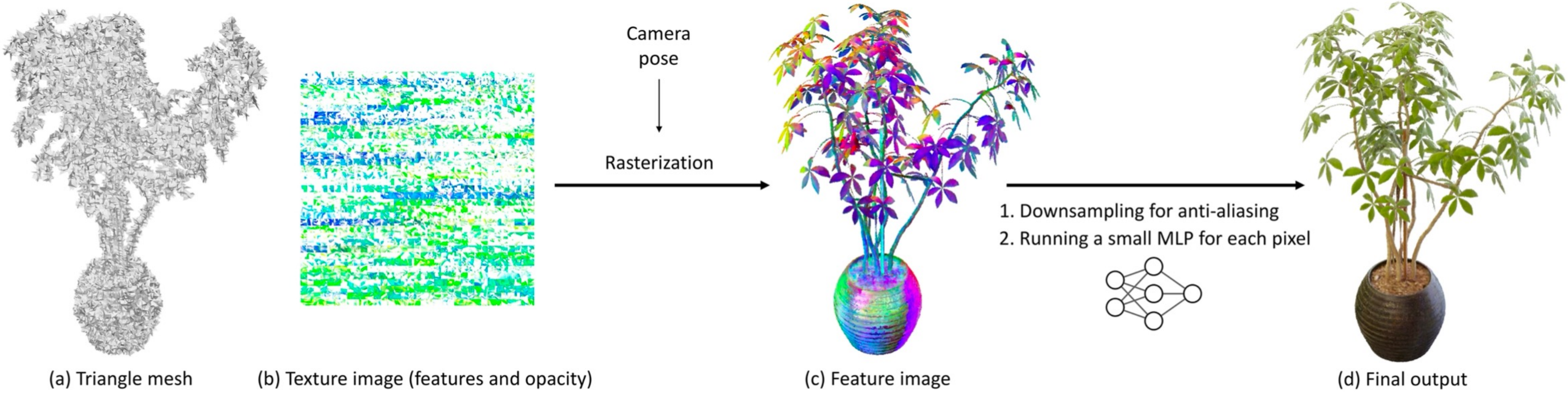
$$\frac{\partial I_i}{\partial u_k} = w_k, \quad \frac{\partial I_i}{\partial \vec{v}_k} = \sum_{m=0}^{2} \frac{\partial I_i}{\partial w_m} \frac{\partial \Omega_m}{\partial \vec{v}_k},$$

$$\frac{\partial L}{\partial u_k} = \sum_{i=1}^{N} \frac{\partial L}{\partial I_i} \frac{\partial I_i}{\partial u_k}, \quad \frac{\partial L}{\partial \vec{v}_k} = \sum_{i=1}^{N} \frac{\partial L}{\partial I_i} \frac{\partial I_i}{\partial \vec{v}_k},$$



MobileNeRF Directly Learns Mesh and Texture

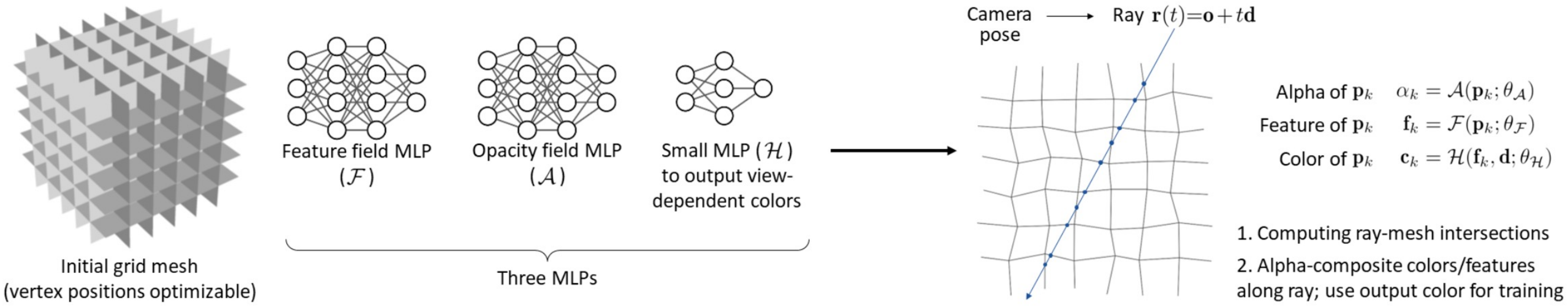
- Exploit the existing graphics rendering pipeline
  - Mesh + texture + viewpoint → rasterization (feature) → shader (MLP) for color
- ~50 frames/sec on iPhone Xs



[MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures]

Training MobileNeRF

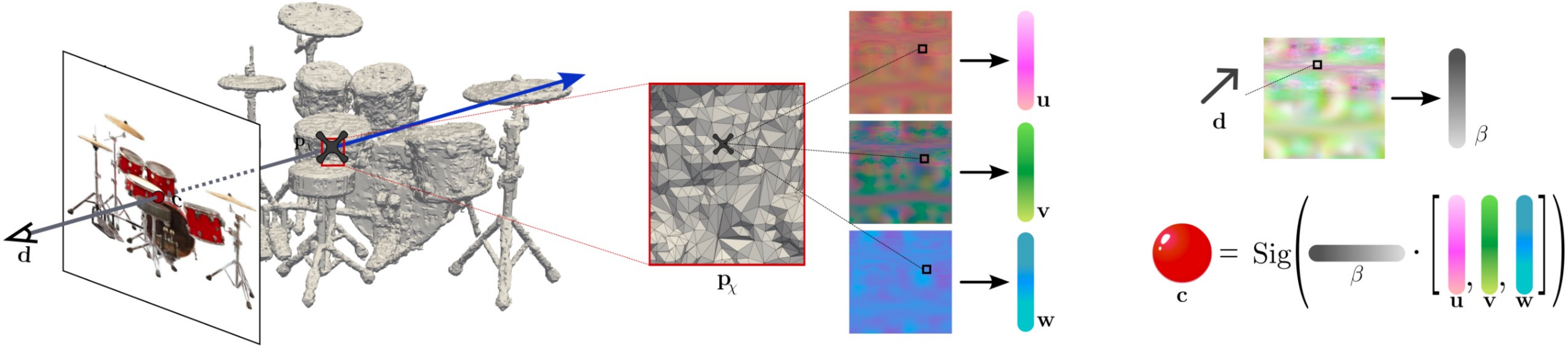
- Start with an initial grid mesh, optimize for vertex positions as well as feature/opacity/color MLPs
- Given a ray
  - Interpolation of texture feature on ray-mesh intersections
  - Interpolated feature → color on sample point → pixel color via alpha composition



MobileNeRF Runs 50 Frames/Sec on iPhoneXS

Given a Trained NeRF Model, Build a Mesh and Exploit Graphics Pipeline: Re-ReND Case

- Mesh (w/ texture) building via Marching Cube method
- At ray-triangle intersection, cheap color computation with texture (dot products of UVW and β)
- Up to ~1000 frames per second on NVIDIA GPU 3090



a) Mesh Rasterization b) Texture Map Look-Up c) Color Rasterization

Agenda

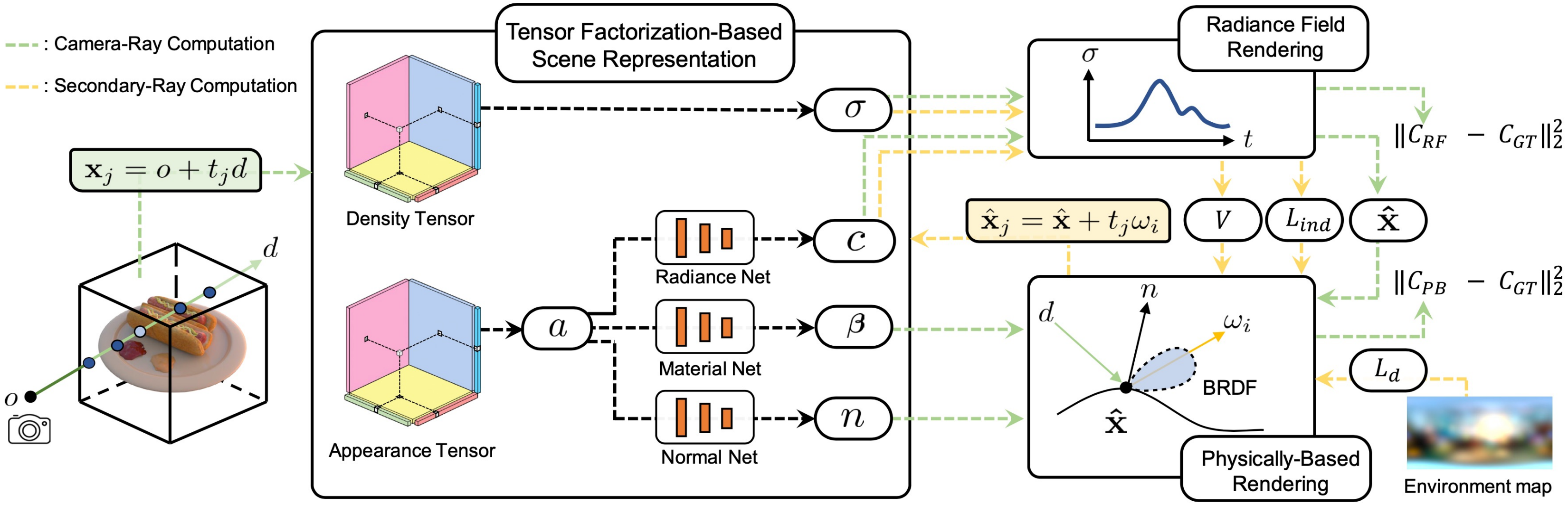
- Introduction to neural rendering
- Problem: high computation cost
- Efficient method #1: voxel grid
- Efficient method #2: mesh
- **Summary and prospect**

Summary and Prospect

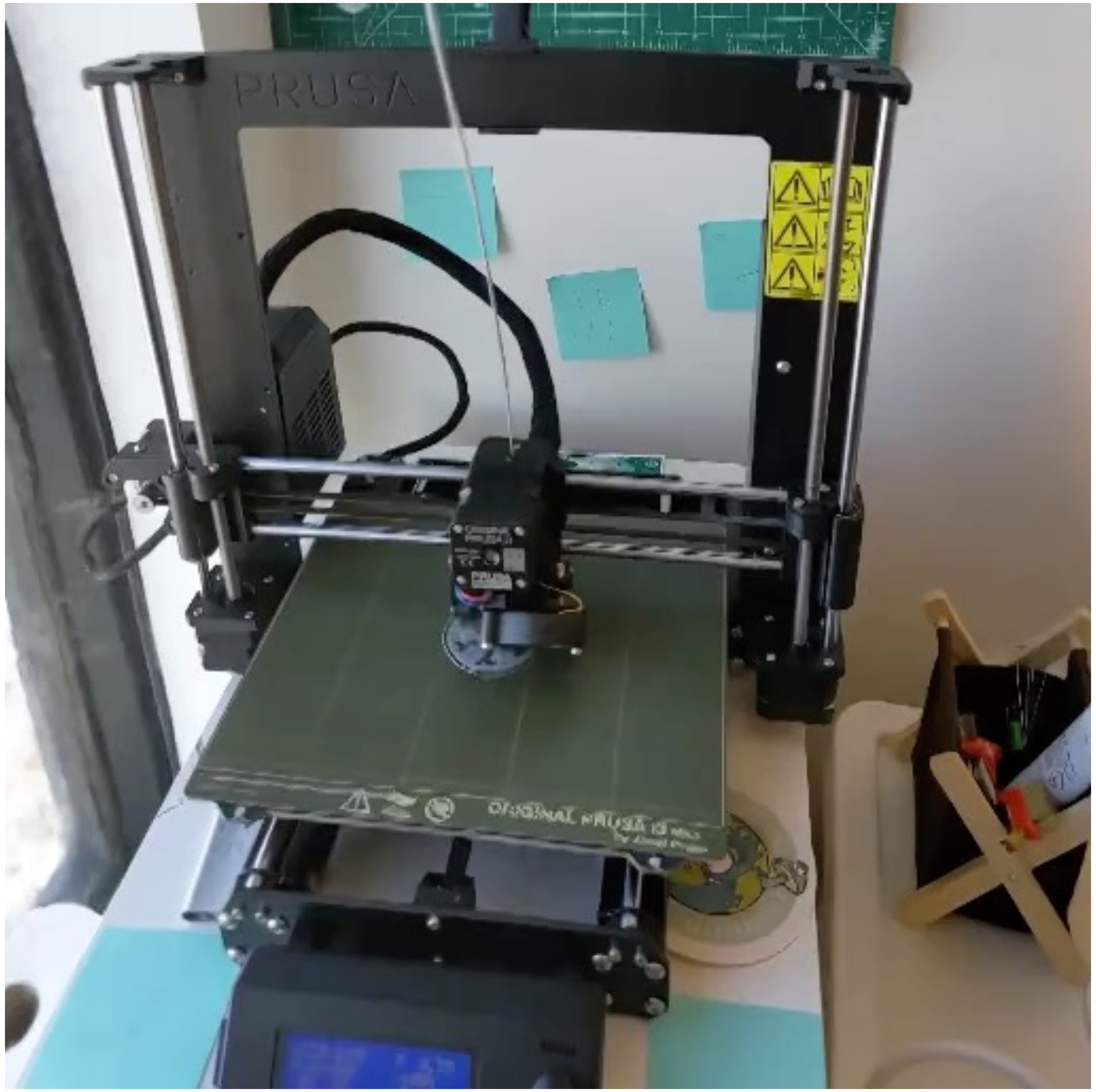
- >1000 neural rendering papers (arxiv) since the NeRF paper in 2020
- Drastic (>1000X) improvements of compute efficiency in only three years
    - Voxel grid to exploit pre-computation
    - Mesh to exploit existing graphics pipeline
    - Light field network to exploit CNN acceleration
- Prospect: Efficient models for real-time **immersive** photo-realism
    - Relighting, dynamic scene, large scale and large language model (LLM)

Towards Relighting: Inverse Rendering on Voxel Grid (TensoIR)

- Can render images on new lighting
  - By separately learning material and shape
  - w.r.t. radiance (color) in NeRF
- Very slow rendering due to high compute cost in rendering eqn
  - Visibility
  - Indirect illumination



Dynamic Scene (aka. Free Viewpoint Video)

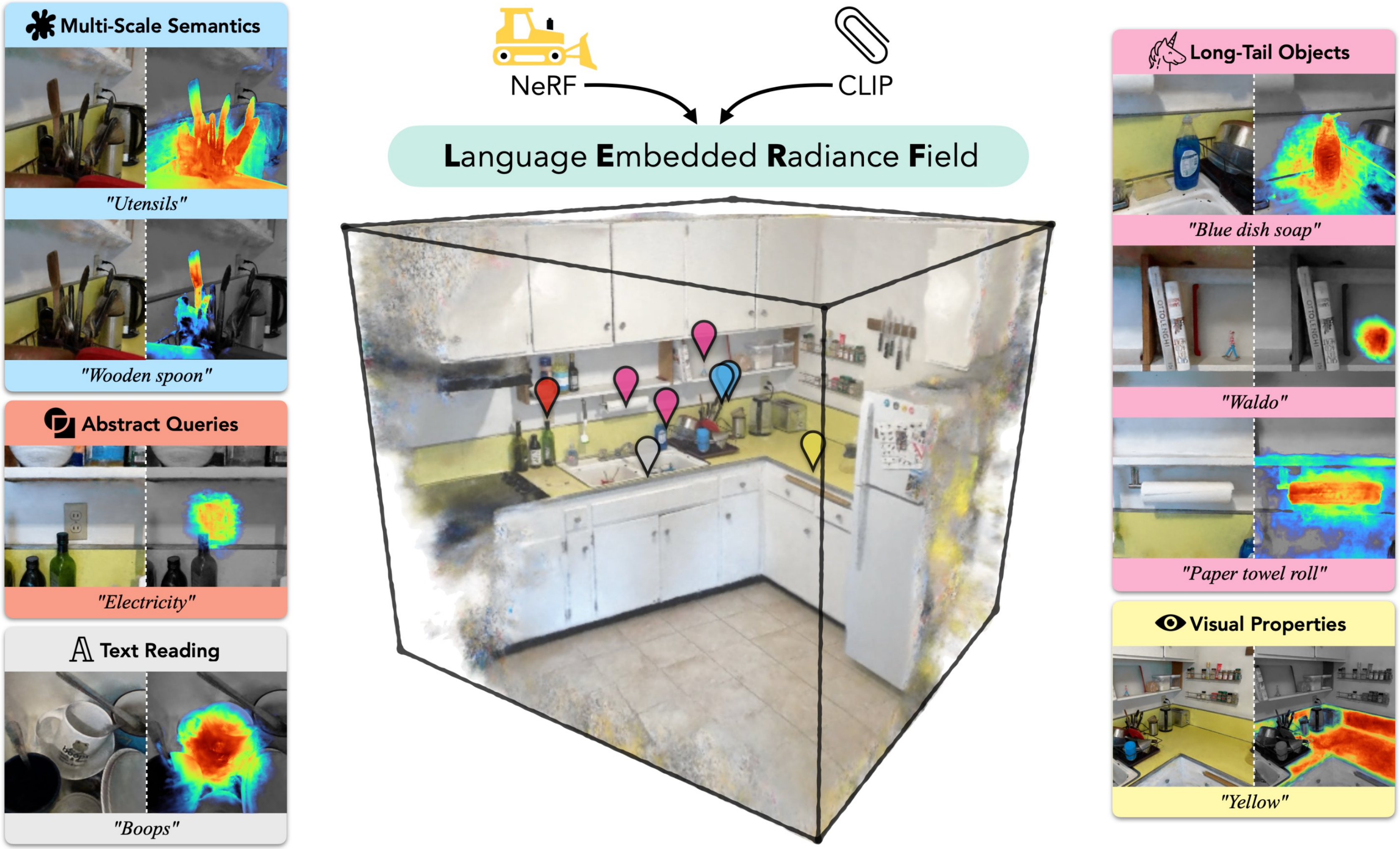


Large-Scale NeRF: Block NeRF

- Consists of small block-level NeRF models
- e.g., city center in San Francisco

[Block-NeRF: Scalable Large Scene Neural View Synthesis]

Multi-Modal 3D Model
e.g., Text Input based Object Localization



Summary and Prospect

- >1000 neural rendering papers since the NeRF paper in 2020
- Drastic (>1000X) improvements of compute efficiency in only three years
  - Voxel grid to exploit pre-computation
  - Mesh to exploit existing graphics pipeline
  - Light field network to exploit CNN acceleration
- Prospect: Efficient models for real-time **immersive** photo-realism
  - Relighting, dynamic scene, large scale and large language model (LLM)
  - **>1000X more efficient models and implementations are needed**
    - Compute cost, model size, training speed, …

Thank You!

TINY ML ASIA
Seoul, South Korea
November 16, 2023

Copyright Notice

www.tinyml.org