<Recent Advances in Rendering>
Monte Carlo Noise Reduction

CS380 – Introduction to Computer Graphics

TA: Kyu Beom Han

qbhan@kaist.ac.kr

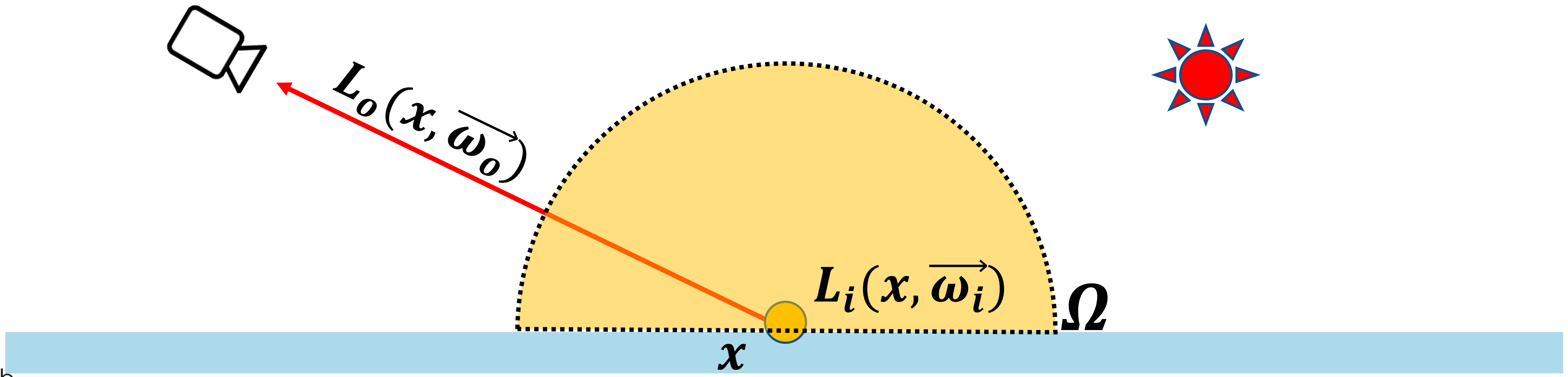
SGVR Lab KAIST

Today's Content

- **Reviews on Monte Carlo(MC) ray tracing and MC noise**
- Path-space MC noise reduction
- Image-space MC noise reduction
- Learning-based MC noise reduction

Review - Rendering Equation

$$\underbrace{L_o(x,\overrightarrow{\omega_o})}_{\text{Outgoing Radiance}} = \underbrace{L_e(x,\overrightarrow{\omega_o})}_{\text{Emitting Radiance}} + \int_{\Omega} \underbrace{f_r(x,\overrightarrow{\omega_i},\overrightarrow{\omega_o})}_{\text{Material Property (e.g., BRDF)}} \underbrace{L_i(x,\overrightarrow{\omega_i})}_{\text{Incoming Radiance}} (\overrightarrow{\omega_i}\cdot\vec{n})\,d\overrightarrow{\omega_i}$$



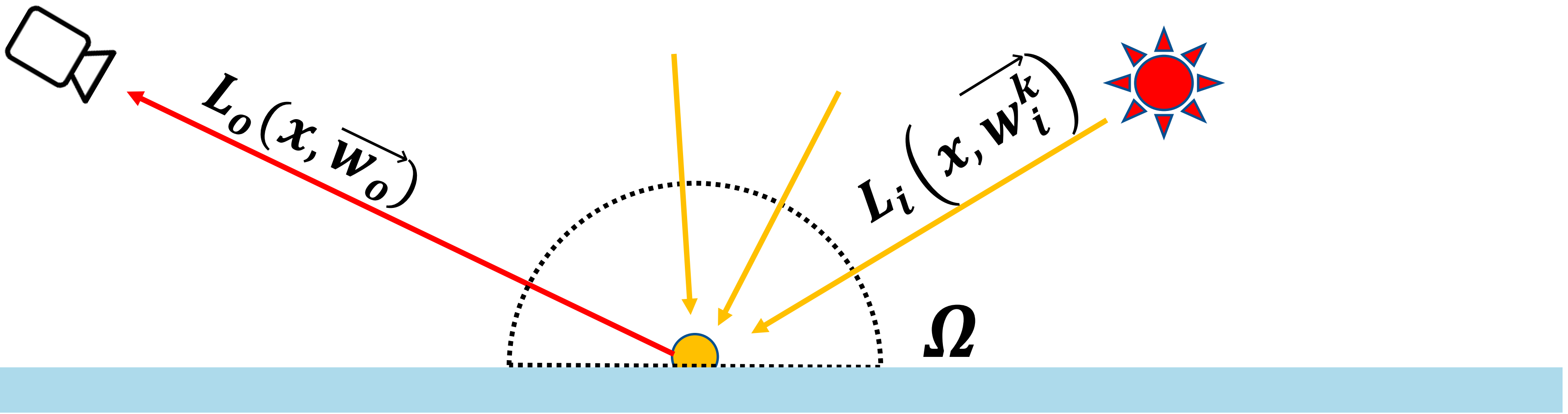
SGVR Lab
KAIST

Review – MC Ray Tracing

- For fast convergence, we need to…
  - Shoot more samples (Large N)
  - Find a good pdf $p\left(\overrightarrow{w_i^k}\right) \sim f_r\left(x, \overrightarrow{w_i^k}, \overrightarrow{w_o}\right) L_i\left(x, \overrightarrow{w_i^k}\right)\left(\overrightarrow{w_i^k} \cdot \vec{n}\right)$

$$L_o(x, \overrightarrow{w_o}) = L_e(x, \overrightarrow{w_o}) + \int_\Omega f_r(x, \overrightarrow{w_i}, \overrightarrow{w_o}) L_i(x, \overrightarrow{w_i})(\overrightarrow{w_i} \cdot \vec{n})\, d\overrightarrow{w_i}$$

$$\sim L_e(x, \overrightarrow{w_o}) + \frac{1}{N} \sum_{k=1}^{N} \frac{f_r\left(x, \overrightarrow{w_i^k}, \overrightarrow{w_o}\right) L_i\left(x, \overrightarrow{w_i^k}\right)\left(\overrightarrow{w_i^k} \cdot \vec{n}\right)}{p\left(\overrightarrow{w_i^k}\right)}$$

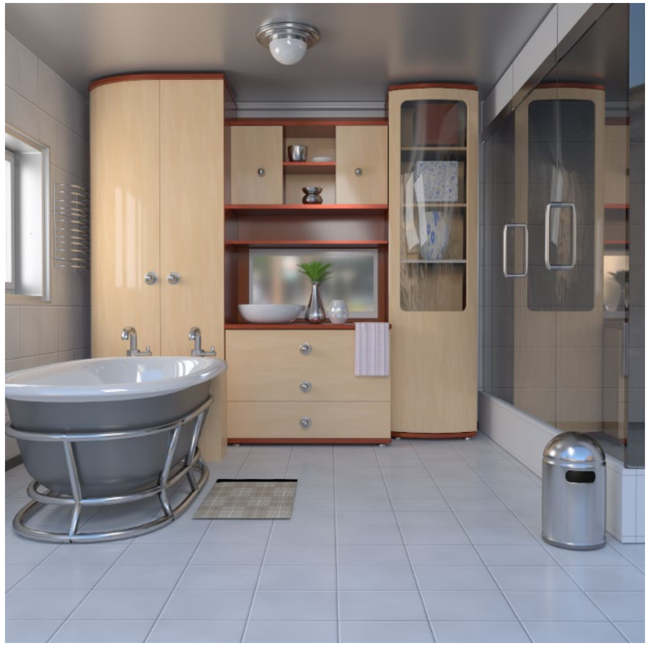


Review – MC Ray Tracing and MC Noise

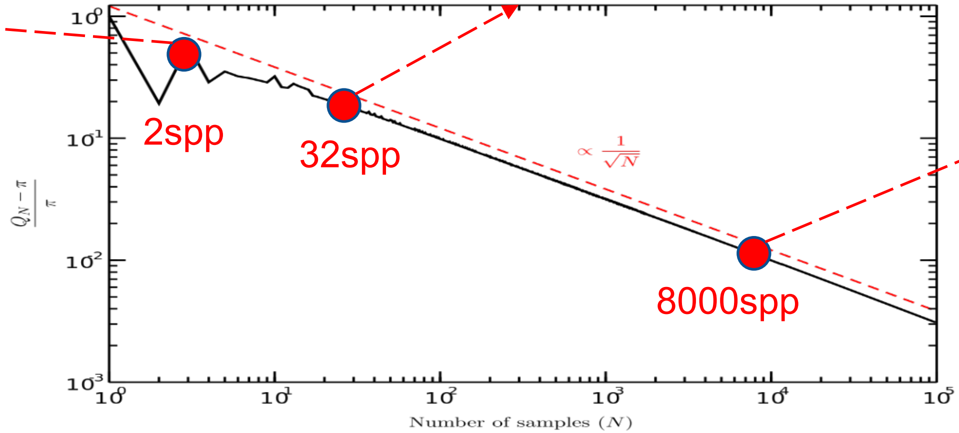
- Shooting few samples per pixel (spp) leads to noisy radiance estimation

2.5s

8.9s



18m

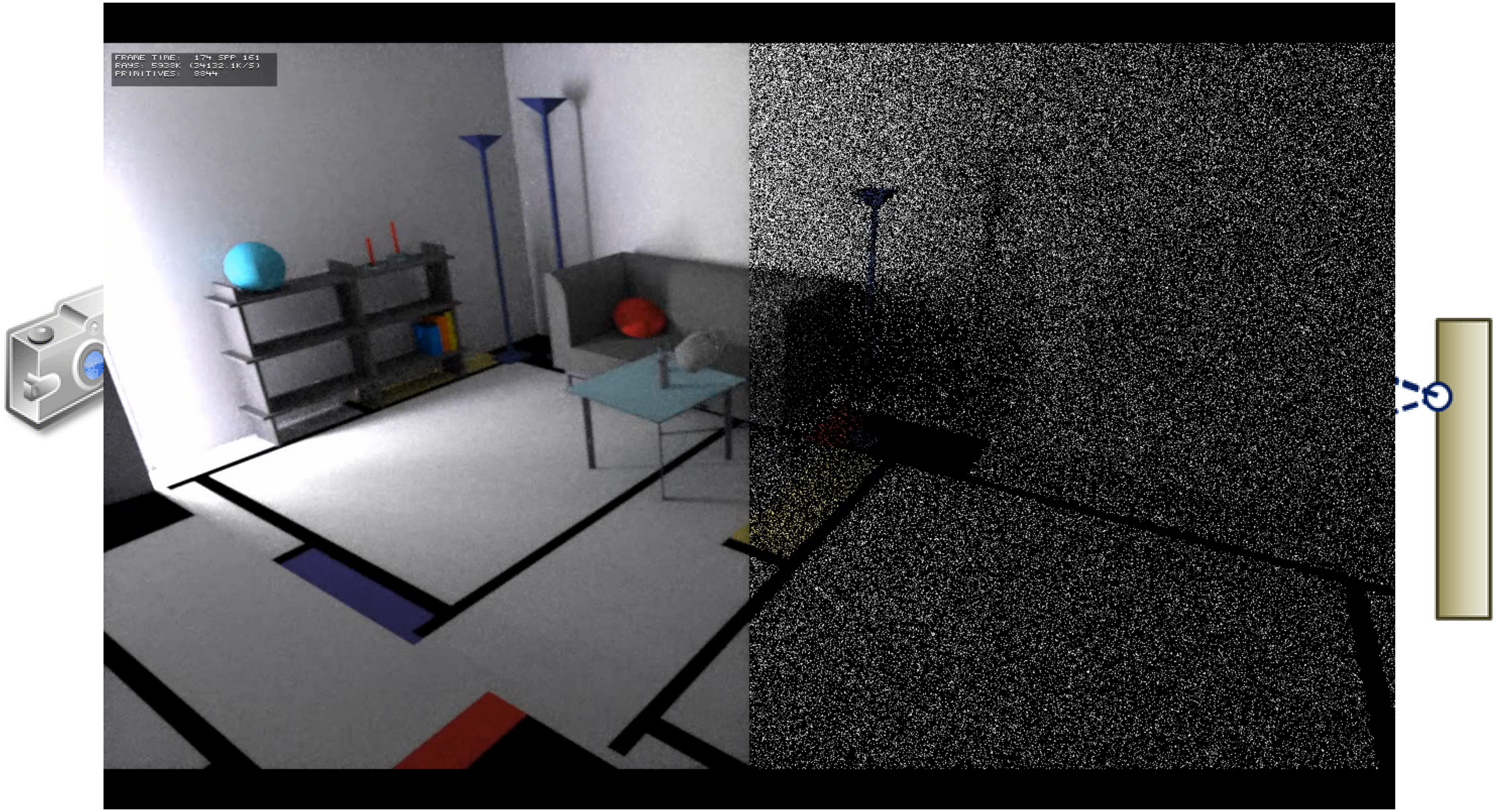


Content

- Reviews on Monte Carlo(MC) ray tracing and MC noise
- **Path-space MC noise reduction**
- Image-space MC noise reduction
- Learning-based MC noise reduction

Review - Metropolis Light Transport (MLT)

- Using advanced sampling technique (Metropolis-Hasting algorithm) to generate valid (important) samples.
- Beneficial for scenes with complex geometry and indirect lighting.

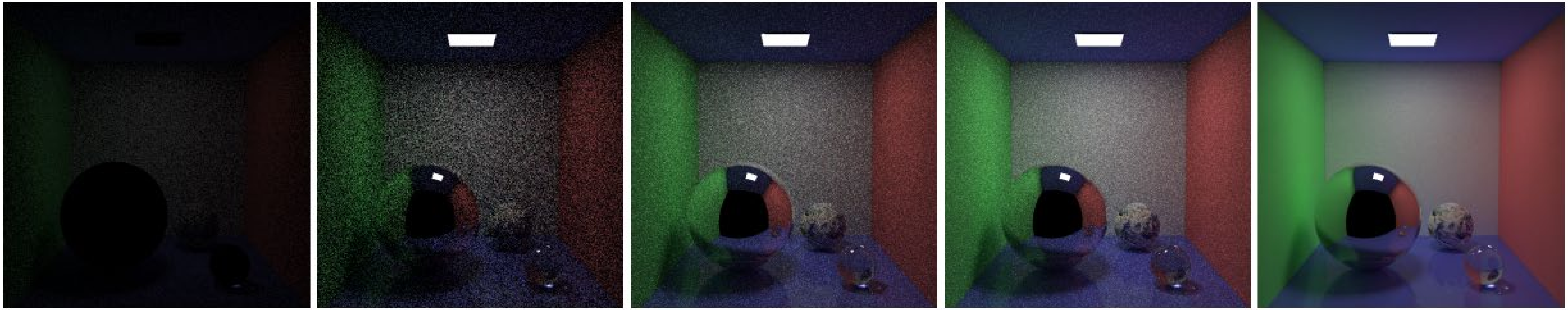


Ray Tracey's blog: Real-time Metropolis Light Transport on the GPU: it works!!!!

Physically based Computer Graphics for Realistic Image Formation to Simulate Optical Measurement Systems, Retzlaff et al., JSSS 2017

Review - Bidirectional Path Tracing (BDPT)

- Combining rays traced from the camera and light sources
- Beneficial for scenes with complex geometry and indirect lighting



(a) Light tracing (b) Path tracing (c) PT with DI (d) BDPT (e) BDPT with MIS

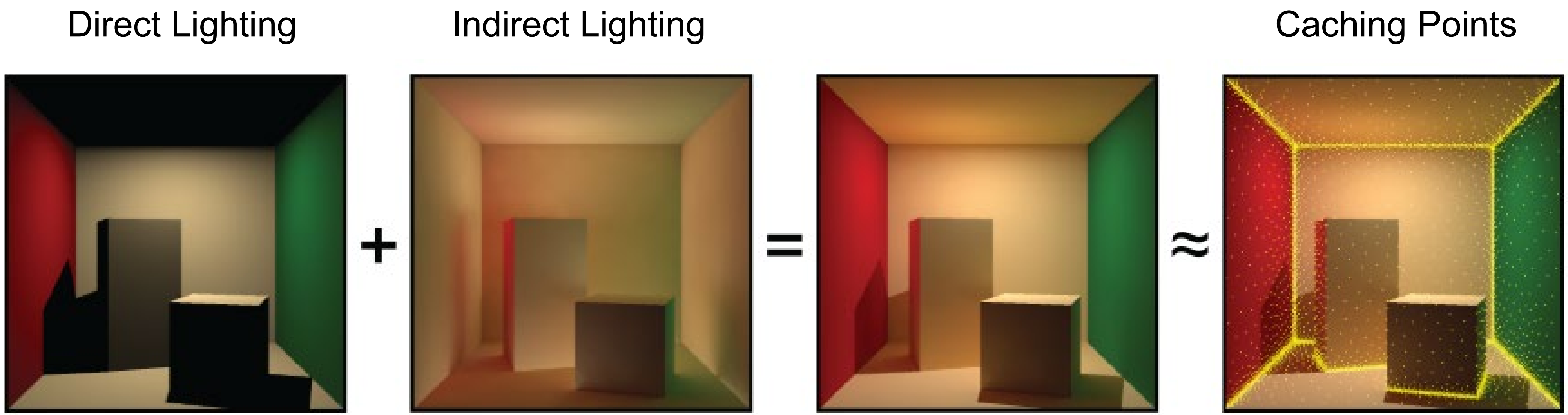
eye point

light source

Bidirectional Path Tracing, Michal Vlnas, Proceedings of CESCG 2018

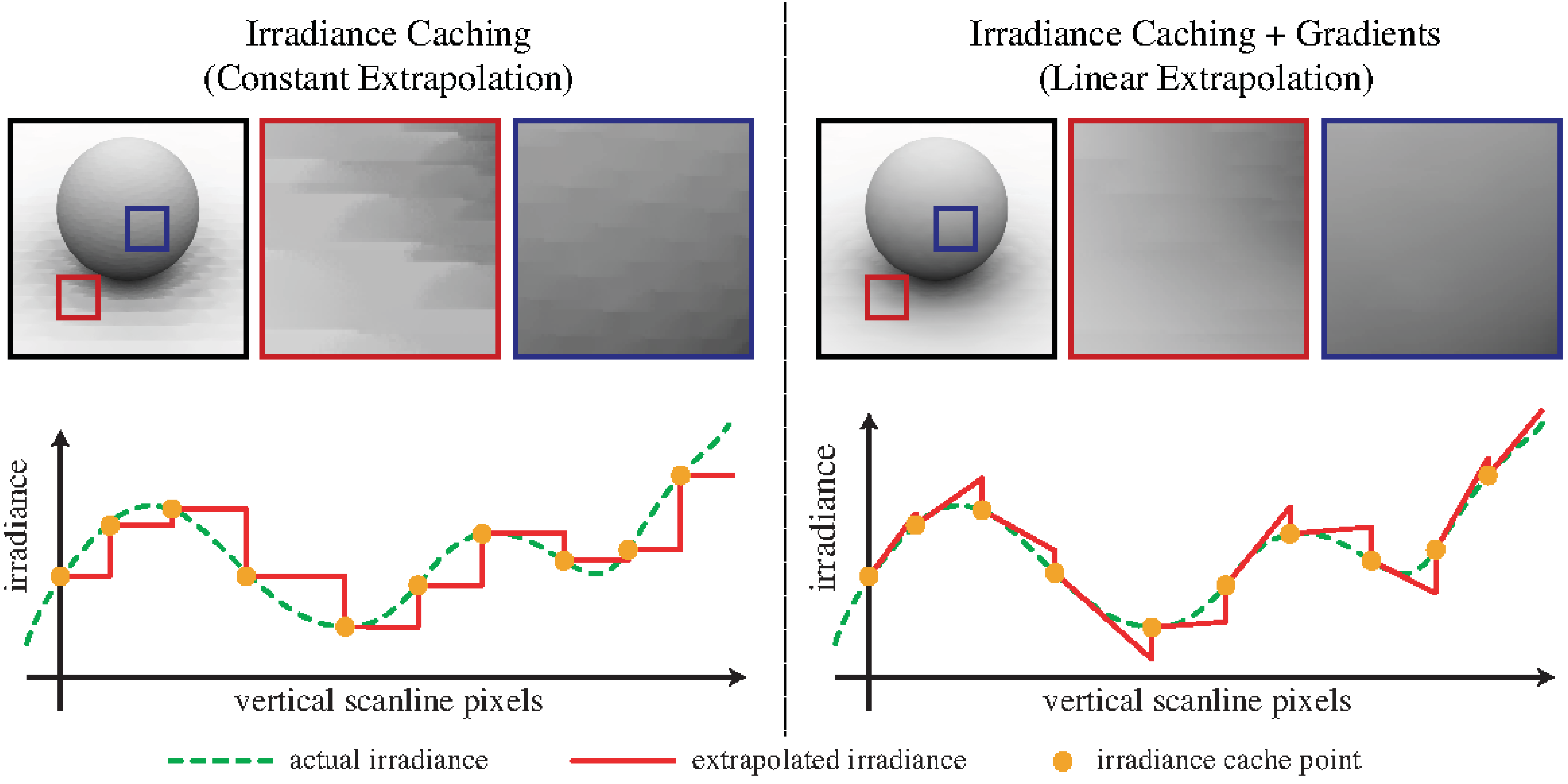
Review - Irradiance Caching

- Caching irradiance (and its gradient) of the points visible from camera
- Intuition: Indirect lighting is mostly smooth → Sparse computation is enough



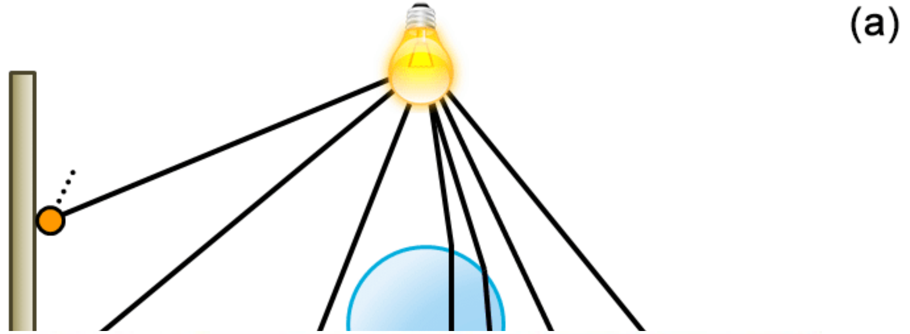
Review - Irradiance Caching

- Caching irradiance (and its gradient) of the points visible from camera
- Intuition: Indirect lighting is mostly smooth → Sparse computation is enough



Review - Photon Mapping

- Shoot photons from the light source and save information (energy, position, direction, etc.) (a)
- Use K-nearest photons for estimating the radiance of the query point (b)



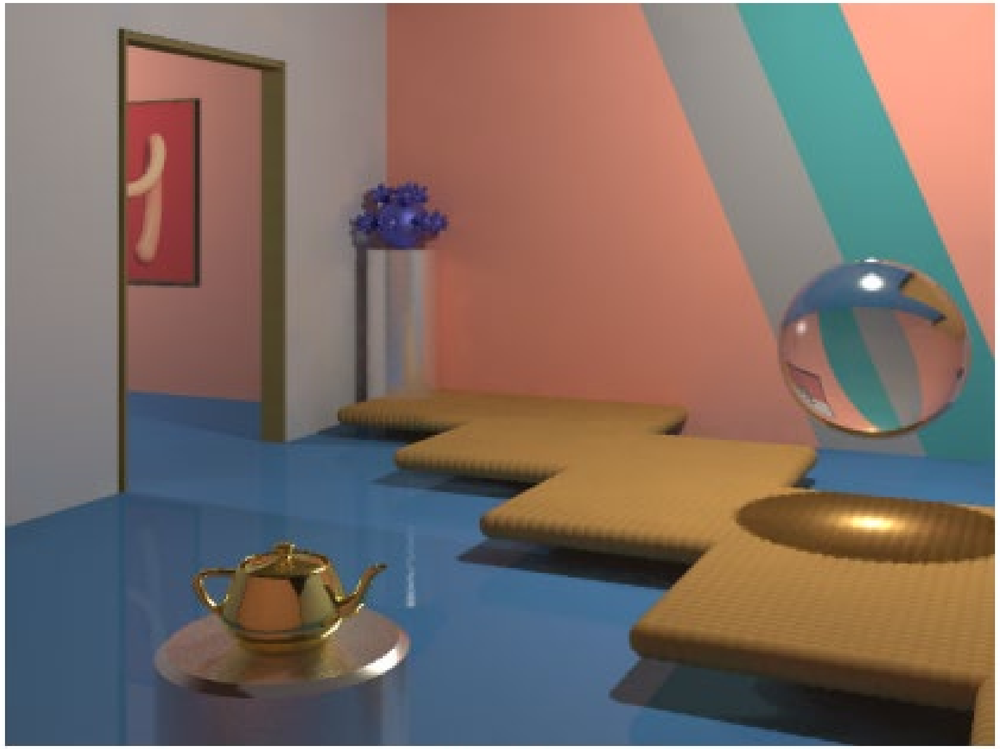
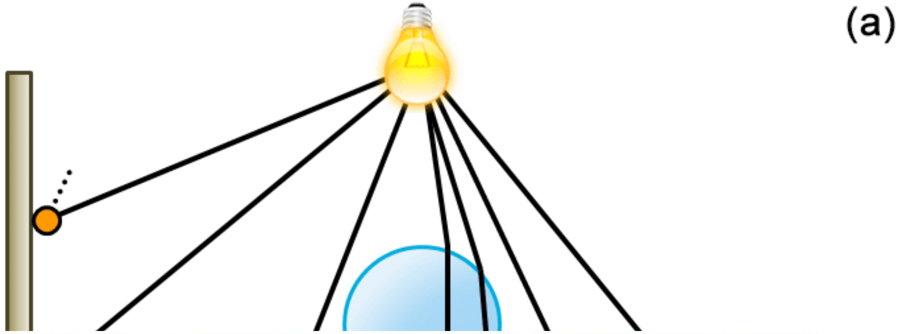


Figure 3: The Museum scene

Figure 4: Direct visualization of the global photon map in the Museum scene

Review - Photon Mapping

- Shoot photons from the light source and save information (energy, position, direction, etc.) (a)
- Use K-nearest photons for estimating the radiance of the query point (b)



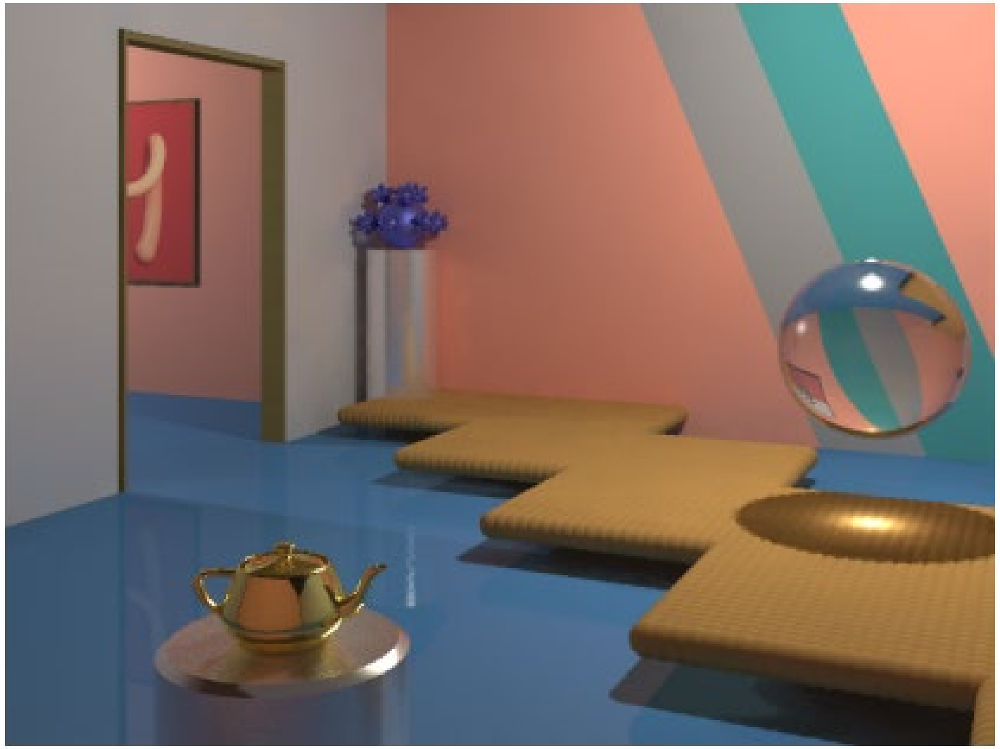
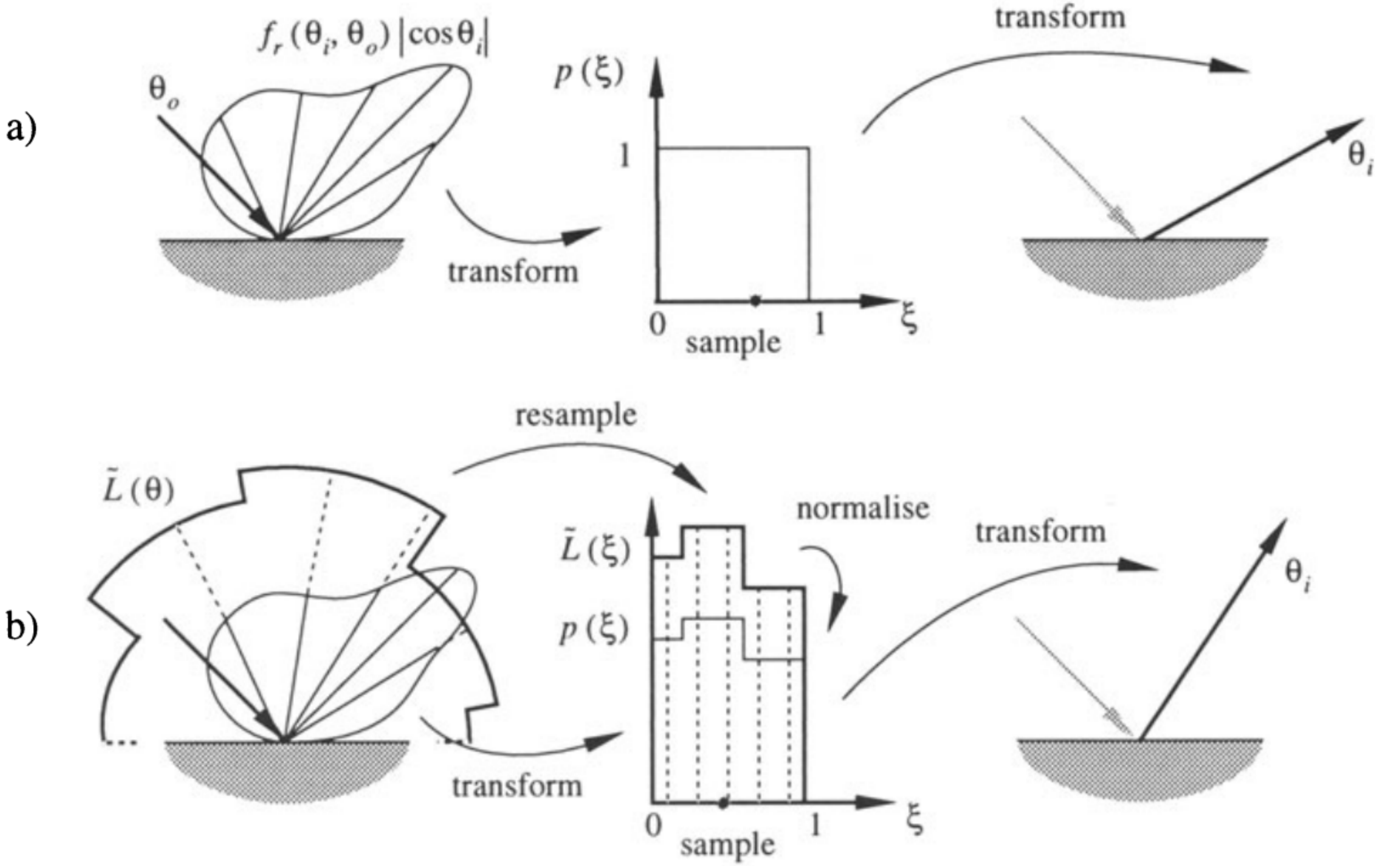


Figure 3: The Museum scene

Figure 4: Direct visualization of the global photon map in the Museum scene

Path Guiding

- Guiding samples to certain position or direction
- Giving higher probability to position/direction with higher radiance (or any other metric)



Finding a good PDF!

Importance Driven Path Tracing using the Photon Map, Jensen, EG Rendering Techniques 1995
A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing, Lafortune et al., EG Rendering Techniques 1995

Path Guiding

- PDFs stored on various grid- or tree-like structures
  - PDF as 2D map (θ, ϕ)

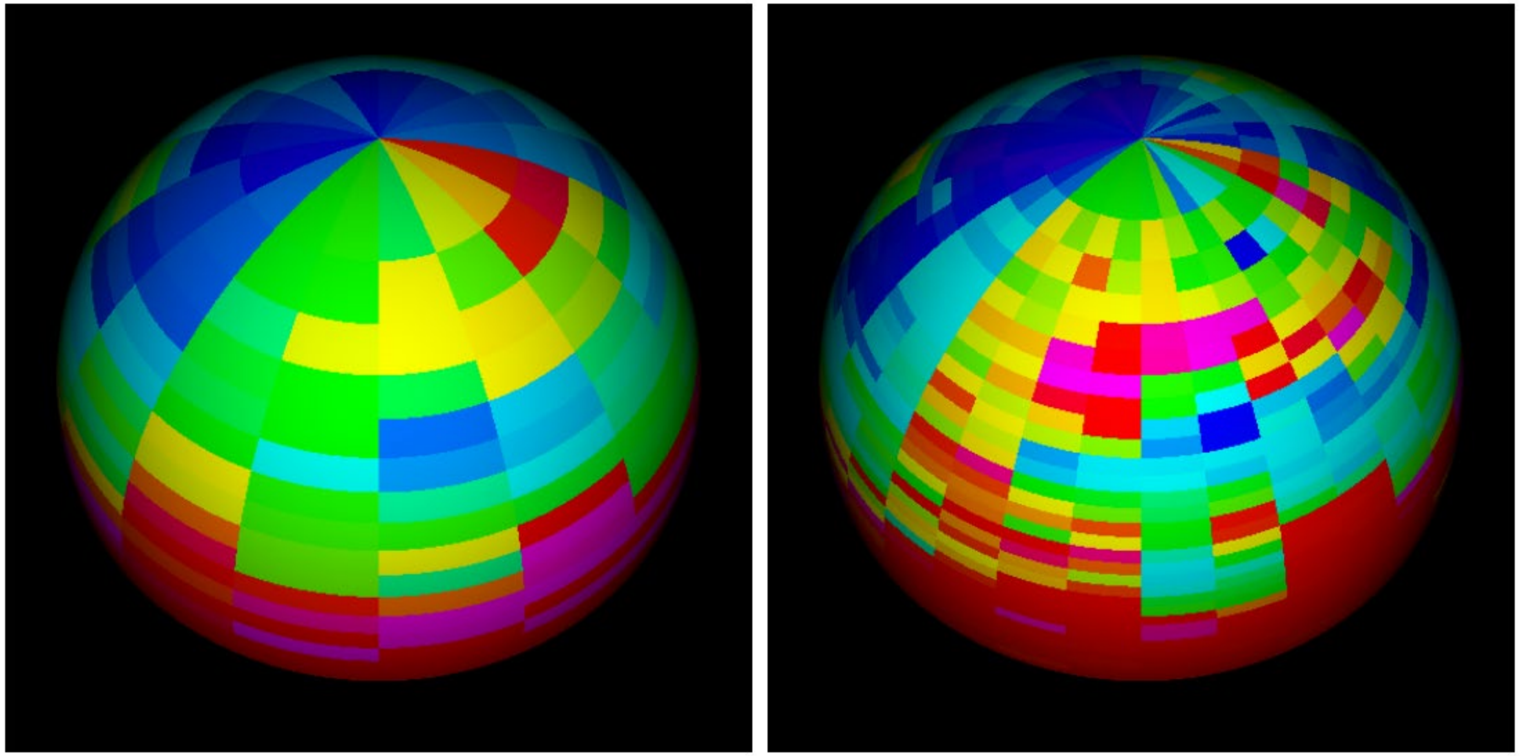
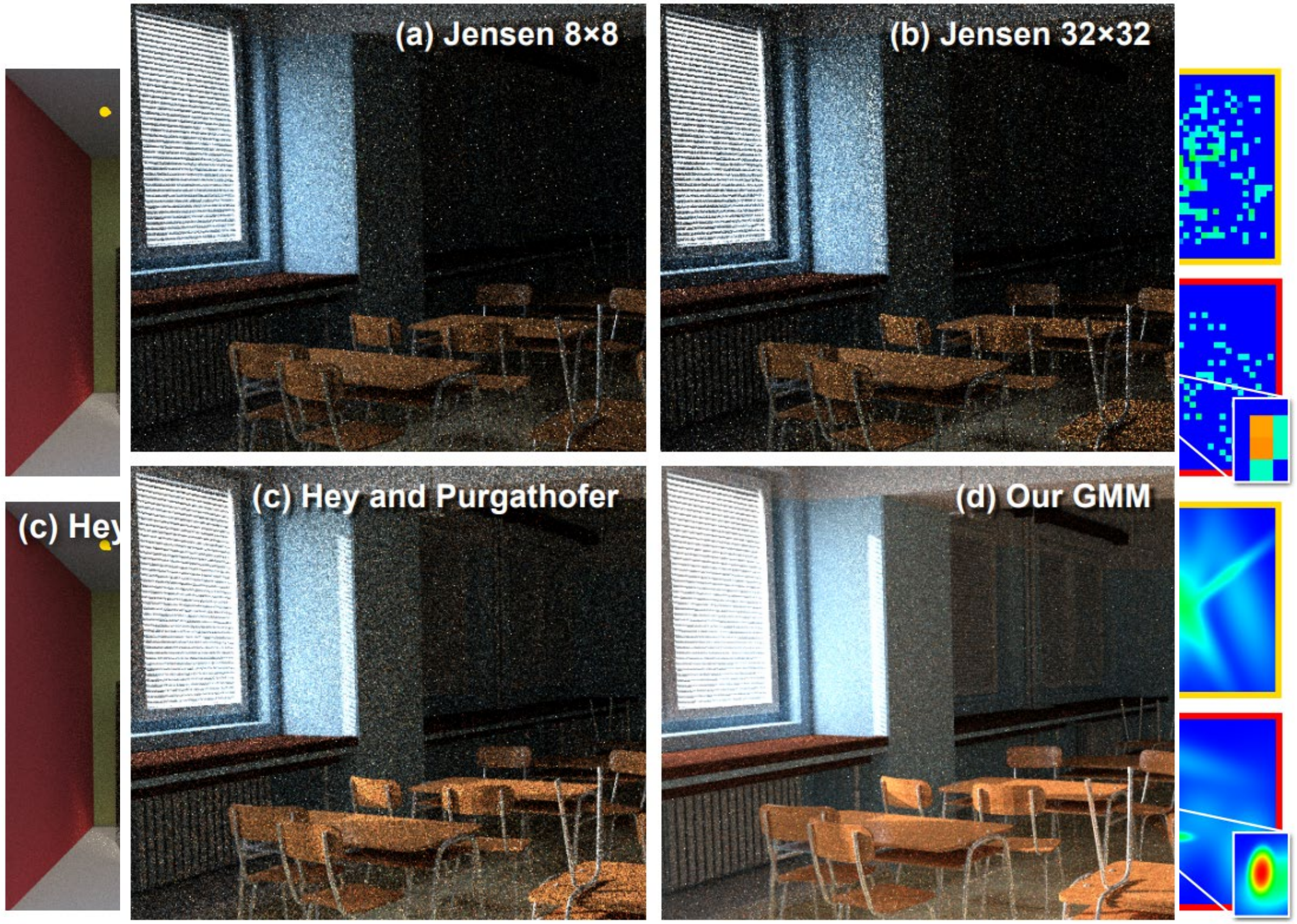


Figure 2: Color-mapped scalar values of predicate p at 2 different refinement stages.

Towards Interactive Global Illumination Effects via Sequential Monte Carlo Adaptation, Pegoraro et al., Symp. Inter. RT 2008

Path Guiding

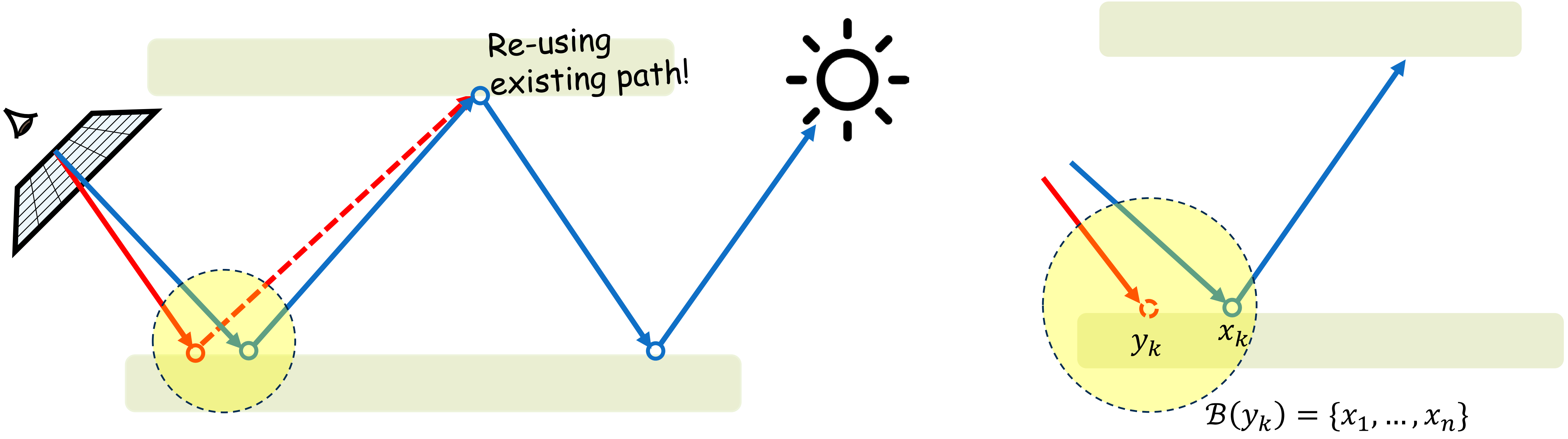
- PDFs stored on various grid- or tree-like structures
  - PDF as 2D map (θ, ϕ)
  - Advanced structures (e.g., mixture models) also possible



On-line Learning of Parametric Mixture Models for Light Transport Simulation, Vorba et al., ToG 2014

Path-space Filtering (or Path Reuse)

- Estimating an appropriate denoising filter (kernel) to be applied on each bounce of samples
  - $\bar{L}(y_k) = \sum_{j=1}^{n} L(x_j) \cdot \frac{w(x_j)}{p(x_j)}, x_i \in B(y_k)$
  - $\bar{L}(\cdot)$: prefiltered radiance, $L(\cdot)$: prefiltered radiance, $p(\cdot)$: probability of sampled vertex, w$(\cdot)$: weight calculated by balance heuristic



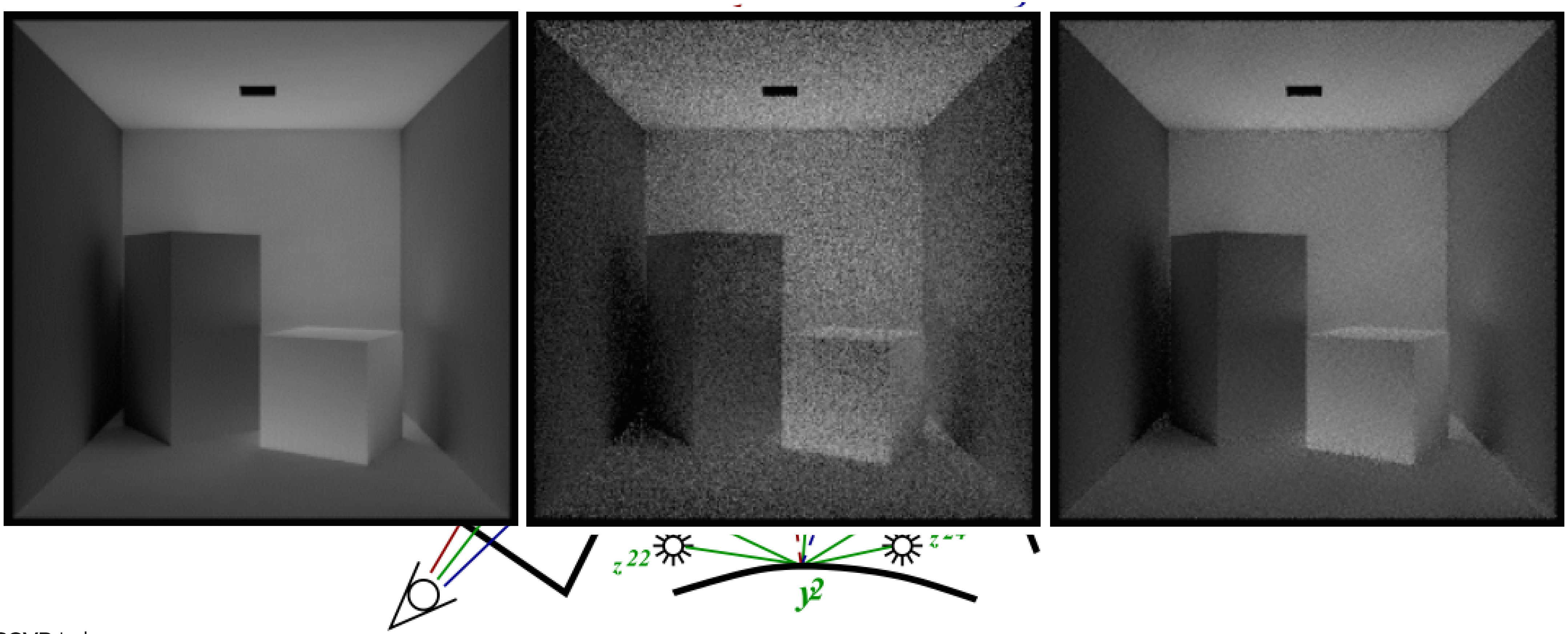
SGVR Lab
KAIST

Accelerating Path Tracing by Re-Using Paths, Bekaert et al., Eurographics Workshop on Rendering 2002

Path-space Filtering (or Path Reuse)

- Estimating an appropriate denoising filter (kernel) to be applied on each bounce of samples
- Can involve various indirect illumination (dashed lines)

Clean Image | 15 spp w/o reuse | 15 spp w/ reuse



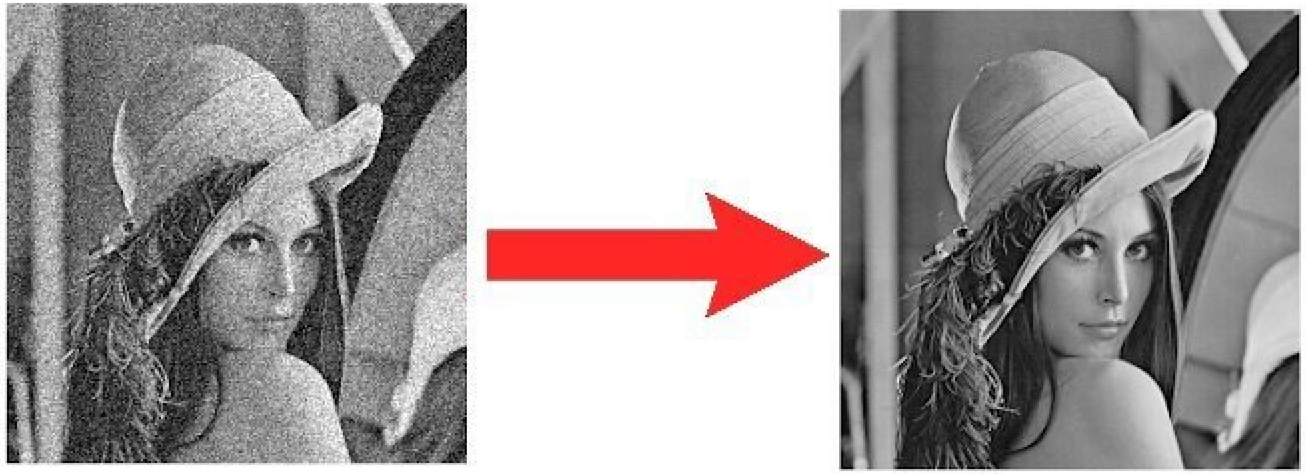
Accelerating Path Tracing by Re-Using Paths, Bekaert et al., Eurographics Workshop on Rendering 2002

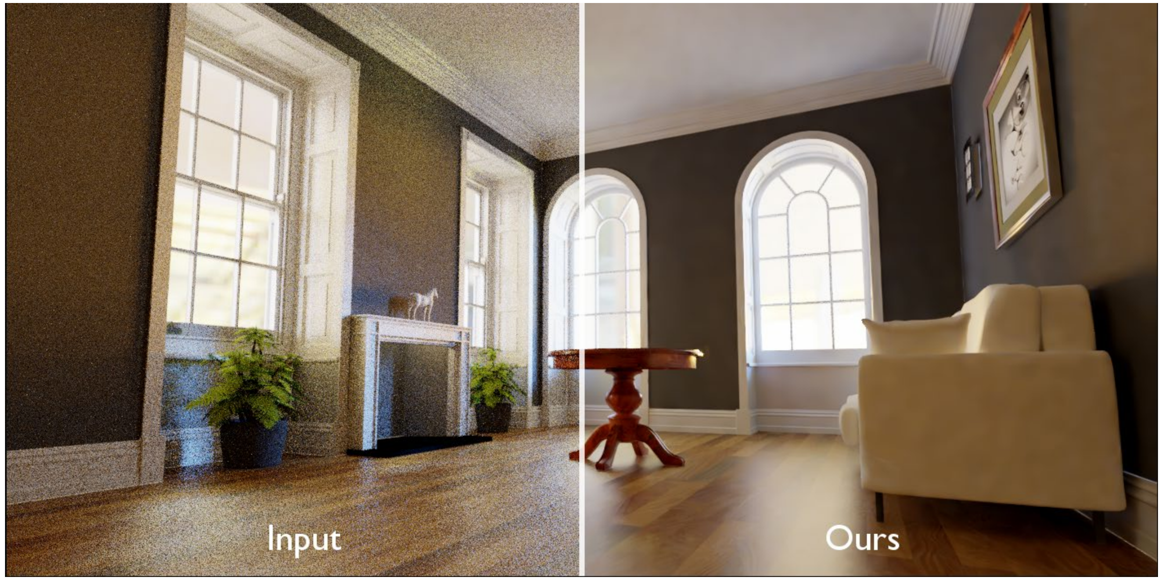
Content

- Reviews on Monte Carlo(MC) ray tracing and MC noise
- Path-space MC noise reduction
- **Image-space MC noise reduction**
- Learning-based MC noise reduction

Image-space MC Noise Reduction

- Efficiently dealing noise on image-space, similar to general image denoising
- Reducing working space from N-dim path-space to 2-dim image space

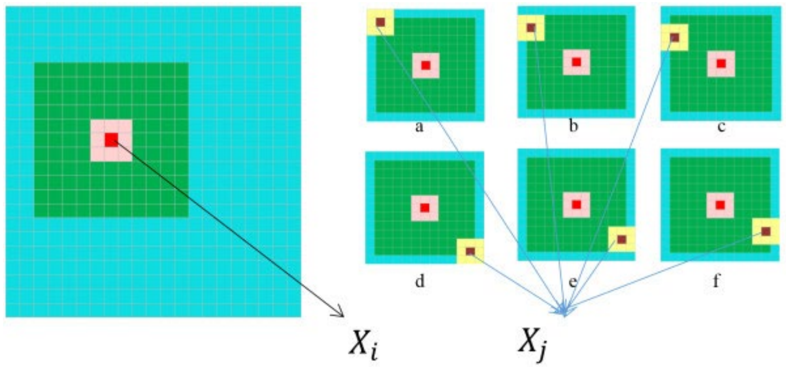




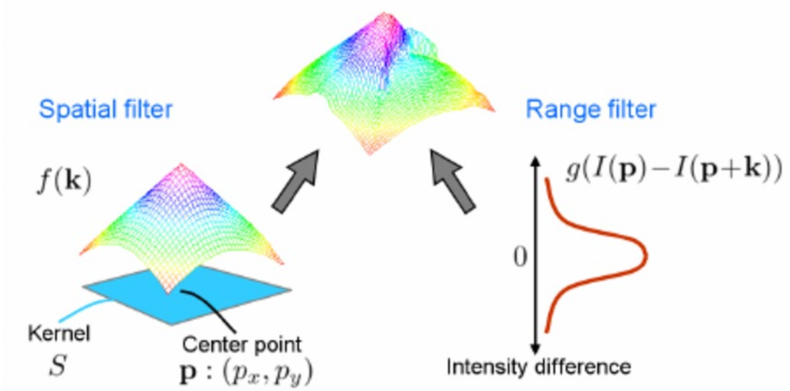
Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings, Bitterli et al., CGF 2016

General Image Denoising Algorithms for MC Rendering

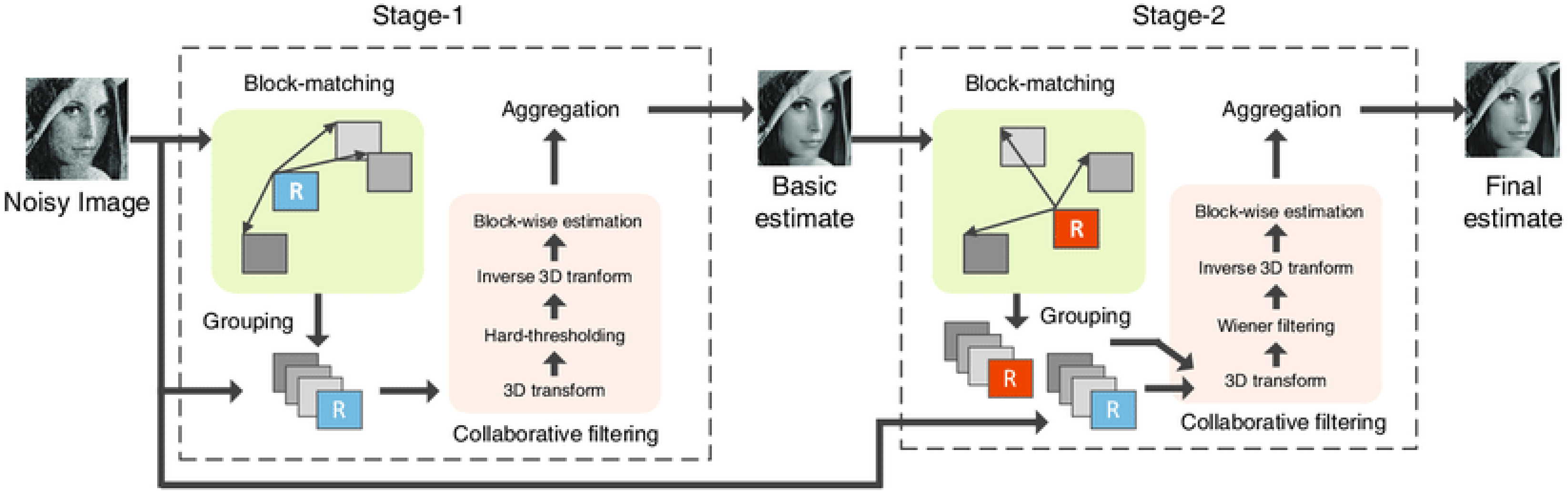
- Efficiently dealing noise on image-space, similar to general image denoising
- Reducing working space from N-dim path-space to 2-dim image space



Non-local Means Filter



Bilateral Filter



Block-Matching 3D (BM3D)

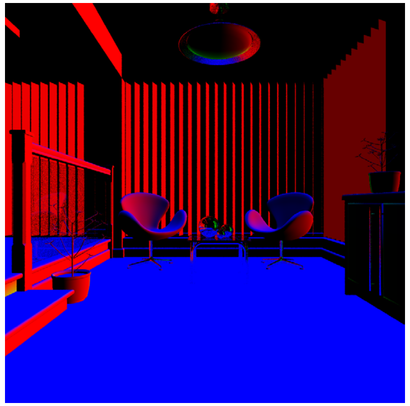
Removing the Noise in Monte Carlo Rendering with General Image Denoising Algorithms, Kalantari et al., CGF 2013

General Image Denoising Algorithms for MC Rendering

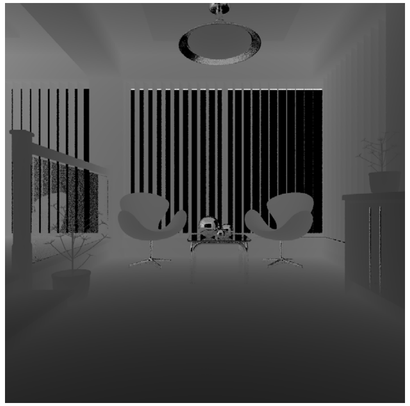
- Efficiently dealing noise on image-space, similar to general image denoising
- Reducing working space from N-dim path-space to 2-dim image space
- Filter weights determined based on similarity in RGB, G-buffers

RGB

Albedo



Normal



Depth

$$w_{ij} = \exp\left[-\frac{1}{2\sigma_{\mathbf{p}}^2} \sum_{1 \le k \le 2} (\bar{\mathbf{p}}_{i,k} - \bar{\mathbf{p}}_{j,k})^2\right] \times$$ Pixel position

$$\exp\left[-\frac{1}{2\sigma_{\mathbf{c}}^2} \sum_{1 \le k \le 3} \alpha_k (\bar{\mathbf{c}}_{i,k} - \bar{\mathbf{c}}_{j,k})^2\right] \times$$ RGB

$$\exp\left[-\frac{1}{2\sigma_{\mathbf{f}}^2} \sum_{1 \le k \le m} \beta_k (\bar{\mathbf{f}}_{i,k} - \bar{\mathbf{f}}_{j,k})^2\right],$$ G-buffers (Albedo, normal, depth, etc.)

On Filtering the Noise from the Random Parameters in Monte Carlo Rendering, Sen et al., ToG 2013

Content

- Reviews on Monte Carlo(MC) ray tracing and MC noise

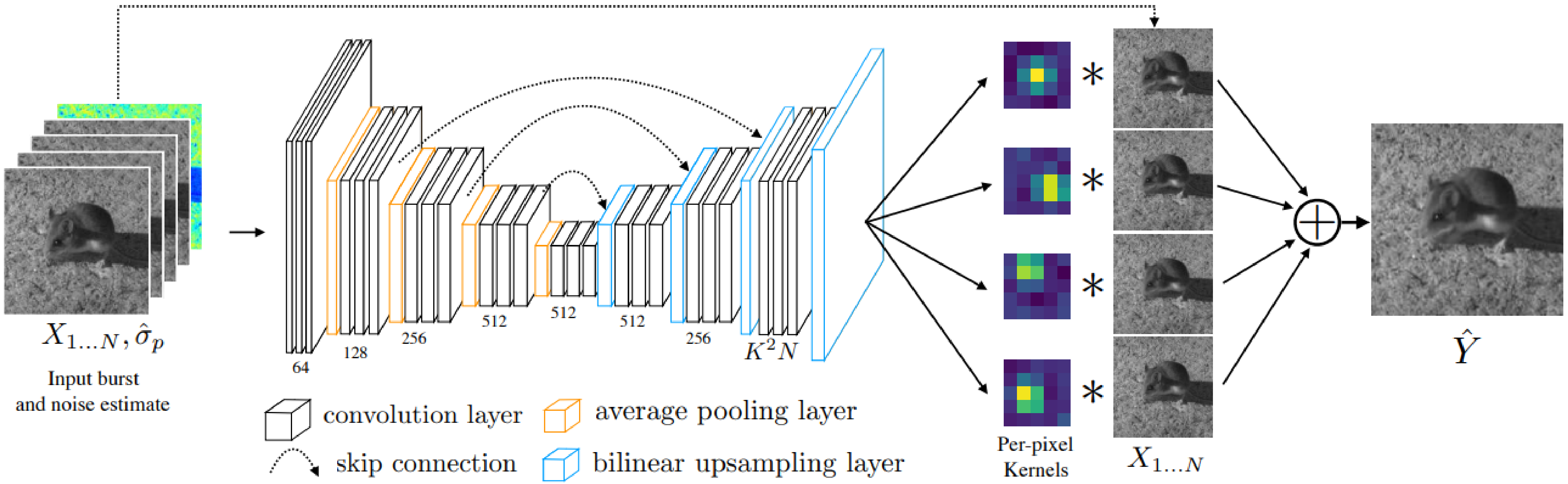
- Path-space MC noise reduction

- Image-space MC noise reduction

- **Learning-based MC noise reduction**
  - **Image-space**
  - **Sample-space**
  - **Path Guiding**
  - **Post-post processing**

Deep-learning Era for Image-space Denoising

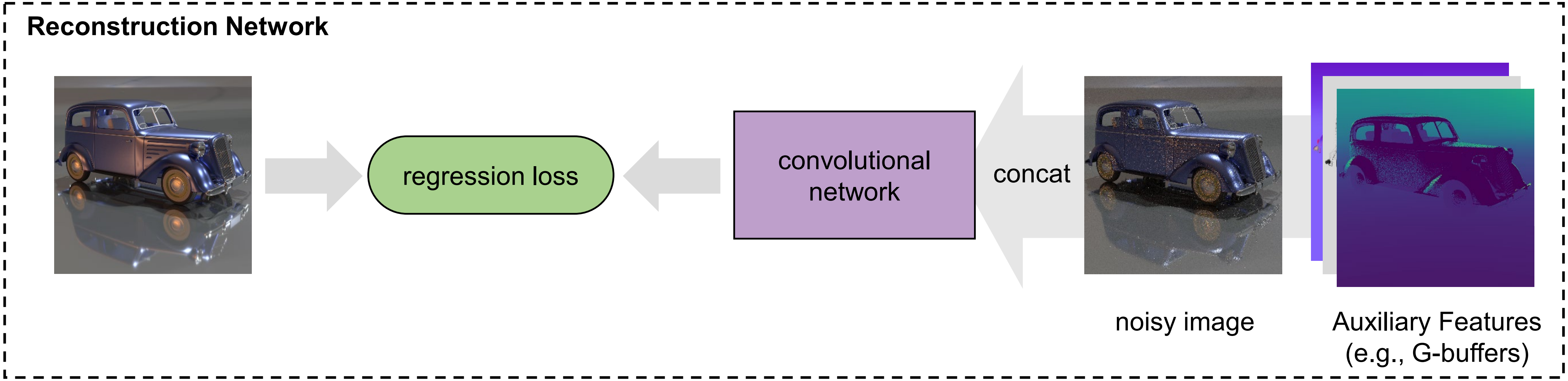
- Various neural networks (MLP, ConvNets, Transformers, etc.) and training strategies (supervised, self-supervised, unsupervised, etc.) are introduced during the last decade
- Reduce design biases of traditional denoising filters



Burst Denoising with Kernel Prediction Networks, Mildenhall et al., CVPR 2018

Conventional Configuration for Learning-based Methods

- Training a neural network to predict the clean image based on the input noisy image and auxiliary features (e.g., G-buffers)

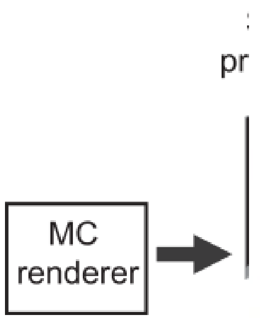


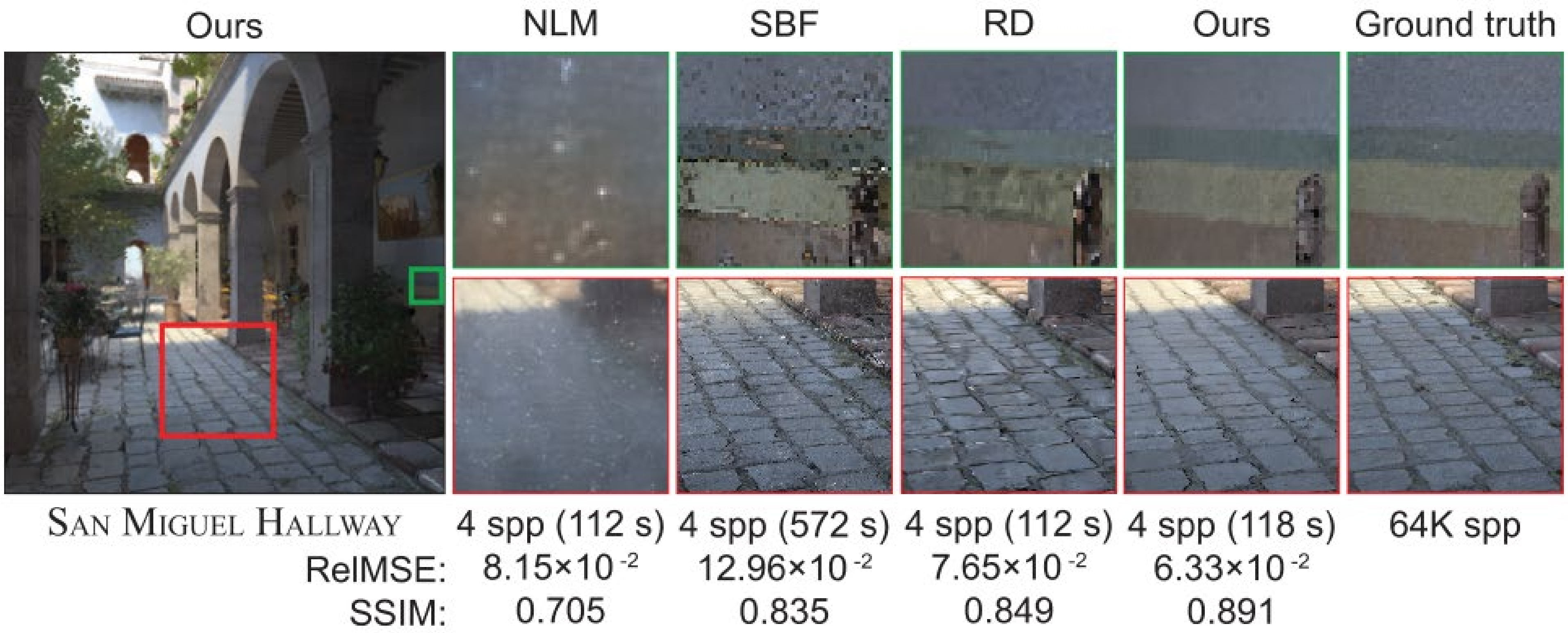
SGVR Lab KAIST

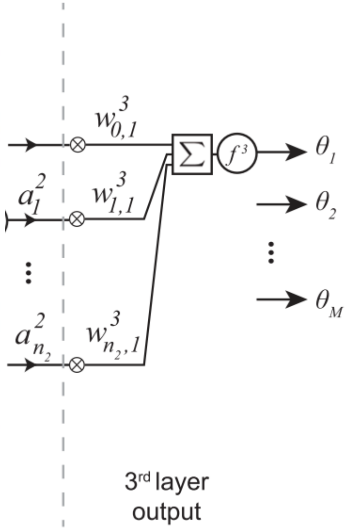
Weakly-supervised Contrastive Learning in Path Manifold for Monte Carlo Image Reconstruction, Cho et al., ToG 2021

Deep-learning for Image-space MC Noise Reduction

- Estimating parameters from cross-bilateral filters using MLP and a large dataset
  - Input : G-buffers, world position, visibility, mean/standard/mean deviation, gradients, spp





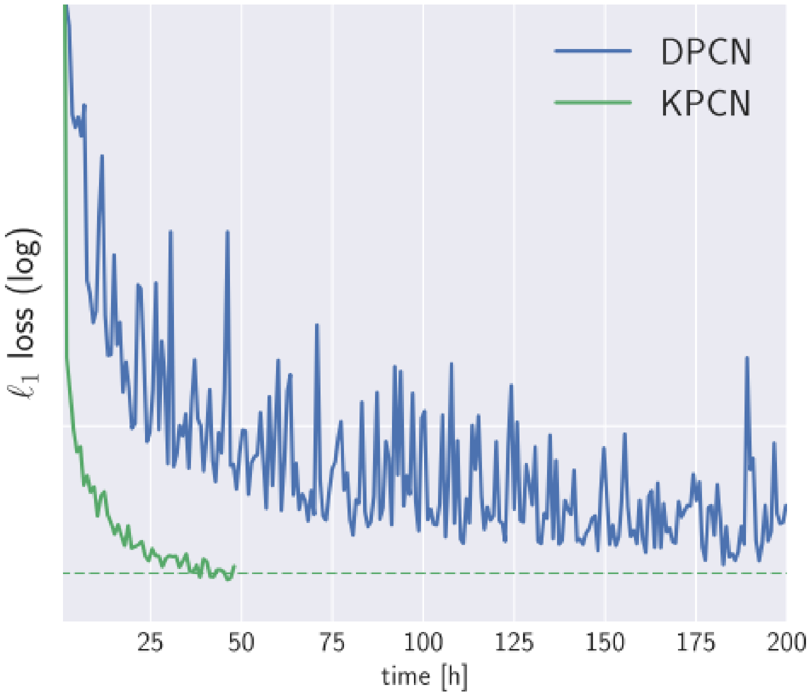


SGVR Lab
KAIST

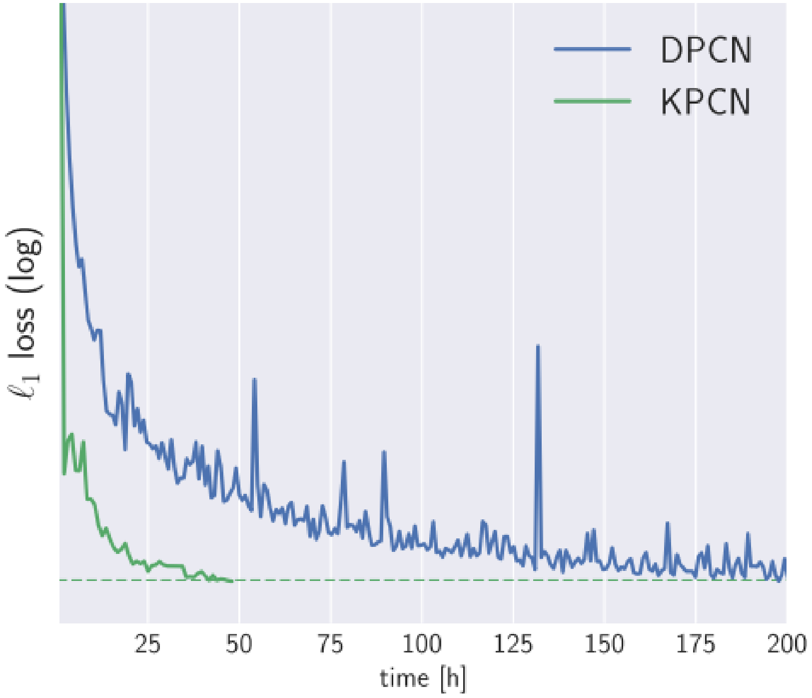
A Machine Learning Approach for Filtering Monte Carlo Noise, Kalantari et al., ToG 2015

Predicting Kernel Weights using CNN

- Robust training by training the network to predict the denoising kernels (KPCN) instead of denoised pixel value (DPCN)
  - Reduces the search space (pixel radiance : 0 ~ unlimited, kernel weights: 0~1)



(a) Diffuse

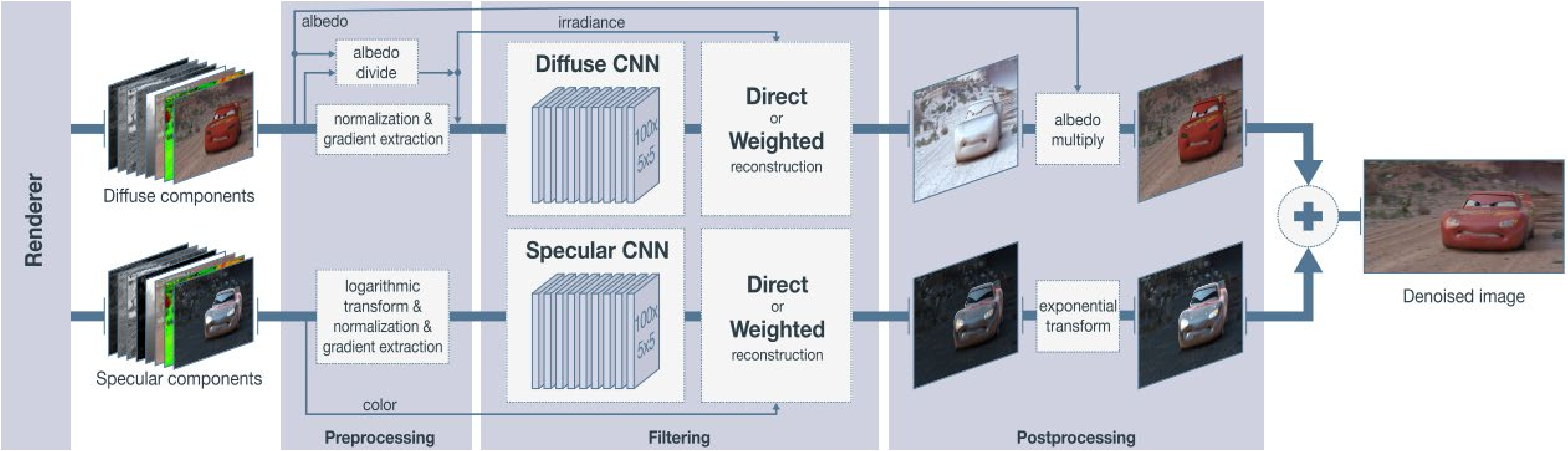


(b) Specular

Kernel-predicting Convolutional Networks for Denoising Monte Carlo Renderings, Bako et al., ToG 2017

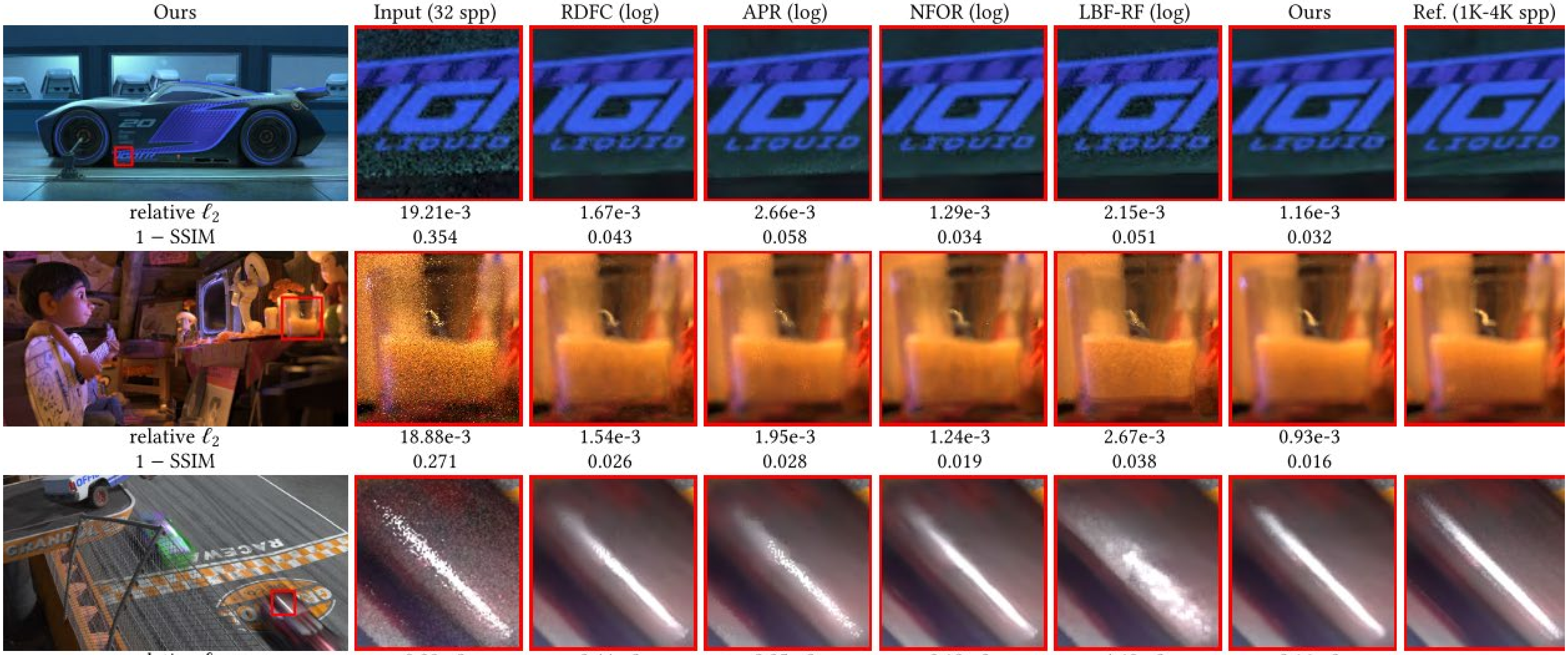
Decompose to Diffuse and Specular

- Train each denoising CNNs to deal with separate lighting effects
  - Diffuse: Geometry dependent, Smooth & low range
  - Specular: View dependent, High range



Kernel-predicting Convolutional Networks for Denoising Monte Carlo Renderings, Bako et al., ToG 2017

Kernel-predicting Convolutional Network (KPCN)

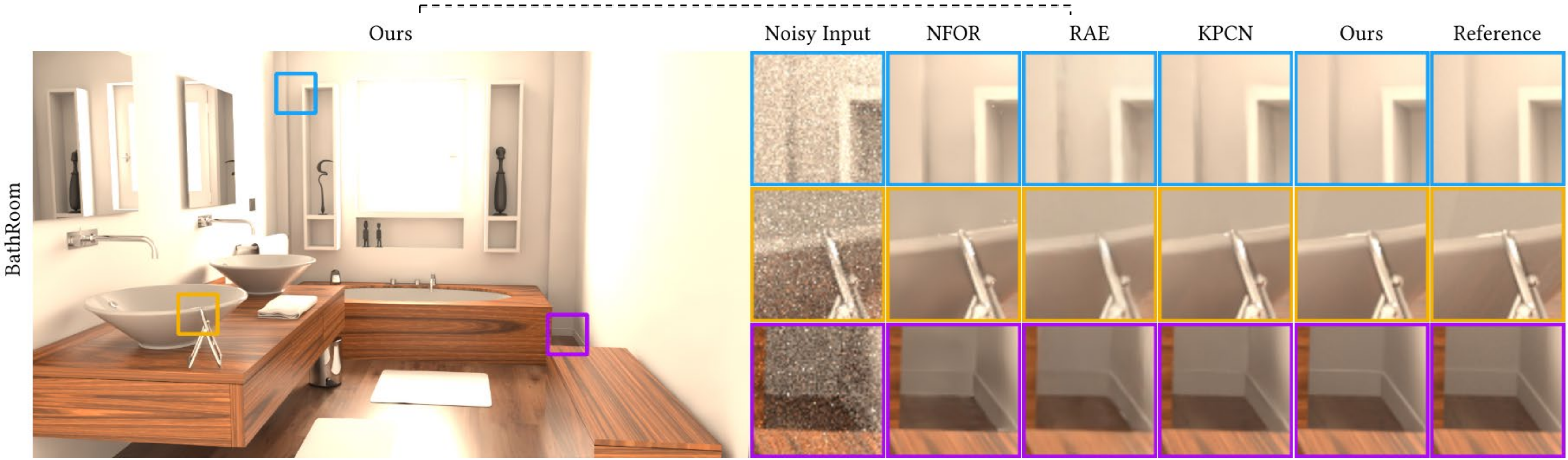


	Input (32 spp)	RDFC (log)	APR (log)	NFOR (log)	LBF-RF (log)	Ours	Ref. (1K-4K spp)
relative ℓ_2	19.21e-3	1.67e-3	2.66e-3	1.29e-3	2.15e-3	1.16e-3	
$1 - \mathrm{SSIM}$	0.354	0.043	0.058	0.034	0.051	0.032	
relative ℓ_2	18.88e-3	1.54e-3	1.95e-3	1.24e-3	2.67e-3	0.93e-3	
$1 - \mathrm{SSIM}$	0.271	0.026	0.028	0.019	0.038	0.016	
relative ℓ_2	9.28e-3	2.44e-3	3.35e-3	2.12e-3	4.69e-3	2.16e-3	
$1 - \mathrm{SSIM}$	0.090	0.023	0.030	0.019	0.027	0.019	

Kernel-predicting Convolutional Networks for Denoising Monte Carlo Renderings, Bako et al., ToG 2017

Adversarial Training for Direct Pixel Denoising (AdvMCD)

- Jointly train the denoising networks and critic networks
- The critic networks are trained to guess whether the input image is clean or noisy (denoised)
- Denoising networks are trained to fool the critic network



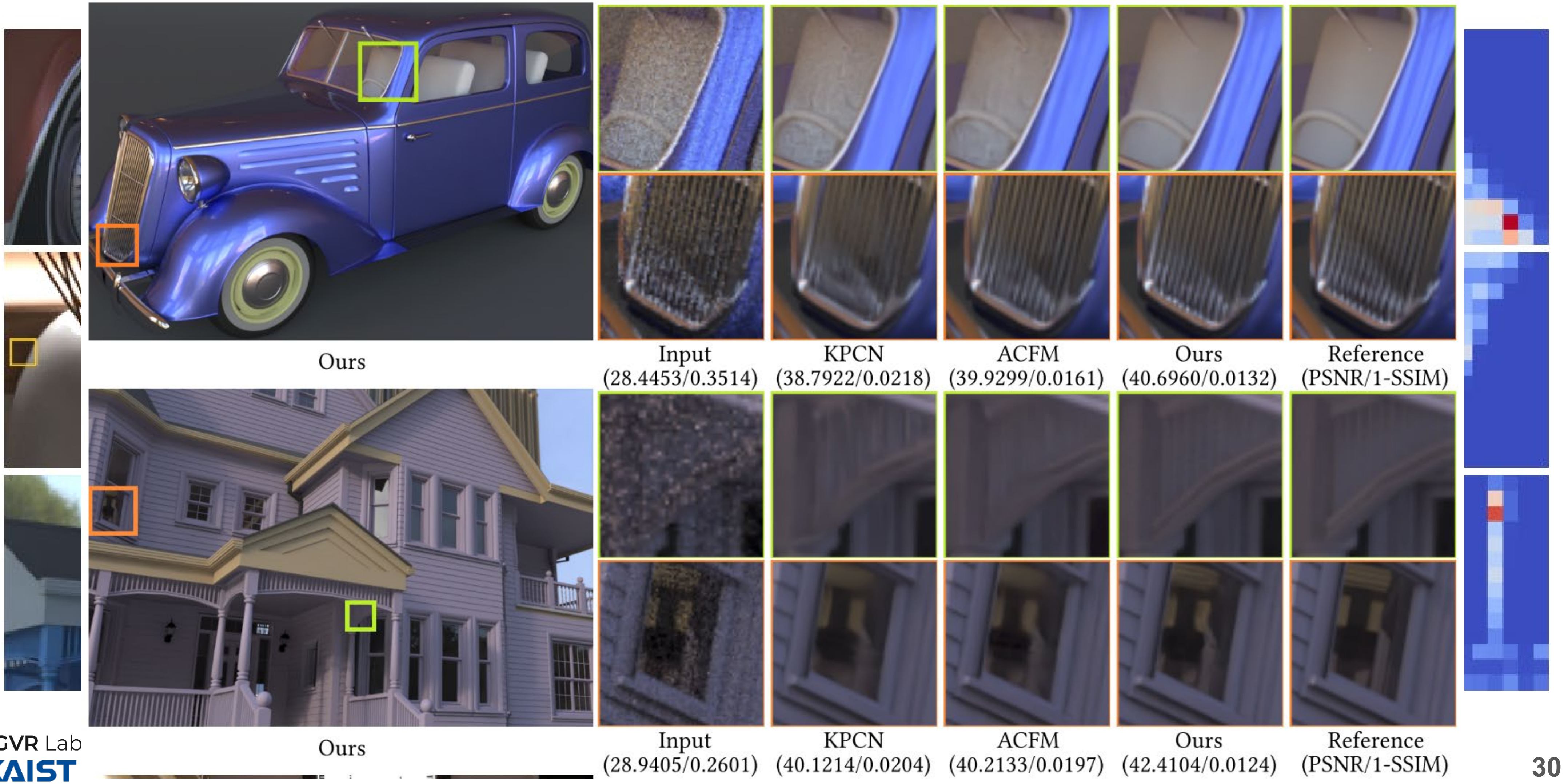
	Noisy Input	NFOR	RAE	KPCN	Ours
1-SSIM	(0.498707)	(0.023738)	(0.036035)	(0.019587)	(0.015482)
RMSE	(0.015005)	(0.000601)	(0.000909)	(0.000411)	(0.000254)
PSNR	(23.433917)	(38.139083)	(36.114381)	(39.167406)	(41.241905)

SGVR Lab KAIST

Adversarial Monte Carlo Denoising with Conditioned Auxiliary Feature Modulation, Xu et al., ToG 2019

Feature-guided Self-attention (AFGSA)

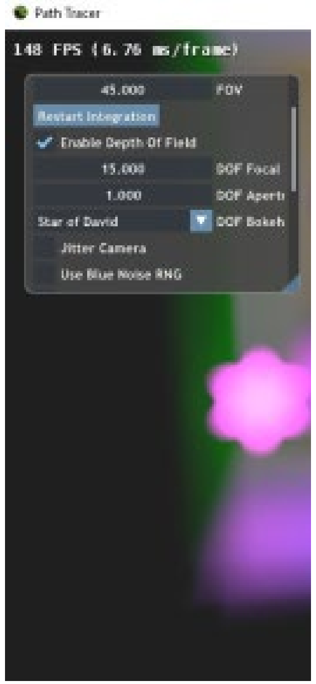
- Stack multiple transformer blocks that creates self-attention map from input image and auxiliary features

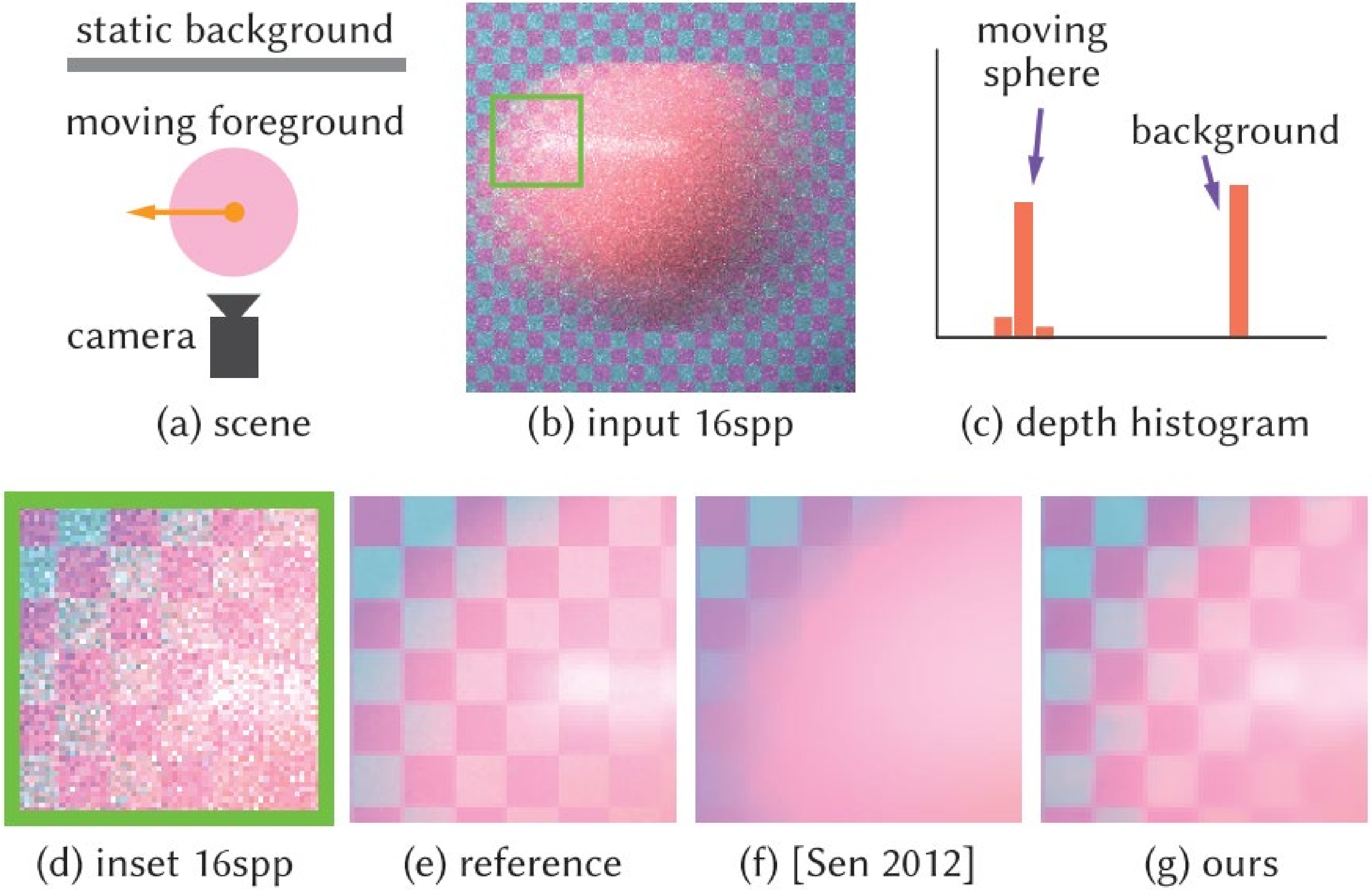


Adversarial Monte Carlo Denoising with Conditioned Auxiliary Feature Modulation, Yu et al., ToG 2021

Jumping from Image-space to Sample-space

- Ray tracing allows to naturally generate blurring effects
- How to reduce the noise while preserving these effects?

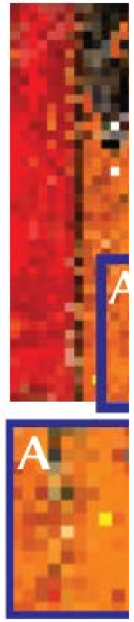


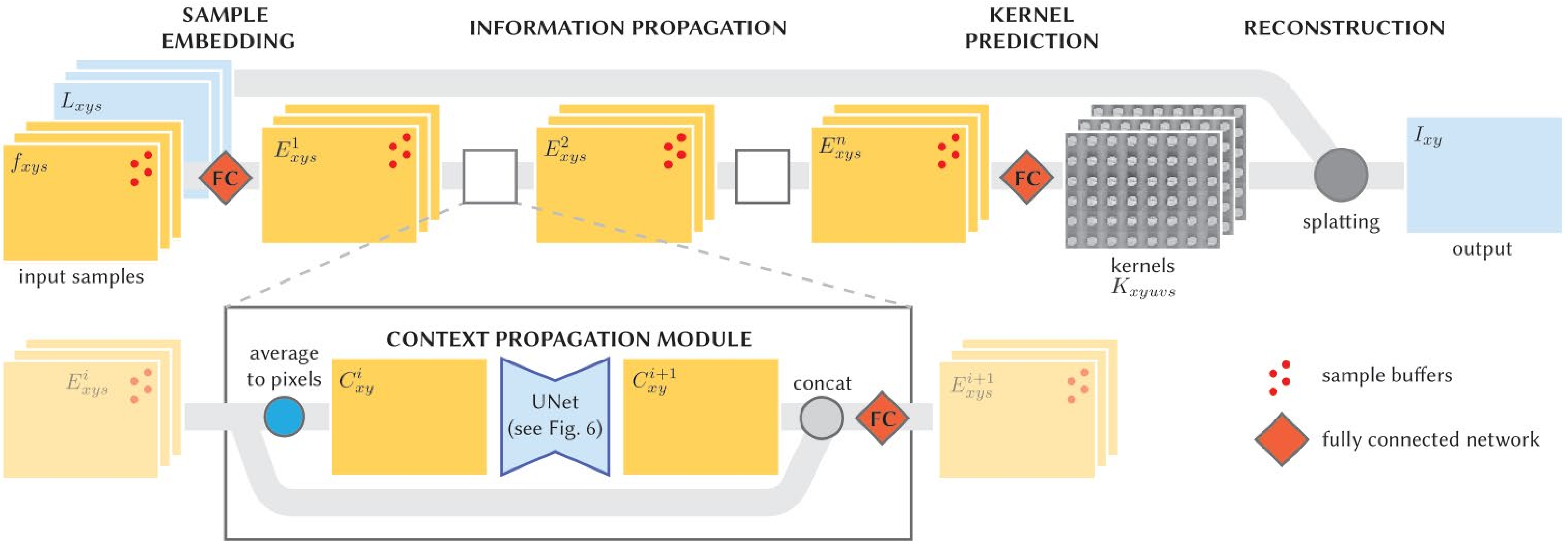


Sample-based Monte Carlo Denoising using a Kernel-splatting Network, Gharbi et al., ToG 2019

Splatting Kernel for Samples

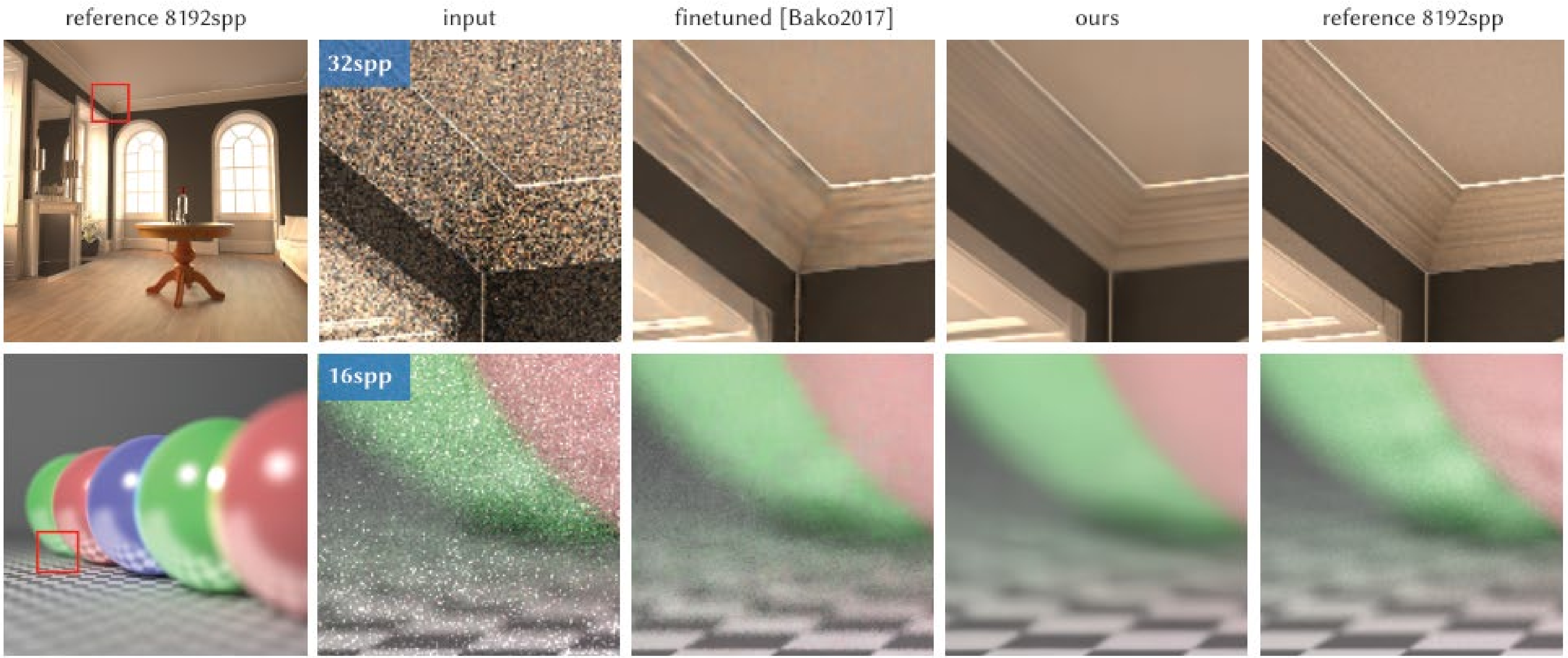
- Conventional kernels: Gathers nearby pixels (samples) with assigned weights
  - Denoised Pixel : Is the i_th sample of my j_th neighbor an outlier?
- Splatting Kernels: Pixels (samples) contributes to nearby pixels with assigned weights
  - Noisy Pixel (sample) : Am I an outlier to my j_th neighbor?
- Intuitive & permutation invariant





Sample-based Monte Carlo Denoising using a Kernel-splatting Network, Gharbi et al., ToG 2019

Sample-based Monte Carlo Denoising (SBMC)

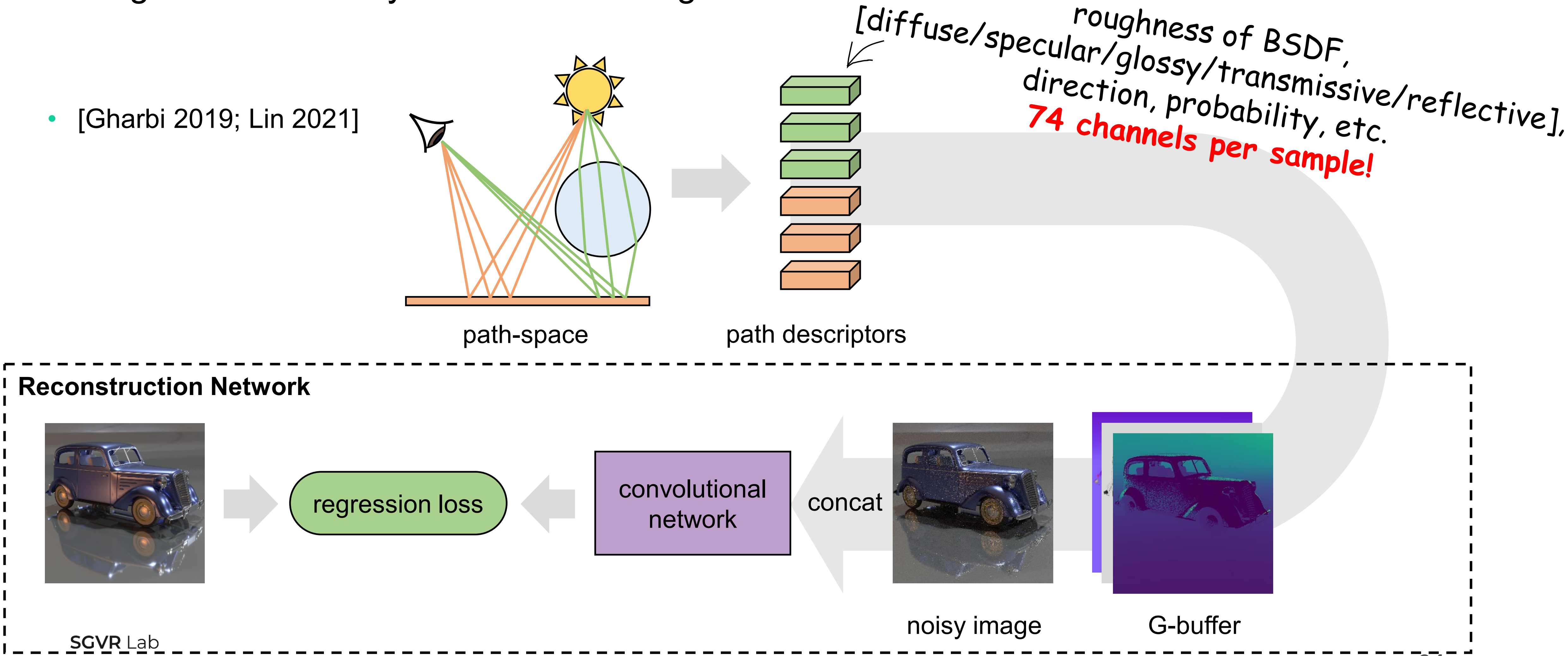


Sample-based Monte Carlo Denoising using a Kernel-splatting Network, Gharbi et al., ToG 2019

Path-space Features for Denoising

- Multi-bounce features are useful for reconstructing complex lighting details
- High-dimensionality harms the training of neural network

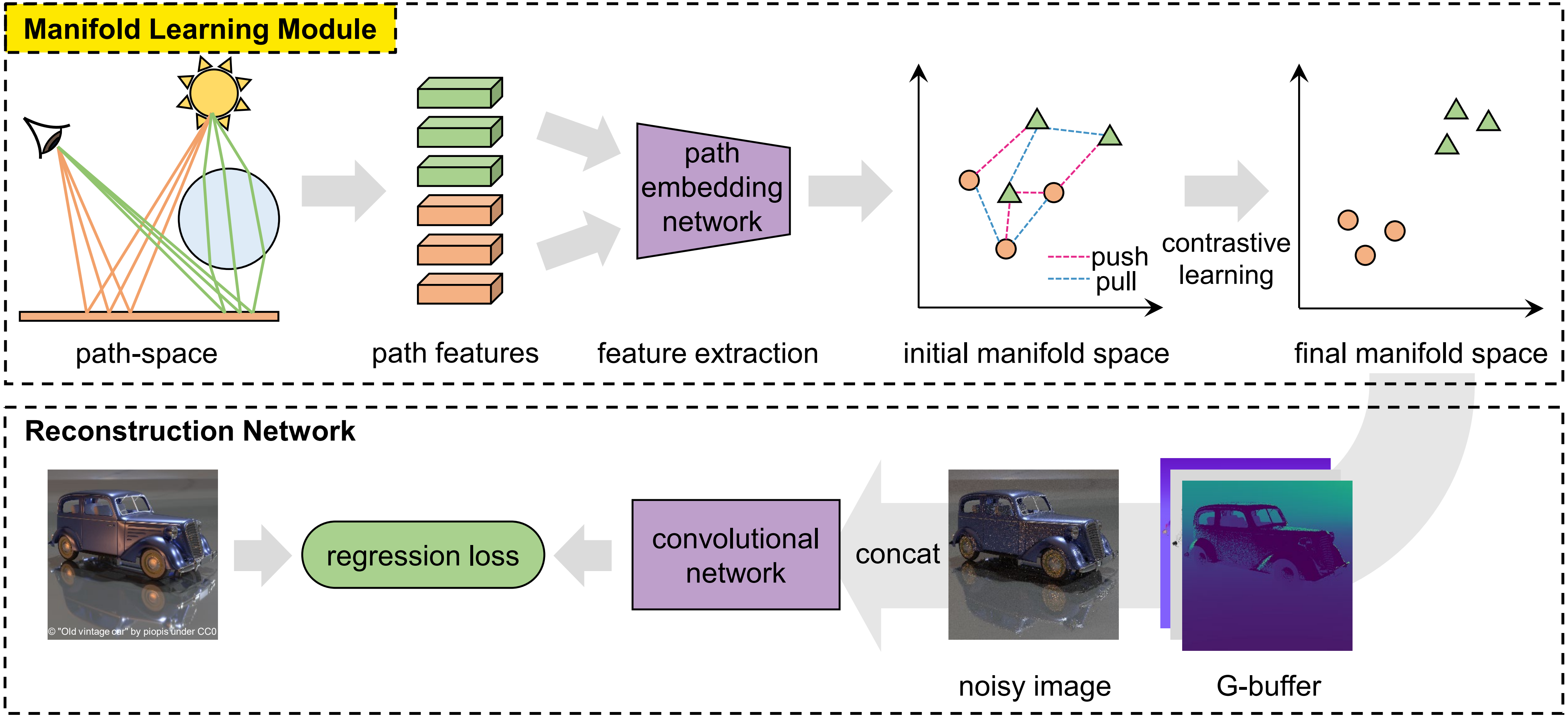
- [Gharbi 2019; Lin 2021]



Weakly-supervised Contrastive Learning in Path Manifold for Monte Carlo Image Reconstruction, Cho et al., ToG 2021

Manifold Learning for Path-space Features

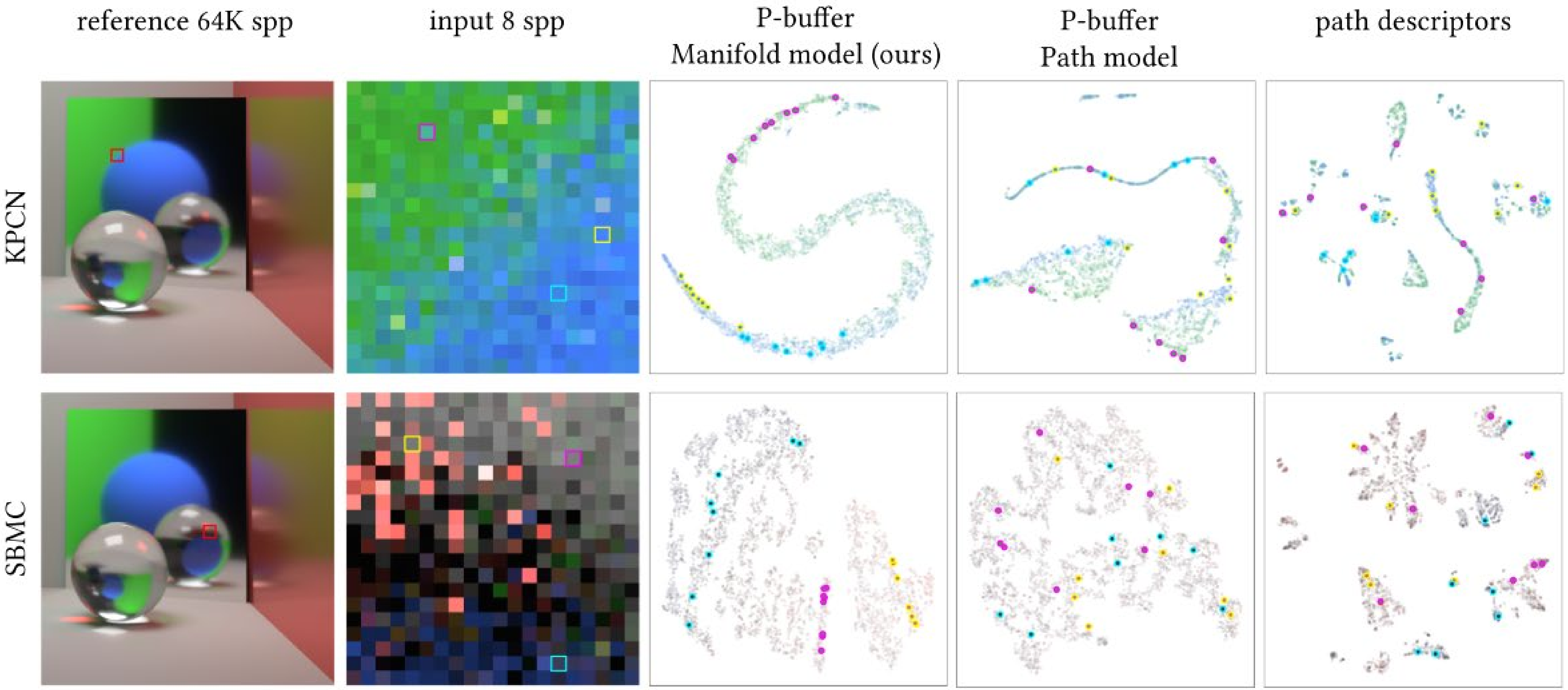
- Embed path features to low-dimensional space



Weakly-supervised Contrastive Learning in Path Manifold for Monte Carlo Image Reconstruction, Cho et al., ToG 2021

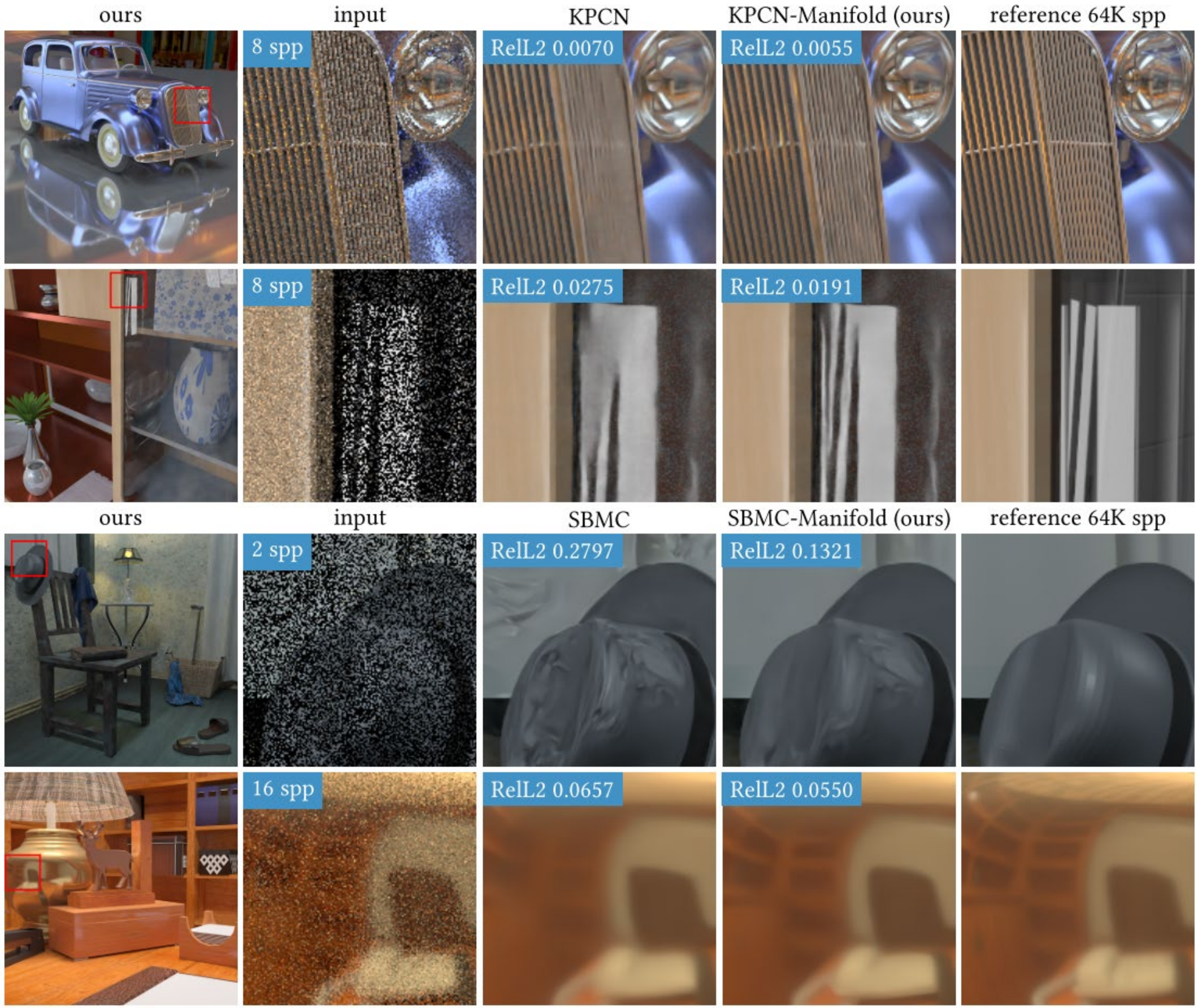
Manifold Learning for Path-space Features

- Use pixel colors as pseudo-labels
- Embed path features based on pixel-color similarity using contrastive learning



Weakly-supervised Contrastive Learning in Path Manifold for Monte Carlo Image Reconstruction, Cho et al., ToG 2021

Manifold Learning for Path-space Features



SGVR Lab KAIST

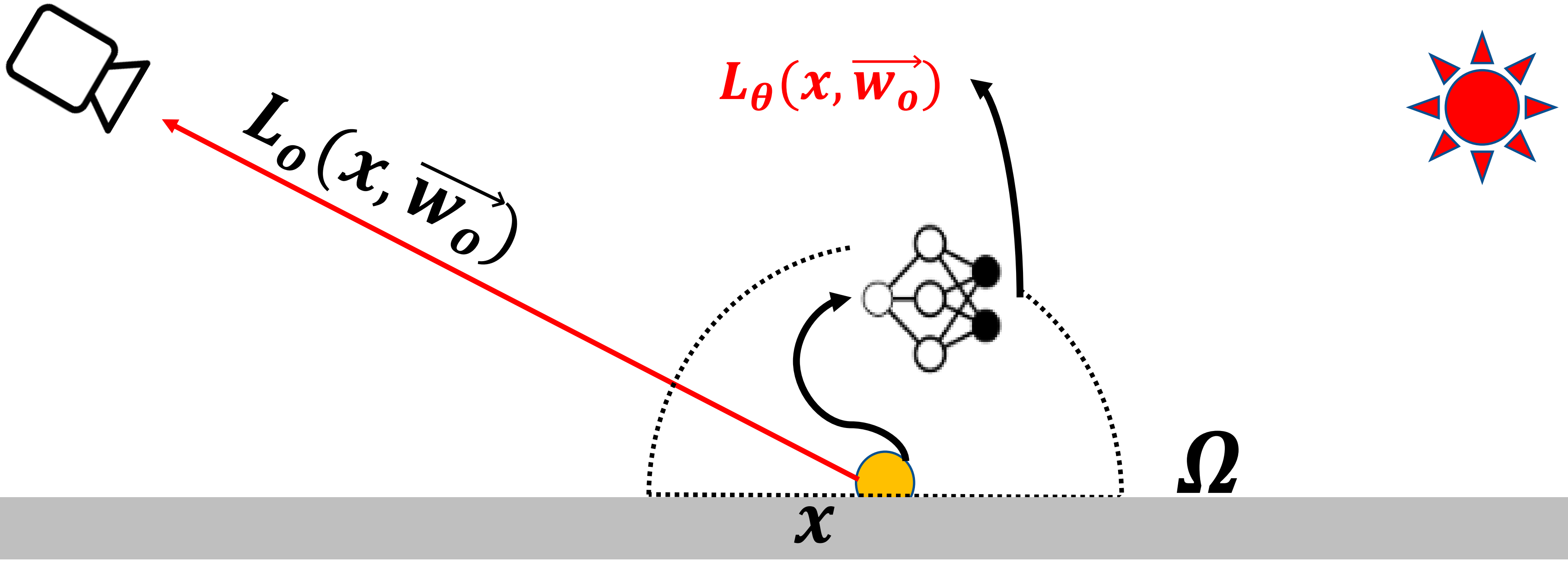
Weakly-supervised Contrastive Learning in Path Manifold for Monte Carlo Image Reconstruction, Cho et al., ToG 2021

Neural Radiance Caching

- Solving rendering equation via Radiance-predicting Neural Network L_θ

$$L_o(x,\overrightarrow{w_o}) = L_e(x,\overrightarrow{w_o}) + \int_\Omega f_r(x,\overrightarrow{w_i},\overrightarrow{w_o}) L_i(x,\overrightarrow{w_i})(\overrightarrow{w_i}\cdot\vec{n})d\overrightarrow{w_i}$$

$$\sim L_e(x,\overrightarrow{w_o}) + L_\theta(x,\overrightarrow{w_o})$$



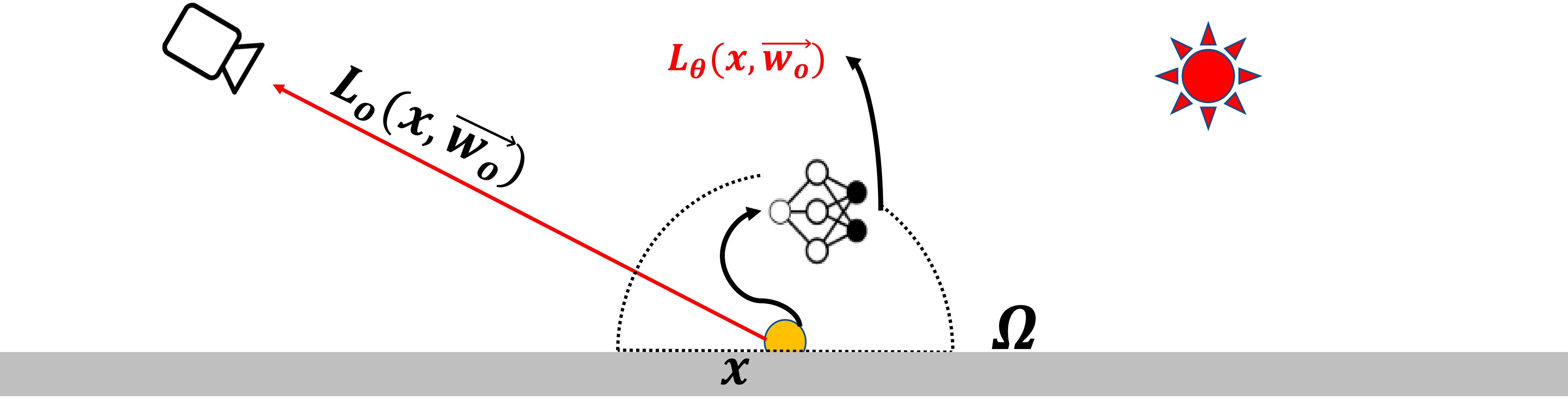
Real-time Neural Radiance Caching for Path Tracing, Muller et al., ToG 2021

Neural Radiance Caching

Train the neural network → Cache, Estimate the radiance → Interpolate

$$L_o(x,\overrightarrow{w_o}) = L_e(x,\overrightarrow{w_o}) + \int_\Omega f_r(x,\overrightarrow{w_i},\overrightarrow{w_o})L_i(x,\overrightarrow{w_i})(\overrightarrow{w_i}\cdot\vec{n})d\overrightarrow{w_i}$$

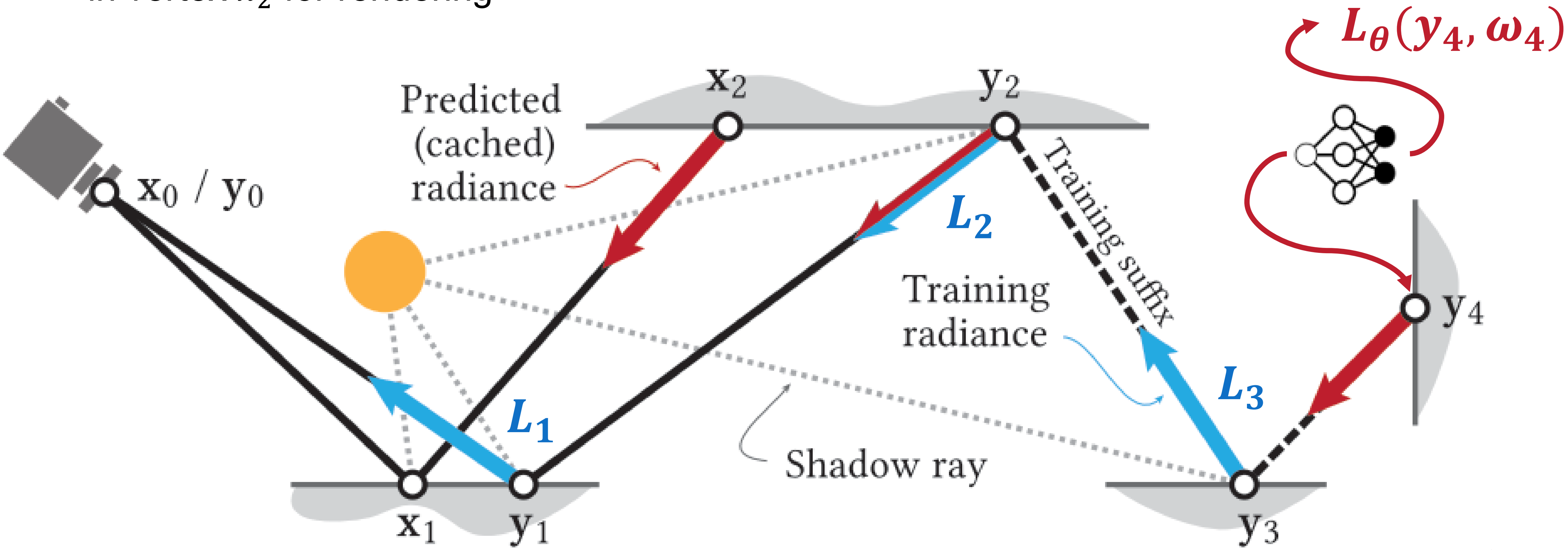
$$\sim L_e(x,\overrightarrow{w_o}) + L_\theta(x,\overrightarrow{w_o})$$



Real-time Neural Radiance Caching for Path Tracing, Muller et al., ToG 2021

Self-training for Neural Radiance Cache

- Minimize the loss between the calculated radiances and the estimated radiances of the preceding vertices
- $Loss = relL2\big(L_1, L_\theta(y_1, \omega_1)\big) + relL2\big(L_2, L_\theta(y_2, \omega_2)\big) + relL2\big(L_3, L_\theta(y_3, \omega_3)\big)$
- Trace a short rendering path $(x_0 x_1 x_2)$ where we used the cached(estimated) radiance in vertex x_2 for rendering



Real-time Neural Radiance Caching for Path Tracing, Muller et al., ToG 2021

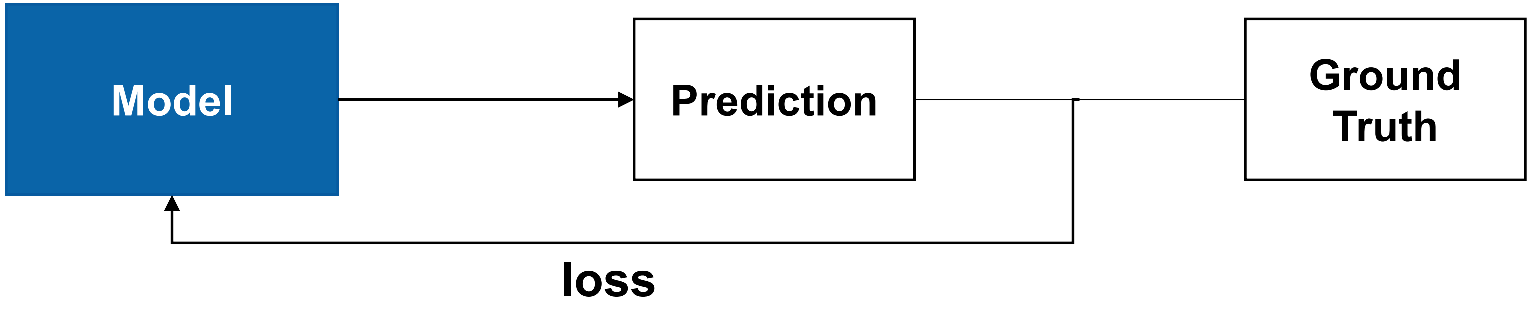
Result – 1spp Video

Result – 1spp Video w/ Iimage Denoising

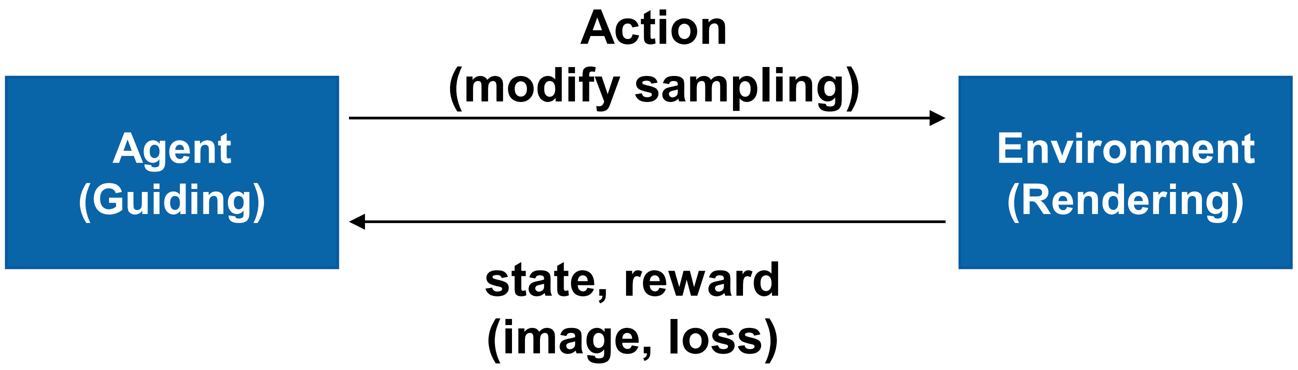
Real-time Neural Radiance Caching for Path Tracing, Muller et al., ToG 2021

Path Guiding using Reinforcement Learning

- Supervised Learning

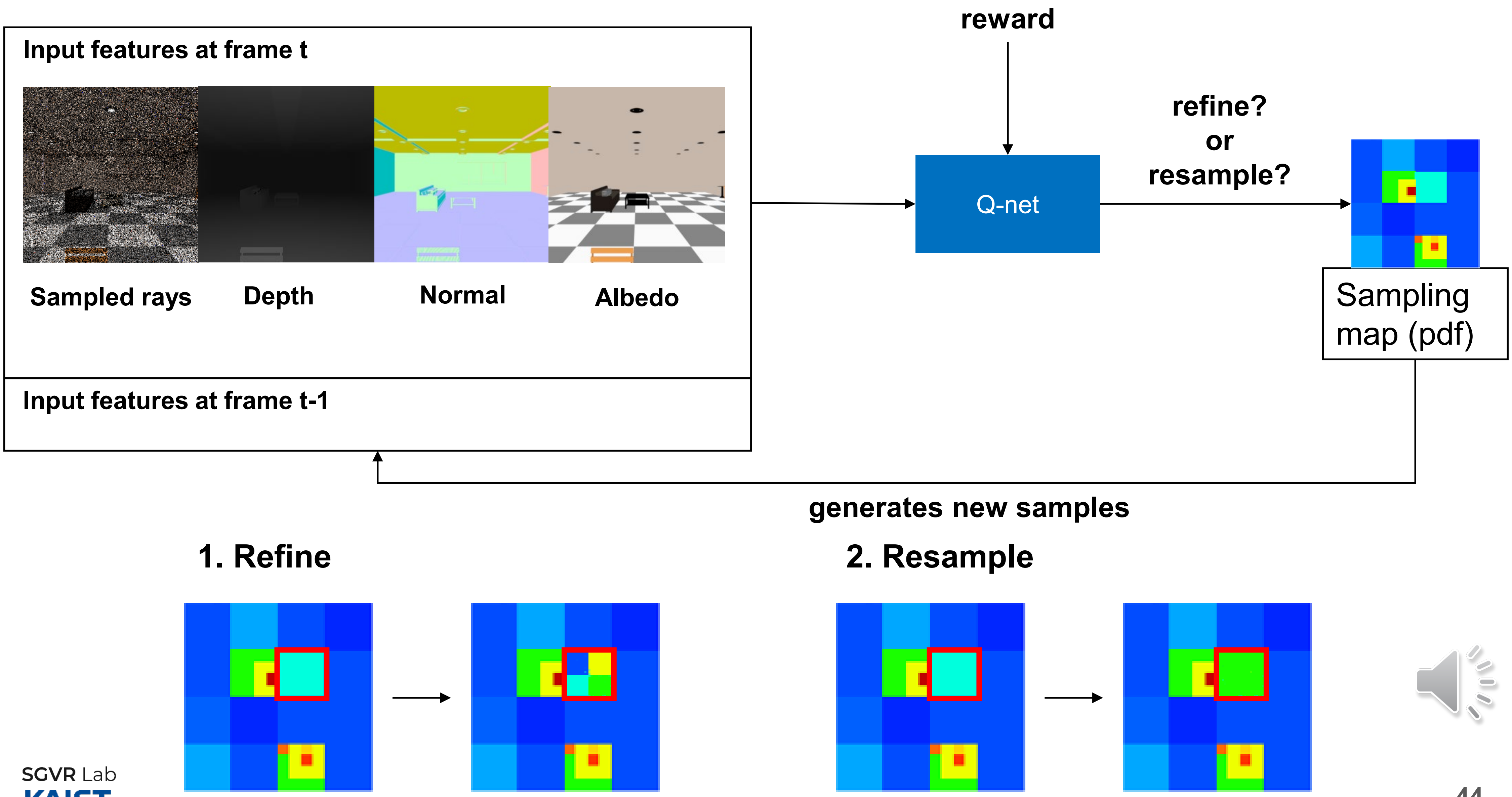


- Reinforcement Learning
  - Find a policy $\boldsymbol{\pi}: \boldsymbol{S} \rightarrow \boldsymbol{A}$ with maximum rewards



Adaptive Iincident Radiance Field Sampling and Reconstruction Using Deep Reinforcement Learning, Huo et al., ToG 2020

Sampling Map Generation (Q-Network)

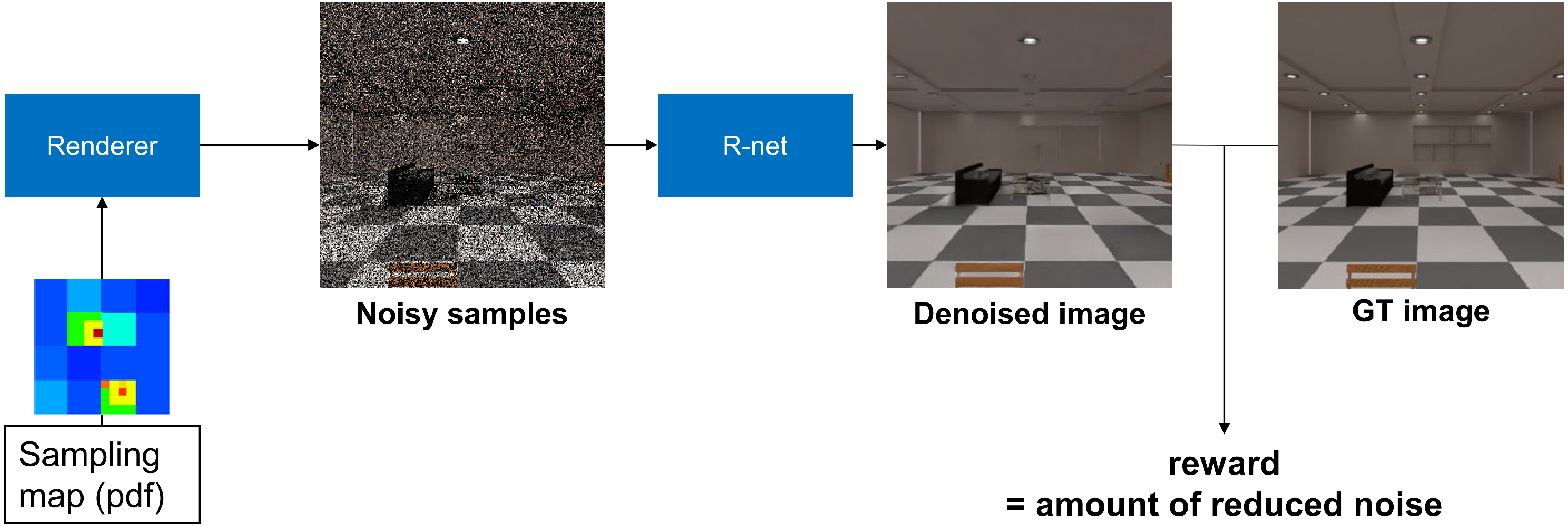


SGVR Lab
KAIST

Adaptive Iincident Radiance Field Sampling and Reconstruction Using Deep Reinforcement Learning, Huo et al., ToG 2020

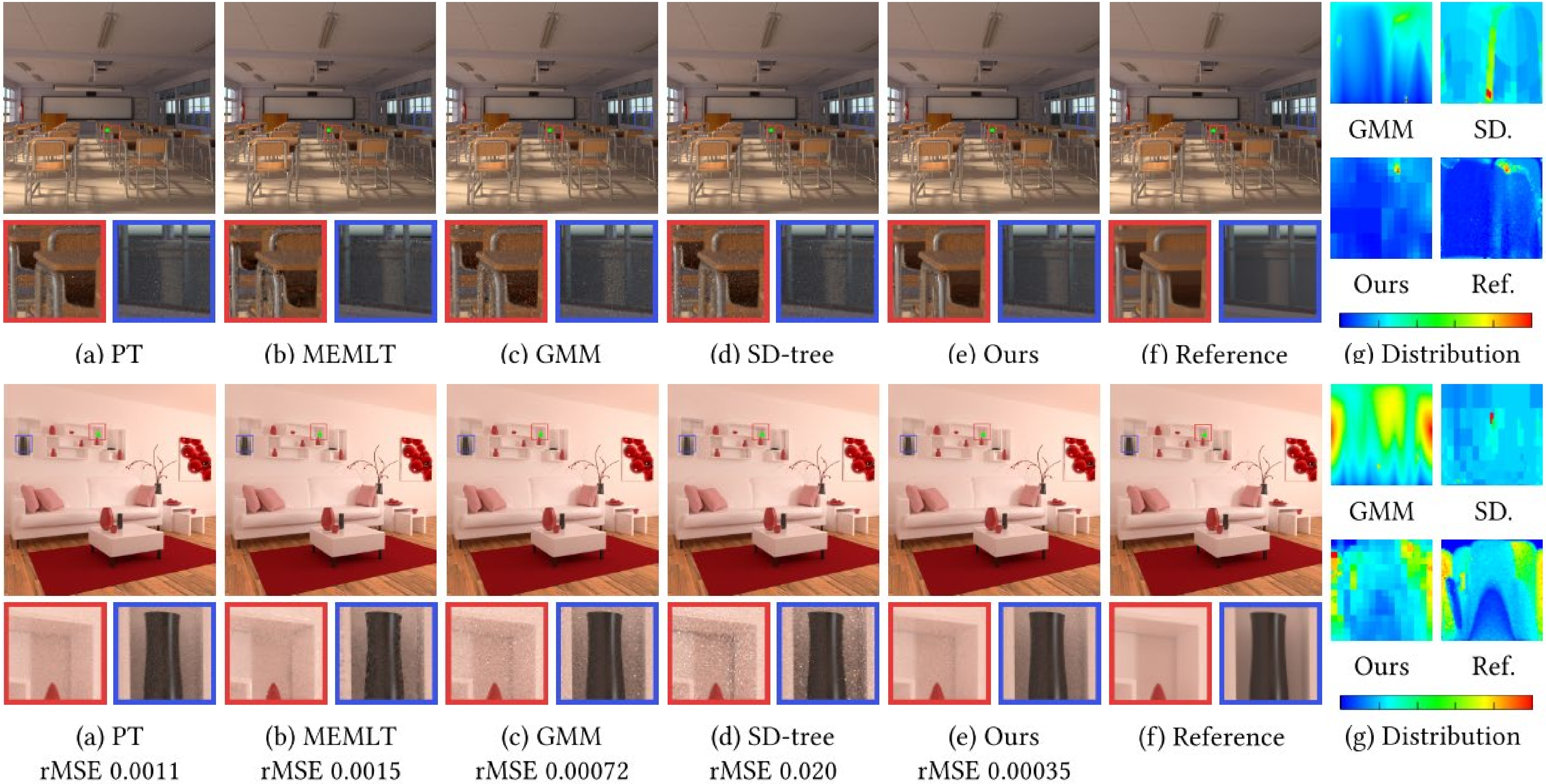
Rendering and Reconstruction (R-Network)

- Q-Net trained to estimate a sampling network that maximizes the reward



Adaptive Iincident Radiance Field Sampling and Reconstruction Using Deep Reinforcement Learning, Huo et al., ToG 2020

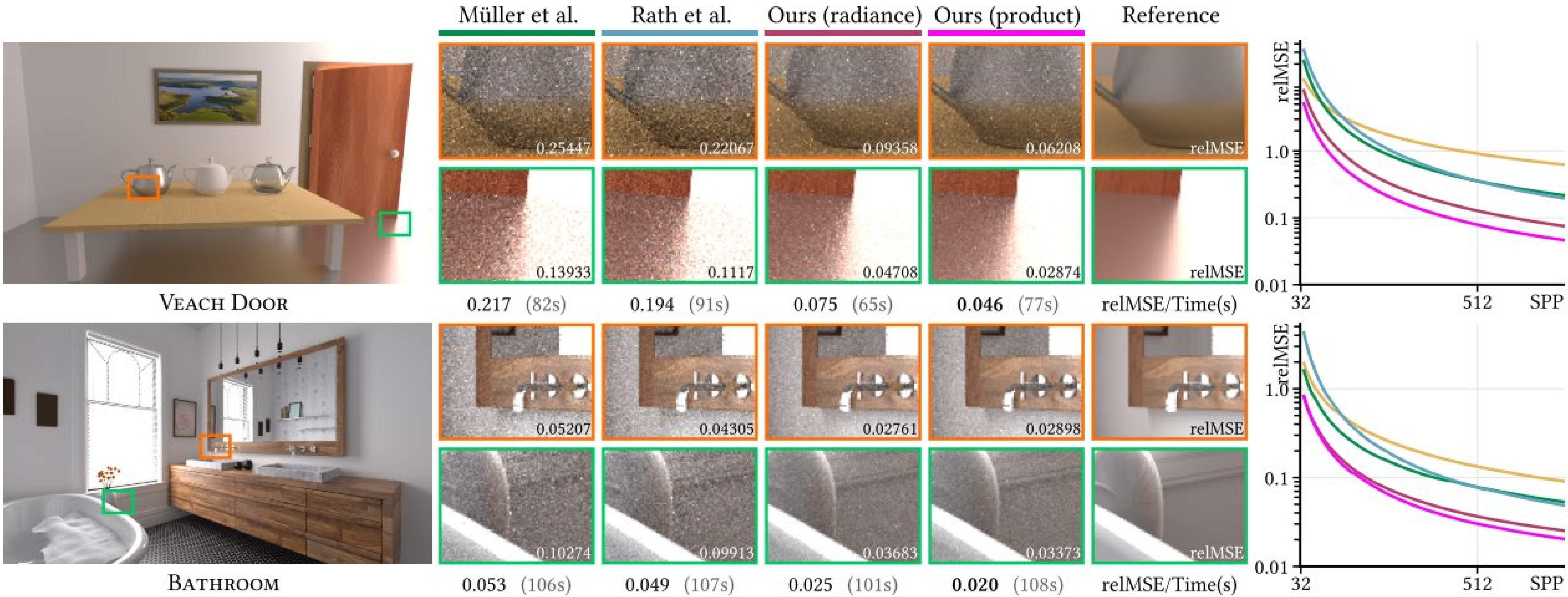
Path Guiding using Reinforcement Learning



Adaptive Iincident Radiance Field Sampling and Reconstruction Using Deep Reinforcement Learning, Huo et al., ToG 2020

Implicit Neural Representation for Path Guiding

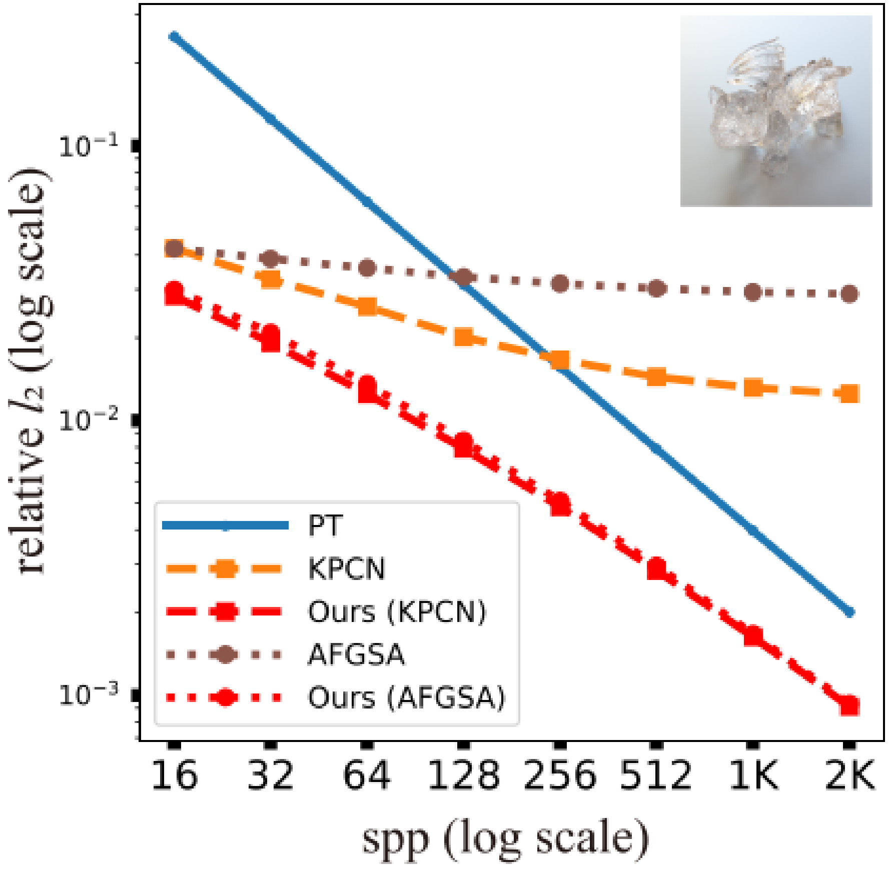
- Using implicit neural representation that encodes parameters for mixture models
- Using vMF (von Mises-Fisher) distribution $v(\omega \mid \mu, \kappa) = \frac{\kappa}{4\pi \sinh \kappa} \exp\left(\kappa \mu^T \omega\right)$



Neural Parametric Mixtures for Path Guiding, Dong et al., SIGGRAPH 2023 Conference Track

Post-processing the Denoiser (Post-post Processing)

- Denoising models trained on certain noise level is biased to the noise level & dataset
- Cannot show consistent performance throughout noise levels



(a) DRAGON

SGVR Lab KAIST

Neural James-Stein Combiner for Unbiased and Biased Renderings, Gu et al., ToG 22

Combining Biased and Unbiased Estimates

- Path Traced Result X : Noisy but Unbiased (Bias ↓, Variance ↑)
- Denoised Result Y: Smooth but Biased (Bias ↑, Variance ↓)
- James-Stein Estimator shrinks X towards Y as $\delta(X,Y) = Y + \left(1 - \frac{(p-2)\sigma^2}{\|X-Y\|^2}\right)(X - Y)$
  - p : Dimension of estimation (3 = RGB channel), σ: variance of radiance
- Performs better than sample mean (in our case, X) if $p \geq 3$

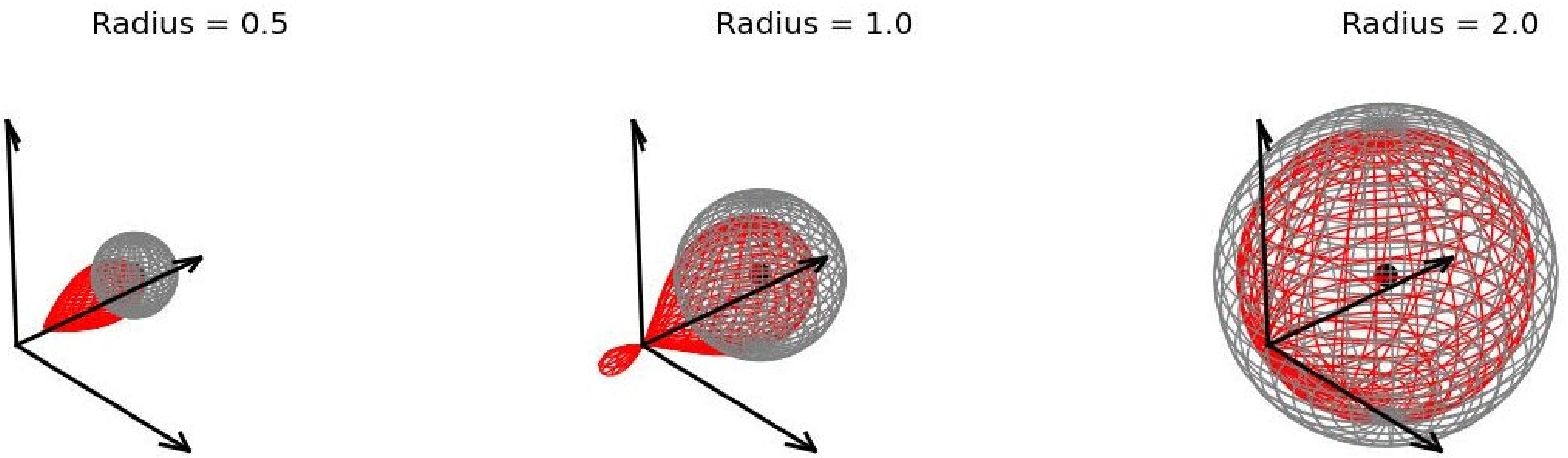
$$MSE = (BIAS)^2 + VARIANCE$$

Leaving some space on BIAS, James-Stein Estimator reduces the VARIANCE
by shrinking the points to be dense

Neural James-Stein Combiner for Unbiased and Biased Renderings, Gu et al., ToG 22

Combining Biased and Unbiased Estimates

- Path Traced Result X : Noisy but Unbiased (Bias ↓, Variance ↑)
- Denoised Result Y: Smooth but Biased (Bias ↑, Variance ↓)
- James-Stein Estimator shrinks X towards Y as $\delta(X,Y) = Y + \left(1 - \frac{(p-2)\sigma^2}{\|X-Y\|^2}\right)(X - Y)$
  - p : Dimension of estimation (3 = RGB channel), σ: variance of radiance
- Performs better than sample mean (in our case, X) if $p \geq 3$



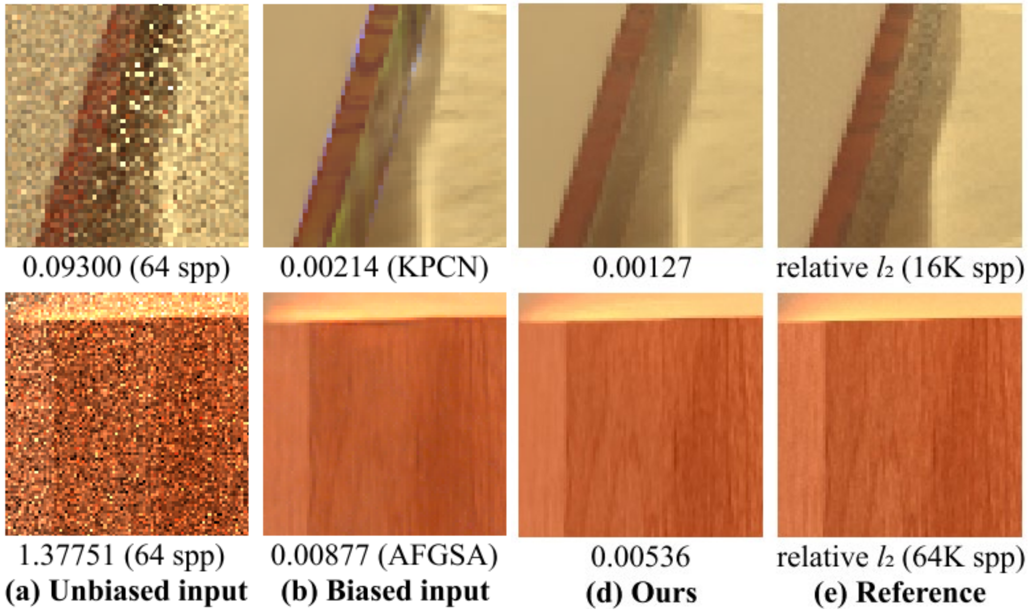
James-Stein Estimator shows less MSE error
Grey – Sampled points on radius sphere with center (1, 1, 1)
Red – James-Stein estimator applied on sampled points

SGVR Lab KAIST

Neural James-Stein Combiner for Unbiased and Biased Renderings, Gu et al., ToG 22

Combining Biased and Unbiased Estimates

- Path Traced Result X : Noisy but Unbiased (Bias ↓, Variance ↑)
- Denoised Result Y: Smooth but Biased (Bias ↑, Variance ↓)
- James-Stein Estimator shrinks X towards Y as $\delta(X,Y) = Y + \left(1 - \frac{(p-2)\sigma^2}{\|X-Y\|^2}\right)(X - Y)$
  - p : Dimension of estimation (3 = RGB channel), σ: variance of radiance
- Performs better than sample mean (in our case, X) if $p \geq 3$



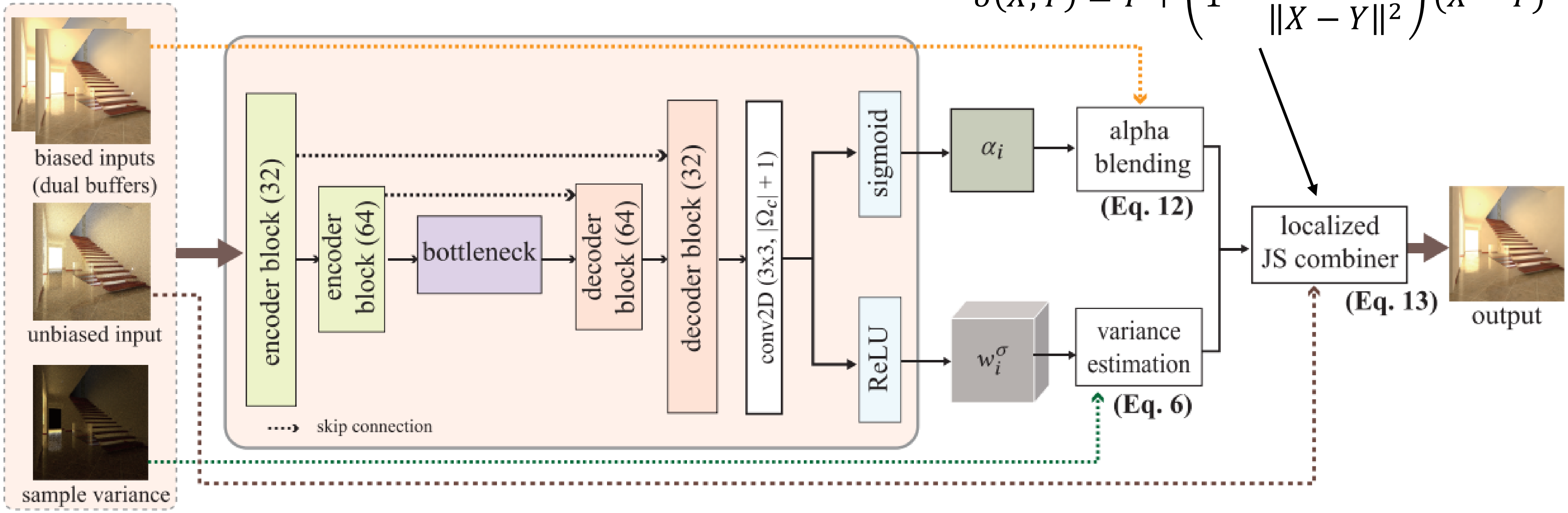
<Intuition>
Balancing the bias and variance of between path traced result and denoised result

Neural James-Stein Combiner for Unbiased and Biased Renderings, Gu et al., ToG 22

Neural James-Stein Combiner

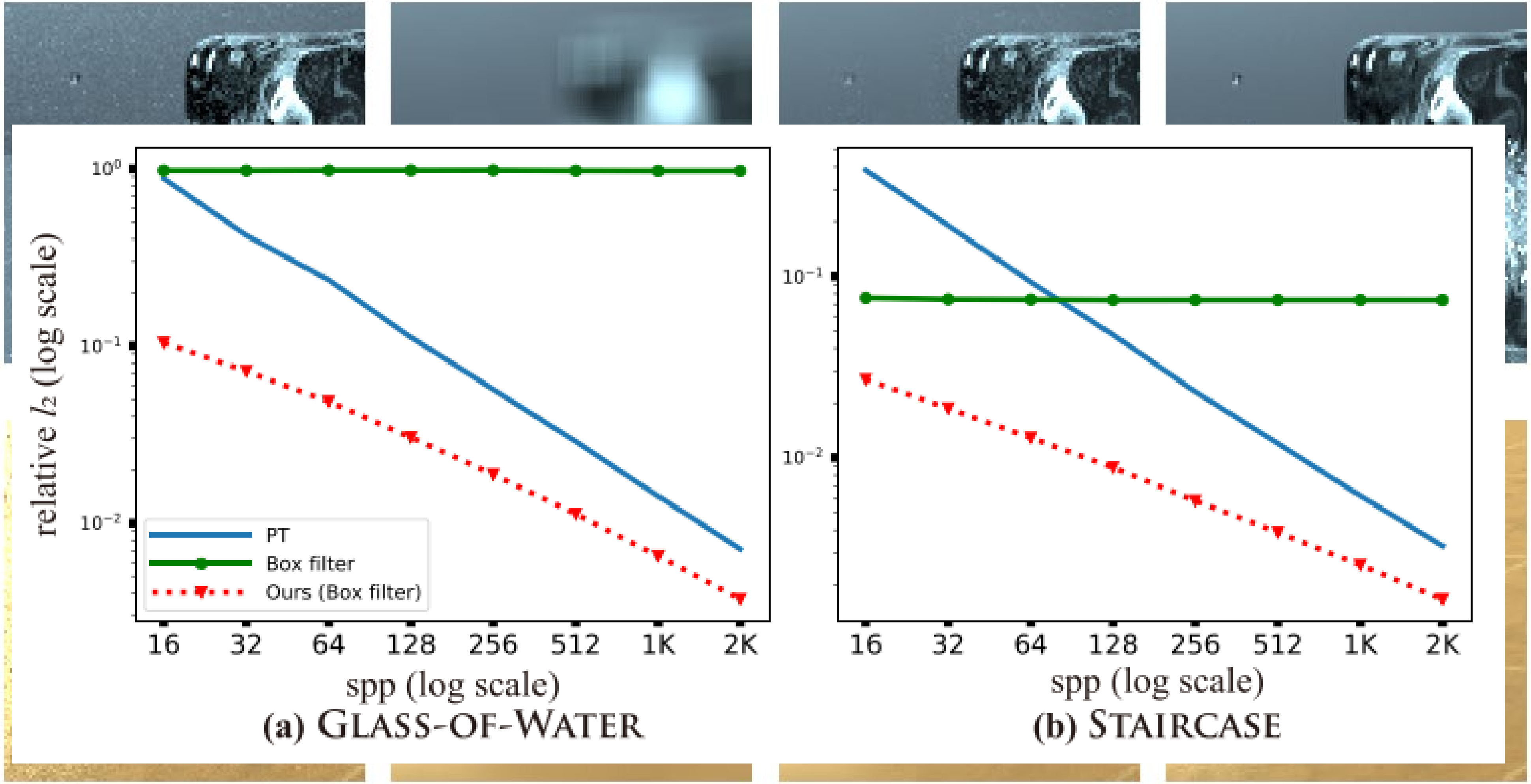
- Small U-Net to estimate weights for James-Stein Combiner

$$\delta(X,Y) = Y + \left(1 - \frac{(p-2)\sigma^2}{\|X-Y\|^2}\right)(X-Y)$$



Neural James-Stein Combiner for Unbiased and Biased Renderings, Gu et al., ToG 22

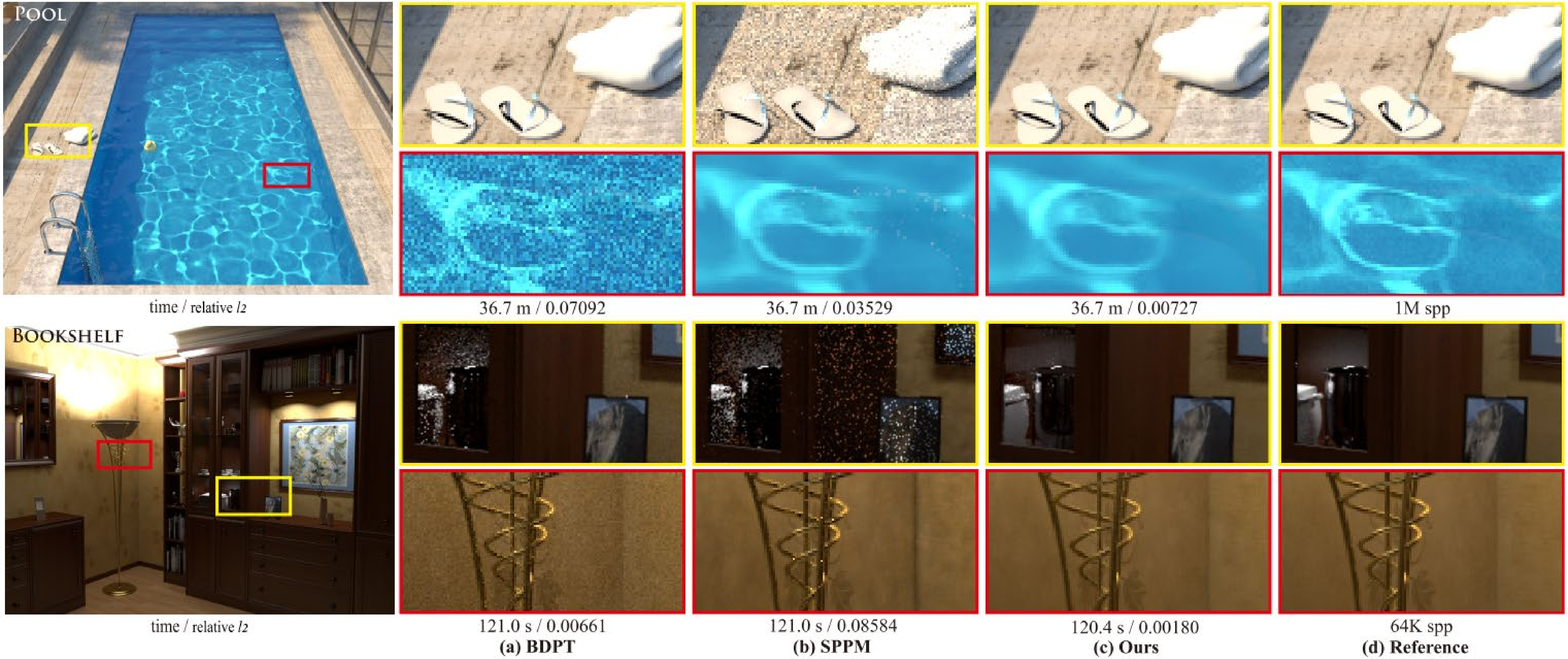
Neural James-Stein Combiner



0.00617 (1K spp)	0.07395	0.00257	relative l_2
(a) Unbiased Input	**(b) Biased Input**	**(c) Combined result**	**Reference**

Neural James-Stein Combiner for Unbiased and Biased Renderings, Gu et al., ToG 22

Neural James-Stein Combiner



Neural James-Stein Combiner for Unbiased and Biased Renderings, Gu et al., ToG 22

What We Covered

- Path-space MC noise reduction
  - Path Guiding
  - Path Reuse (Path-space Filtering)
- Image-space MC noise reduction
  - Image Denoising
  - Adaptive Sampling
- Learning-based MC noise reduction
  - Image-space methods
  - Sample- & Path-space methods
  - Path-guiding
  - Post-post processing