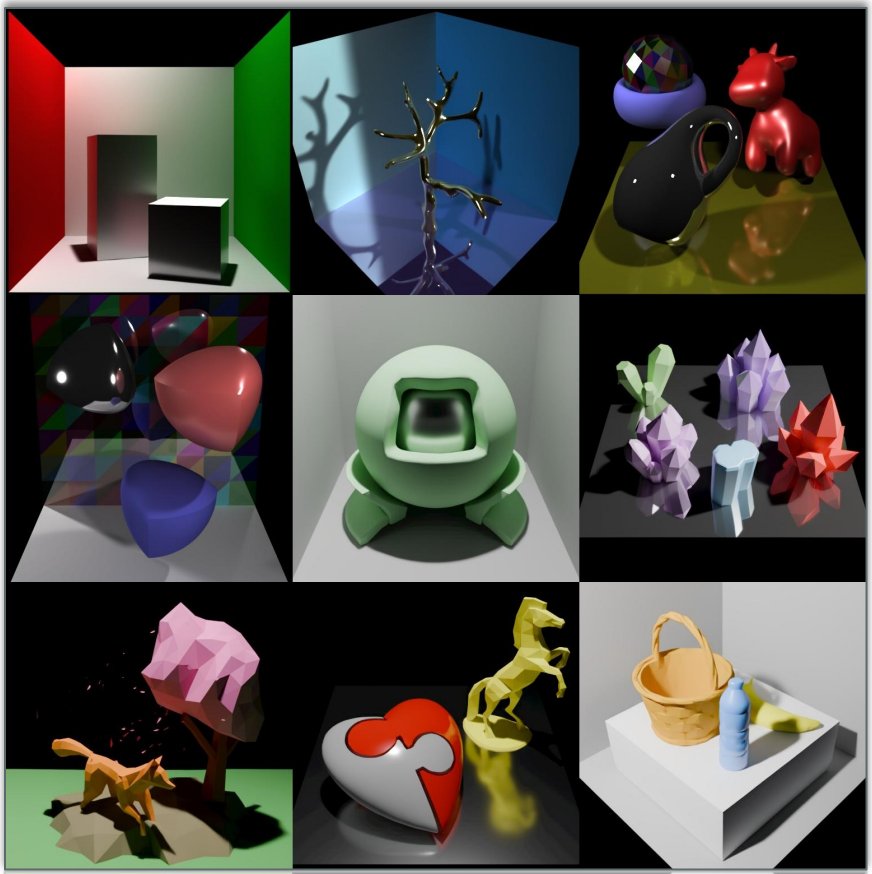
RenderFormer

Paper Presentation

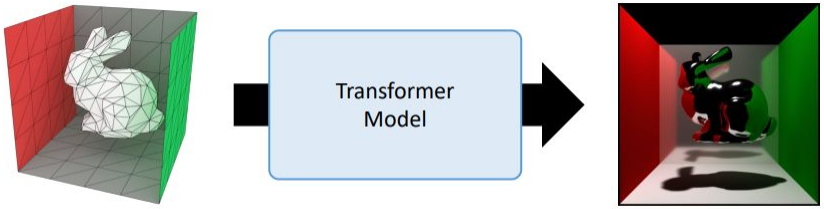
17th Nov, 2025

Team 4
Kyaw Ye Thu, Janghyun



https://microsoft.github.io/renderformer/

RenderFormer: A Fully Transformer Rendering Pipeline



3D Model

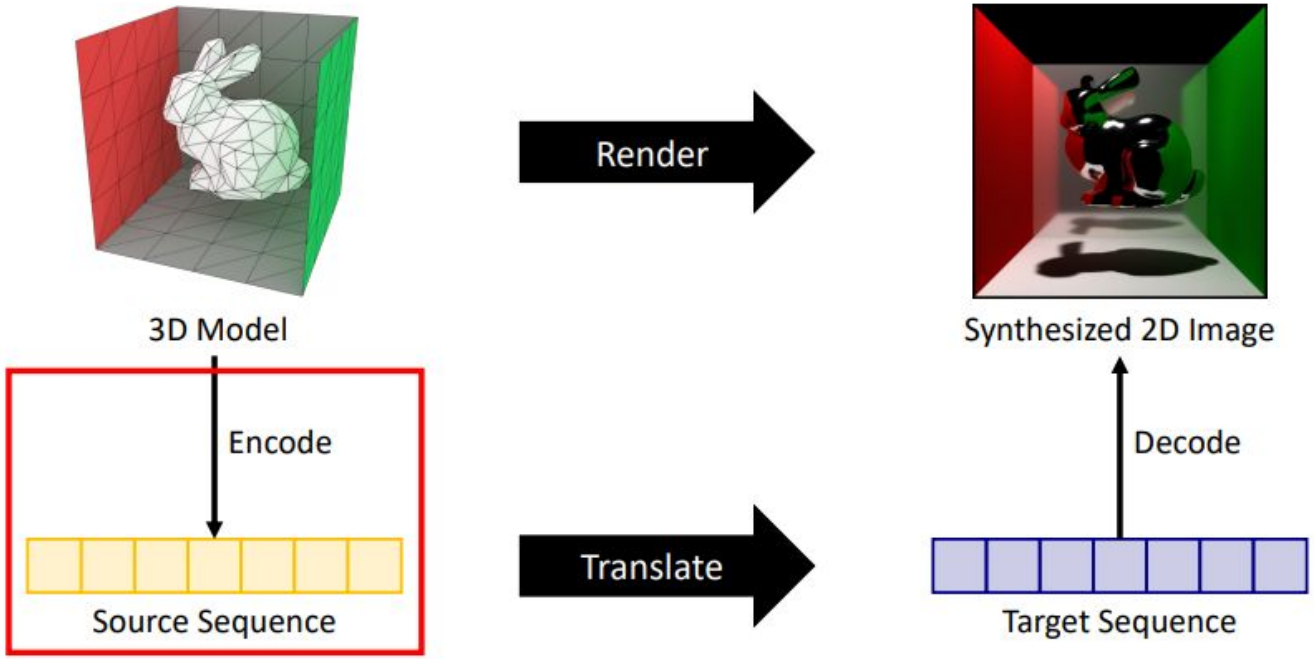
Synthesized 2D Image

Outline

- Detailed Architecture of RenderFormer
- Training
- Results

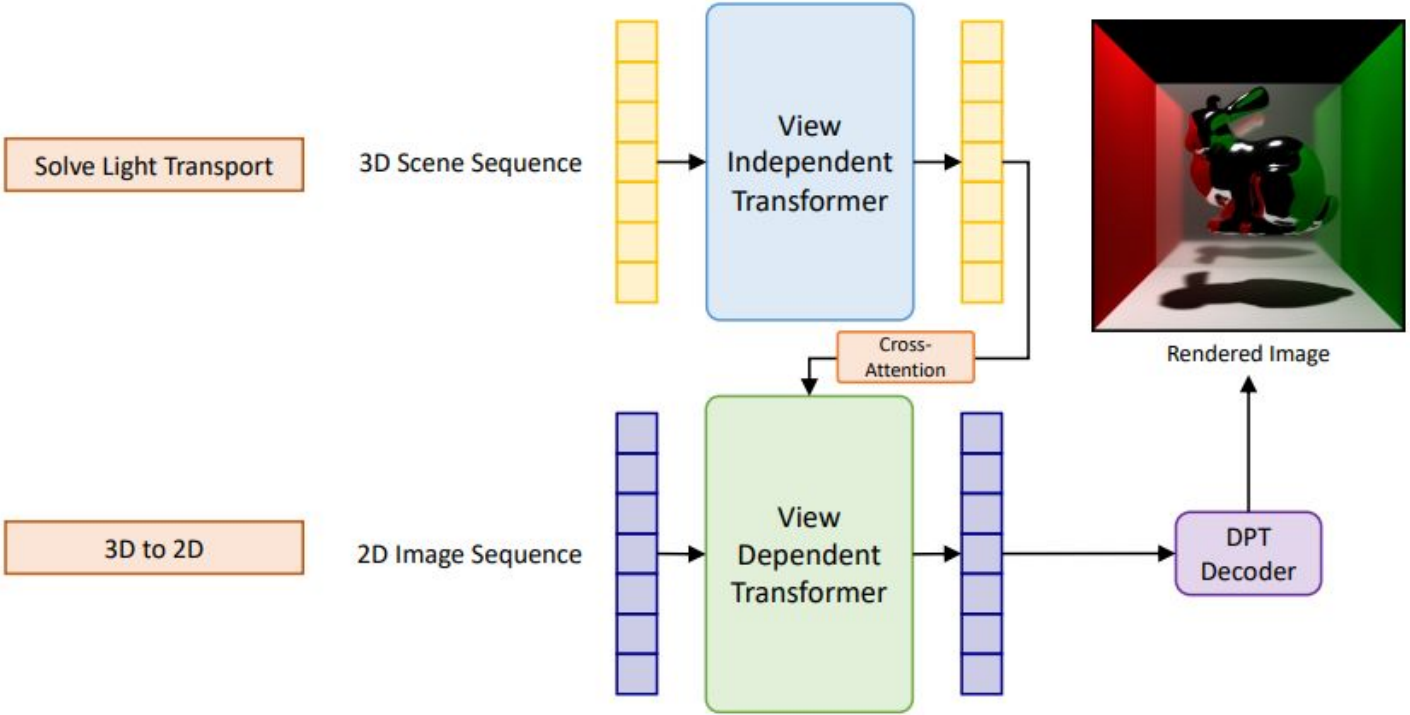
Renderformer Architecture

Idea: 3D Rendering = Translating 3D to 2D



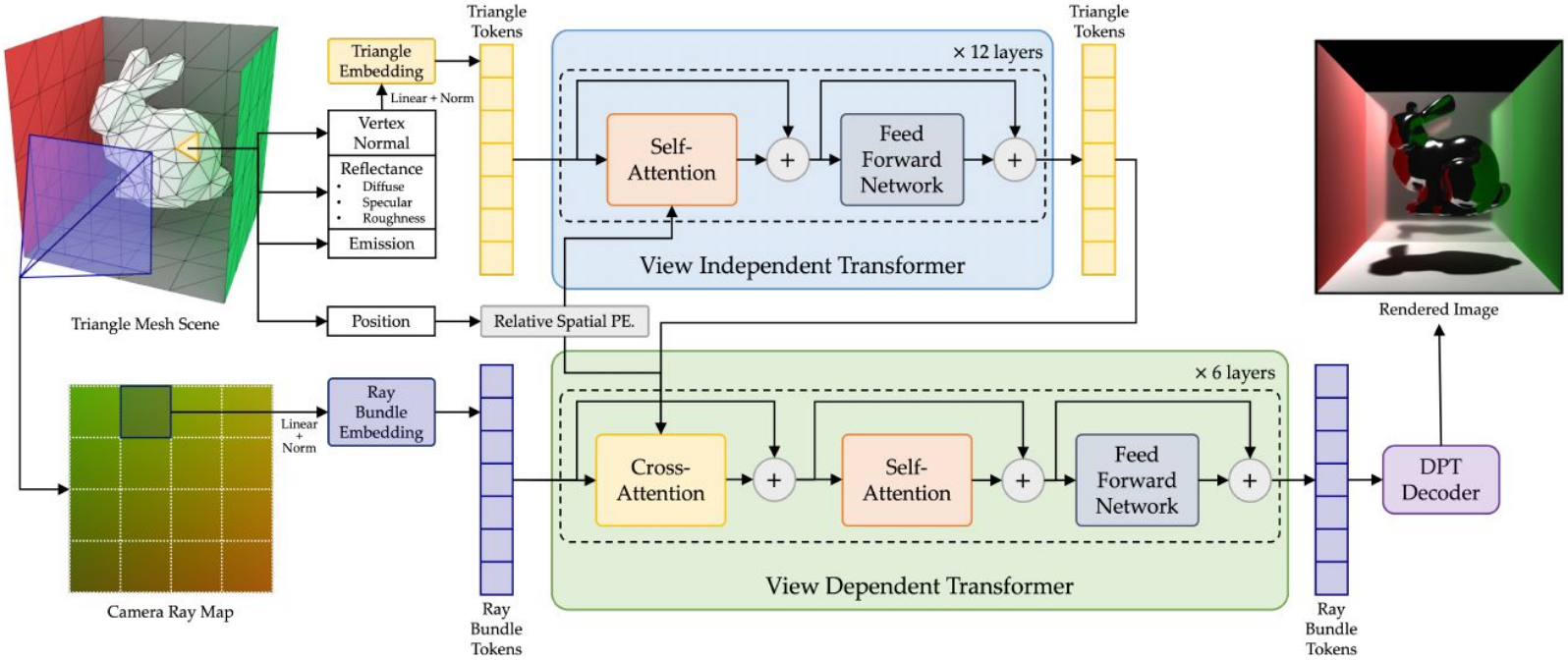
Slide Source: https://microsoft.github.io/renderformer/

Rendering with Transformer Architecture



Slide Source: https://microsoft.github.io/renderformer/

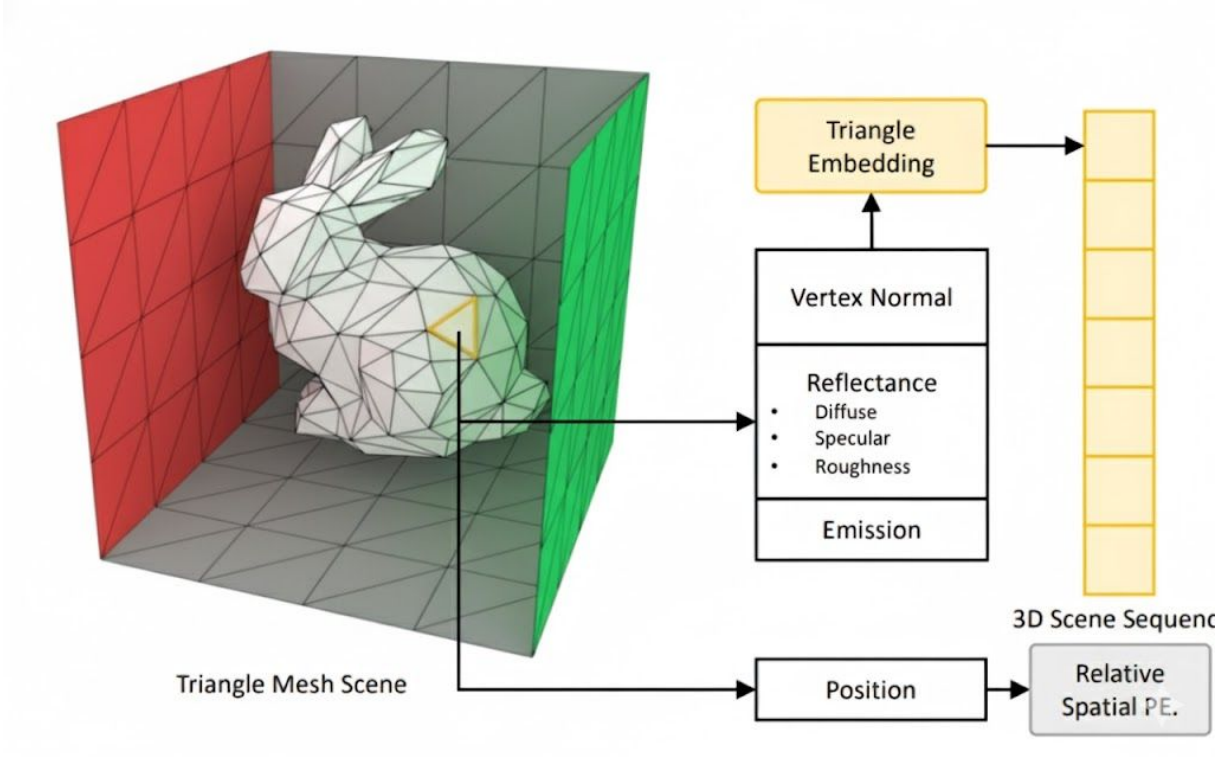
Full Pipeline



Slide Source: https://microsoft.github.io/renderformer/

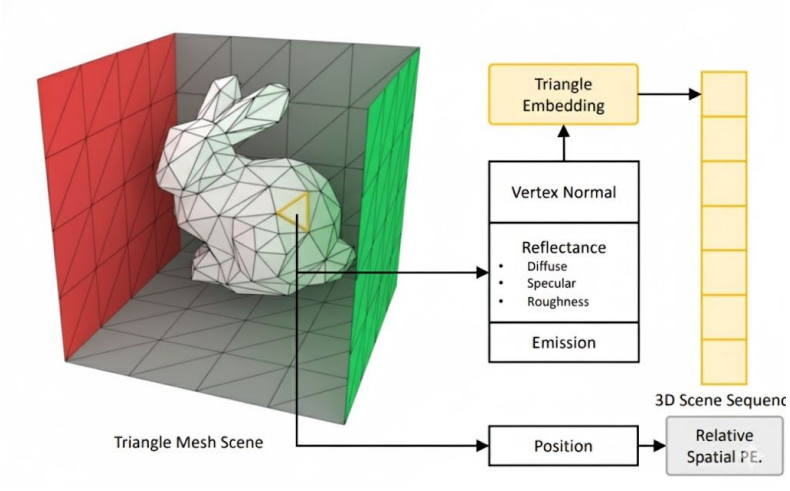
View-Independent Stage

3D Sequence: Tokenize Mesh Scene by Triangles



Slide Source: https://microsoft.github.io/renderformer/

Triangle Embedding



For each triangle,

9x1

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \\ x_3 \\ y_3 \\ z_3 \end{bmatrix}$$

1 normal per vertex
3 normals per triangle

NeRF encoding -> Linear -> RMS norm

768x1

$$\begin{bmatrix} vn_1 \\ vn_2 \\ \dots \\ \dots \\ \dots \\ vn_{768} \end{bmatrix}$$

10x1

$$\begin{bmatrix} diff_r \\ diff_g \\ diff_b \\ spec_r \\ spec_g \\ spec_b \\ emis_r \\ emis_g \\ emis_b \\ rough \end{bmatrix}$$

Linear -> RMS norm

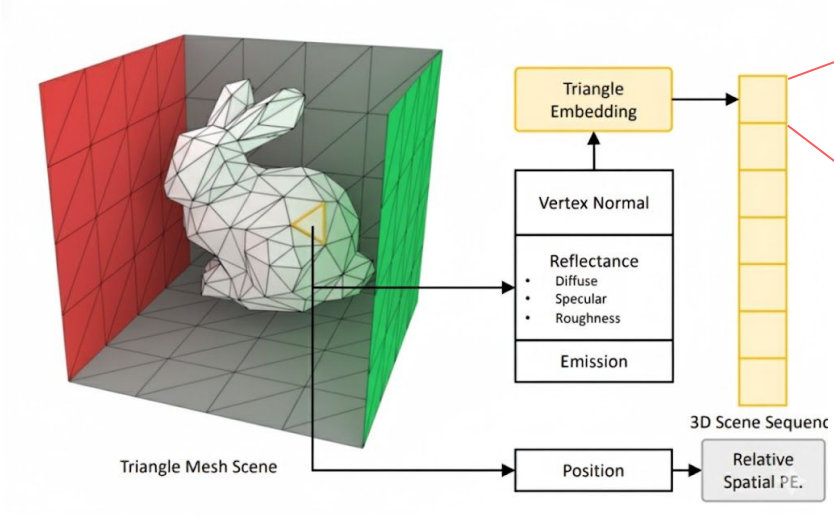
768x1

$$\begin{bmatrix} tex_1 \\ tex_2 \\ \dots \\ \dots \\ \dots \\ tex_{768} \end{bmatrix}$$

Reflectance
Emission

Triangle Embedding

For each triangle,

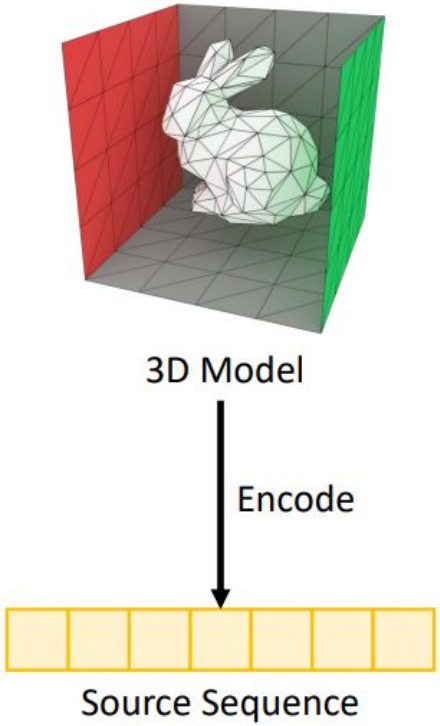


768x1

$$\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ \dots \\ \dots \\ x_{768} \end{bmatrix} = \begin{bmatrix} tri_1 \\ tri_2 \\ \dots \\ \dots \\ \dots \\ tri_{768} \end{bmatrix} + \begin{bmatrix} vn_1 \\ vn_2 \\ \dots \\ \dots \\ \dots \\ vn_{768} \end{bmatrix} + \begin{bmatrix} tex_1 \\ tex_2 \\ \dots \\ \dots \\ \dots \\ tex_{768} \end{bmatrix}$$

tri = triangle token (learnable parameter)
vn = vertex normal
tex = texture
x = triangle embedding

Relative Positional Encoding



The position of the triangle relative to others in the virtual world matters!

✅Relative Positional Encoding
❌Absolute Positional Encoding

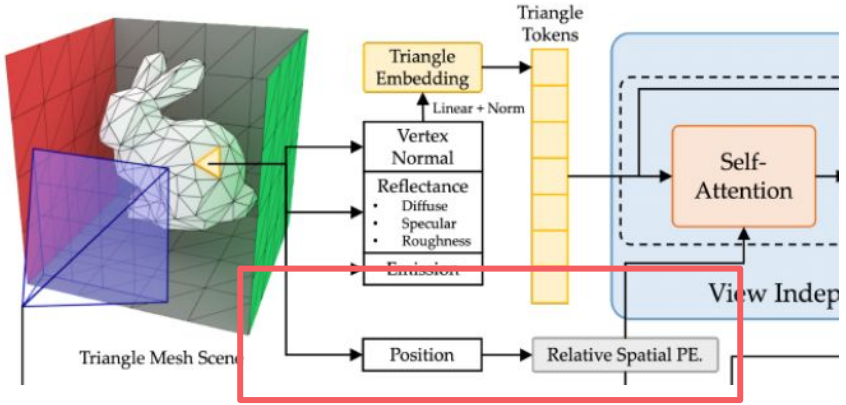
The index position of the token (i.e., triangle) in the sequence does NOT matter.
(Swapping two triangles in the sequence should produce the same result.)

RoPE (Rotary Positional Encoding)

Adding the positional information directly to token embedding pollute the semantic information with the positional information

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ \cdots \\ \cdots \\ x_{768} \end{bmatrix} = \begin{bmatrix} tri_1 \\ tri_2 \\ \cdots \\ \cdots \\ \cdots \\ tri_{768} \end{bmatrix} + \begin{bmatrix} vn_1 \\ vn_2 \\ \cdots \\ \cdots \\ \cdots \\ vn_{768} \end{bmatrix} + \begin{bmatrix} tex_1 \\ tex_2 \\ \cdots \\ \cdots \\ \cdots \\ tex_{768} \end{bmatrix} + \begin{bmatrix} pos_1 \\ pos_2 \\ \cdots \\ \cdots \\ \cdots \\ pos_{768} \end{bmatrix}$$

To preserve the semantic content while encoding relative distances,



During attention computation, the triangle's 3D spatial info is used to rotate the query and key vectors

RoPE (Rotary Positional Encoding)

To preserve the semantic content while encoding relative distances,

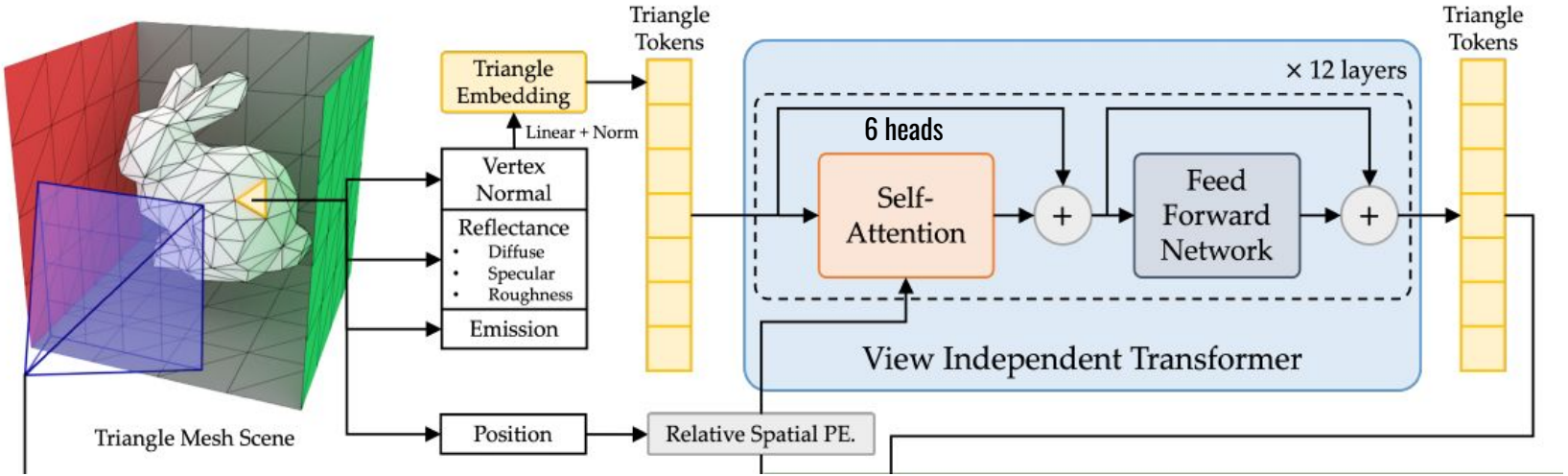
Let **q** be our input vector at position *p, the rotated q is*

$$R(\mathbf{q}, p) = \begin{pmatrix} \mathbf{M_1} & & & \\ & \mathbf{M_2} & & \\ & & \ddots & \\ & & & \mathbf{M_{d/2}} \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_d \end{pmatrix}$$

where $\mathbf{M_i} = \begin{bmatrix} \cos(\omega_i p) & \sin(\omega_i p) \\ -\sin(\omega_i p) & \cos(\omega_i p) \end{bmatrix}$

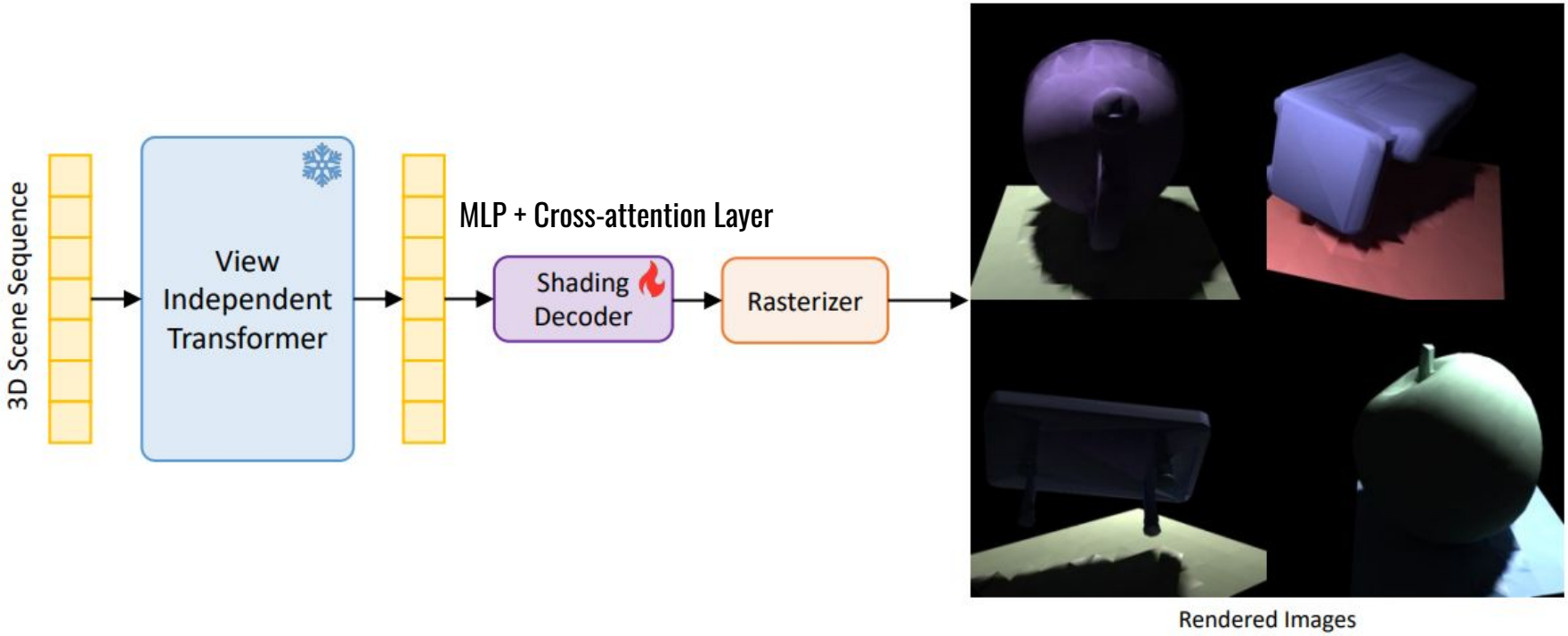
```
q_rotated = R(q, p), k_rotated = R(k, p)
```

The Entire View-Independent Stage



Follows original transformer architecture with full bidirectional self-attention.

View-Independent Stage Resolves Diffuse Light Transport

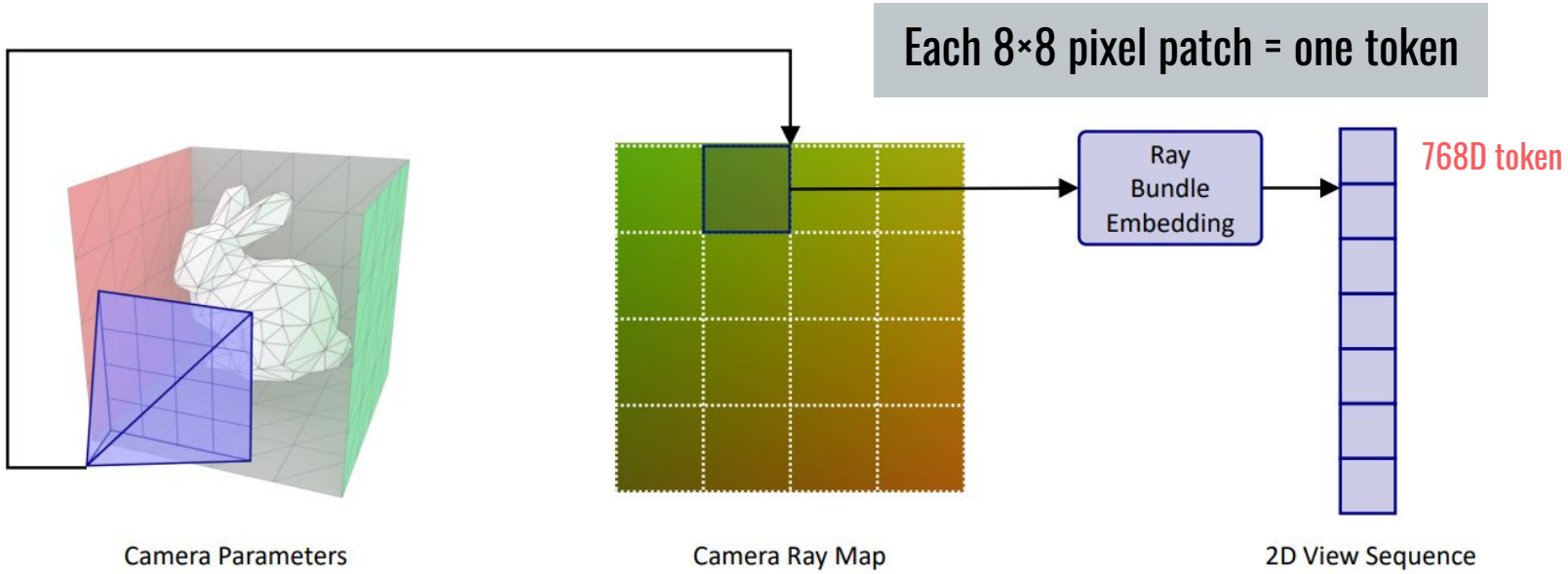


Slide Source: https://microsoft.github.io/renderformer/

View-Dependent Stage

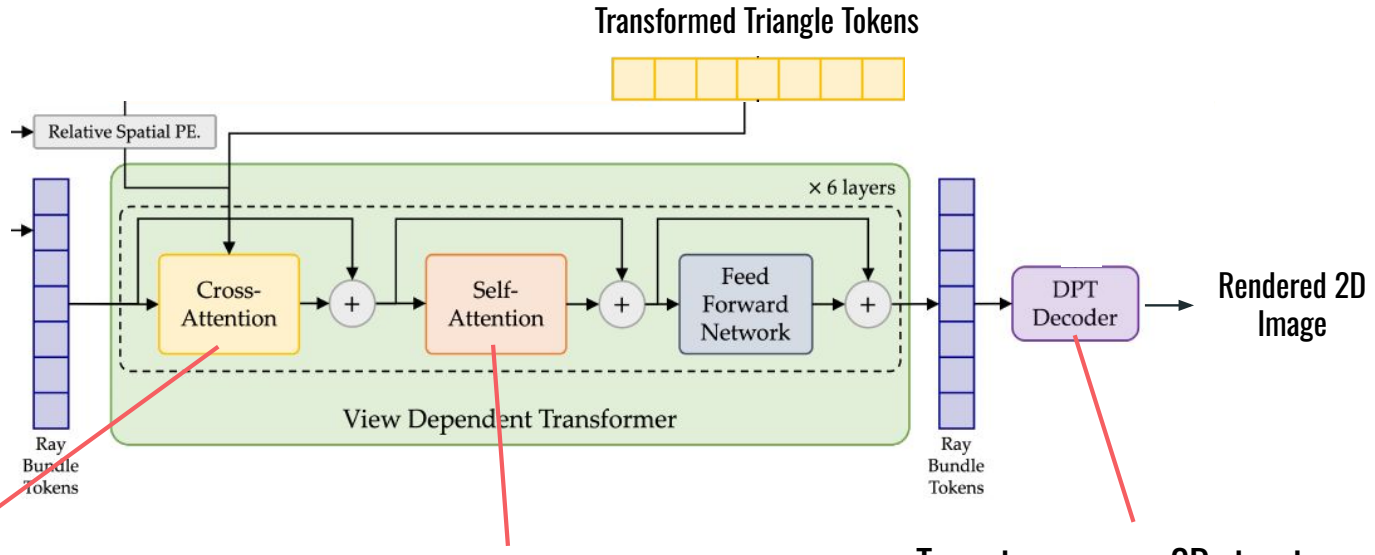
2D Sequence: Tokenize View using Ray Bundles

Goal: To transform the triangle tokens transformed by the previous view-independent stage to radiance pixel values corresponding to a given virtual camera.



Slide Source: https://microsoft.github.io/renderformer/

Dissection View-Dependent Stage

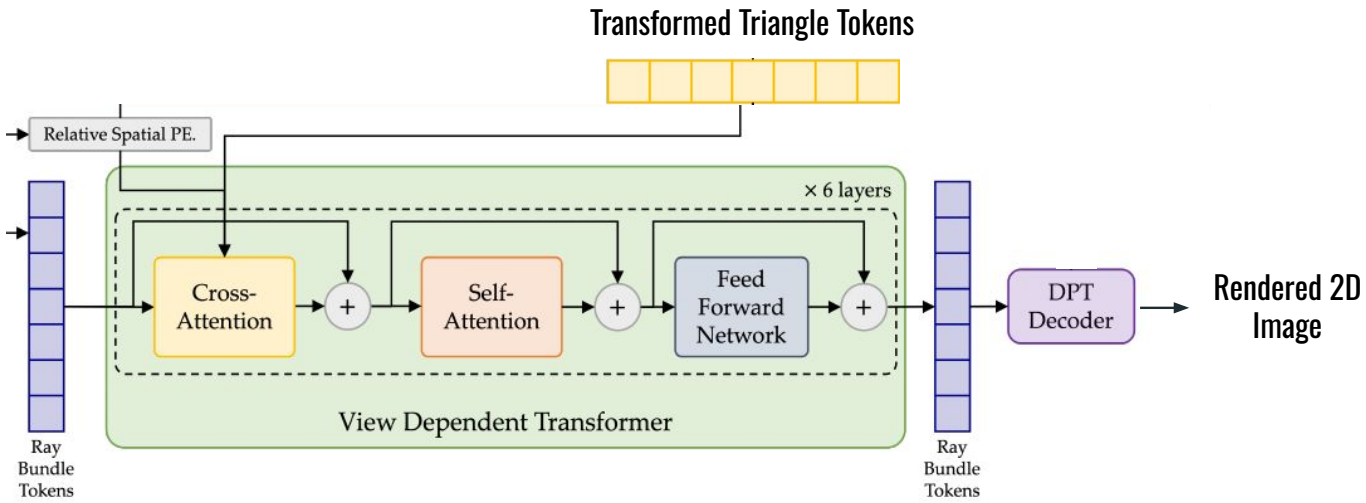


To find the triangles related to the rays in the ray-bundle

To share information between view rays

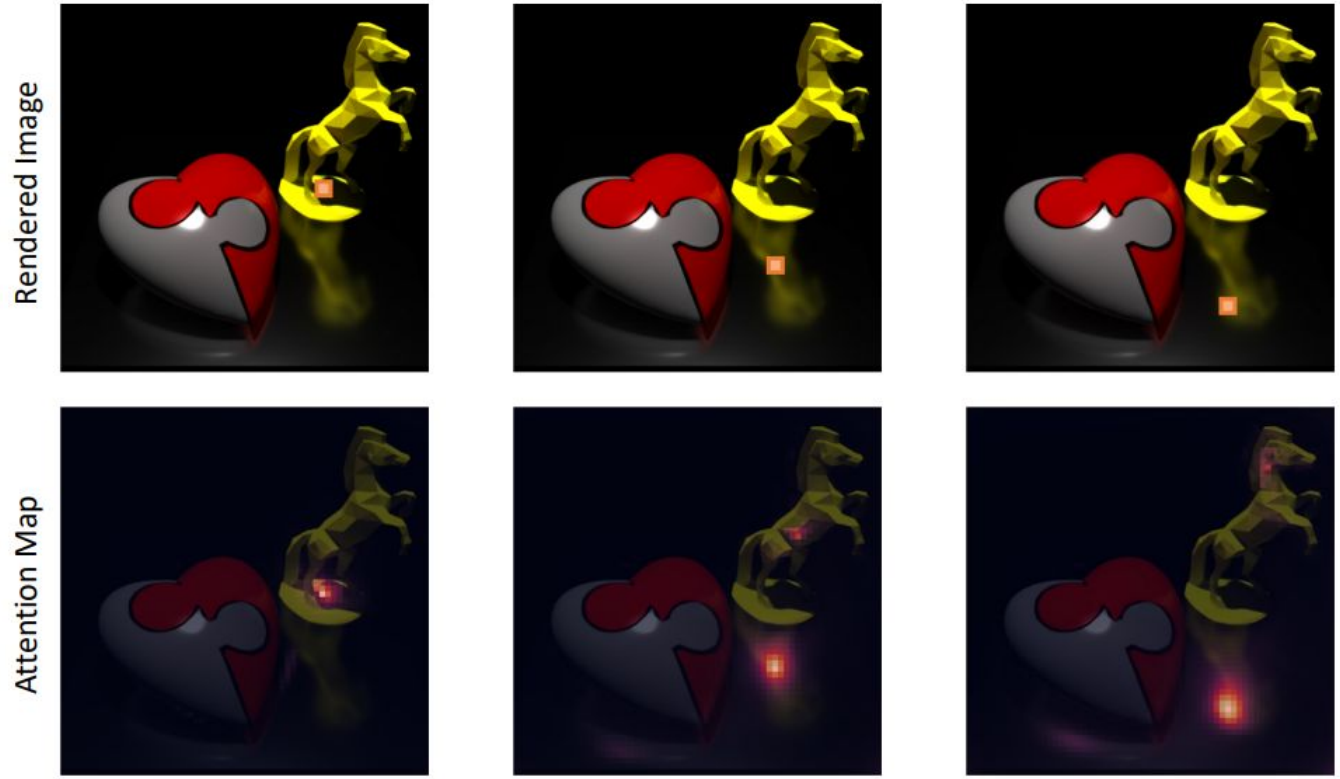
To restore proper 2D structure (compensate tokenizing pixels into patches)

Ablation Study on View Dependent Stage



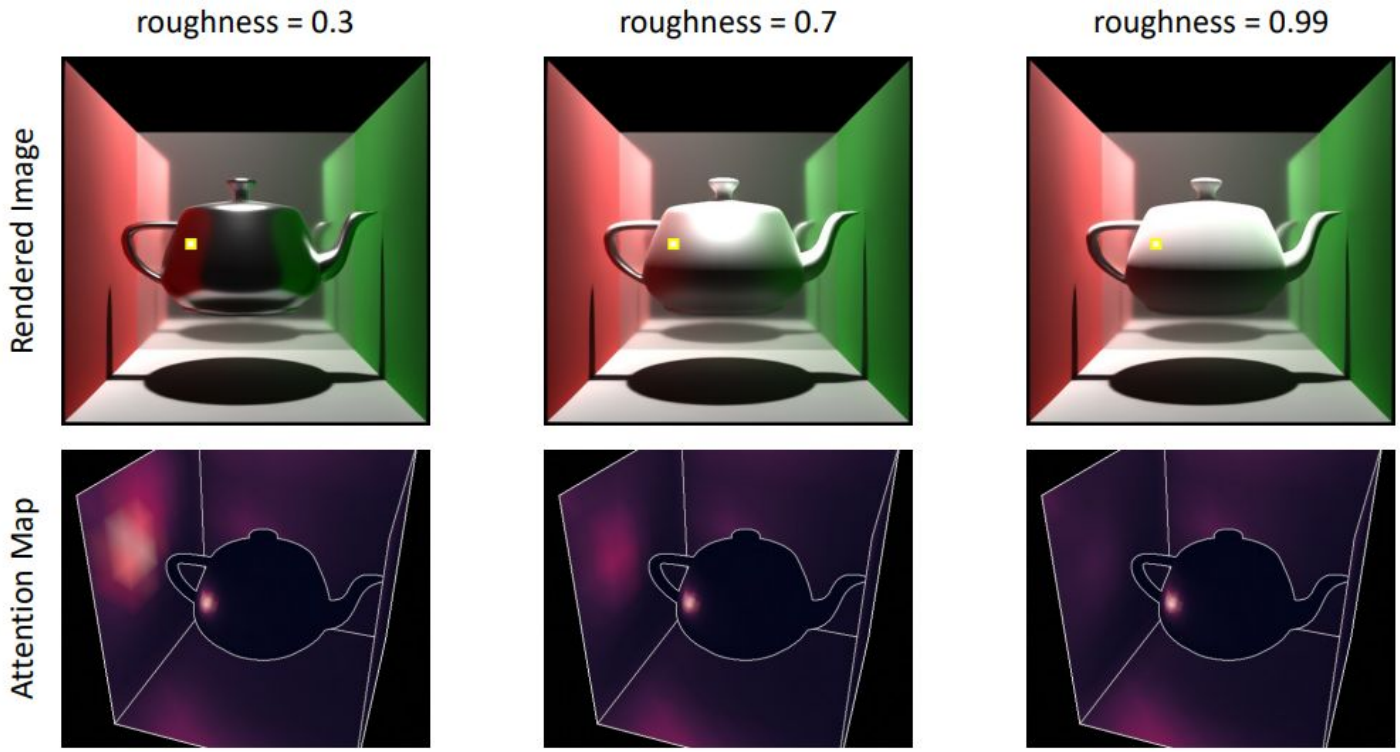
Variant	PSNR ↑	SSIM ↑	LPIPS ↓	FLIP ↓
full view-dependent stage	**29.77**	**0.9526**	**0.05514**	**0.1751**
w/o dpt	29.75	0.9476	0.05519	0.1806
w/o self-attention	29.70	0.9503	0.05703	0.1766
w/o dpt & w/o self-attention	29.15	0.9396	0.06407	0.1836

View-Dependent Stage Resolves Triangle-Ray Intersection



Slide Source: https://microsoft.github.io/renderformer/

View-Dependent Stage Resolves Triangle-Ray Intersection

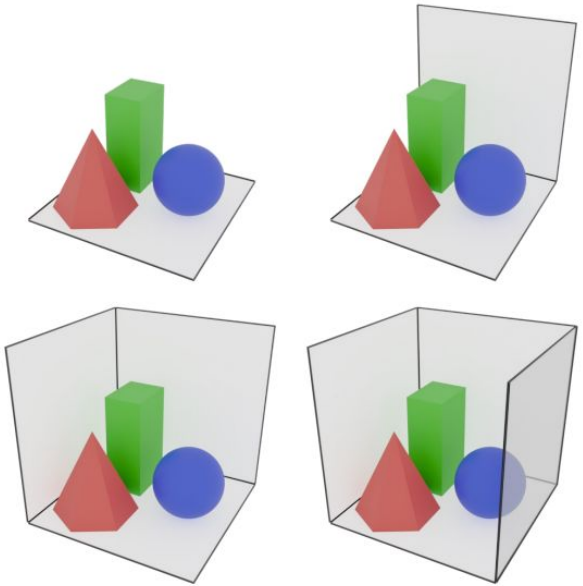


Slide Source: https://microsoft.github.io/renderformer/

Training

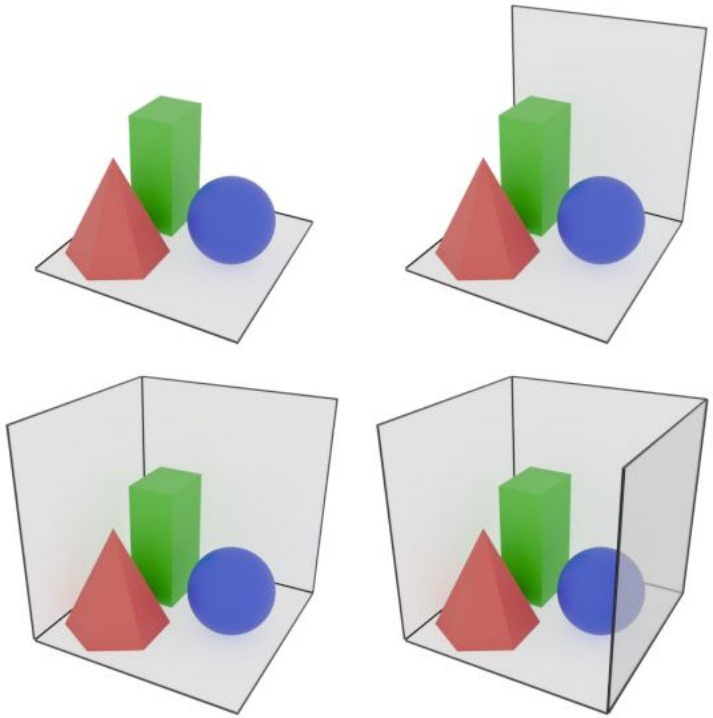
Dataset

1~3 randomly selected objects
one of 4 random template scenes



Slide Source: https://microsoft.github.io/renderformer/

Training Data – Template-based Scene Generation

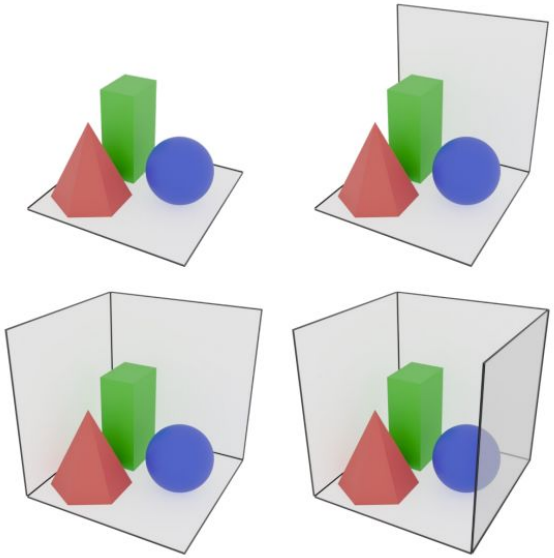


4,096 Max #Triangles	**0.01-1.0** Roughness Range
8 Max #Lights	**2.1-2.7** Light Distance
30-60 Camera FOV	**1.5-2.0** Camera Distance

Slide Source: https://microsoft.github.io/renderformer/

Facts about Training

2M synthetic scenes



8M HDR training images
(4 different viewpoint per scene)

8 NVIDIA A100 GPUs with 40GB of memory

Flash-Attention 2 and Liger Kernel for speed-up

256x256 resolution
Max mesh size of 1,536 ▲

5 days (8M HDR images)

512x512 resolution
Max mesh size of 4,096 ▲

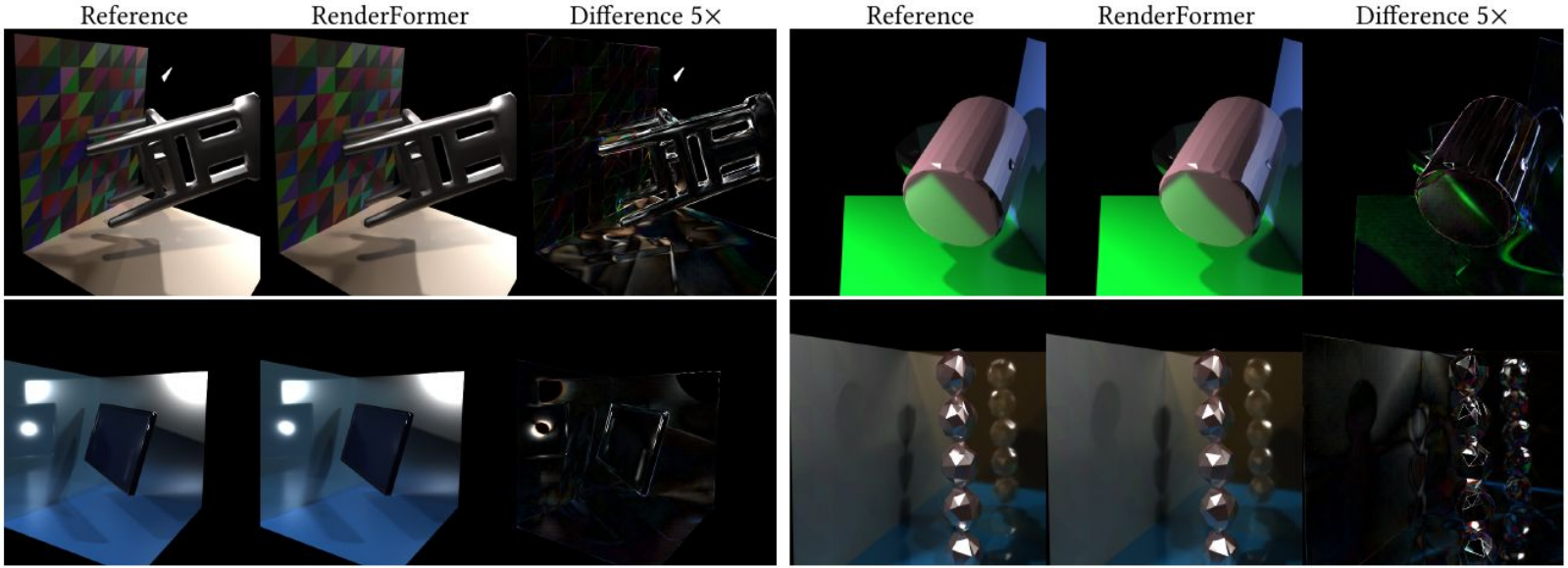
3 days (8M HDR images)

Loss Function

Apply log transform to the images first

$$loss_{L1} + 0.05 \cdot loss_{LPIPS}$$

(Learned Perceptual Image Patch Similarity)
Minimize perceptual differences



Results

Visual Results

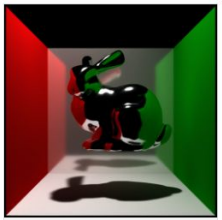
https://microsoft.github.io/renderformer/

Rendering Gallery

Examples of scenes rendered with RenderFormer demonstrating various lighting conditions, materials, and geometric complexity, without any per-scene training or fine-tuning. Check out the **reference images** for more details.

Cornell Box

Cornell University Program of Computer Graphics

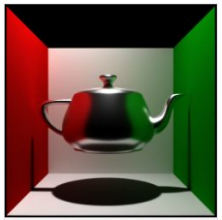


Stanford Bunny in Cornell Box

Stanford University Computer Graphics Laboratory

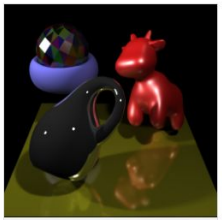
Lucy Statue

Stanford University Computer Graphics Laboratory



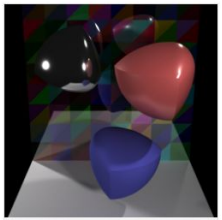
Utah Teapot

University of Utah, Utah Model Repository



Composed Scene

Fausto Javier Da Rosa and Keenan Crane

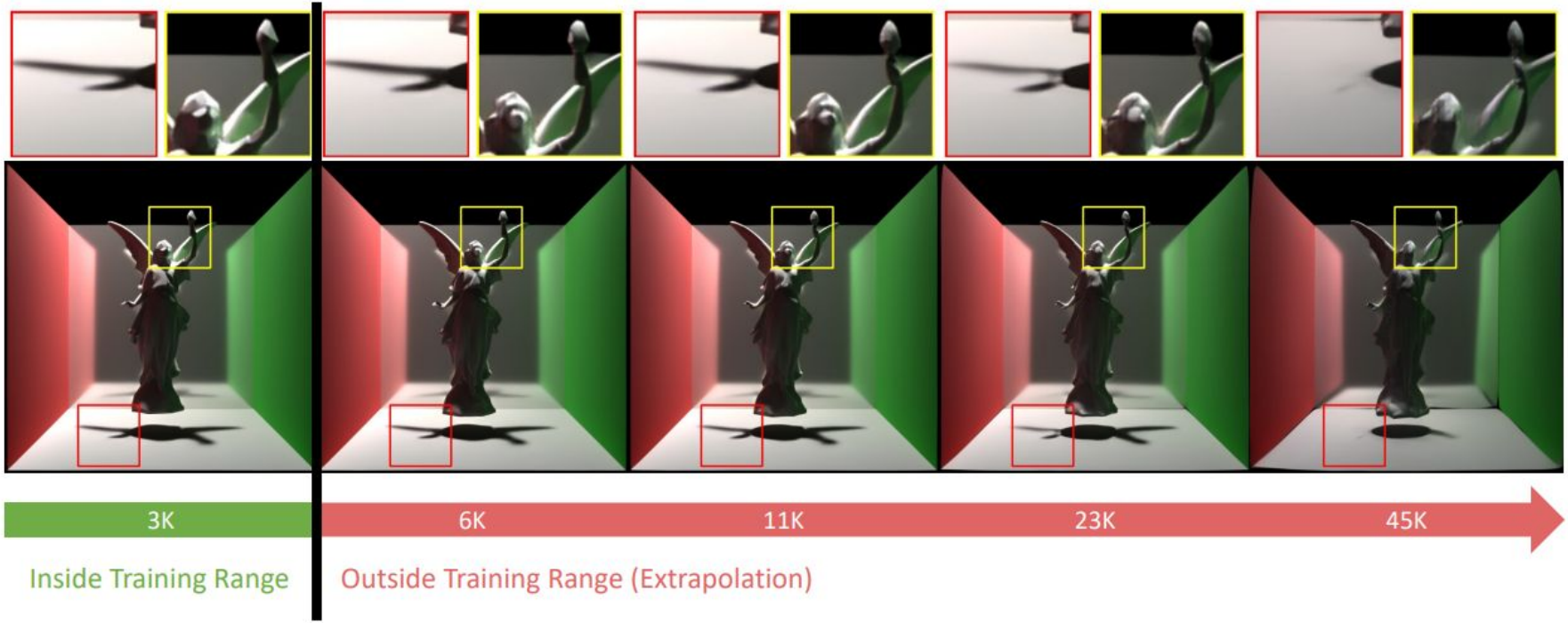


Constant Width Bodies

Keenan Crane

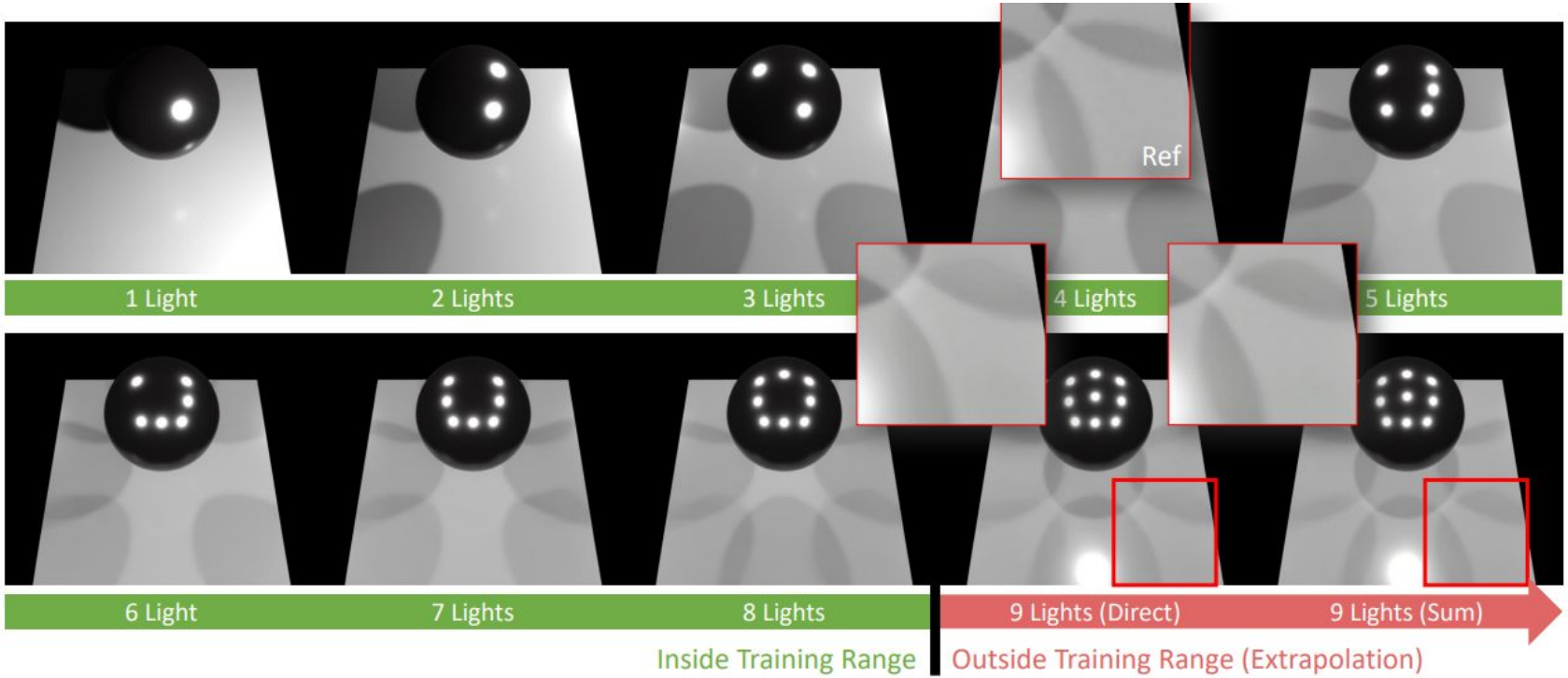
Slide Source: https://microsoft.github.io/renderformer/

Generalization (# Triangles)



Slide Source: https://microsoft.github.io/renderformer/

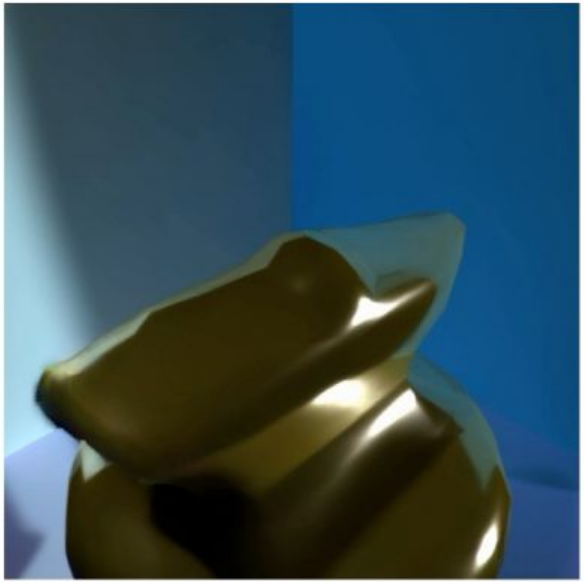
Generalization (# Light Sources)



Slide Source: https://microsoft.github.io/renderformer/

Generalization (Camera Distance)

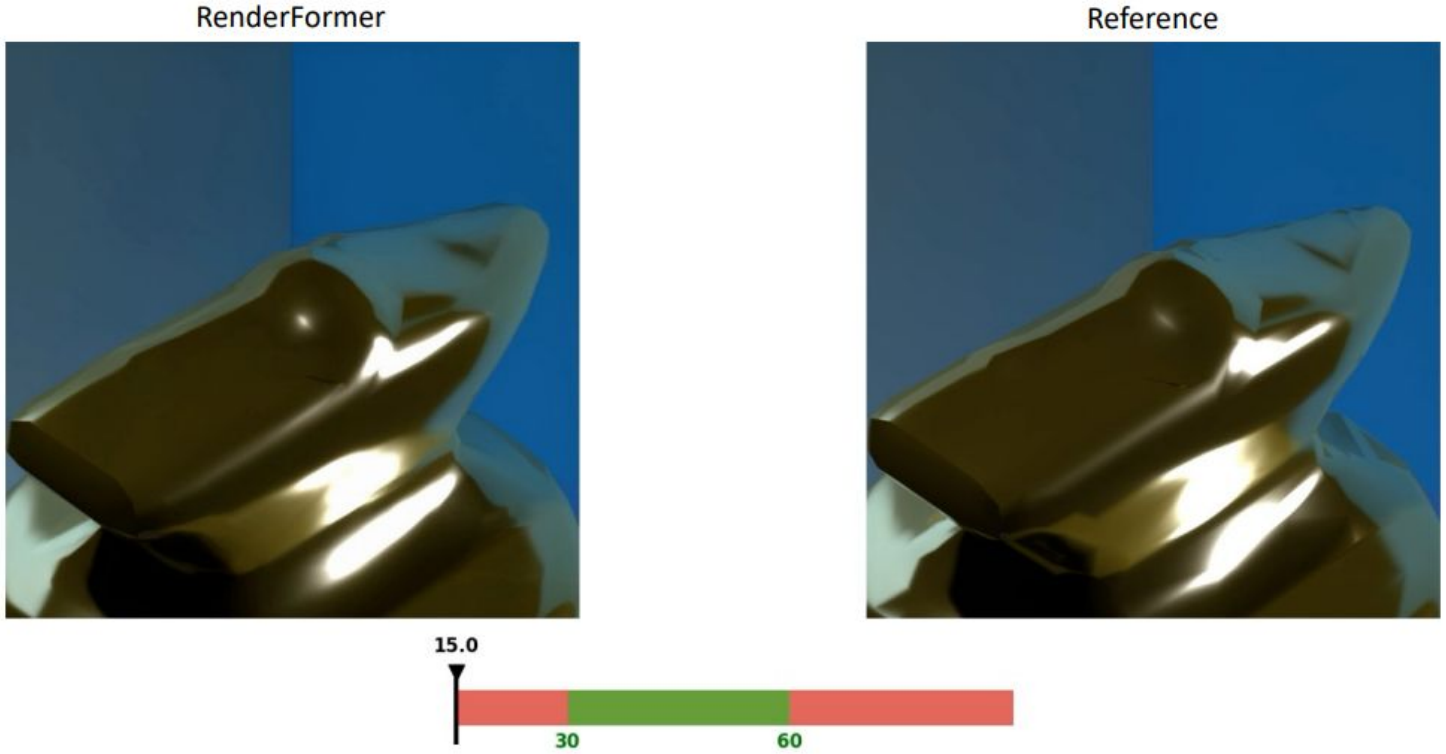
RenderFormer



Reference

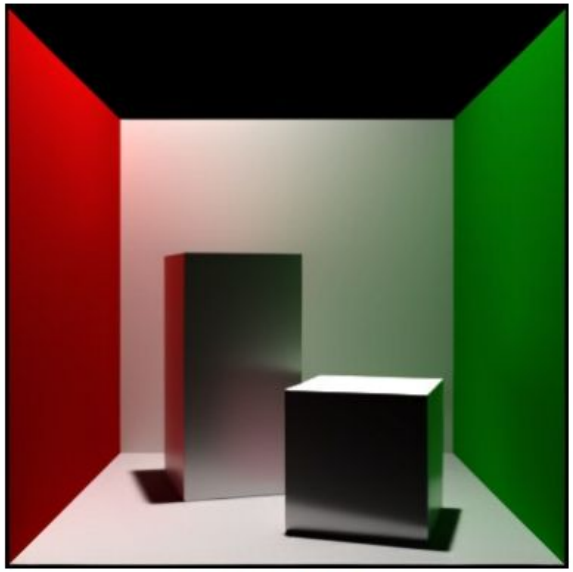
Slide Source: https://microsoft.github.io/renderformer/

Generalization (Camera FoV)

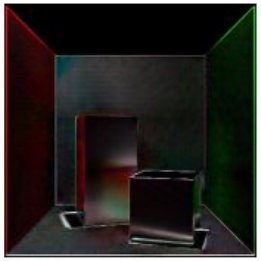


Slide Source: https://microsoft.github.io/renderformer/

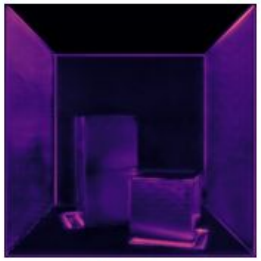
Generalization (Resolution)



Rendered Image



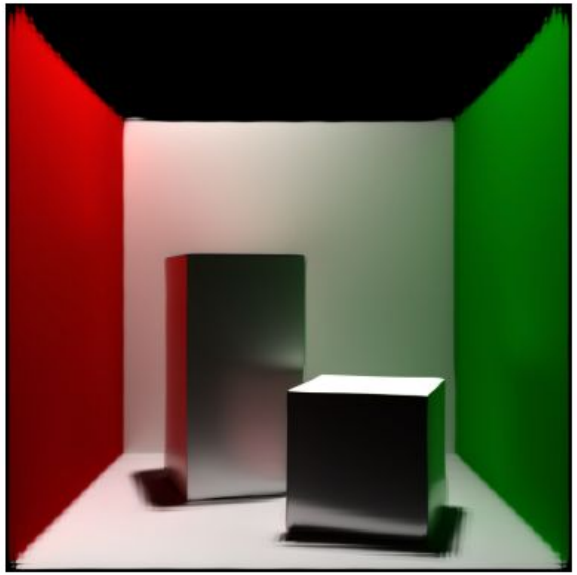
Error Image (x5)



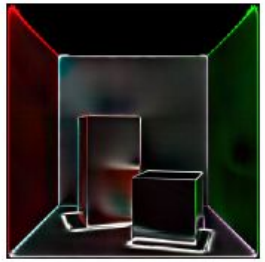
FLIP Error

Slide Source: https://microsoft.github.io/renderformer/

Generalization (Resolution)



Rendered Image

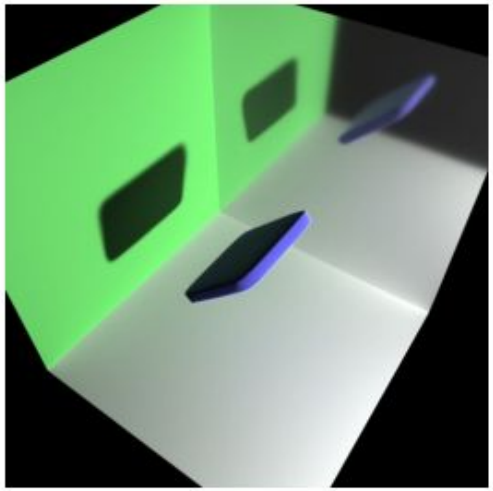


Error Image (x5)

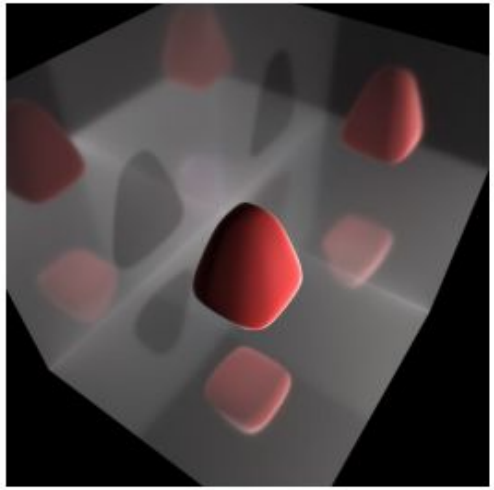
FLIP Error

Slide Source: https://microsoft.github.io/renderformer/

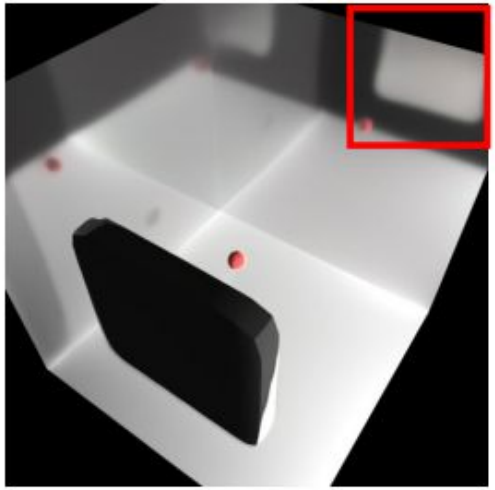
Generalization (Interreflection Depth)



1 Interreflection



2 Interreflections



3 Interreflections

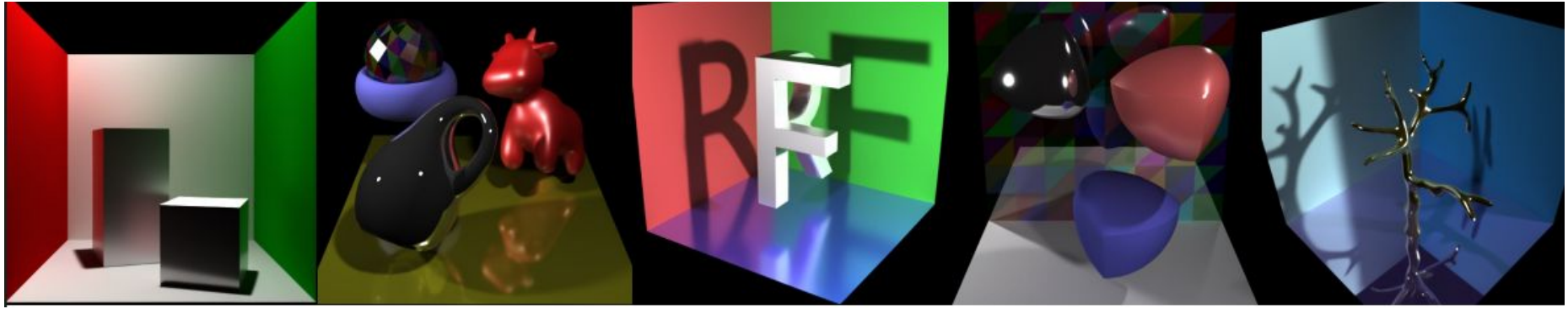
Ref.

Pred.

Slide Source: https://microsoft.github.io/renderformer/

Conclusion

Attention Is All You Need for Rendering



RenderFormer: A Fully Transformer Rendering Pipeline

- ✅ End-to-end with Raw 3D Input
- ✅ No Per-scene Training
- ✅ Full Global Illumination Effects
- ✅ Minimal Prior Constraints

RenderFormer Today: An Initial Step

Future Work

- More Generality
  - Larger Scene
  - Texture
  - Complex Material
  - Environmental Lighting
  - …
- Higher Efficiency

Slide Source: https://microsoft.github.io/renderformer/

Q&A

Quiz

Thank you for Your Attention!

RenderFormer: Transformer-based Neural Rendering of Triangle Meshes with Global Illumination