Neural Two-Level Monte Carlo Real-Time Rendering

Mikhail Dereviannykh, Dmitrii Klepikov, Johannes Hanika, Carsten Dachsbacher

Karlsruhe Institute of Technology

Project Page

$$\int_{H+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_o) \cos\theta_i d\omega_i$$

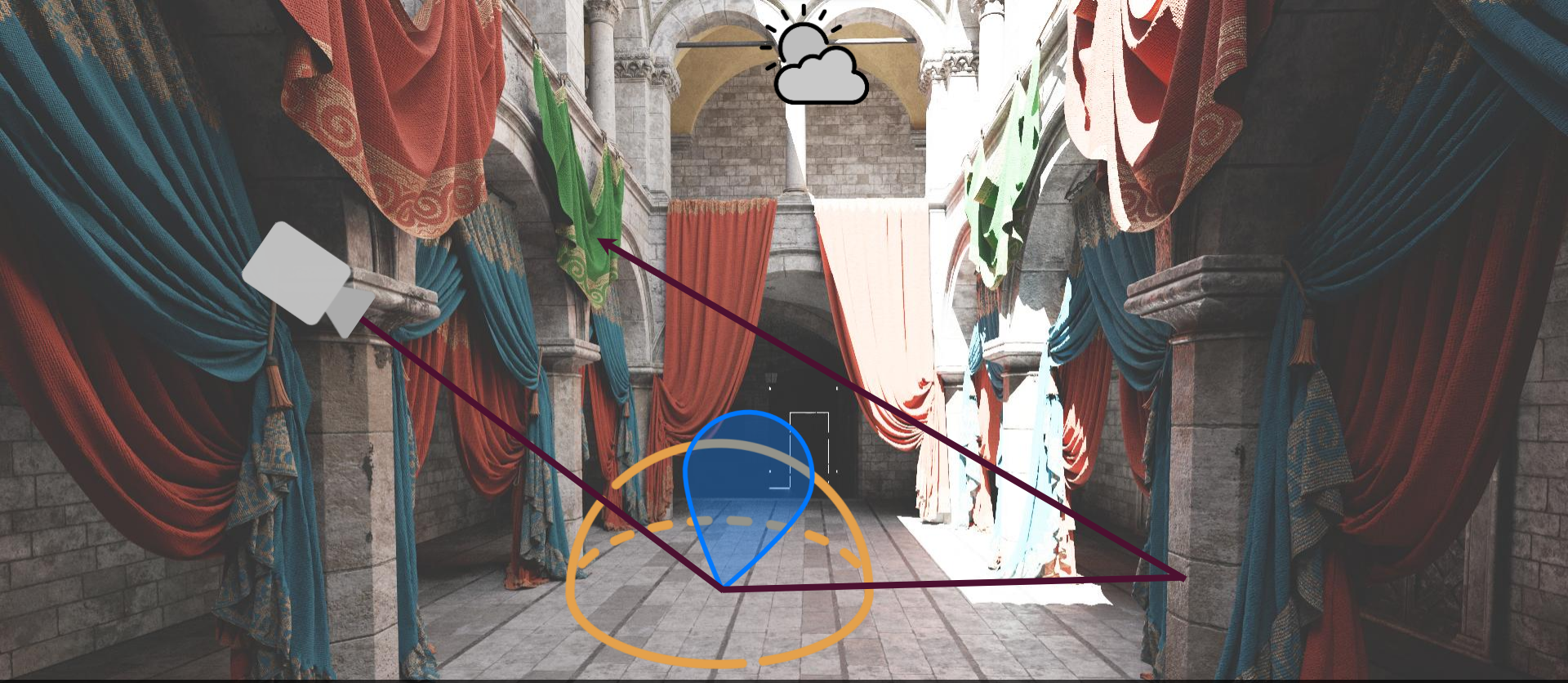
Incident radiance

Materials BRDF

$$\int_{H+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_o) \cos\theta_i d\omega_i$$

Incident radiance

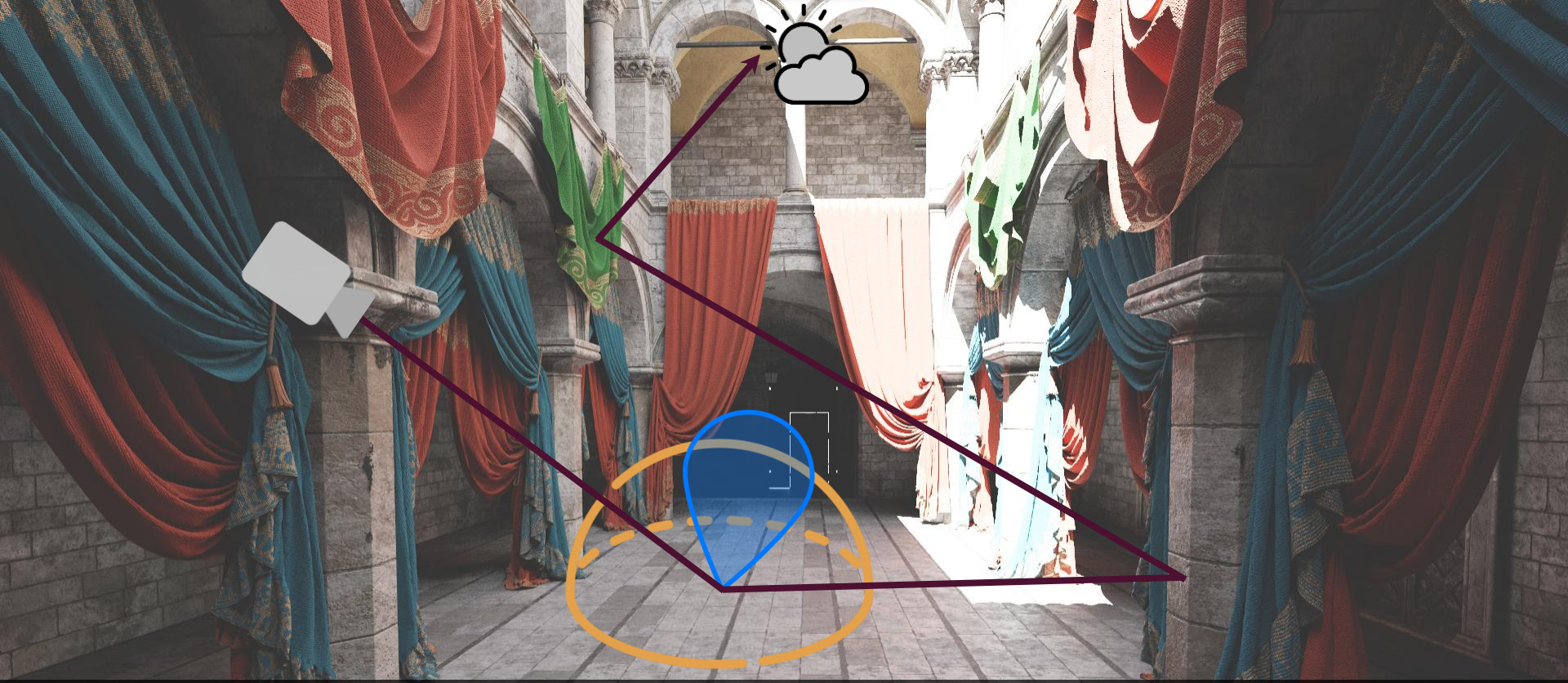
Materials BRDF



$$\int_{H+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_o) \cos\theta_i d\omega_i$$

Incident radiance

Materials BRDF



$$\int_{H+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_o) \cos\theta_i d\omega_i$$

Incident radiance

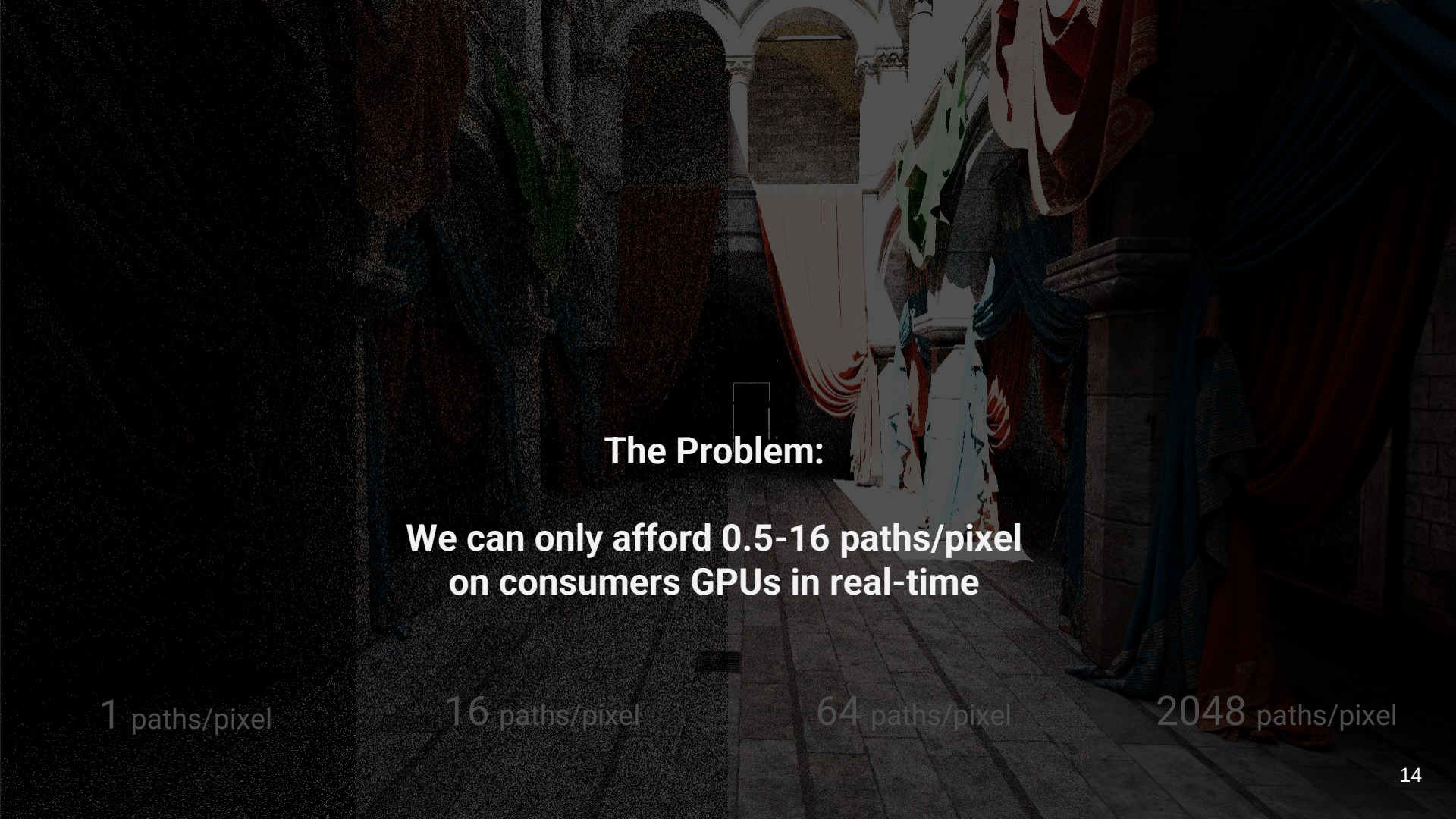
Materials BRDF

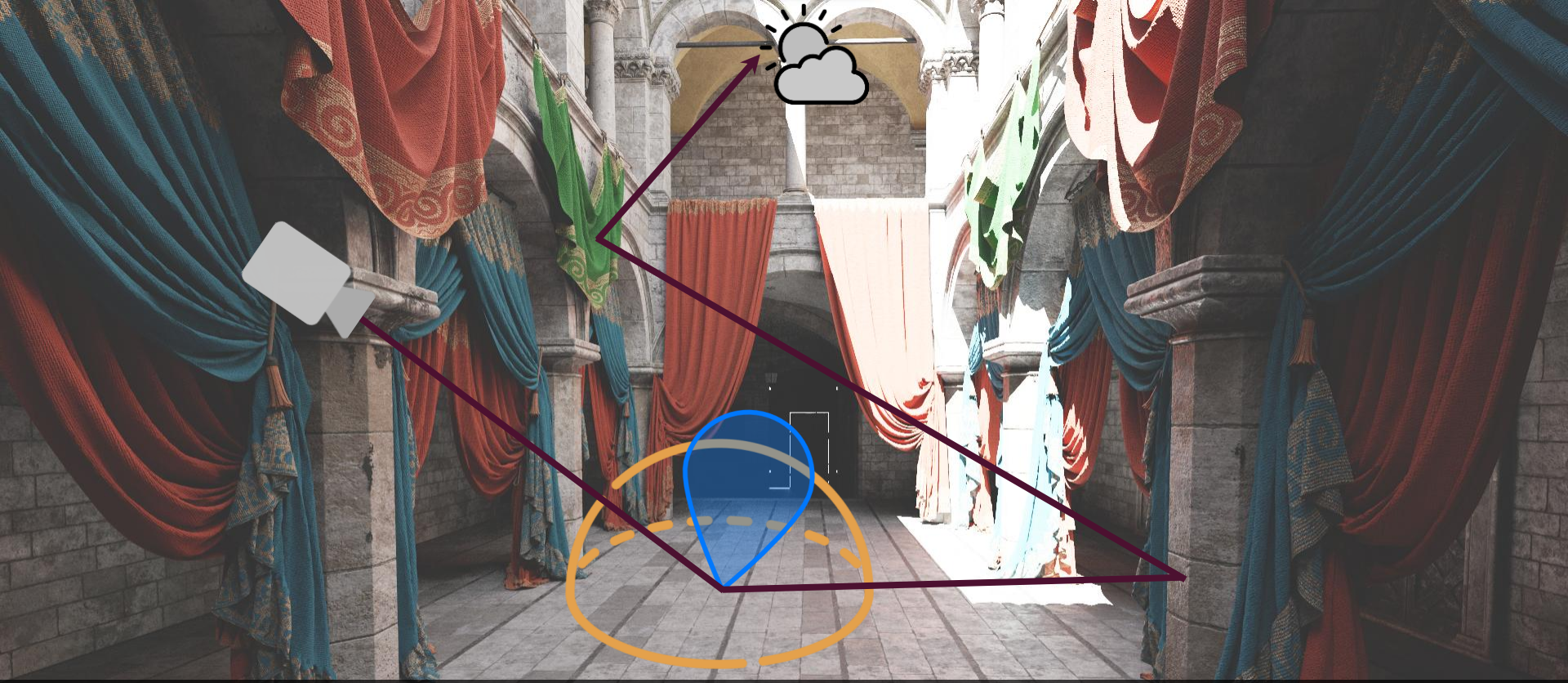

1 path/pixel


1 path/pixel
16 paths/pixel


1 path/pixel
16 paths/pixel
64 paths/pixel


1 path/pixel
16 paths/pixel
64 paths/pixel
2048 paths/pixel

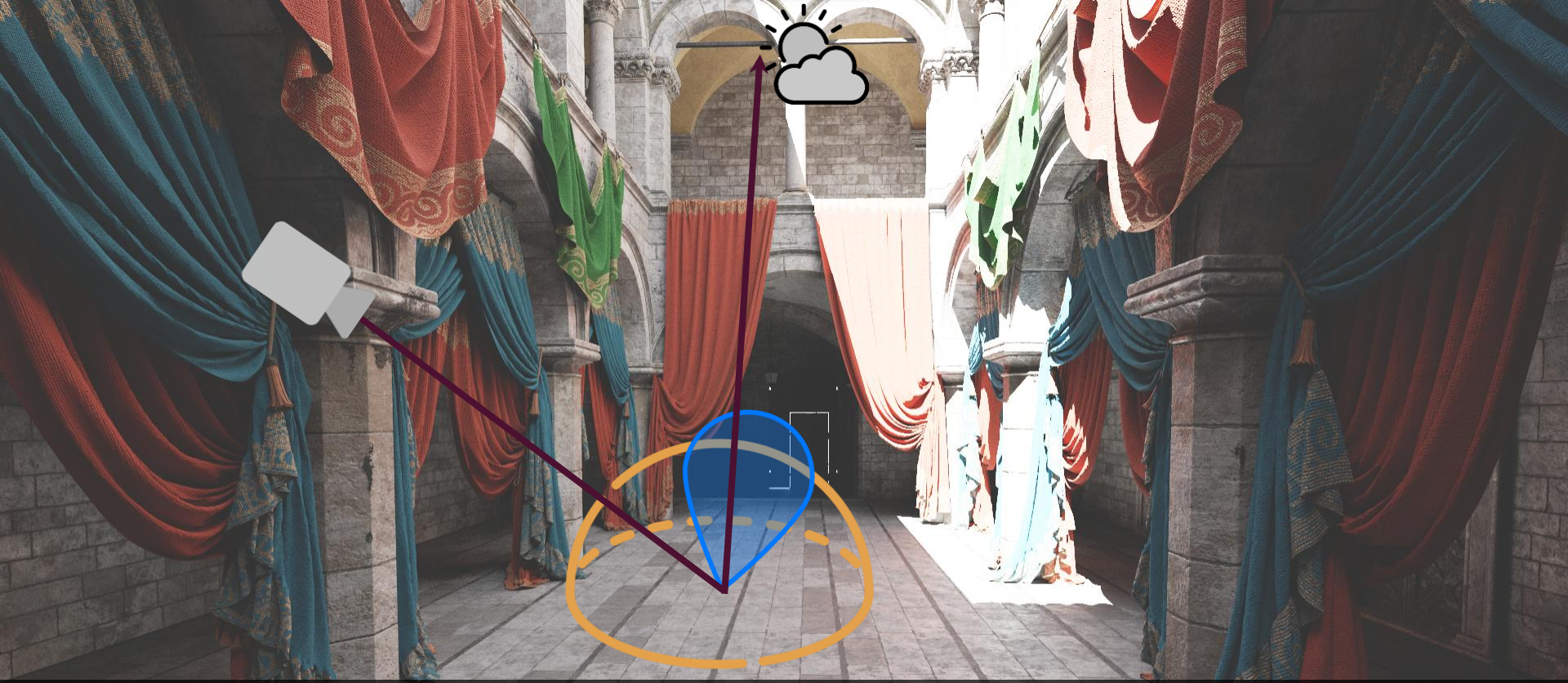

The Problem:
We can only afford 0.5-16 paths/pixel
on consumers GPUs in real-time
1 paths/pixel
16 paths/pixel
64 paths/pixel
2048 paths/pixel



$$\int_{H+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_o) \cos\theta_i d\omega_i$$

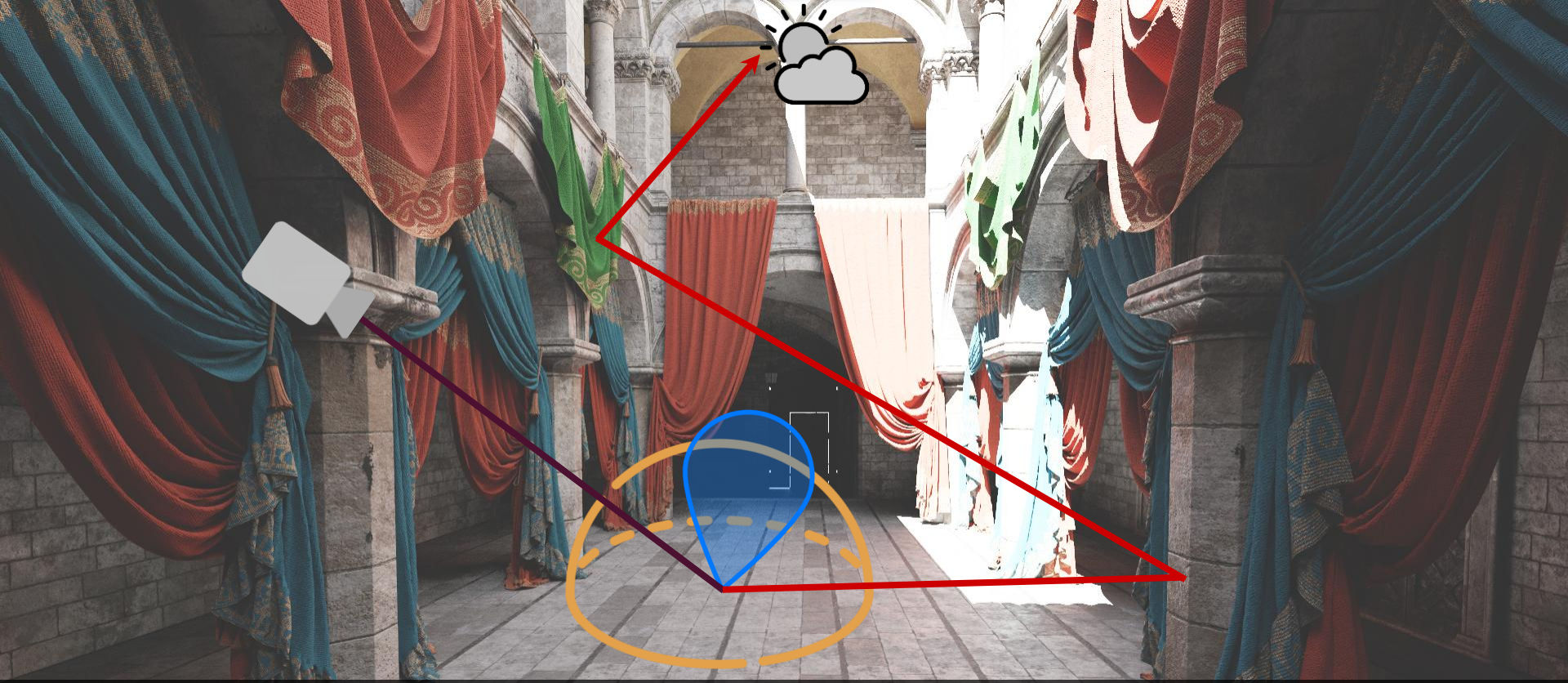
Incident radiance

Materials BRDF

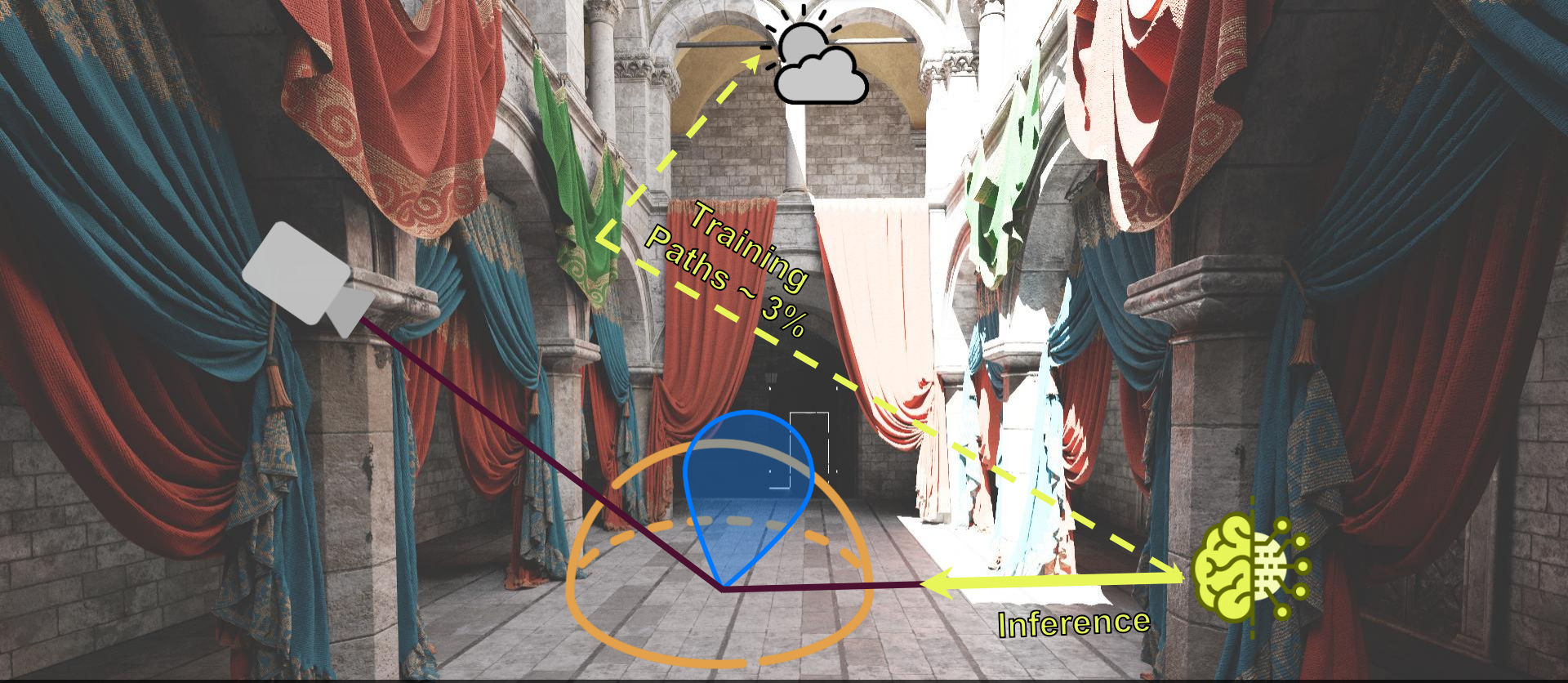


Importance sampling by guiding rays towards incident radiance

[Vorba et al. 2019, Müller et al. 2017, Müller at al. 2019, Rath et al. 2020]



Bounces are expensive
They introduce additional variance

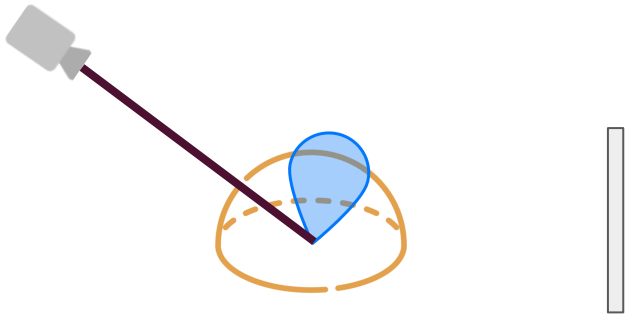


Tiny Neural Networks predict
Outgoing Radiance *[Müller et al. 2021]*

and the other non-neural works [Greger et al. 1998, Majercik et al. 2019, Krivanek et al . 2005]

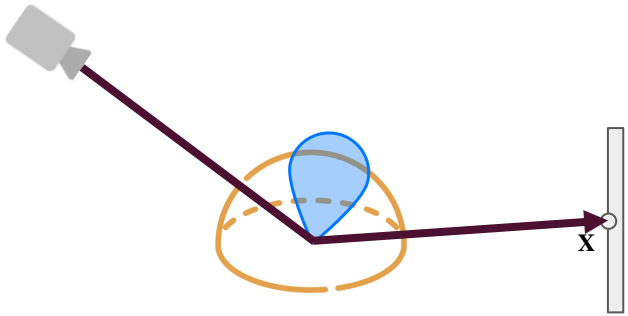
Neural Radiance Cache (NRC) *[Müller et al. 2021, 2023]*

Project Page



Neural Radiance Cache (NRC) *[Müller et al. 2021, 2023]*

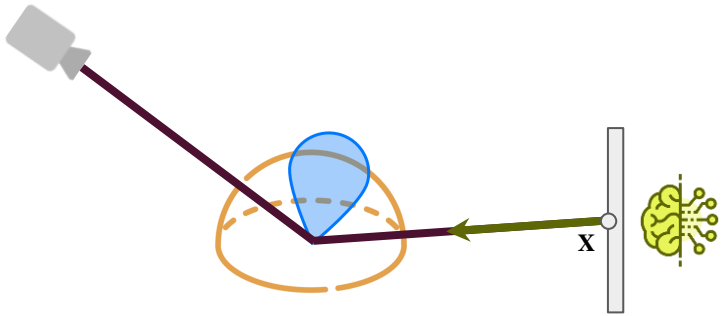
Project Page



[1] Müller et al. 2023

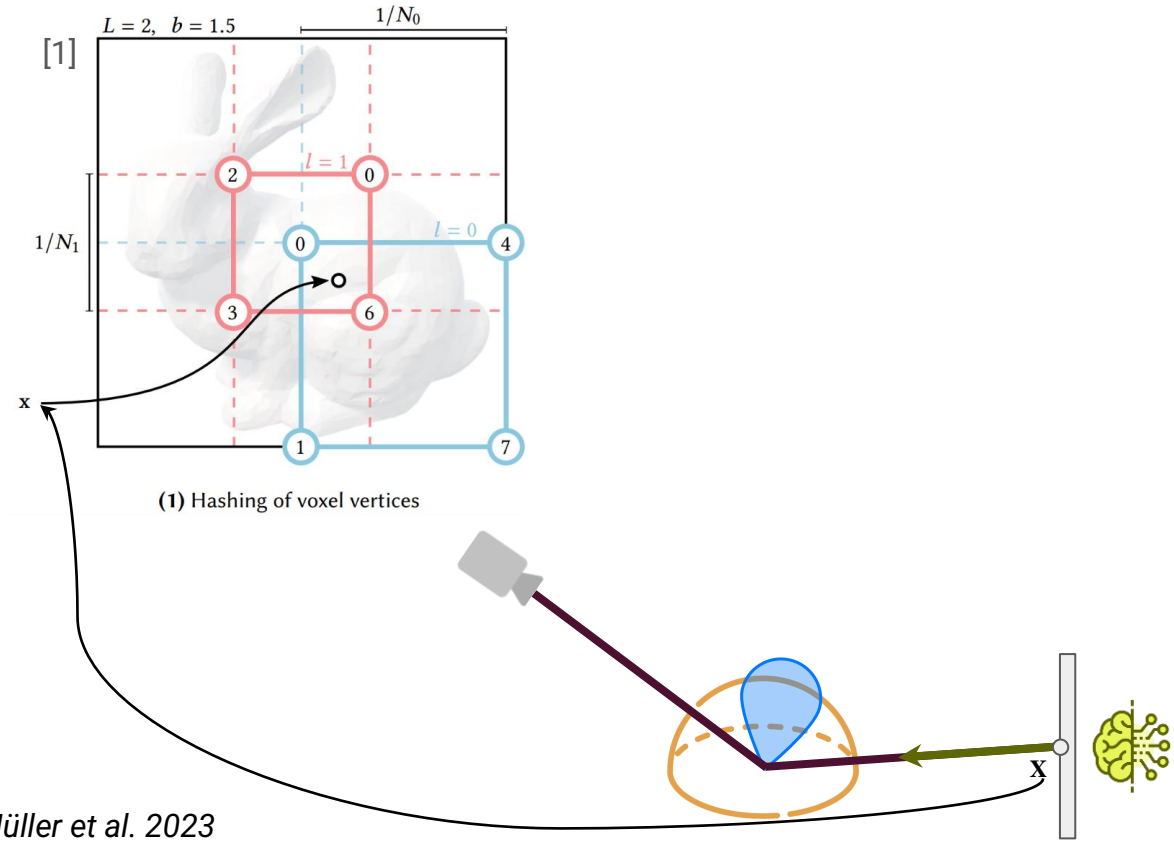
Neural Radiance Cache (NRC) *[Müller et al. 2021, 2023]*

Project Page



Neural Radiance Cache (NRC) *[Müller et al. 2021, 2023]*

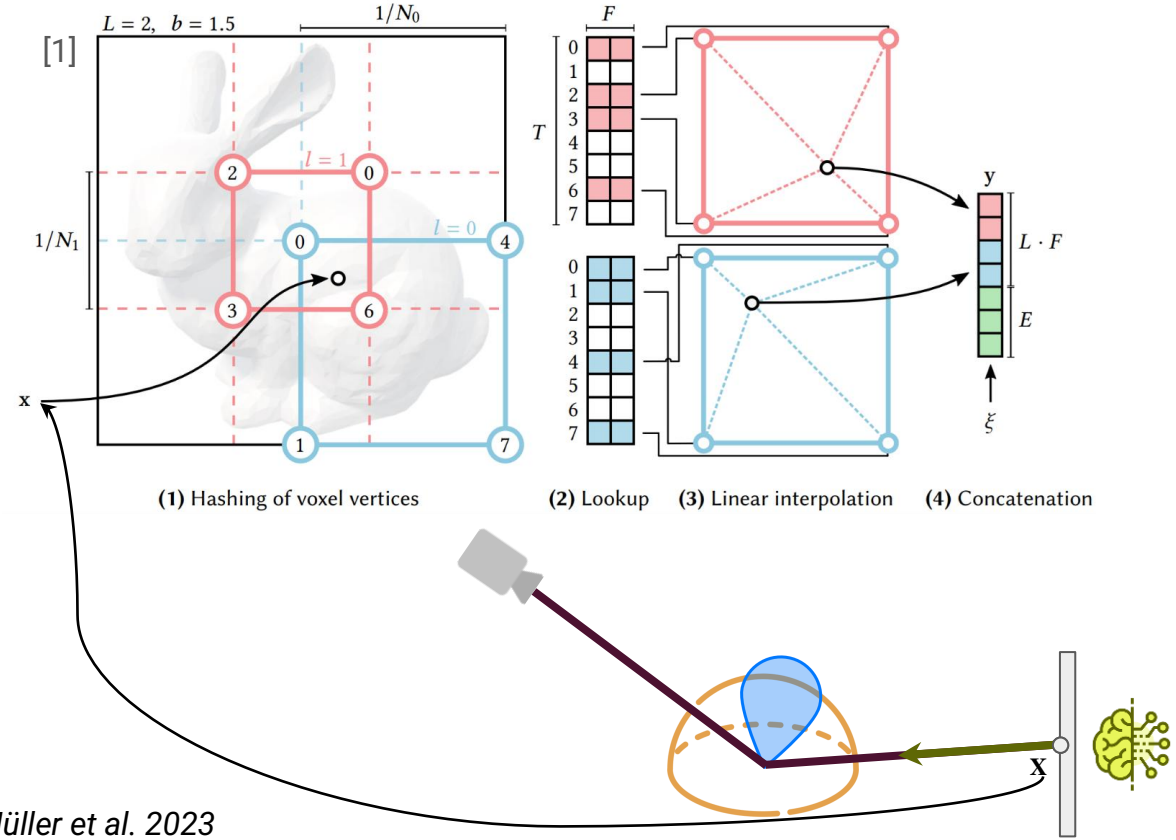
Project Page



[1] Müller et al. 2023

Neural Radiance Cache (NRC) *[Müller et al. 2021, 2023]*

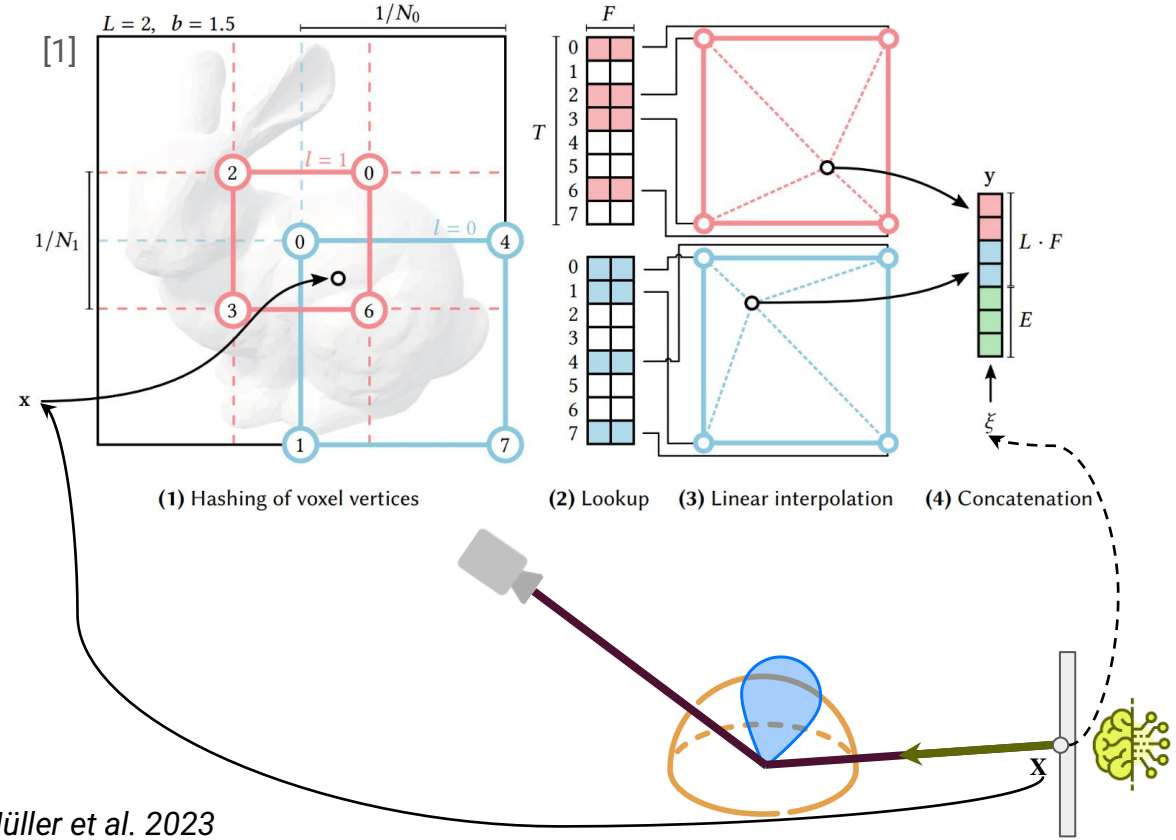
Project Page



[1] Müller et al. 2023

Neural Radiance Cache (NRC) *[Müller et al. 2021, 2023]*

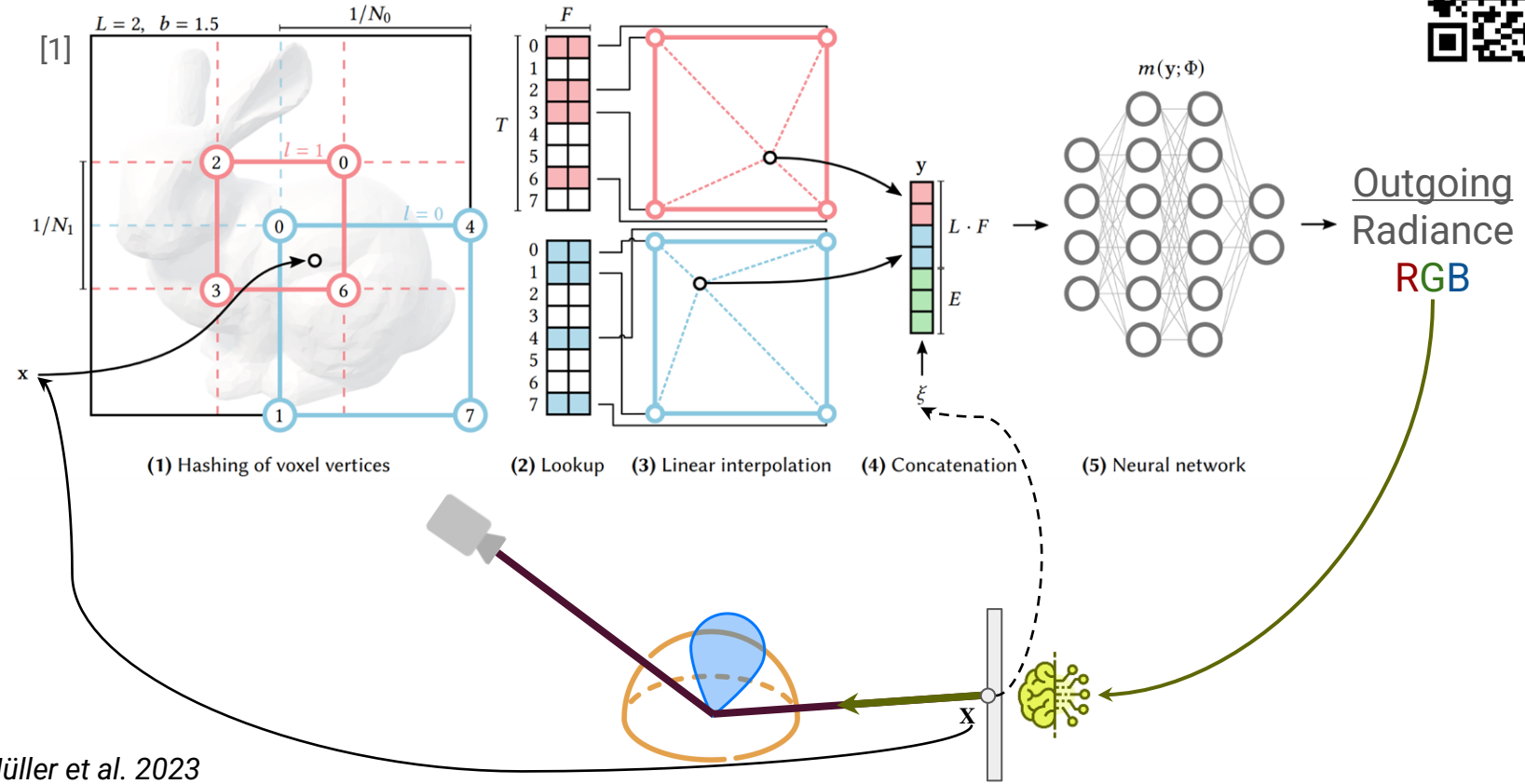
Project Page



[1] Müller et al. 2023

Neural Radiance Cache (NRC) *[Müller et al. 2021, 2023]*

Project Page

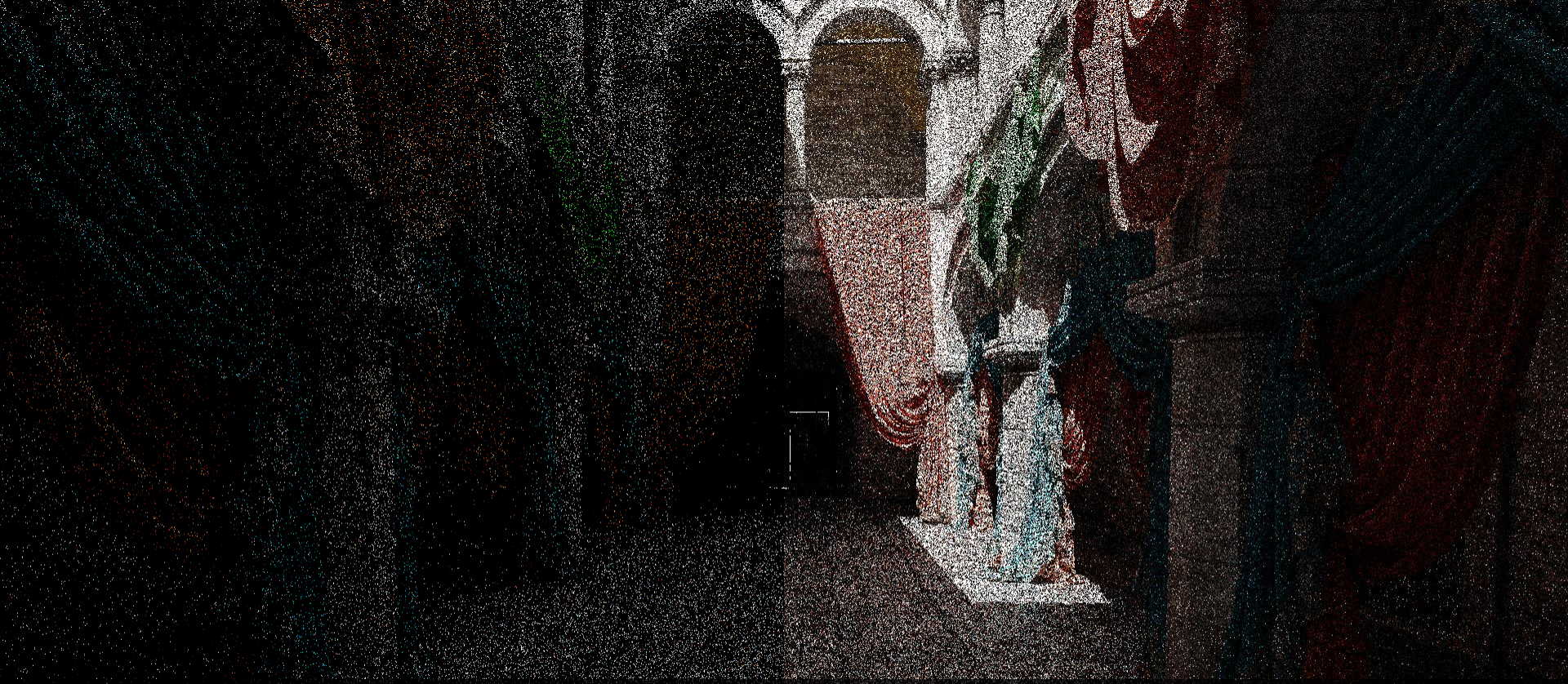


[1] Müller et al. 2023

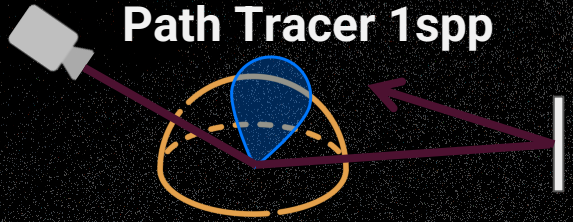
Visualization of NRC

[Müller et al. 2021, Müller et al. 2022]

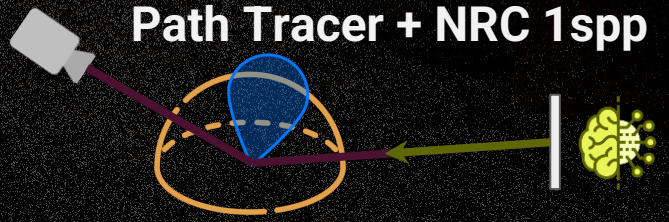

Ground Truth



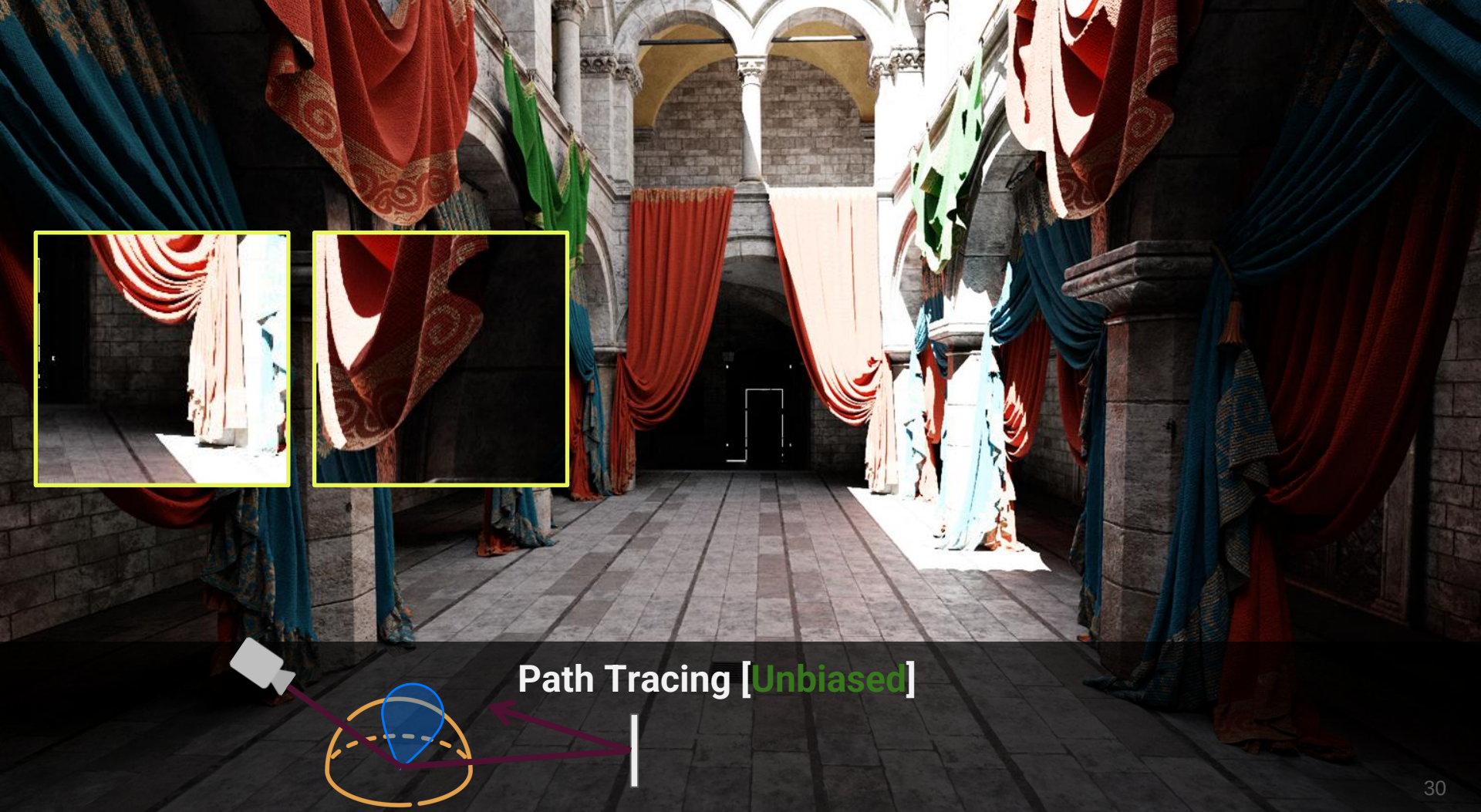
Path Tracer 1spp

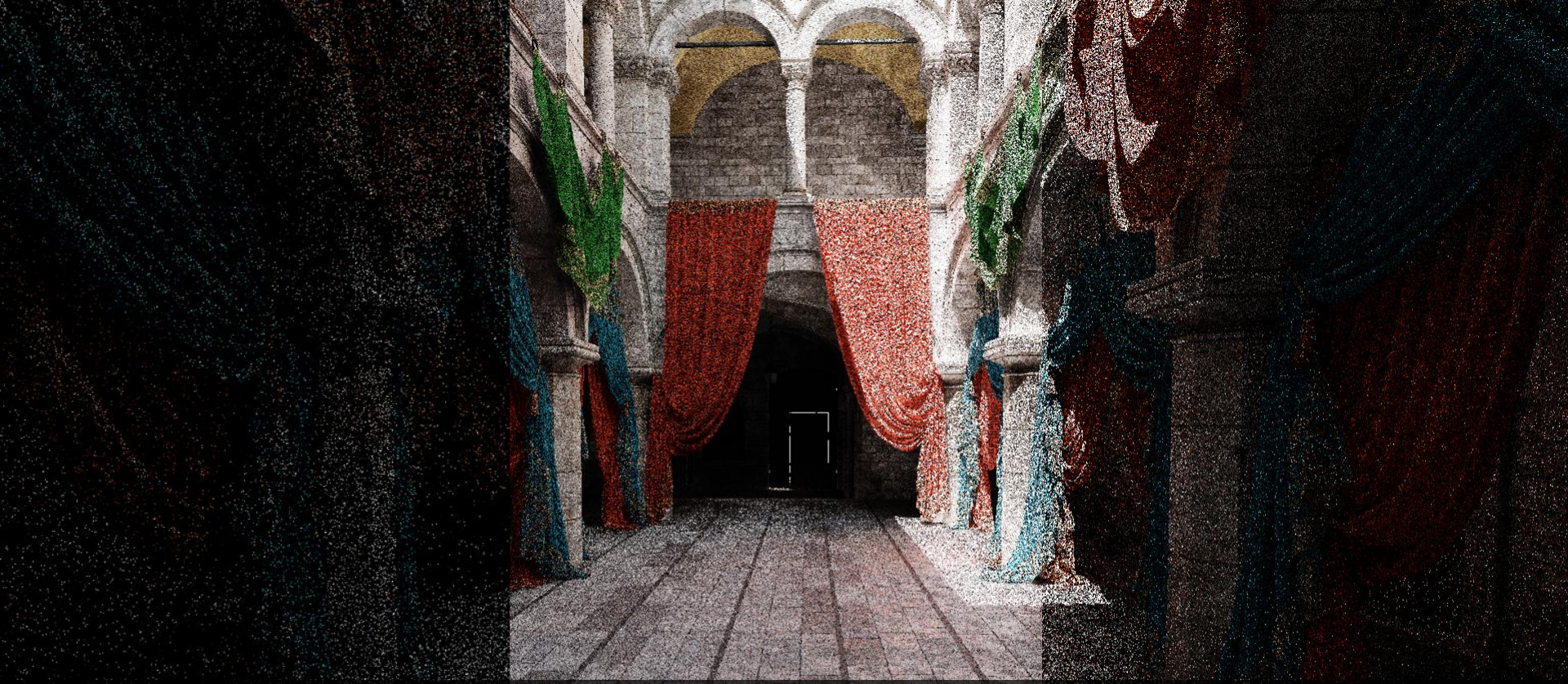


Path Tracer + NRC 1spp



Path Tracing NRC [Biased]


Path Tracing [Unbiased]



Path Tracer 1spp
[Unbiased]

PT + NIRC (Ours), 1spp
[Biased]

PT + NRC, 1spp
[Biased]

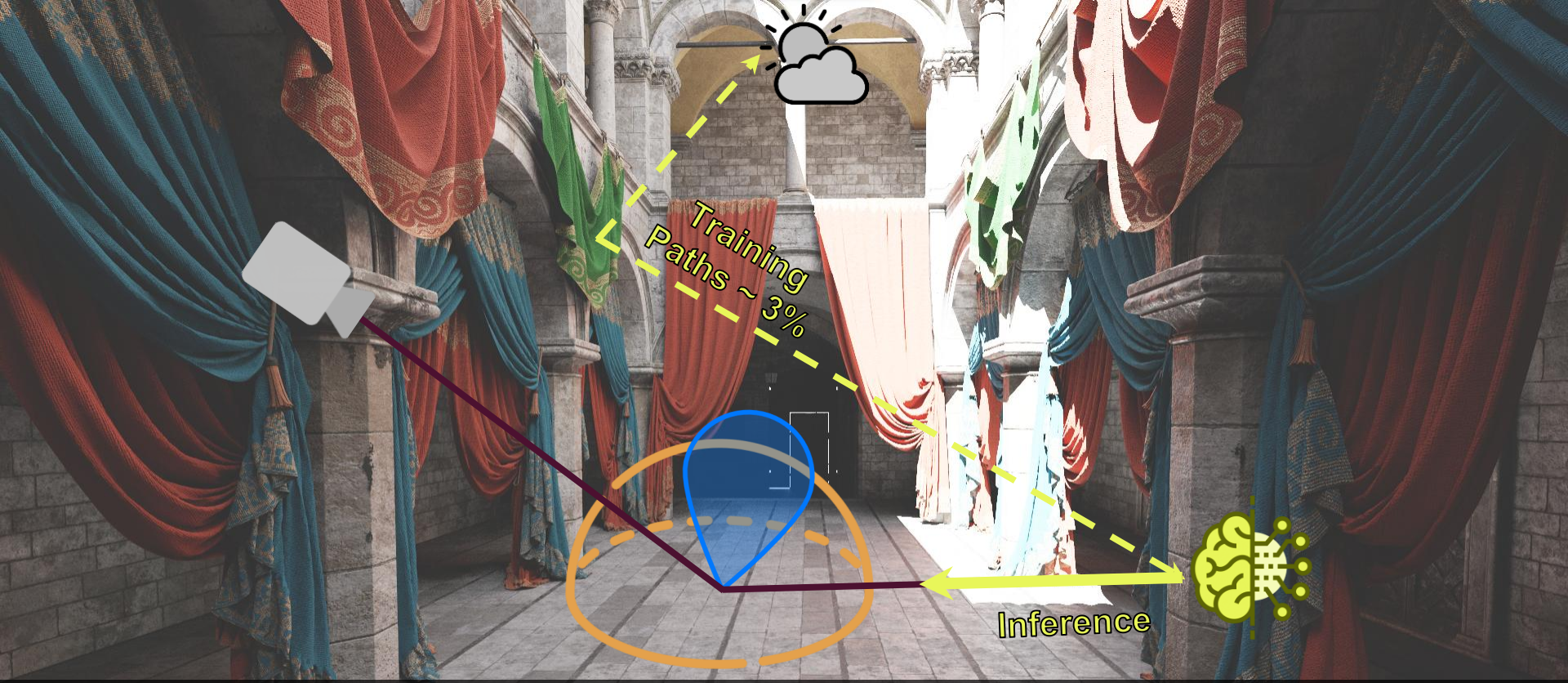
Equal Time Comparison

Path Tracer 2spp
[Unbiased]

PT + NIRC (Ours), 1spp
[Unbiased]

PT + NRC, 2spp
[Biased]

Equal Time Comparison



$$\underbrace{\text{NRC}}_{\text{Brain}} \approx \int_{H+} \underbrace{L_i(x, \omega_i)}_{\text{Incident Radiance}} \underbrace{f_r(x, \omega_i, \omega_o)}_{\text{Materials BRDF}} \cos\theta_i \, d\omega_i$$

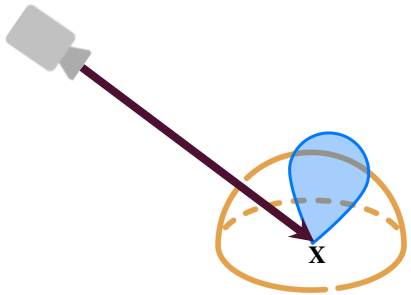
$$\int_{H+} \underbrace{L_i(x, \omega_i)}_{\text{Our Cache}} \underbrace{f_r(x, \omega_i, \omega_o) \cos\theta_i}_{\text{Materials}} d\omega_i$$

NIRC

BRDF

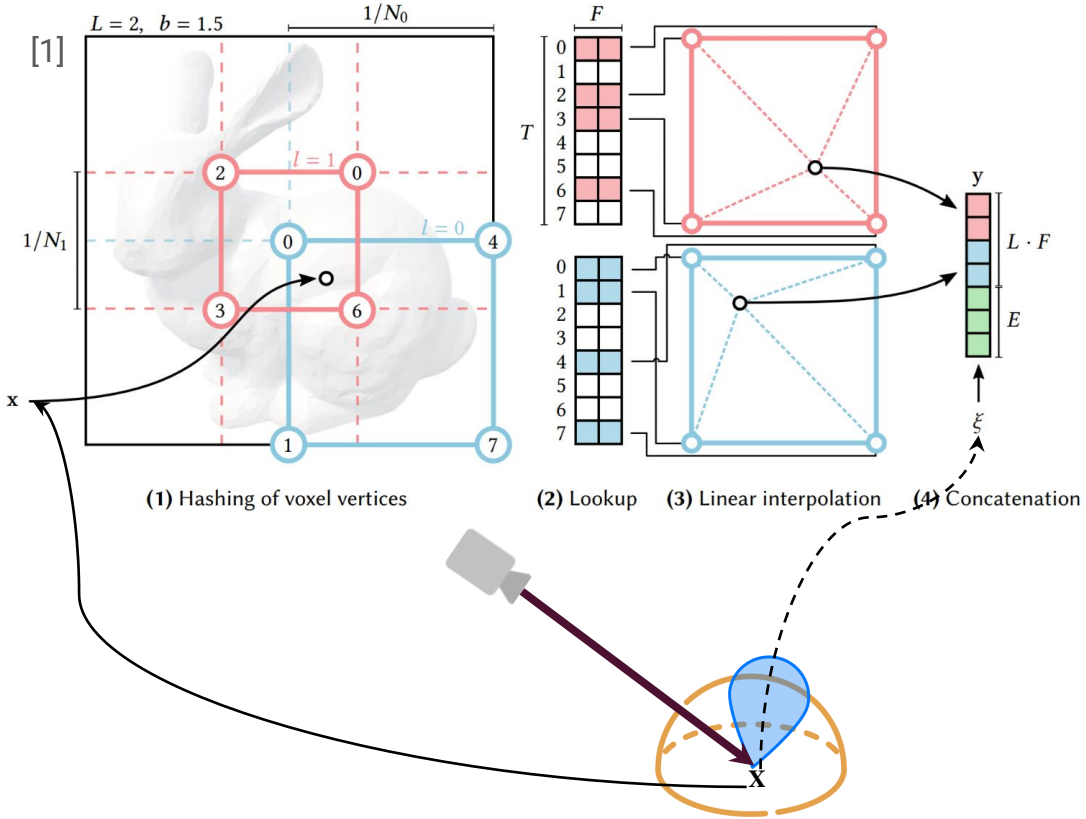
Neural Incident Radiance Cache (NIRC)

Project Page



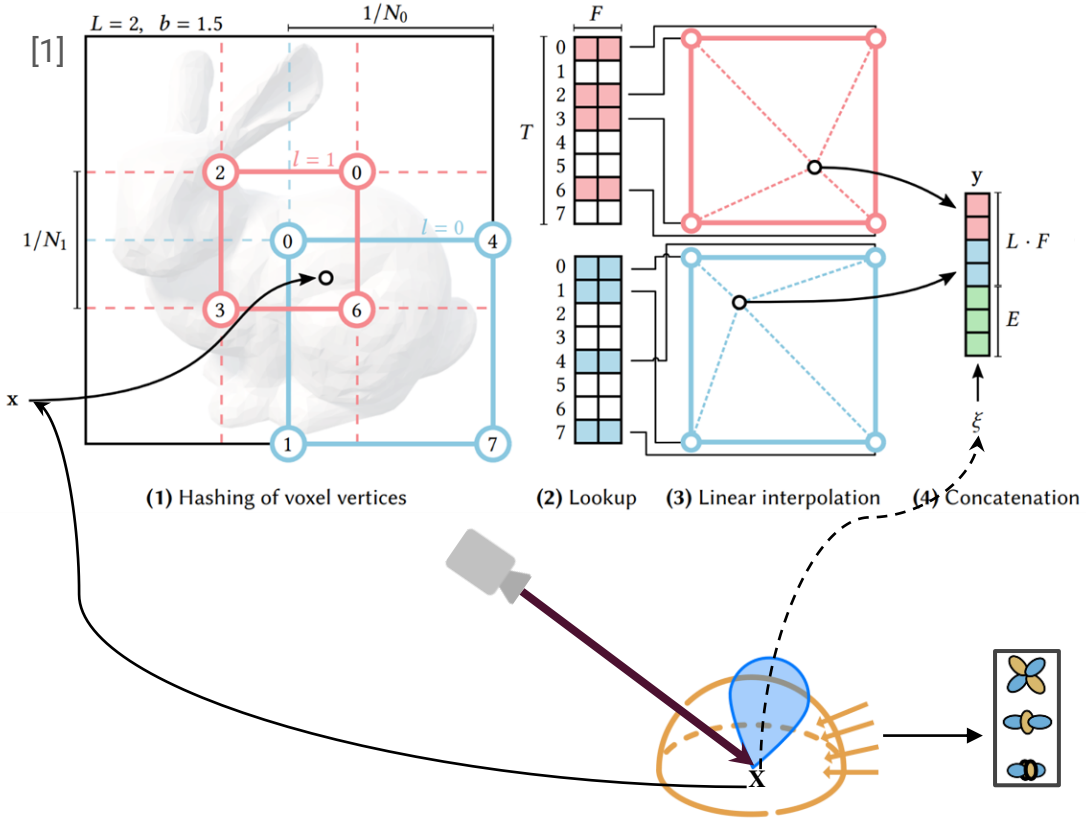
Neural Incident Radiance Cache (NIRC)

Project Page



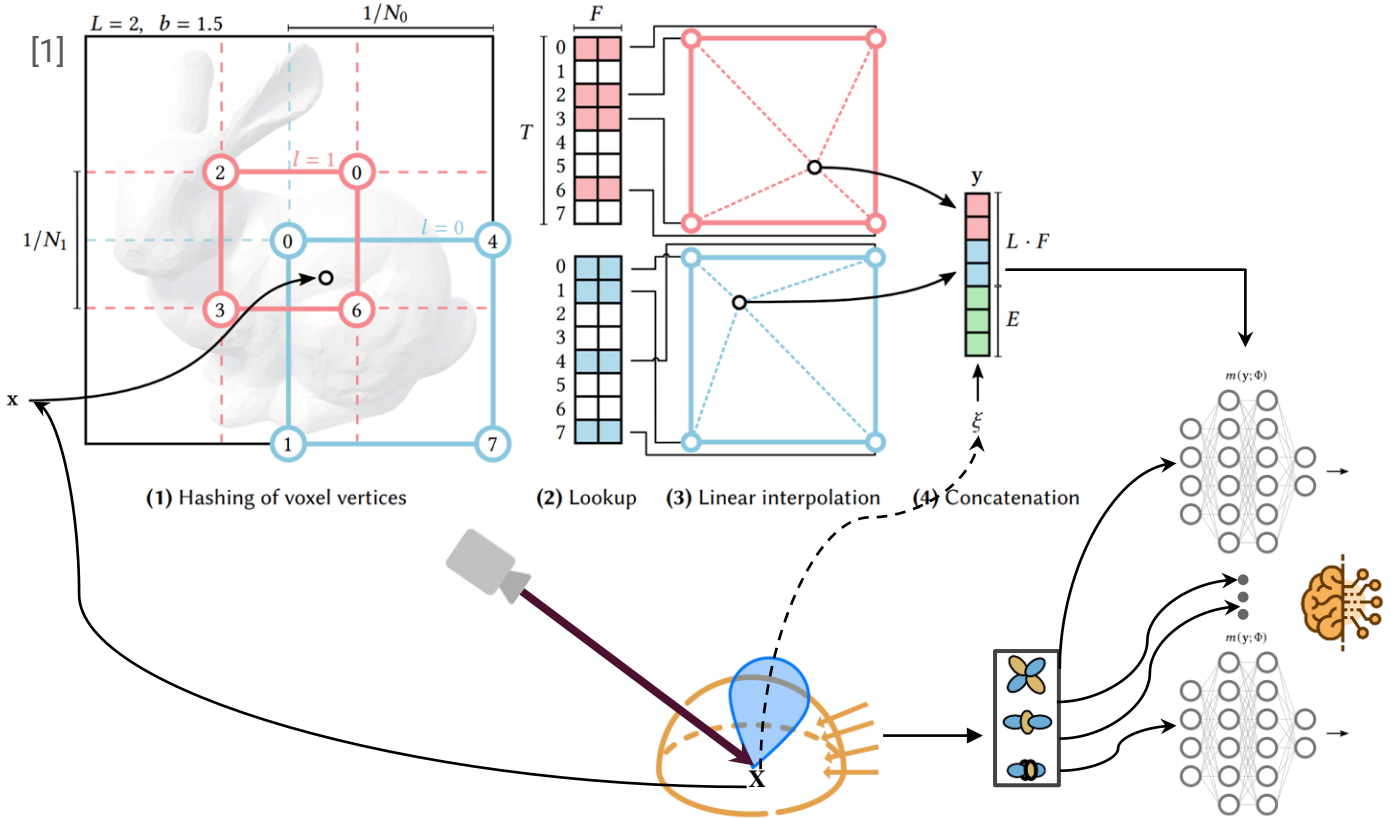
Neural Incident Radiance Cache (NIRC)

Project Page



Neural Incident Radiance Cache (NIRC)

Project Page



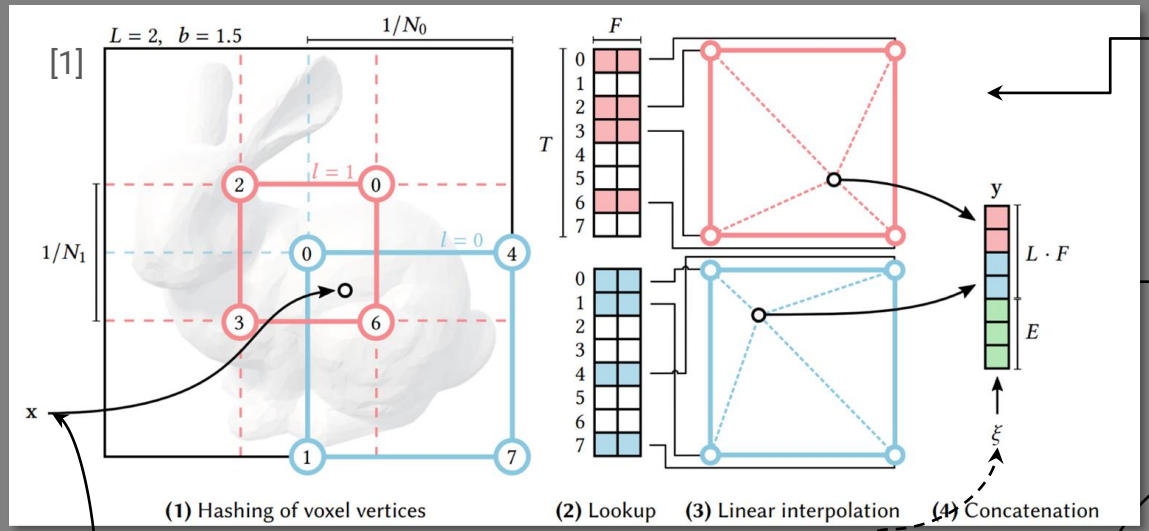
Incident
Radiance
RGB

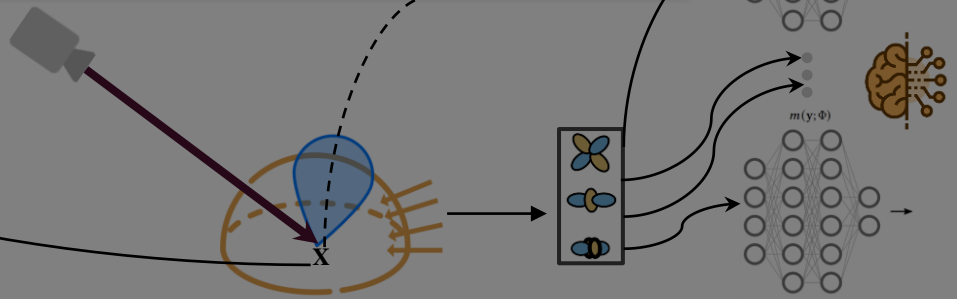
For each sampled direction

Neural Incident Radiance Cache (NIRC)

Hash-Grids take up to 70% of time for original NRC

NIRC samples them once for all directions

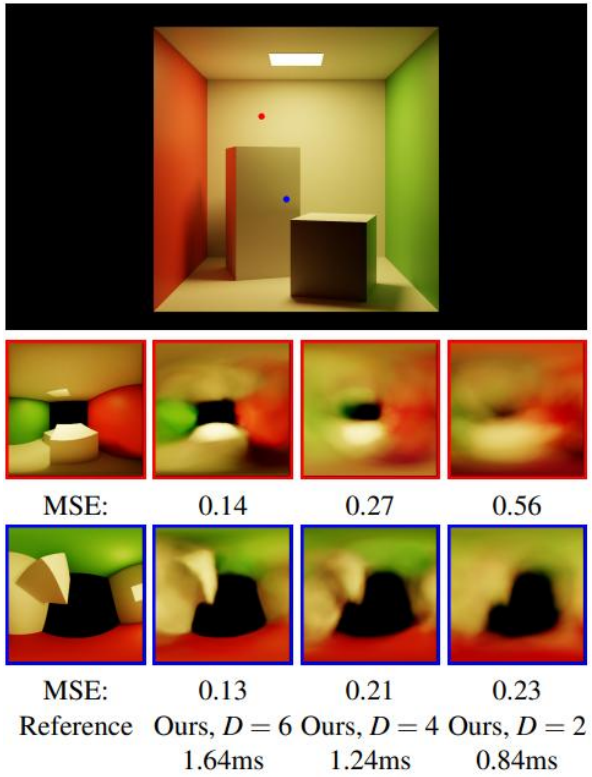




Incident
Radiance
RGB

For each sampled direction

NIRC Visualization



Scales pretty well with the depth of MLP (D = 2/4/6)

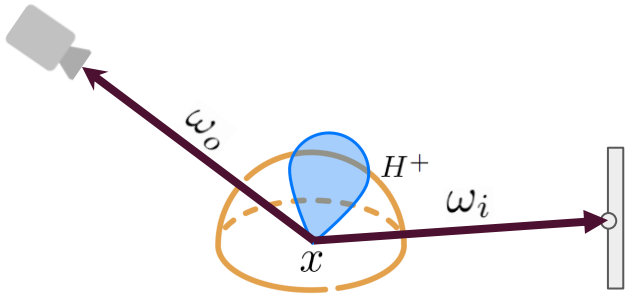
Two-Level Monte Carlo?

Debiasing?

(De)bias?

Project Page

$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_{H^+} \overbrace{L_i(x,\omega_i)}^{\text{Incident Radiance}} \overbrace{f_r(x,\omega_i,\omega_o)\cos\theta_i}^{\text{Materials BRDF}} d\omega_i$$

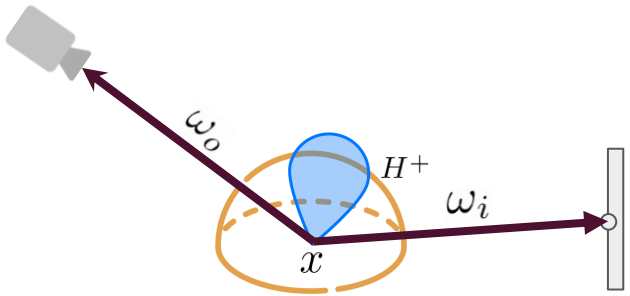


(De)bias?

Project Page

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{H^+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_o) \cos\theta_i d\omega_i$$

$$\hat{L}_o(x, \omega_o) \approx L_e(x, \omega_o) + \frac{1}{N} \sum_{i=1}^{N} \frac{\overbrace{L_i(x, \omega_i)}^{\text{Incident Radiance}} \overbrace{f_r(x, \omega_i, \omega_o)\cos\theta_i}^{\text{Materials BRDF}}}{p(\omega_i)}$$

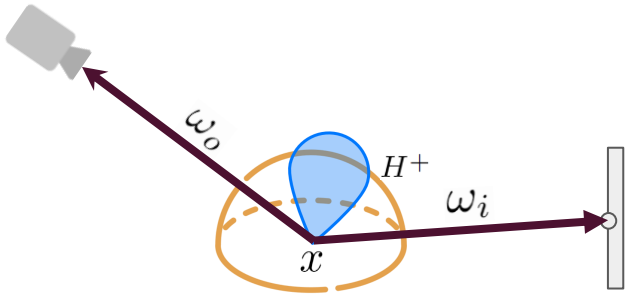


(De)bias?

Project Page

$$\hat{L}_o(x,\omega_o) \approx L_e(x,\omega_o) + \frac{1}{N}\sum_{i=1}^{N} \frac{L_i(x,\omega_i) f_r(x,\omega_i,\omega_o)\cos\theta_i}{p(\omega_i)}$$

$$\hat{L}_o(x,\omega_o) \approx L_e(x,\omega_o) + \frac{1}{N}\sum_{i=1}^{N} \frac{\overbrace{NN_i(x,\omega_i,\hat{\boldsymbol{\theta}})}^{\text{NIRC}} \overbrace{f_r(x,\omega_i,\omega_o)\cos\theta_i}^{\text{Materials BRDF}}}{p(\omega_i)}$$



Two-Level Monte Carlo

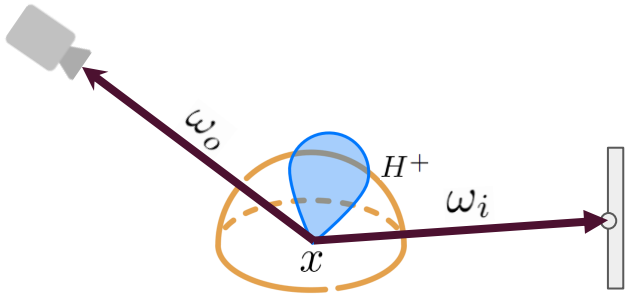
Project Page

Radiance Cache Integral Estimator

$$\hat{L}_c(x, \omega_o, \hat{\boldsymbol{\theta}}) \approx \frac{1}{N_c} \sum_{i=1}^{N_c} \frac{NN_i(x, \omega_i, \hat{\boldsymbol{\theta}}) f_r(x, \omega_i, \omega_o) \cos\theta_i}{p(\omega_i)}$$

trainable weights — *NIRC* — *BRDF*

Final **biased** estimator: $\hat{L}_o(x, \omega_o, \hat{\boldsymbol{\theta}}) \approx L_e(x, \omega_o) + \hat{L}_c(x, \omega_o, \hat{\boldsymbol{\theta}})$



Two-Level Monte Carlo

Project Page

Radiance Cache Integral Estimator

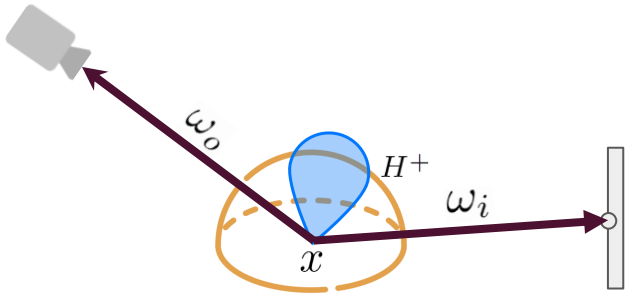
$$\hat{L}_c(x, \omega_o, \hat{\boldsymbol{\theta}}) \approx \frac{1}{N_c} \sum_{i=1}^{N_c} \frac{NN_i(x, \omega_i, \hat{\boldsymbol{\theta}}) f_r(x, \omega_i, \omega_o) \cos\theta_i}{p(\omega_i)}$$

trainable weights — *NIRC* — *BRDF*

Residual Error Integral Estimator

$$\hat{L}_r(x, \omega_o, \hat{\boldsymbol{\theta}}) \approx \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{(L_i(x, \omega_i) - NN_i(x, \omega_i, \hat{\boldsymbol{\theta}})) f_r(x, \omega_i, \omega_o) \cos\theta_i}{p(\omega_i)}$$

Final **biased** estimator: $\hat{L}_o(x, \omega_o, \hat{\boldsymbol{\theta}}) \approx L_e(x, \omega_o) + \hat{L}_c(x, \omega_o, \hat{\boldsymbol{\theta}})$



Two-Level Monte Carlo

Project Page

Radiance Cache Integral Estimator

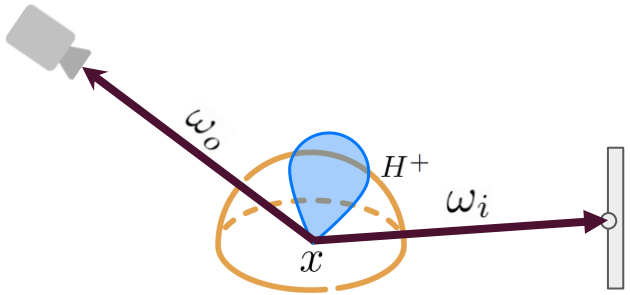
$$\hat{L}_c(x, \omega_o, \hat{\boldsymbol{\theta}}) \approx \frac{1}{N_c} \sum_{i=1}^{N_c} \frac{NN_i(x, \omega_i, \hat{\boldsymbol{\theta}}) f_r(x, \omega_i, \omega_o) \cos\theta_i}{p(\omega_i)}$$

trainable weights — *NIRC* — *BRDF*

Residual Error Integral Estimator

$$\hat{L}_r(x, \omega_o, \hat{\boldsymbol{\theta}}) \approx \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{(L_i(x, \omega_i) - NN_i(x, \omega_i, \hat{\boldsymbol{\theta}})) f_r(x, \omega_i, \omega_o) \cos\theta_i}{p(\omega_i)}$$

Final **unbiased** estimator: $\hat{L}_o(x, \omega_o, \hat{\boldsymbol{\theta}}) = L_e(x, \omega_o) + \hat{L}_c(x, \omega_o, \hat{\boldsymbol{\theta}}) + \hat{L}_r(x, \omega_o, \hat{\boldsymbol{\theta}})$



Variance of Two-Level Monte Carlo

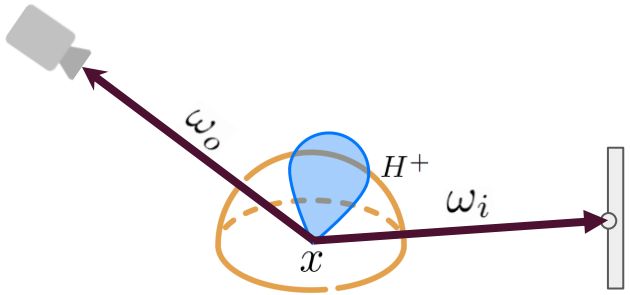
Project Page

Cache Error

$$V(\langle \hat{L}_o \rangle) = V(\langle \hat{L}_c \rangle) + V(\langle \hat{L}_r \rangle) =$$

$$V(\langle \hat{L}_c \rangle) + (\hat{L}_o(x, \omega_o) - L_e(x, \omega_o) - \hat{L}_c(x, \omega_o, \hat{\boldsymbol{\theta}}))^2$$

Final **unbiased** estimator: $\hat{L}_o(x, \omega_o, \hat{\boldsymbol{\theta}}) = L_e(x, \omega_o) + \hat{L}_c(x, \omega_o, \hat{\boldsymbol{\theta}}) + \hat{L}_r(x, \omega_o, \hat{\boldsymbol{\theta}})$



Variance of Two-Level Monte Carlo

Project Page

Cache *Error*

$$V(\langle \hat{L}_o \rangle) = V(\langle \hat{L}_c \rangle) + V(\langle \hat{L}_r \rangle) =$$

$$V(\langle \hat{L}_c \rangle) + (\hat{L}_o(x, \omega_o) - L_e(x, \omega_o) - \hat{L}_c(x, \omega_o, \hat{\boldsymbol{\theta}}))^2$$

The variance stays the same…

Variance of Two-Level Monte Carlo

Project Page

Cache *Error*

$$V(\langle \hat{L}_o \rangle) = V(\langle \hat{L}_c \rangle) + V(\langle \hat{L}_r \rangle) =$$

$$V(\langle \hat{L}_c \rangle) + (\hat{L}_o(x, \omega_o) - L_e(x, \omega_o) - \hat{L}_c(x, \omega_o, \hat{\boldsymbol{\theta}}))^2$$

expensive but vanishes if our cache is good

cheaper with NIRC

The variance stays the same…

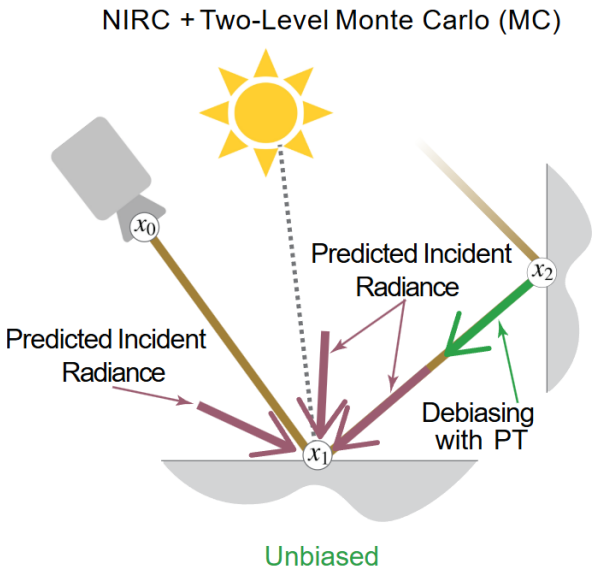
but we can allocate **more cheap samples** to estimate $\hat{L}_c$

Results

Neural Incident Radiance Cache (NIRC)

Project Page

1. Can be used to predict incident radiance or the visibility term

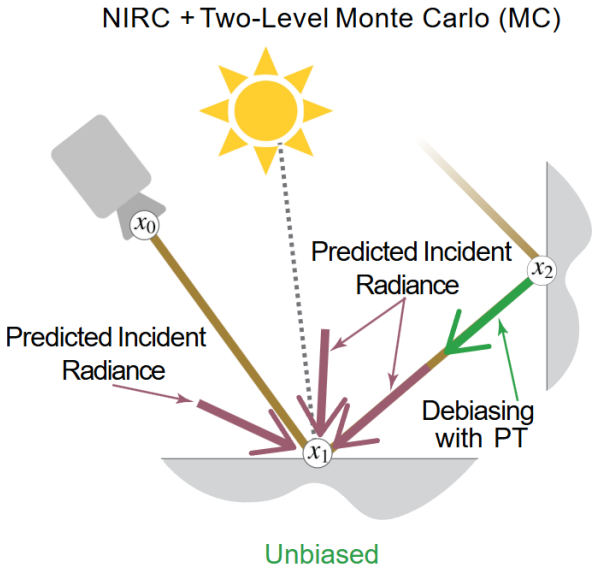


Neural Incident Radiance Cache (NIRC)

Project Page

1. Can be used to predict incident radiance or the visibility term
2. Can be efficiently sampled w/o additional ray tracing and iNGP is amortized

For 1920x1080 neural samples, RTX 4080:
NIRC: ~ 0.5-1.5ms
NRC: ~ 5-10ms (with tracing costs)

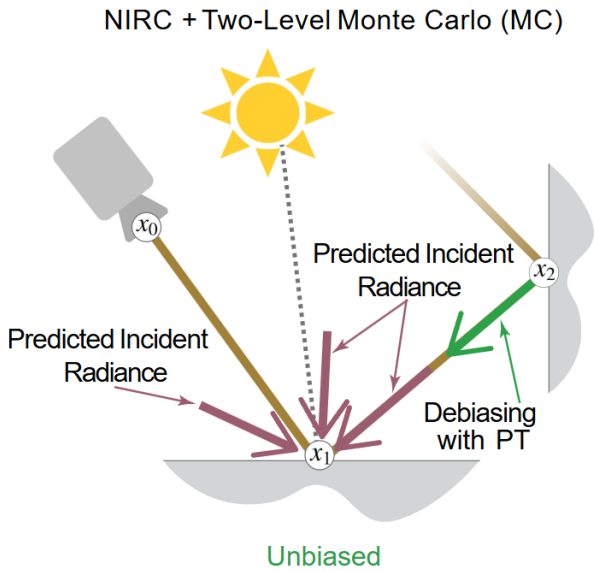


Neural Incident Radiance Cache (NIRC)

Project Page

1. Can be used to predict incident radiance or the visibility term
2. Can be efficiently sampled w/o additional ray tracing and iNGP is amortized
3. Decrease variance for unbiased estimator

For 1920x1080 neural samples, RTX 4080:
NIRC: ~ 0.5-1.5ms
NRC: ~ 5-10ms (with tracing costs)

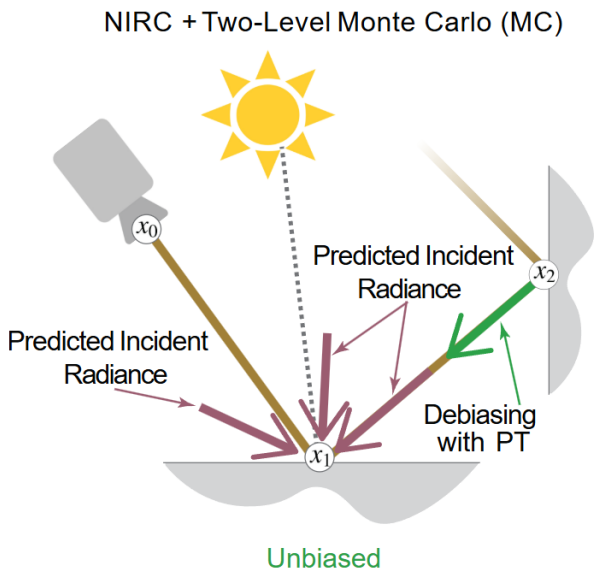


Neural Incident Radiance Cache (NIRC)

Project Page

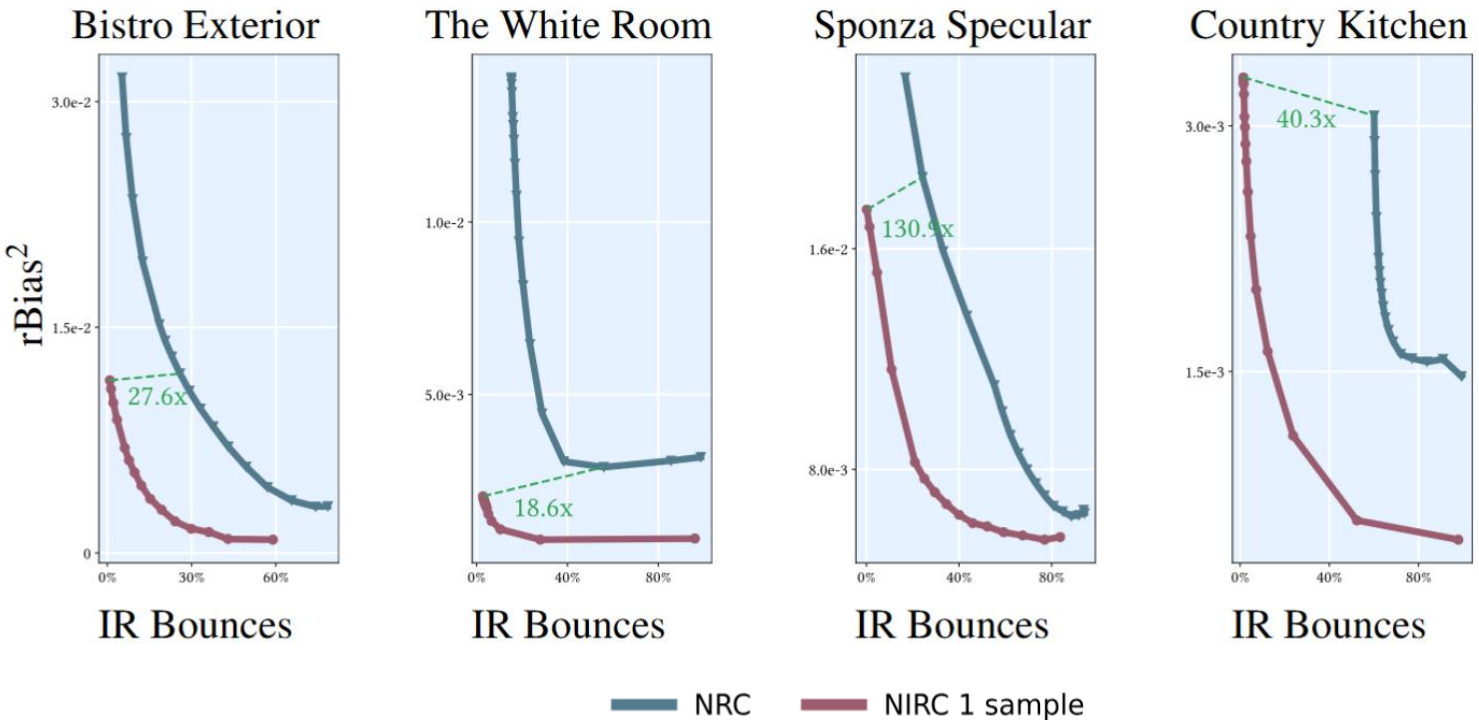
1. Can be used to predict incident radiance or the visibility term
2. Can be efficiently sampled w/o additional ray tracing and iNGP is amortized
3. Decrease variance for unbiased estimator
4. Biased estimator can save some indirect bounces!

For 1920x1080 neural samples, RTX 4080:
NIRC: ~ 0.5-1.5ms
NRC: ~ 5-10ms (with tracing costs)

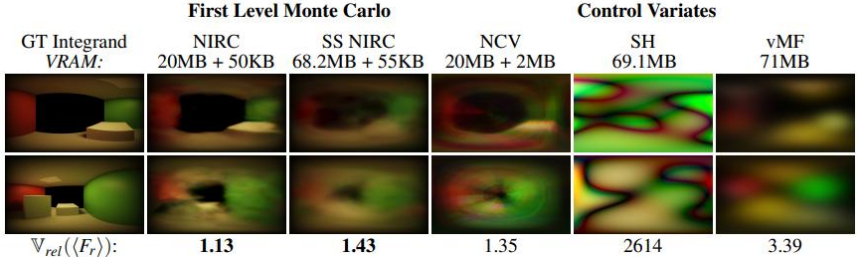


Saving on bounces: Bias vs Indirect Bounces

Project Page

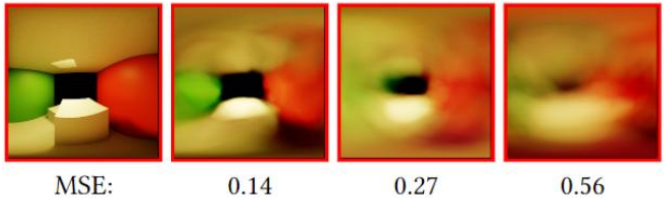


A lot more in the paper/project page:



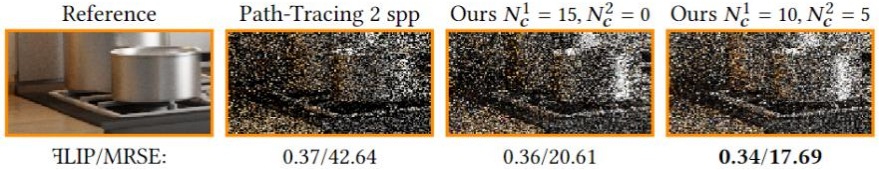
(Neural) Control Variates vs Two-Level MC

Neural Visibility Cache (**NVC**)



NIRC with **LODs** for the
Multi-Level MC

The distribution of Neural Samples in a light path



Please check out our paper

Thanks for attention!

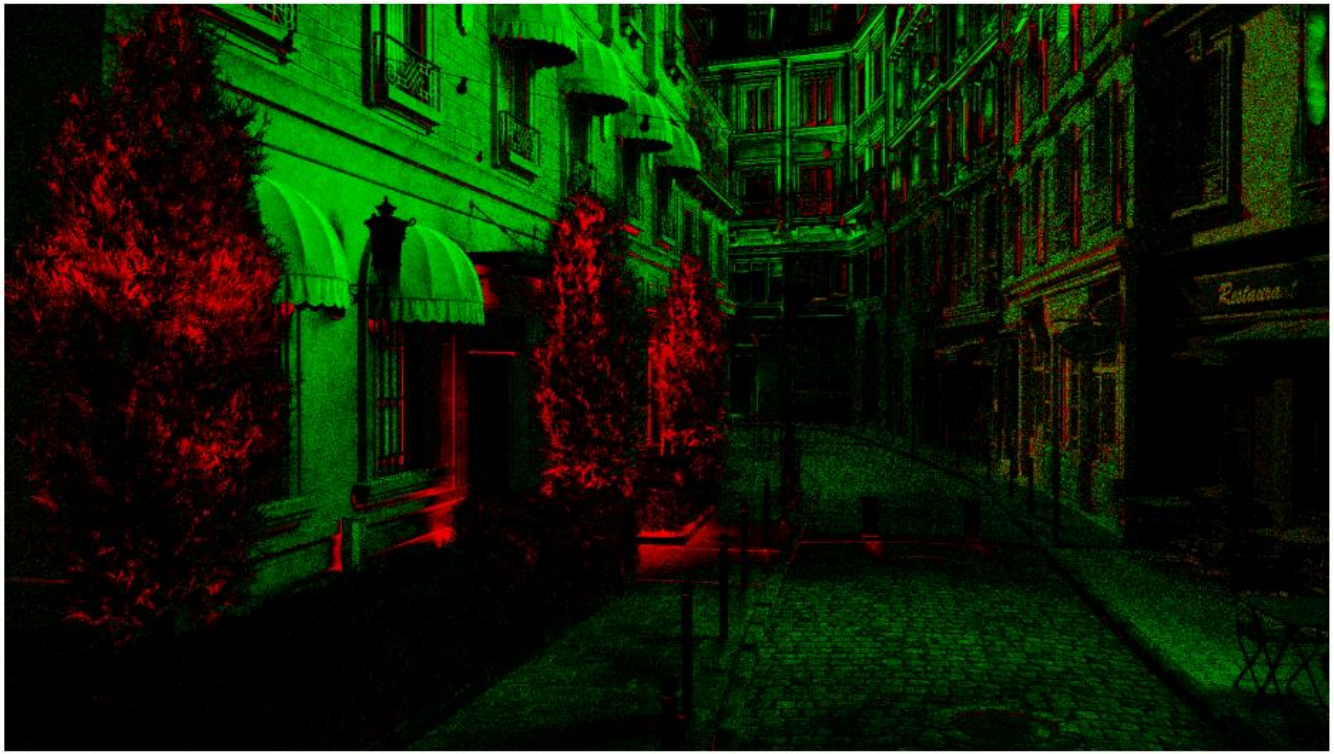
Biased NIRC\NRC\PT 1spp: ~35-40ms
Unbiased NIRC: 65ms

Please check out our paper

Limitations

NIRC is often not accurate enough for metallic surfaces

Limitations



Lower\Higher Variance after application of
2-Level MC estimator with the NIRC

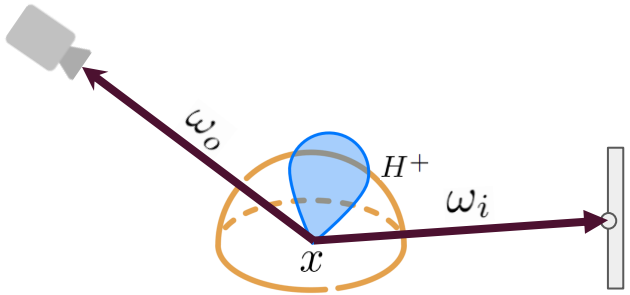
Similarities with Control Variates (CV)

CVs **limits the mathematical basis** which could be used for approximating the integrand in each point (SHs, vMFs, Normalizing flows)
[Pantaleoni 2020][Müller et. al. 2020]

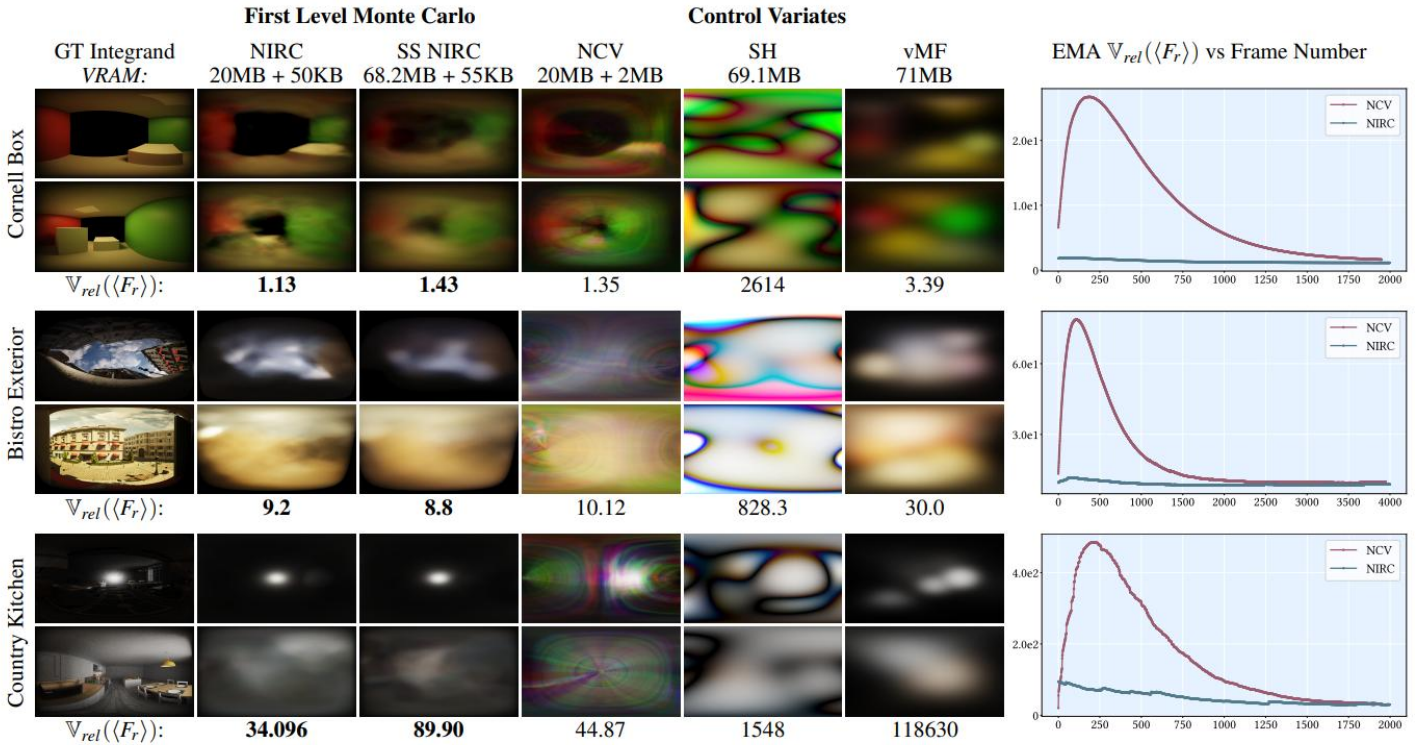
We **can not use** expressive MLPs with nonlinearities for CVs directly :(

analytically calculated with CVs

Final **unbiased** estimator: $\hat{L}_o(x, \omega_o, \hat{\boldsymbol{\theta}}) \approx L_e(x, \omega_o) + \overbrace{\hat{L}_c(x, \omega_o, \hat{\boldsymbol{\theta}})} + \hat{L}_r(x, \omega_o, \hat{\boldsymbol{\theta}})$



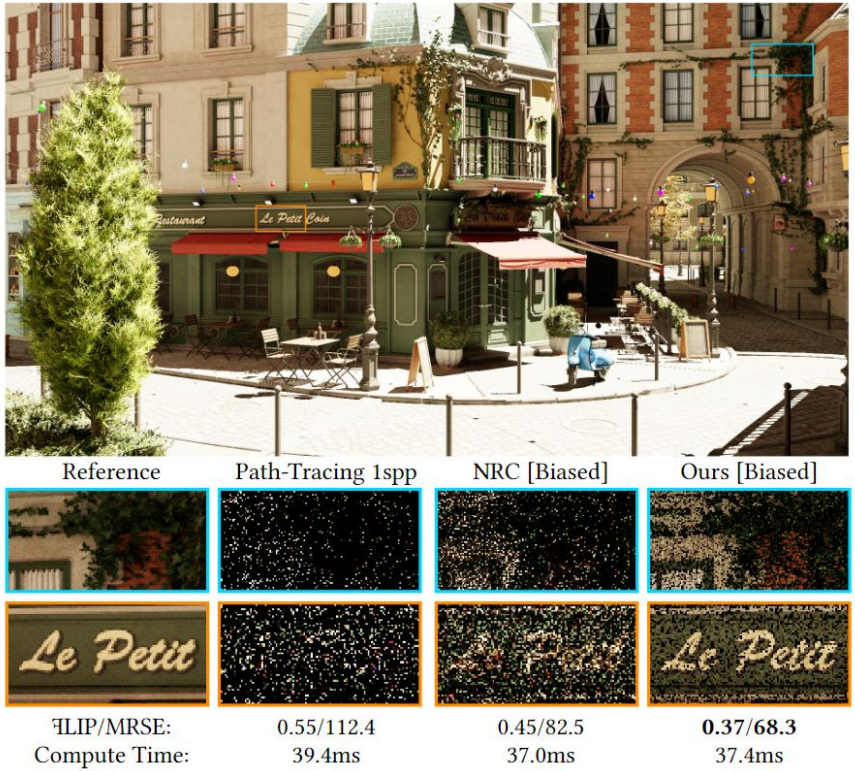
Comparisons with Control Variates (CVs)



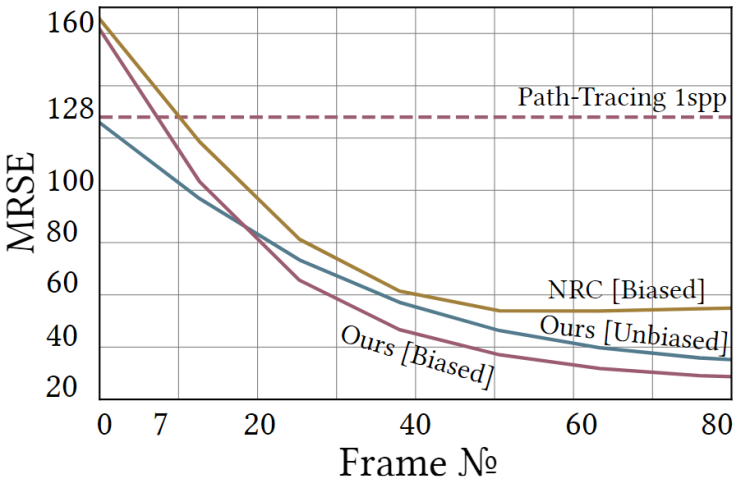
analytically calculated with CVs

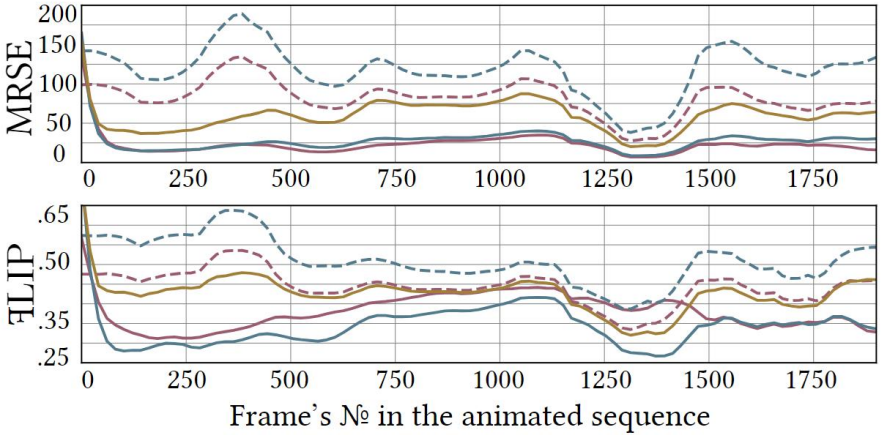
$$\hat{L}_o(x, \omega_o, \hat{\boldsymbol{\theta}}) \approx L_e(x, \omega_o) + \overbrace{\hat{L}_c(x, \omega_o, \hat{\boldsymbol{\theta}})} + \hat{L}_r(x, \omega_o, \hat{\boldsymbol{\theta}})$$

Without the sky illumination

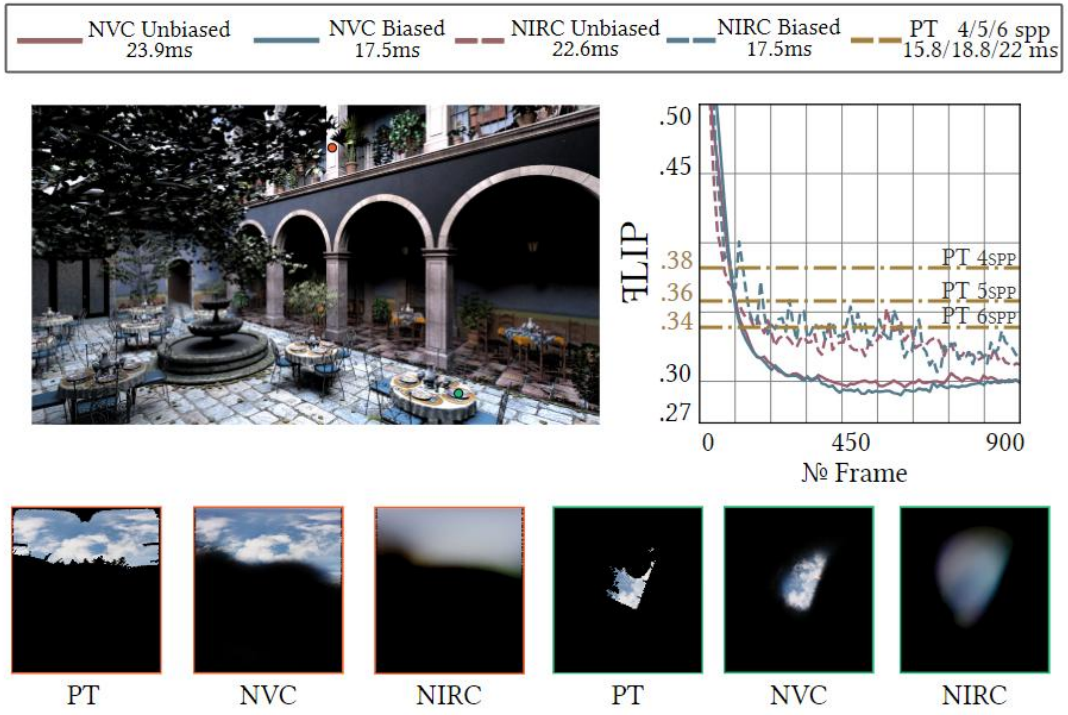


NIRC Convergence

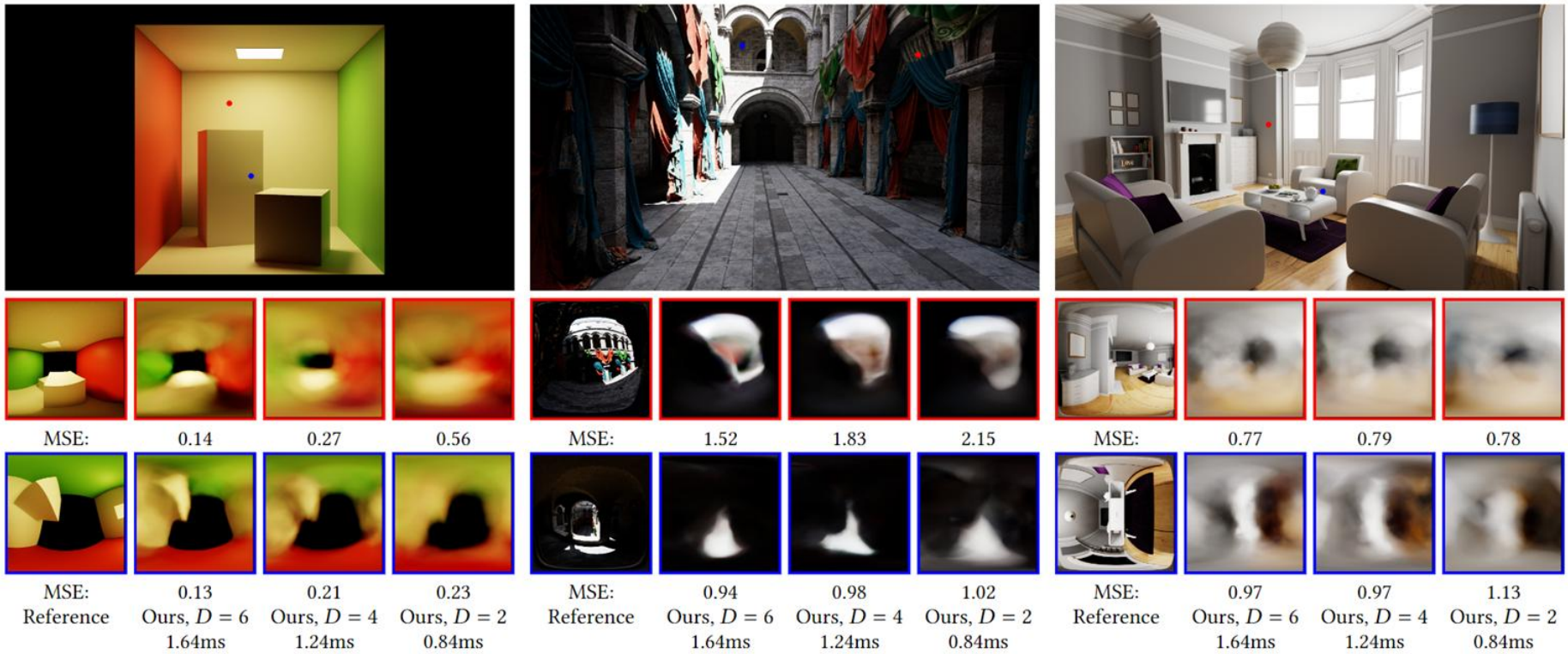




Neural Visibility Cache



LODs of NIRC


MSE: 0.14 0.27 0.56
MSE: 1.52 1.83 2.15
MSE: 0.77 0.79 0.78
MSE: Reference
0.13 Ours, $D = 6$ 1.64ms
0.21 Ours, $D = 4$ 1.24ms
0.23 Ours, $D = 2$ 0.84ms
MSE: Reference
0.94 Ours, $D = 6$ 1.64ms
0.98 Ours, $D = 4$ 1.24ms
1.02 Ours, $D = 2$ 0.84ms
MSE: Reference
0.97 Ours, $D = 6$ 1.64ms
0.97 Ours, $D = 4$ 1.24ms
1.13 Ours, $D = 2$ 0.84ms

Distribution of Neural Samples per light path

Figure 7: We explore the effect of redistribution of the number of neural samples between the first and second vertex of a light path, denoted as N_c^1 and N_c^2, respectively. The experiments show the Sponza and Country Kitchen scenes rendered with up to 5 vertices in a light path without Russian roulette in order to match the render time. We observe that the optimal allocation of neural samples between the first and second vertex is highly scene-dependent, influenced by, for example, material properties and illumination.