<Recent Advances in Rendering>
Monte Carlo Noise Reduction

CS482 – Interactive Computer Graphics

TA: Kyu Beom Han

qbhan@kaist.ac.kr

SGVR Lab KAIST

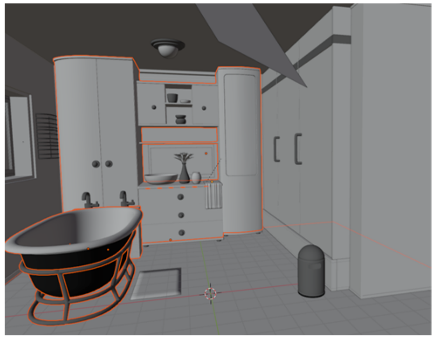
Today’s Content

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

Why Monte Carlo (Rendering) Noise Reduction?

- High complexity (2D v.s. 3D)
- Complex compatibility
  - Renderer type
  - Asset type
  - HW type

3D asset



Graphics Pipeline

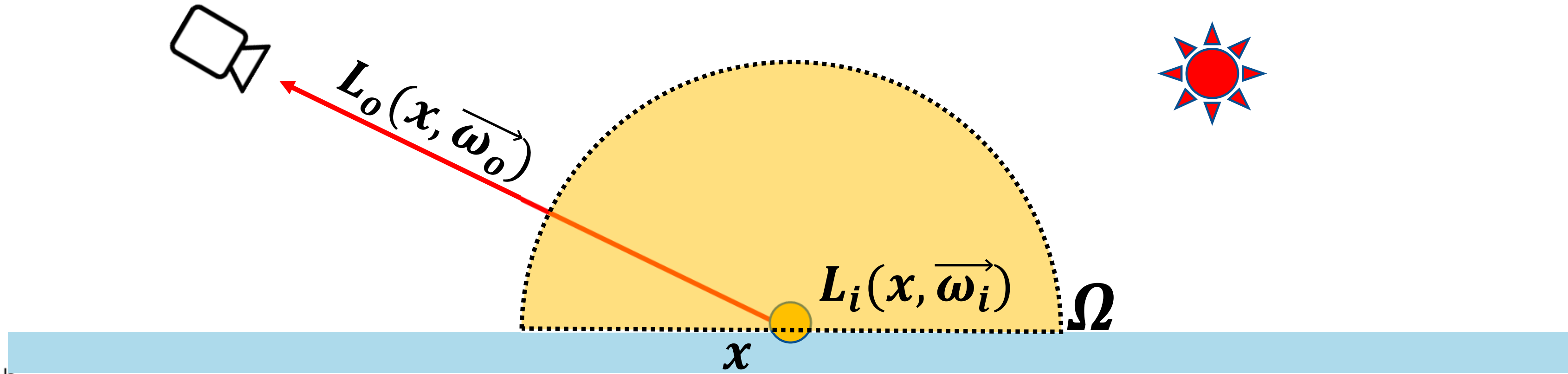
- Vertex Program
- Geometry Program
- Geometry Post-process
- Rasterize Ray-tracing
- Shading

Noisy Image

Clean Image

Review - Rendering Equation

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

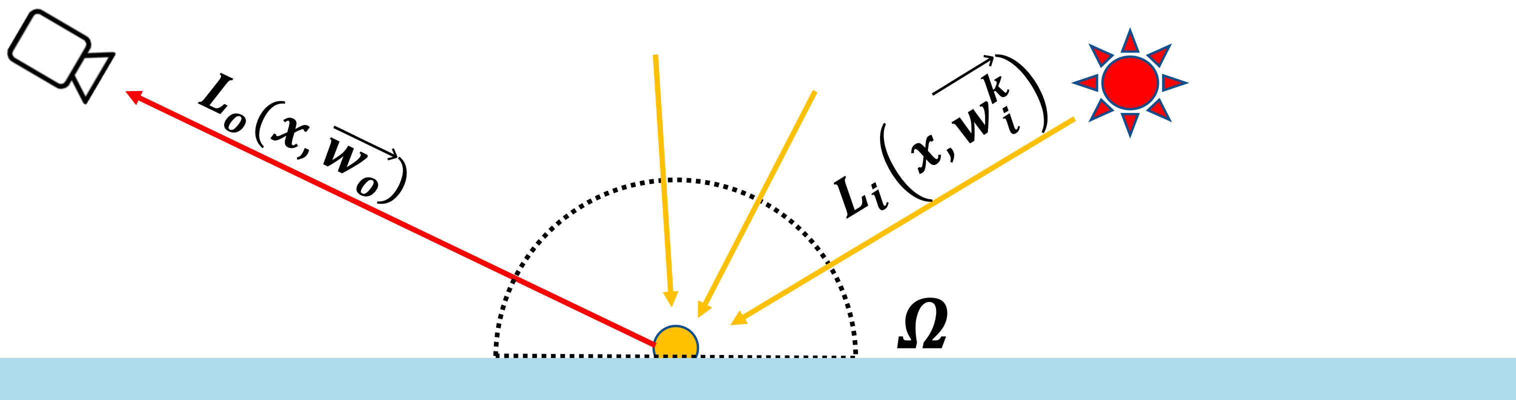


Review – MC Ray Tracing

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

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

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



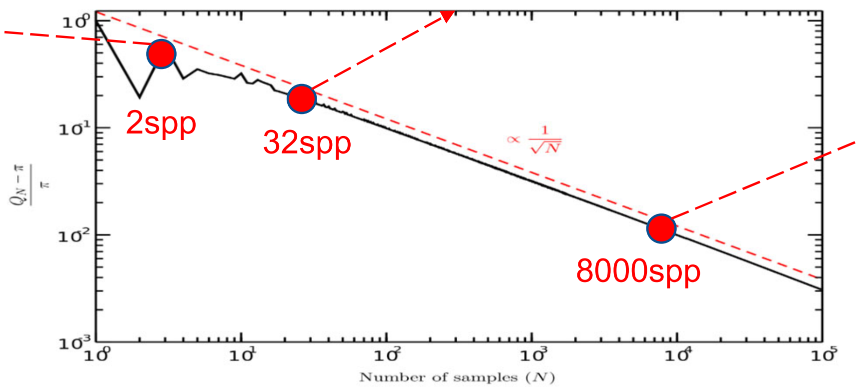
Review – MC Ray Tracing and MC Noise

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

2.5s

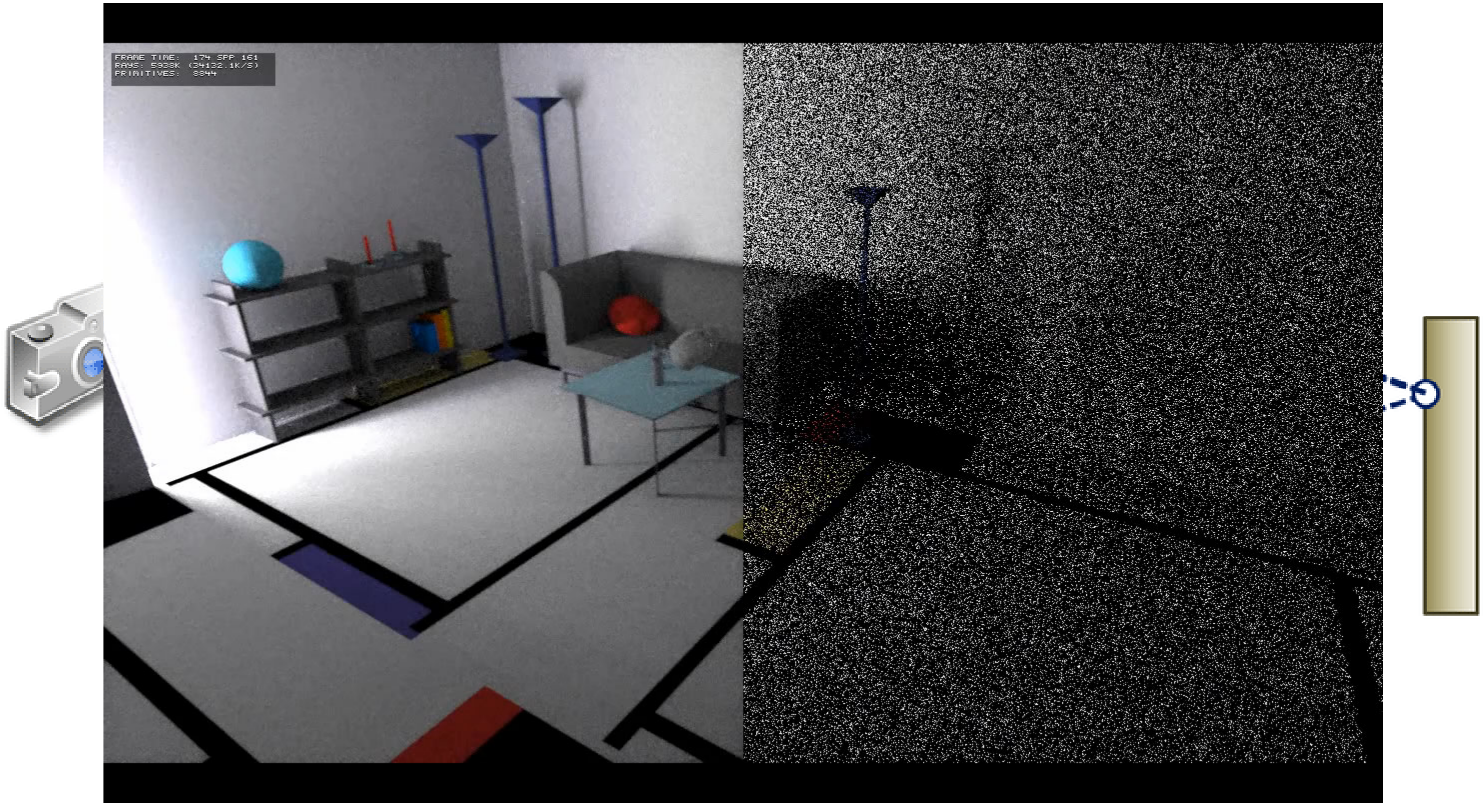
8.9s

18m



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.

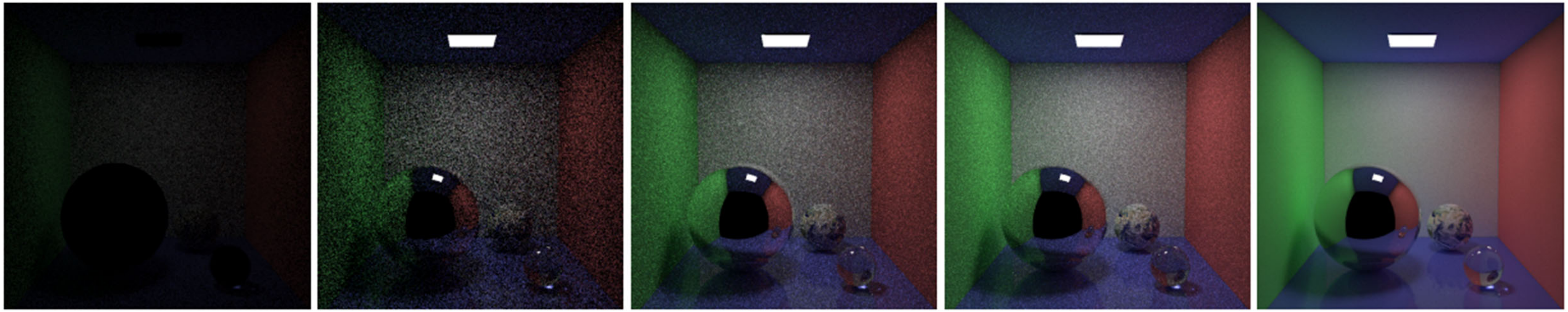


Ray Tracey's blog: Real-time Metropolis Light Transport on the GPU: it works!!!!

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

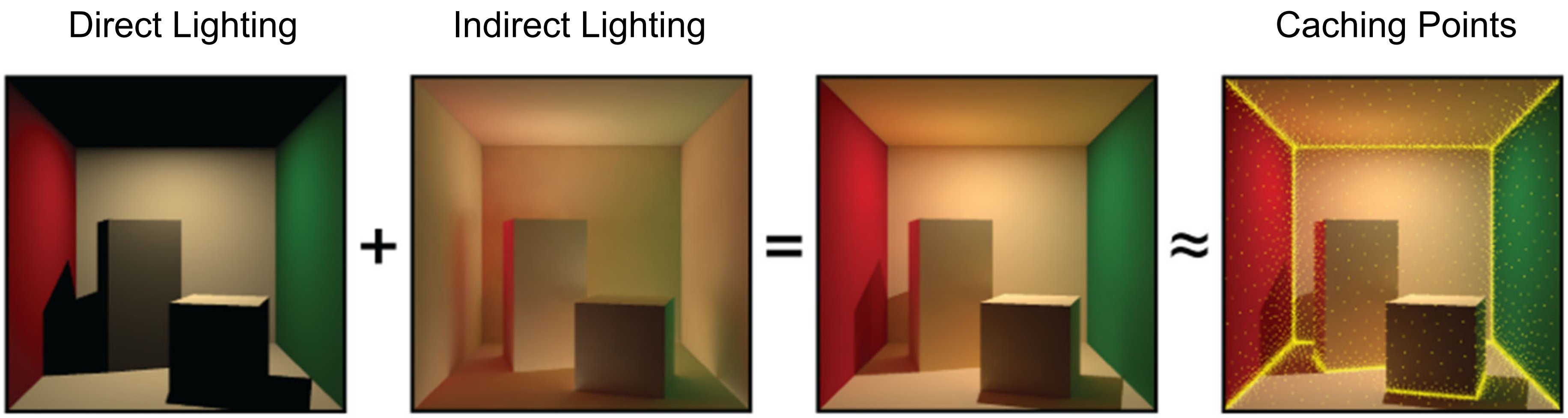
eye point

light source

Bidirectional Path Tracing, Michal Vlnas, Proceedings of CESCG 2018

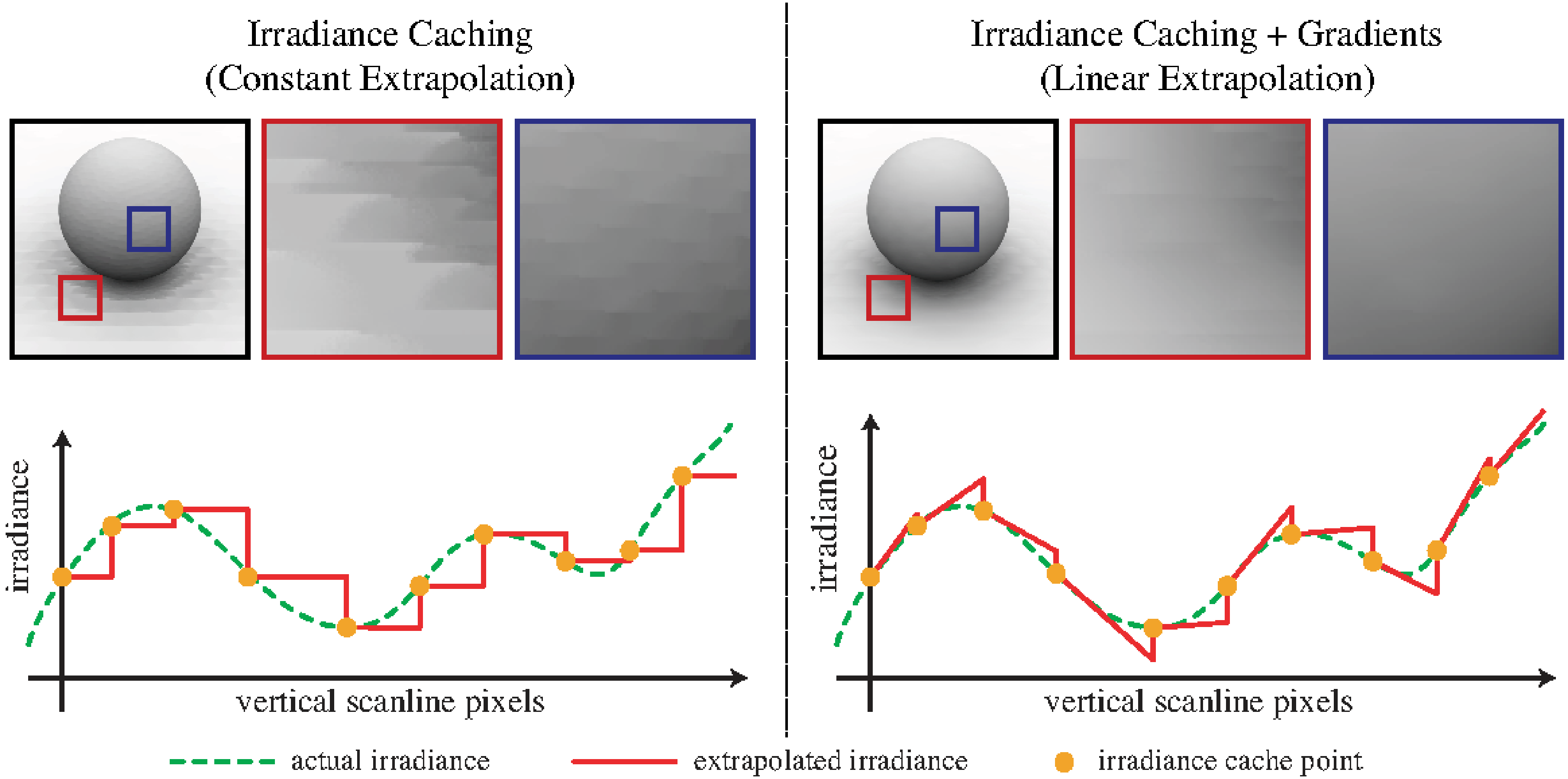
Review - Irradiance Caching

- 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)

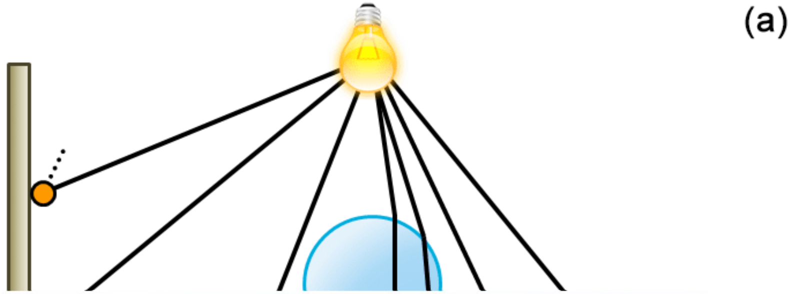


Figure 3: The Museum scene

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

SGVR Lab
KAIST

Global Illumination using Photon Maps, Jensen et al., EGWR 1996
Physically based Computer Graphics for Realistic Image Formation to Simulate Optical Measurement Systems, Retzlaff et al., JSSS 2017

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)

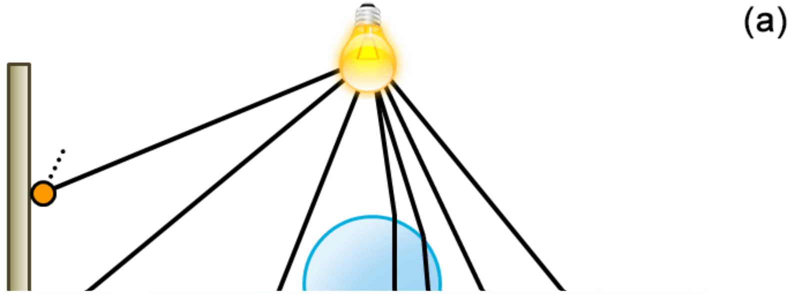


Figure 3: The Museum scene

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

SGVR Lab
KAIST

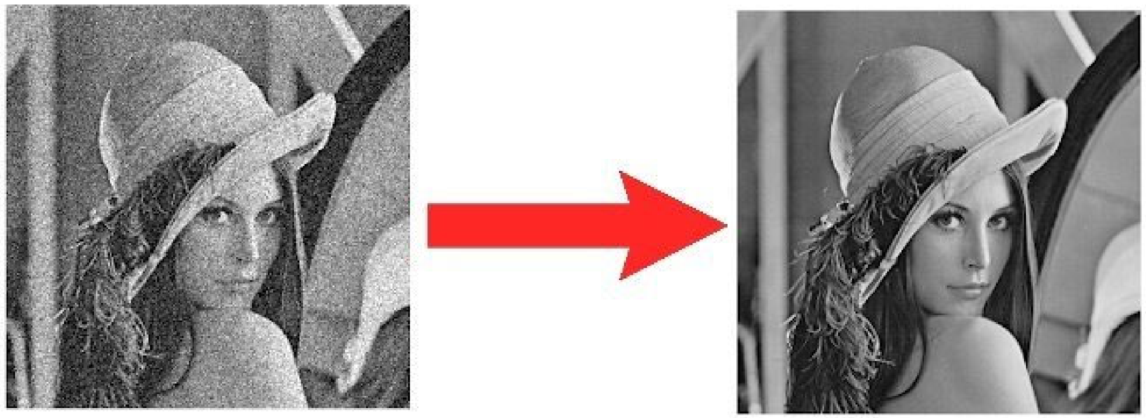
Global Illumination using Photon Maps, Jensen et al., EGWR 1996
Physically based Computer Graphics for Realistic Image Formation to Simulate Optical Measurement Systems, Retzlaff et al., JSSS 2017

Content

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

Image-space MC Noise Reduction

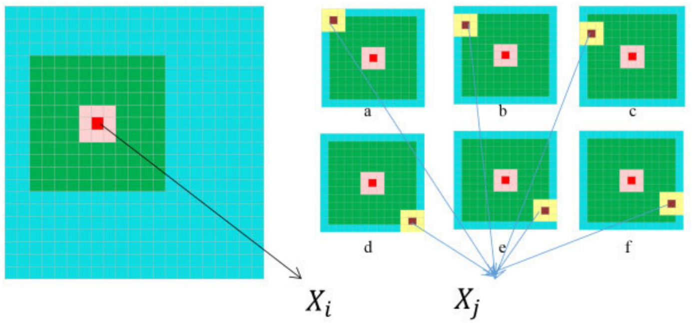
- Efficiently dealing noise on image-space, similar to general image denoising
- Reducing working space from N-dim path-space to 2-dim image space



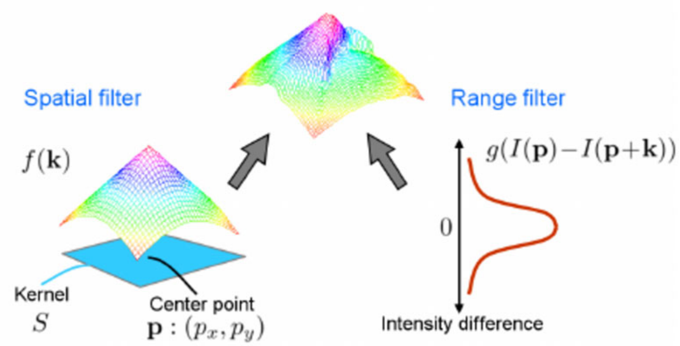
Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings, Bitterli et al., CGF 2016

General Image Denoising Algorithms for MC Rendering

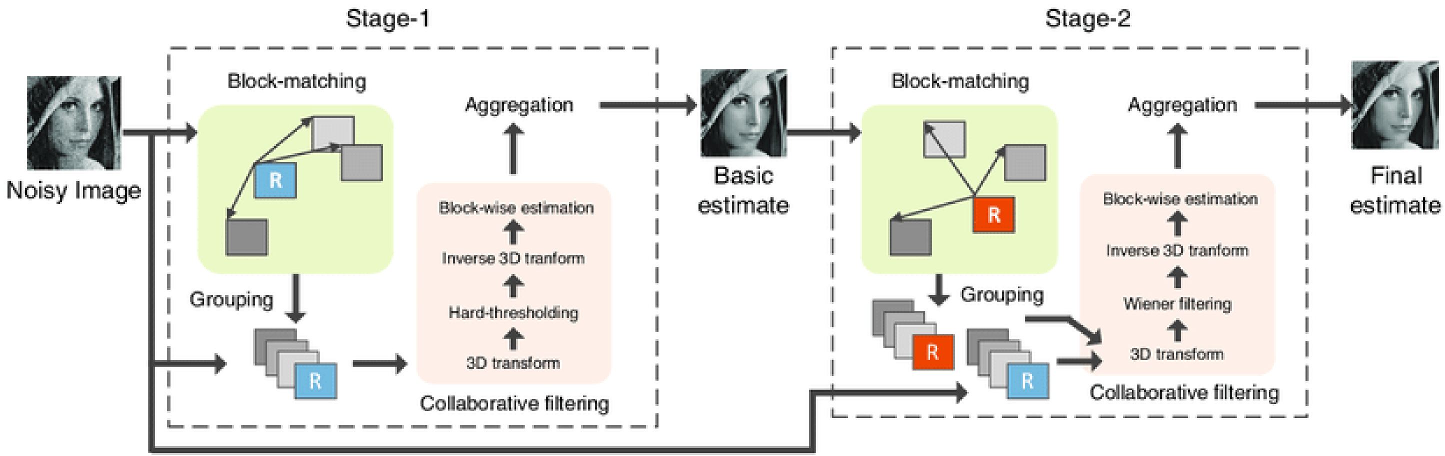
- Efficiently dealing noise on image-space, similar to general image denoising
- Reducing working space from N-dim path-space to 2-dim image space



Non-local Means Filter



Bilateral Filter



Block-Matching 3D (BM3D)

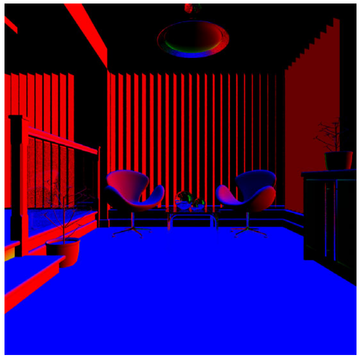
Removing the Noise in Monte Carlo Rendering with General Image Denoising Algorithms, Kalantari et al., CGF 2013

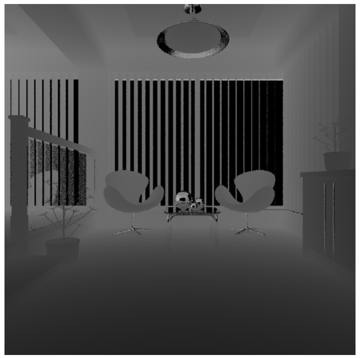
General Image Denoising Algorithms for MC Rendering

- 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

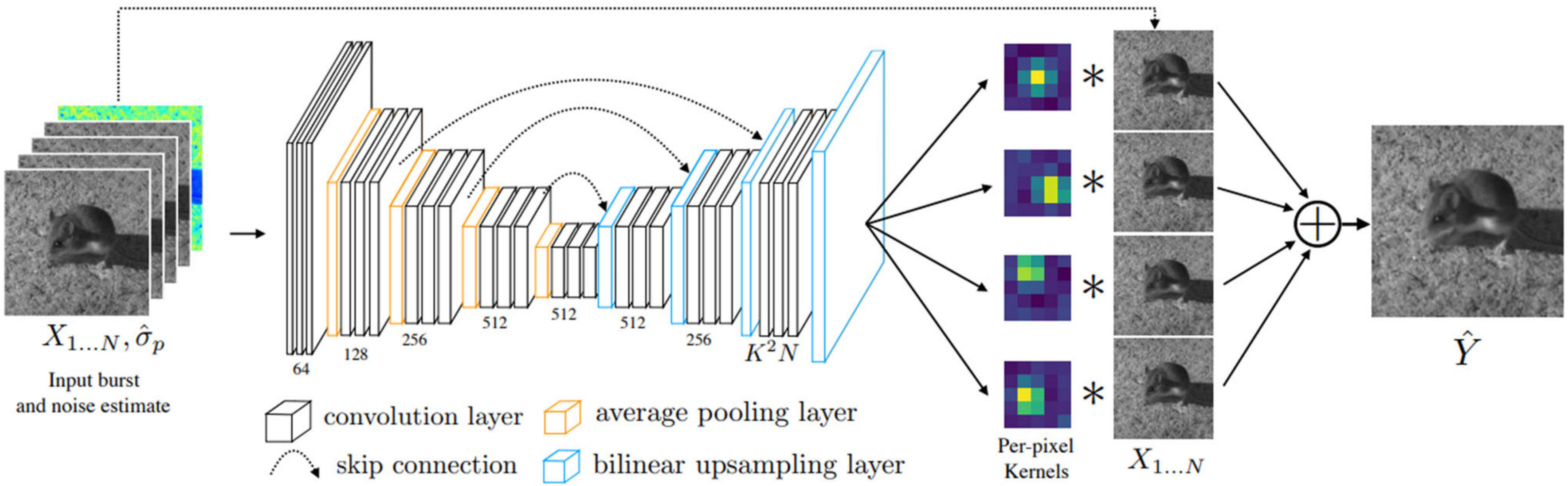
- Path-space MC noise reduction

- 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

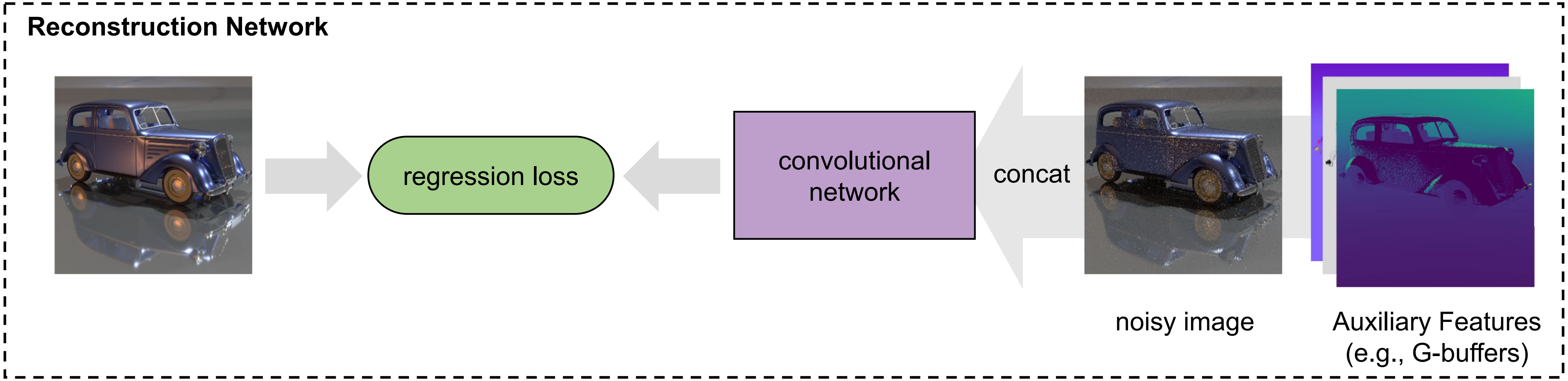
- Various neural networks (MLP, ConvNets, Transformers, etc.) and training strategies (supervised, self-supervised, unsupervised, etc.) are introduced during the last decade
- Reduce design biases of traditional denoising filters



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

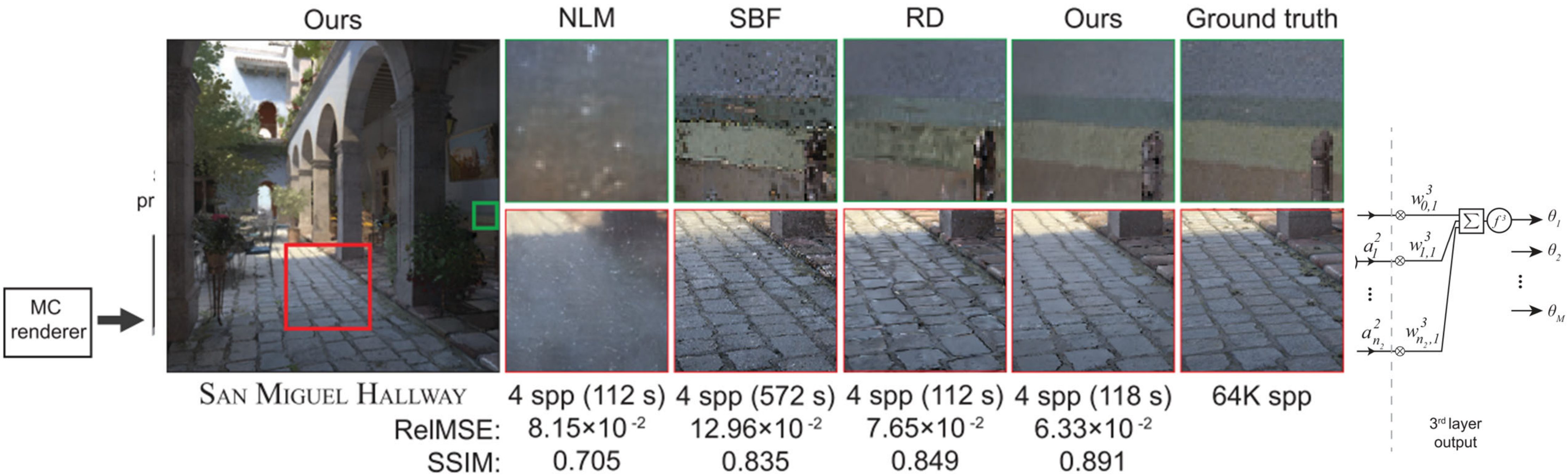
Conventional Configuration for Learning-based Methods

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



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

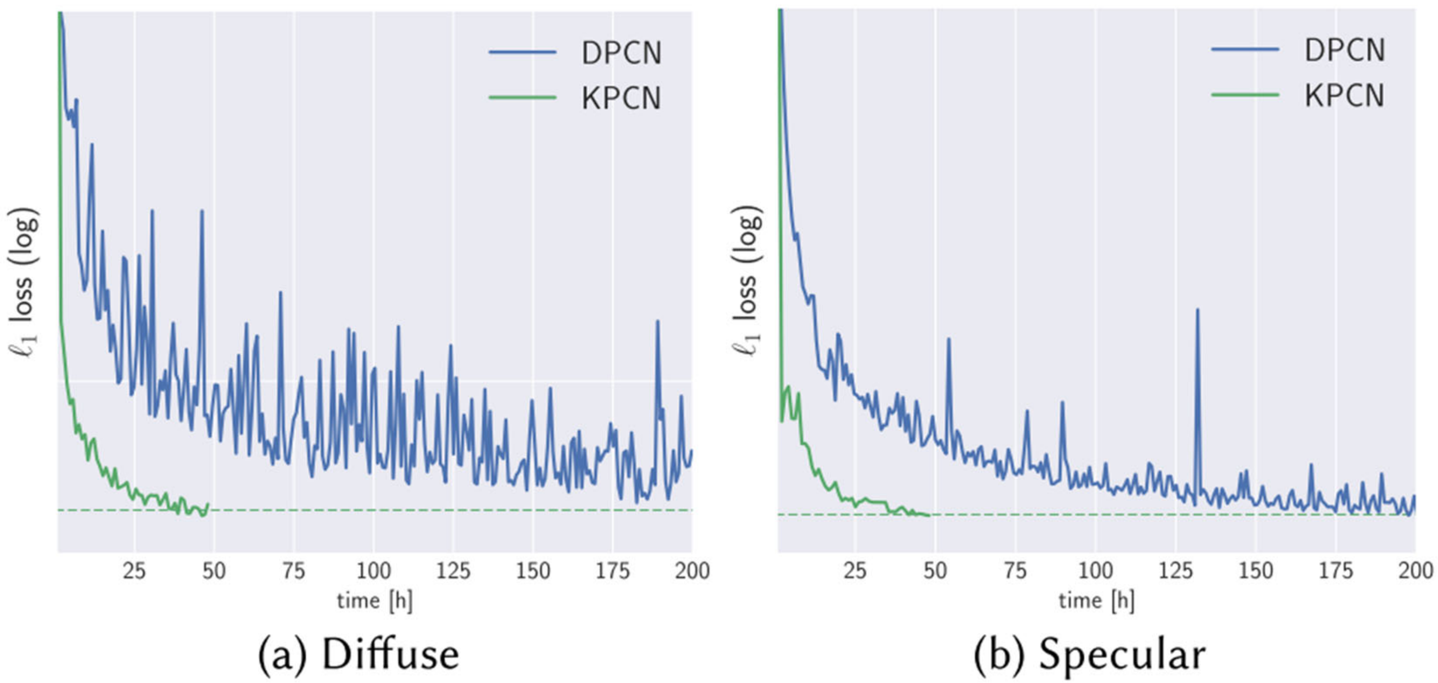


SGVR Lab KAIST

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

Predicting Kernel Weights using CNN

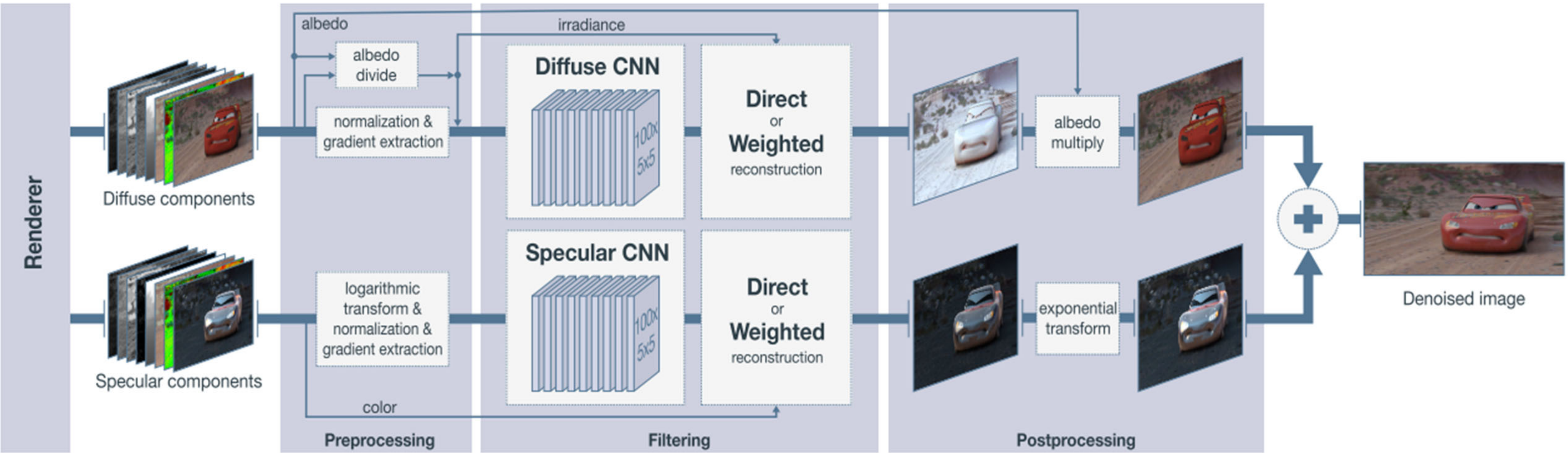
- Robust training by training the network to predict the denoising kernels (KPCN) instead of denoised pixel value (DPCN)
  - Reduces the search space (pixel radiance : 0 ~ unlimited, kernel weights: 0~1)



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

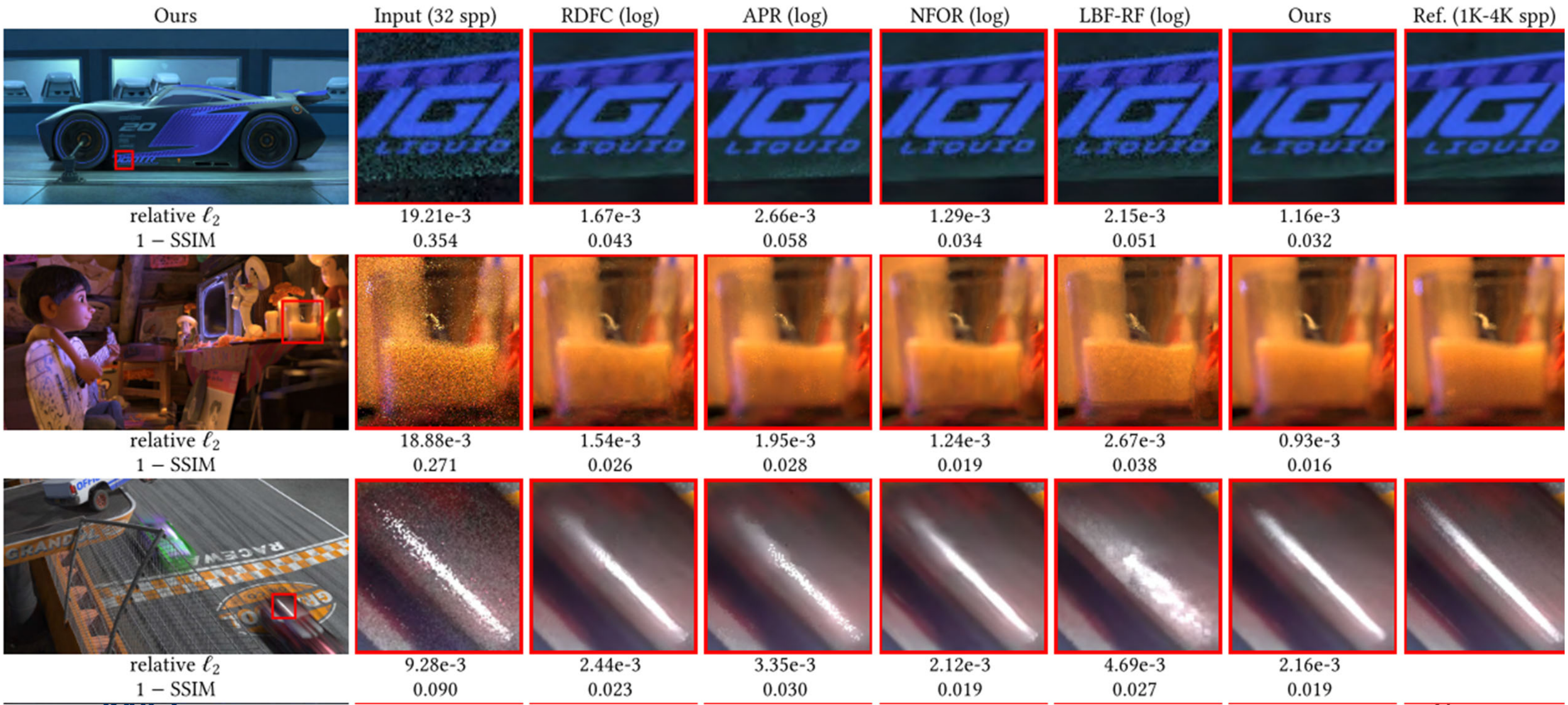
Decompose to Diffuse and Specular

- 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

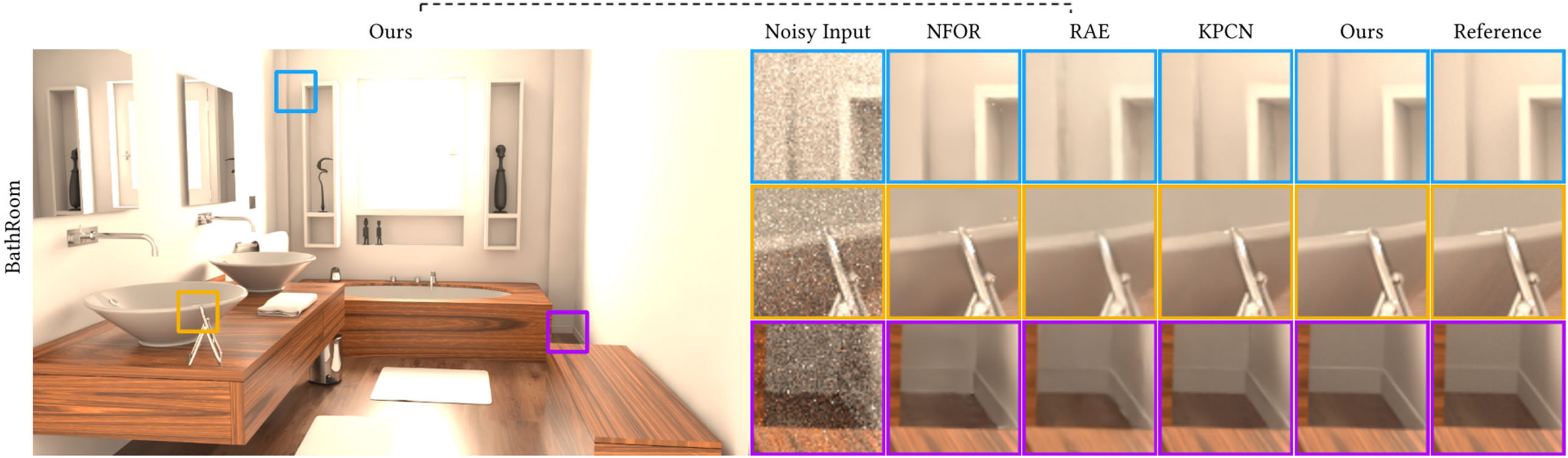
Kernel-predicting Convolutional Network (KPCN)



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



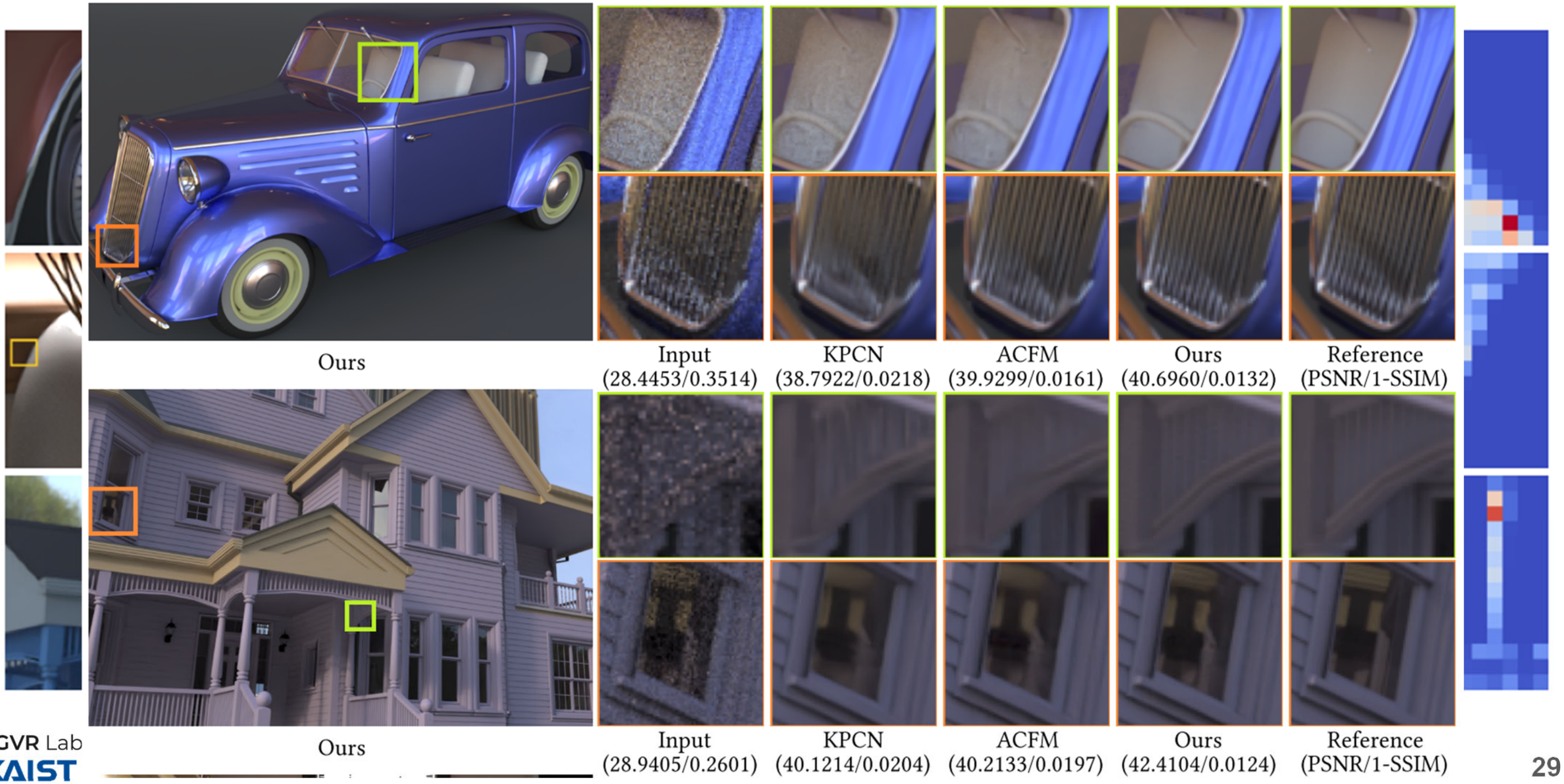
	Noisy Input	NFOR	RAE	KPCN	Ours
1-SSIM	(0.498707)	(0.023738)	(0.036035)	(0.019587)	(0.015482)
RMSE	(0.015005)	(0.000601)	(0.000909)	(0.000411)	(0.000254)
PSNR	(23.433917)	(38.139083)	(36.114381)	(39.167406)	(41.241905)

SGVR Lab KAIST

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

Feature-guided Self-attention (AFGSA)

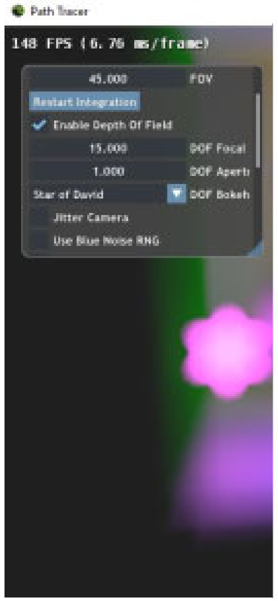
- Stack multiple transformer blocks that creates self-attention map from input image and auxiliary features

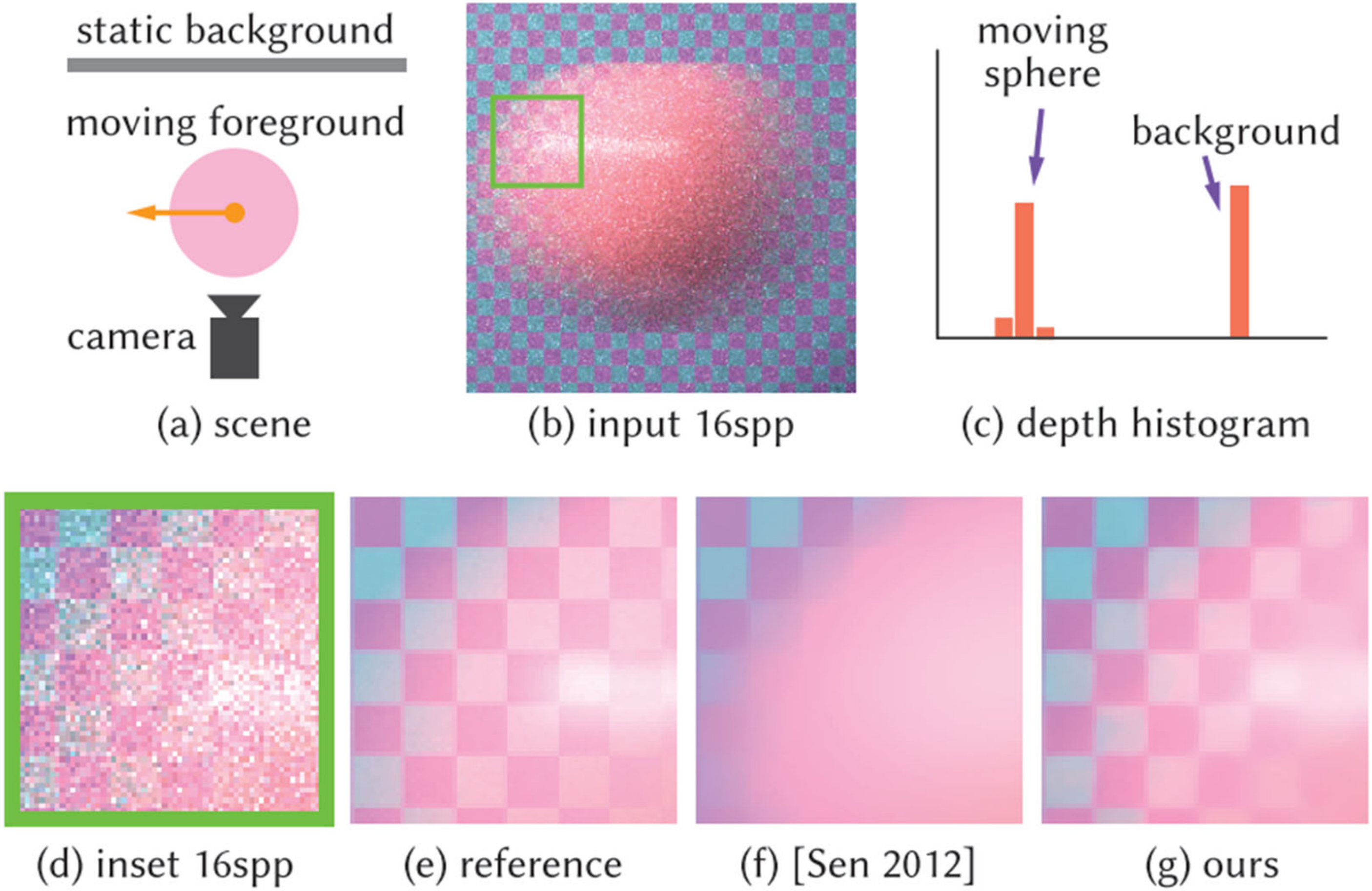


Adversarial Monte Carlo Denoising with Conditioned Auxiliary Feature Modulation, Yu et al., ToG 2021

Jumping from Image-space to Sample-space

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

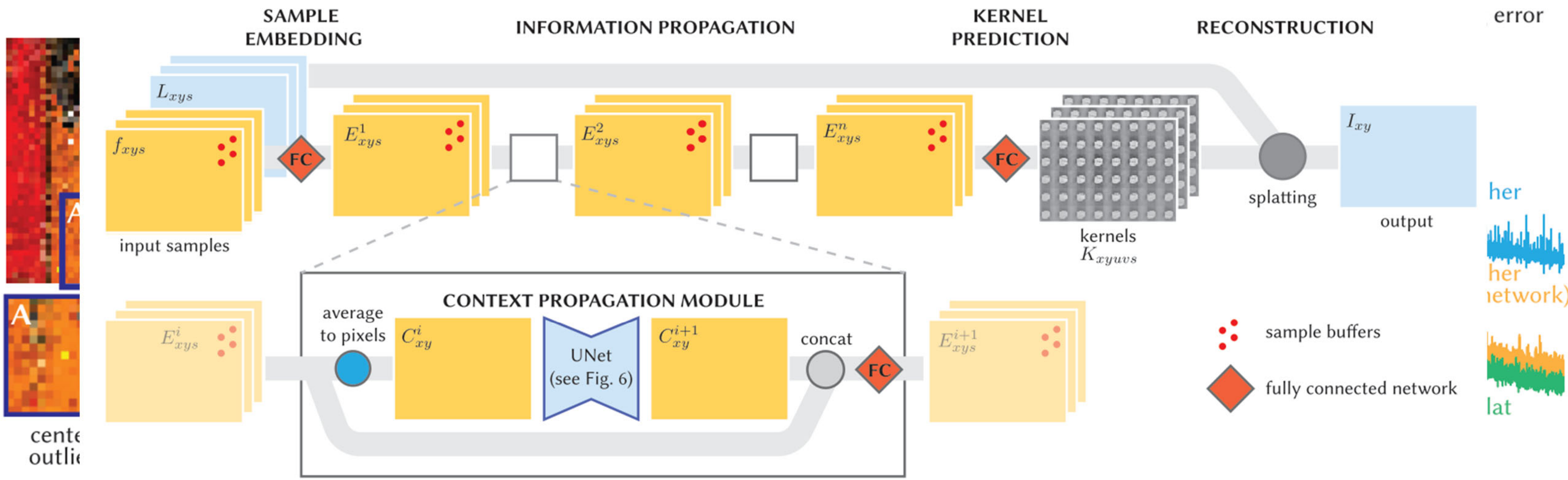




Sample-based Monte Carlo Denoising using a Kernel-splatting Network, Gharbi et al., ToG 2019

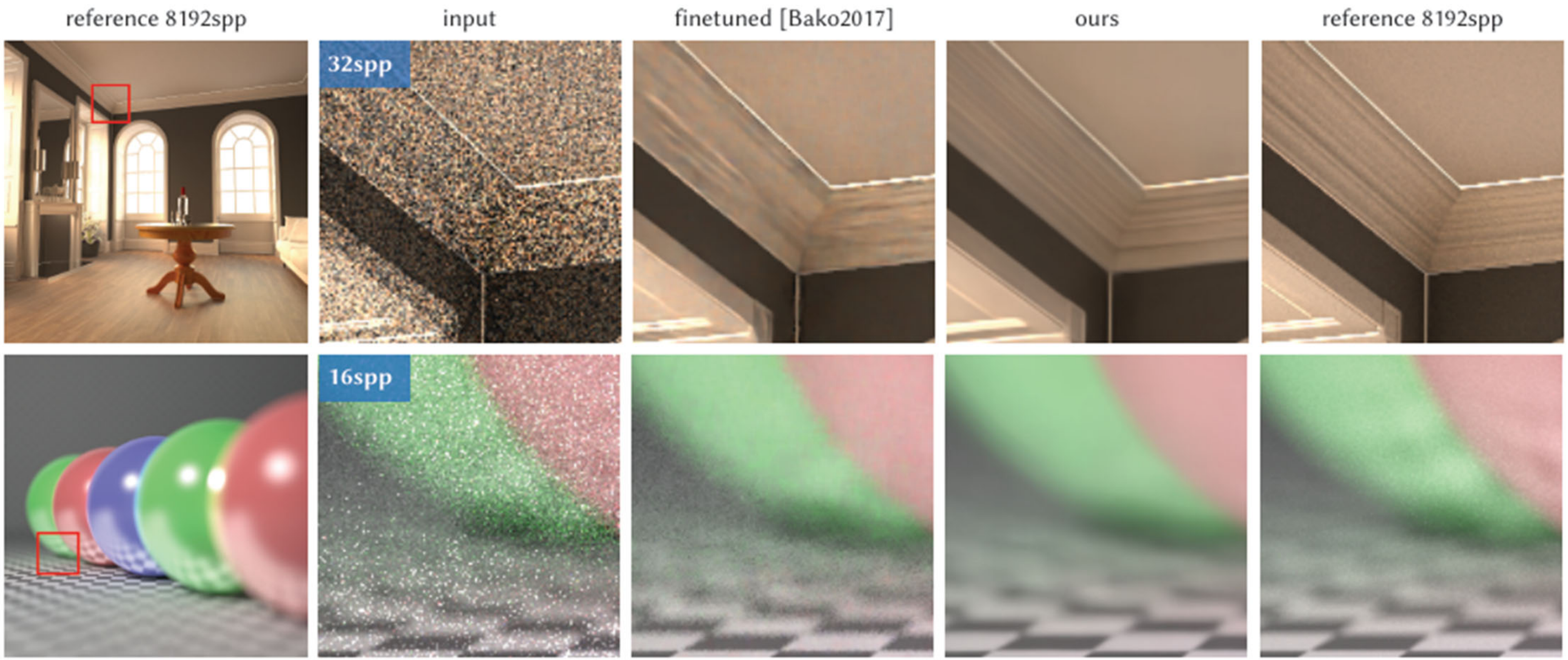
Splatting Kernel for Samples

- 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)

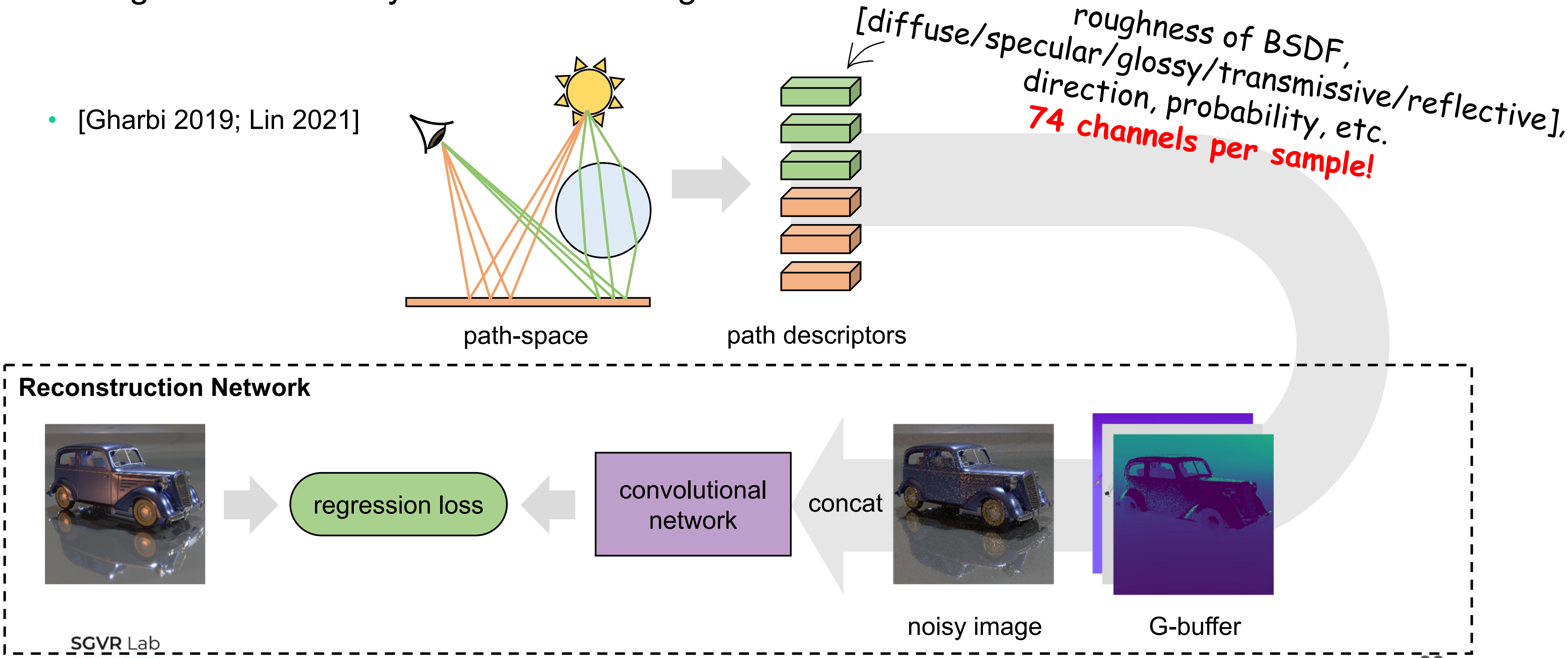


Sample-based Monte Carlo Denoising using a Kernel-splatting Network, Gharbi et al., ToG 2019

Path-space Features for Denoising

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

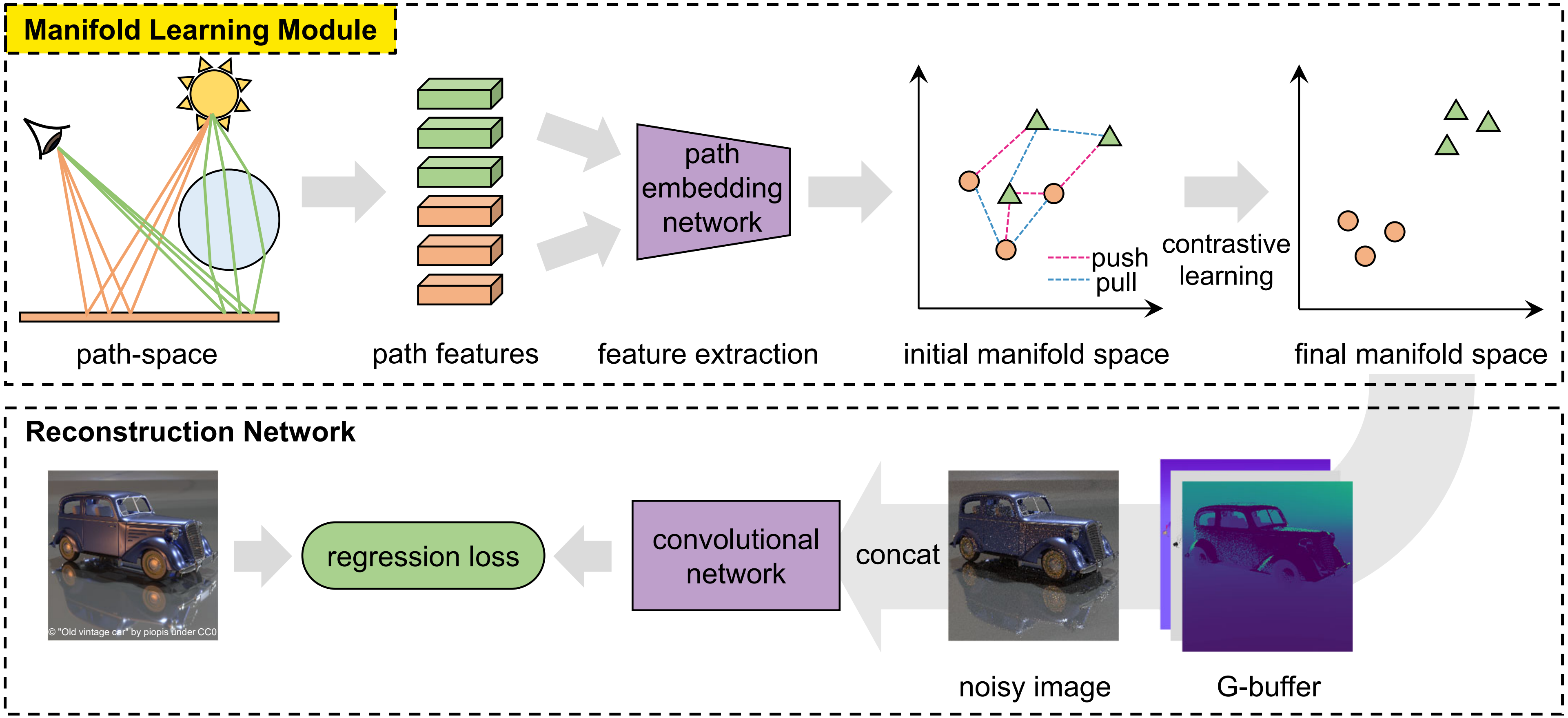
- [Gharbi 2019; Lin 2021]



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

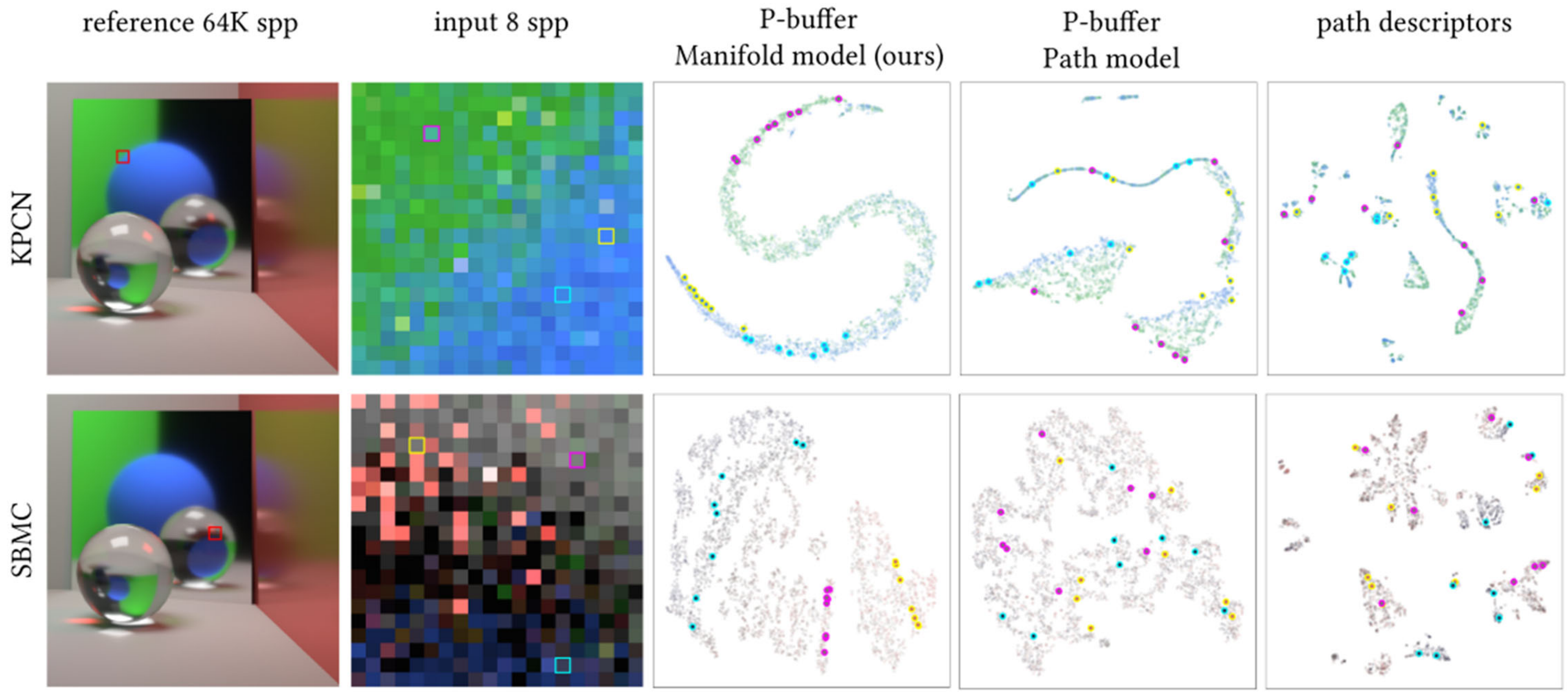
Manifold Learning for Path-space Features

- Embed path features to low-dimensional space

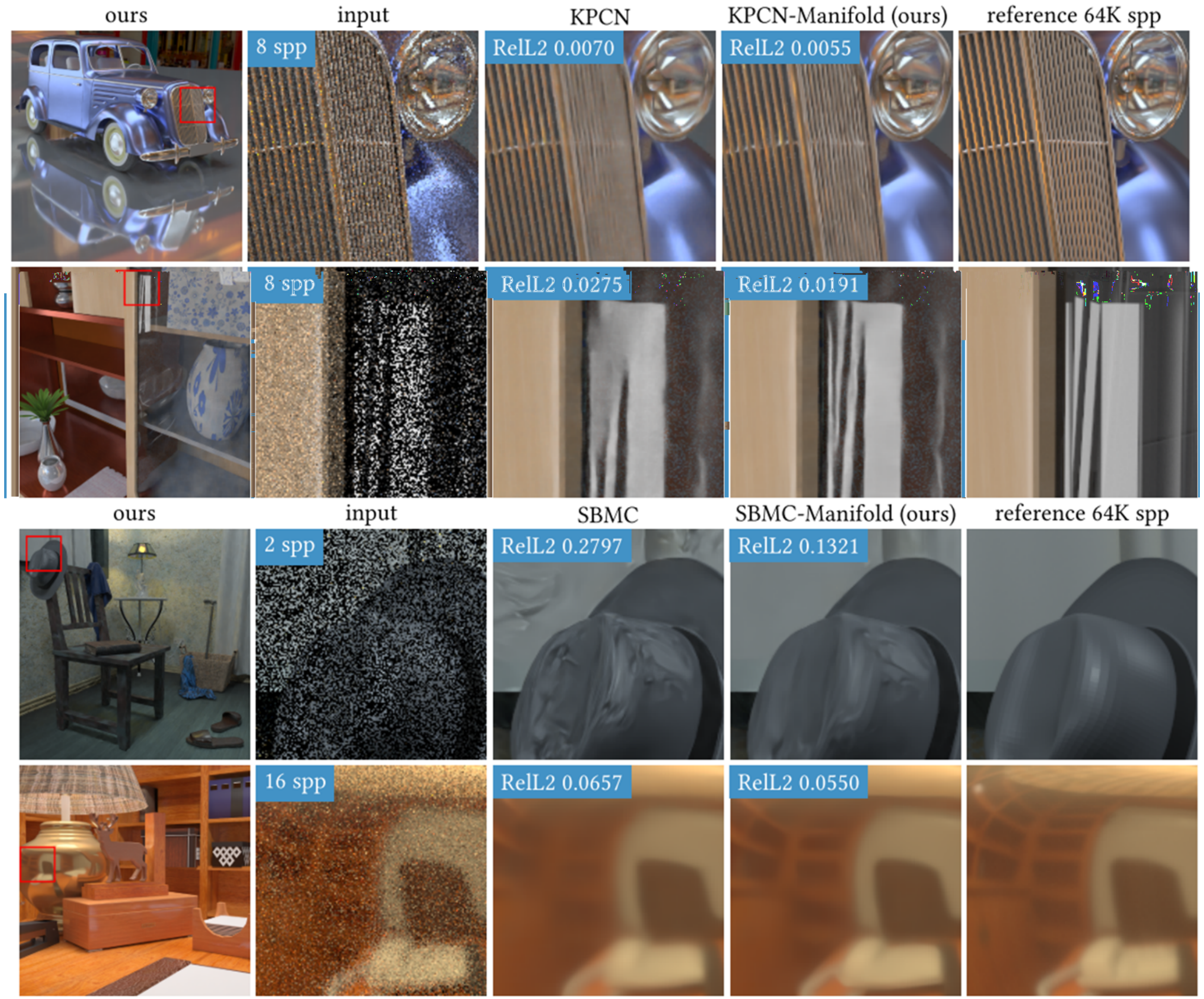


Manifold Learning for Path-space Features

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



Manifold Learning for Path-space Features



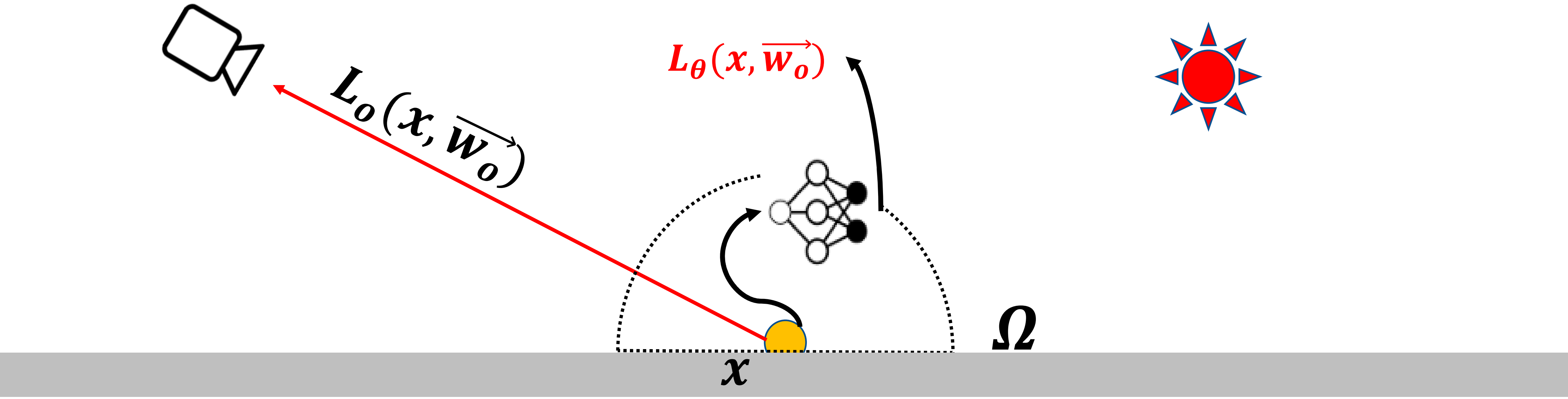
Weakly-supervised Contrastive Learning in Path Manifold for Monte Carlo Image Reconstruction, Cho et al., ToG 2021

Neural Radiance Caching

- Solving rendering equation via Radiance-predicting Neural Network L_θ

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

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



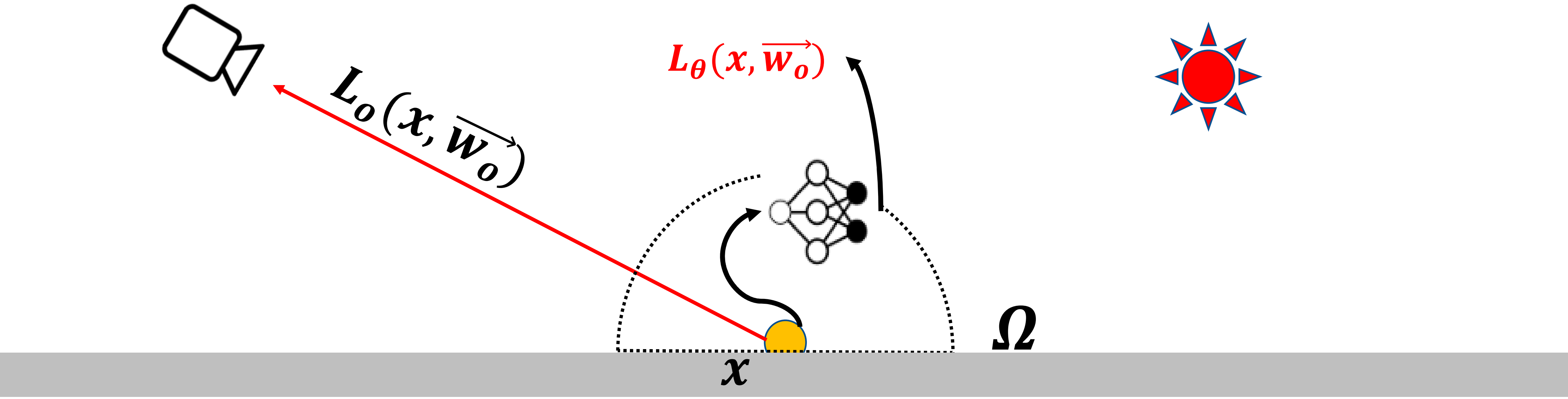
Real-time Neural Radiance Caching for Path Tracing, Muller et al., ToG 2021

Neural Radiance Caching

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

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

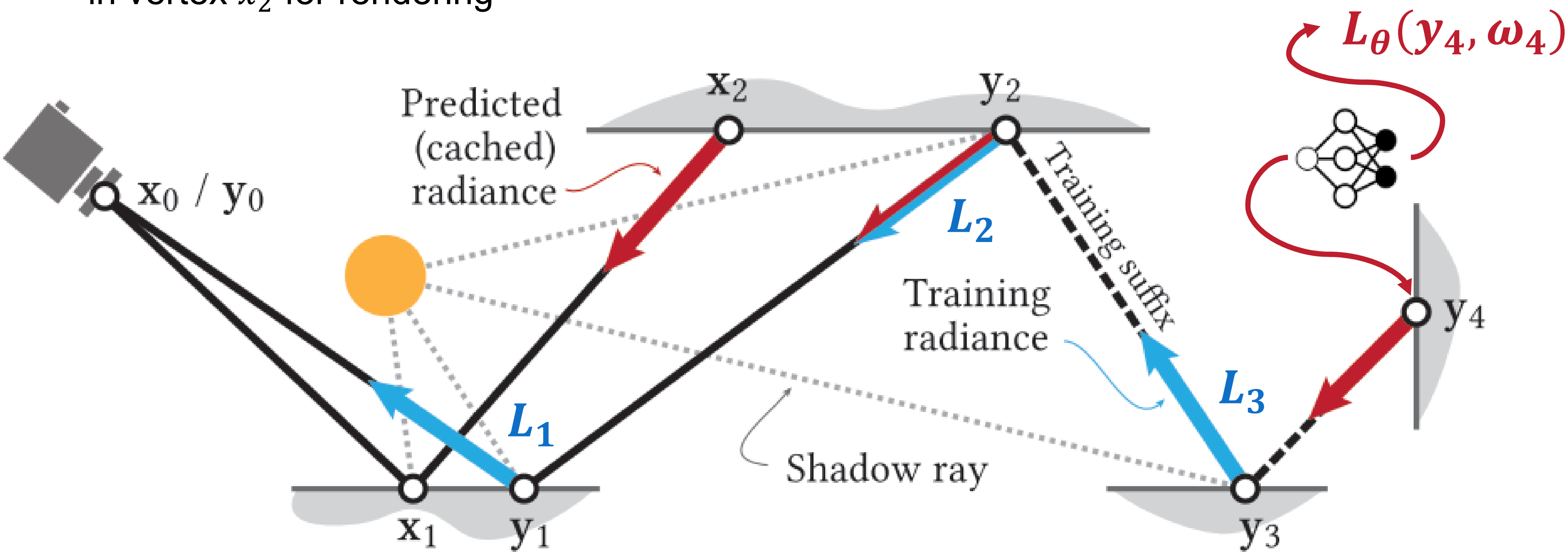
$$\sim L_e(x,\overrightarrow{w_o}) + L_\theta(x,\overrightarrow{w_o})$$



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

Self-training for Neural Radiance Cache

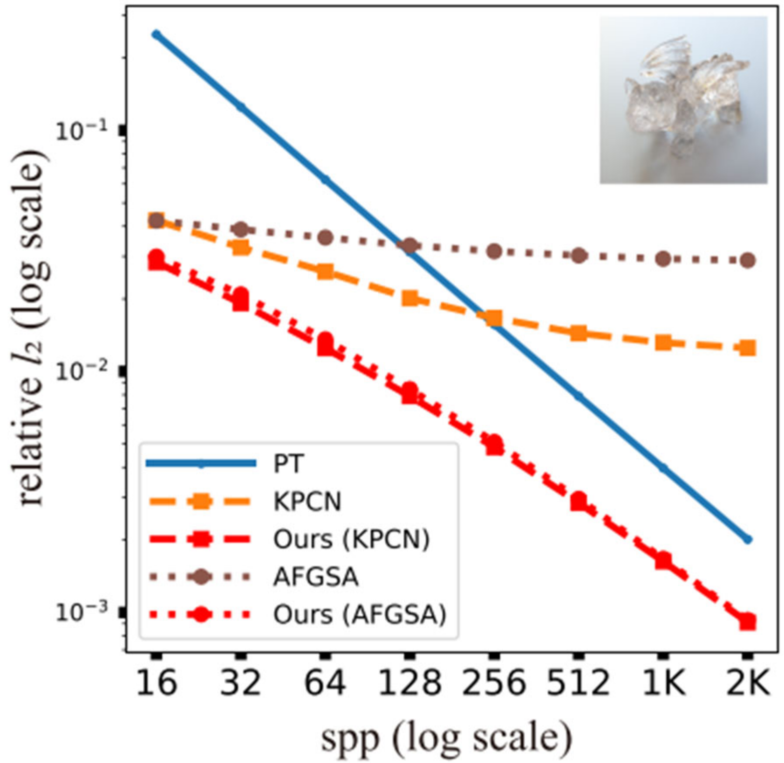
- Minimize the loss between the calculated radiances and the estimated radiances of the preceding vertices
- $Loss = relL2(L_1, L_\theta(y_1, \omega_1)) + relL2(L_2, L_\theta(y_2, \omega_2)) + relL2(L_3, L_\theta(y_3, \omega_3))$
- Trace a short rendering path $(x_0 x_1 x_2)$ where we used the cached(estimated)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



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

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

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

Combining Biased and Unbiased Estimates

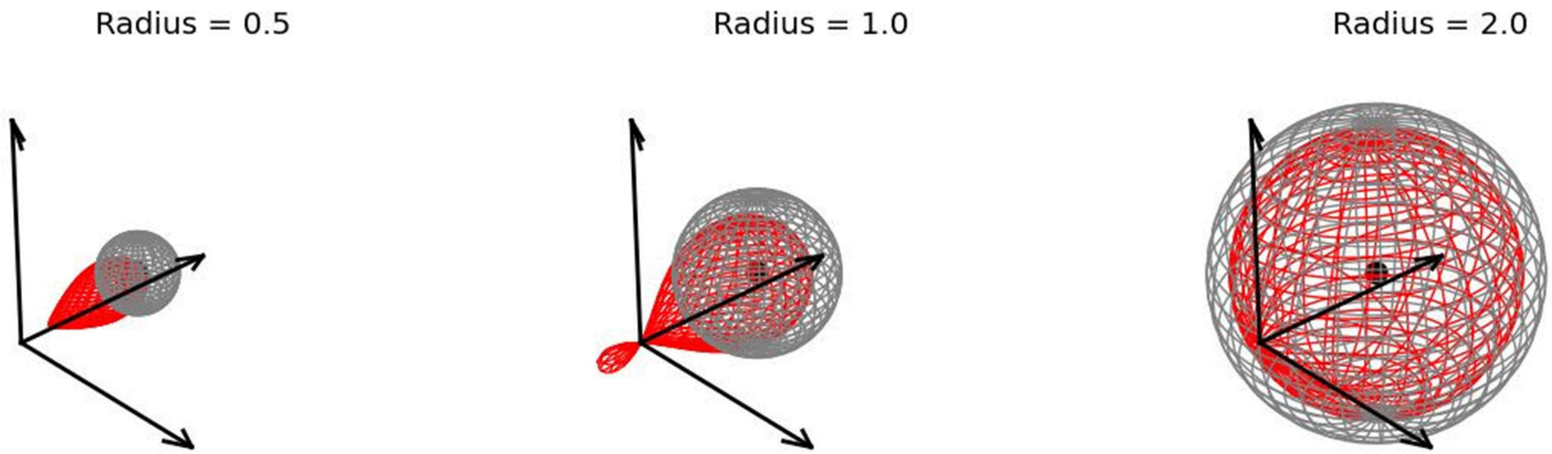
- Path Traced Result X : Noisy but Unbiased (Bias ↓, Variance ↑)
- Denoised Result Y: Smooth but Biased (Bias ↑, Variance ↓)
- James-Stein Estimator shrinks X towards Y as $\delta(X,Y) = Y + \left(1 - \frac{(p-2)\sigma^2}{\|X-Y\|^2}\right)(X - Y)$
  - p : Dimension of estimation (3 = RGB channel), σ: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

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 $p \geq 3$

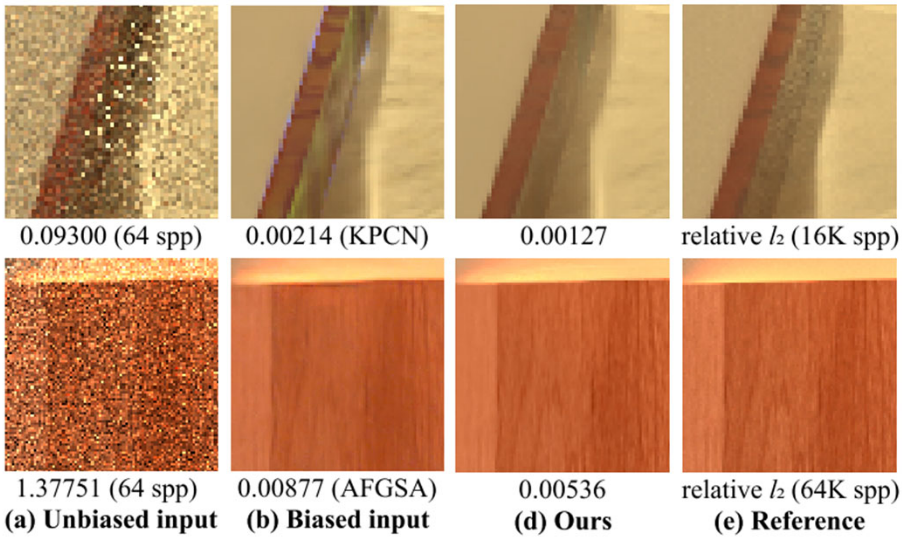


James-Stein Estimator shows less MSE error
Grey – Sampled points on radius sphere with center (1, 1, 1)
Red – James-Stein estimator applied on sampled points

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 $p \geq 3$

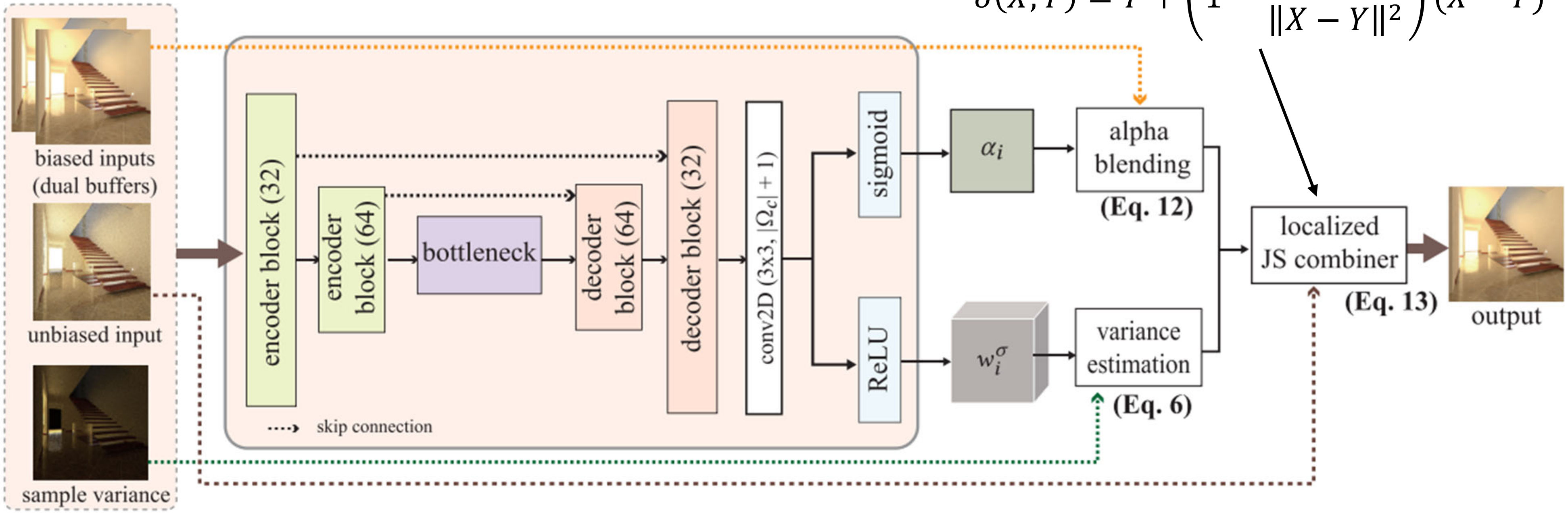


<Intuition>
Balancing the bias and variance of between path traced result and denoised result

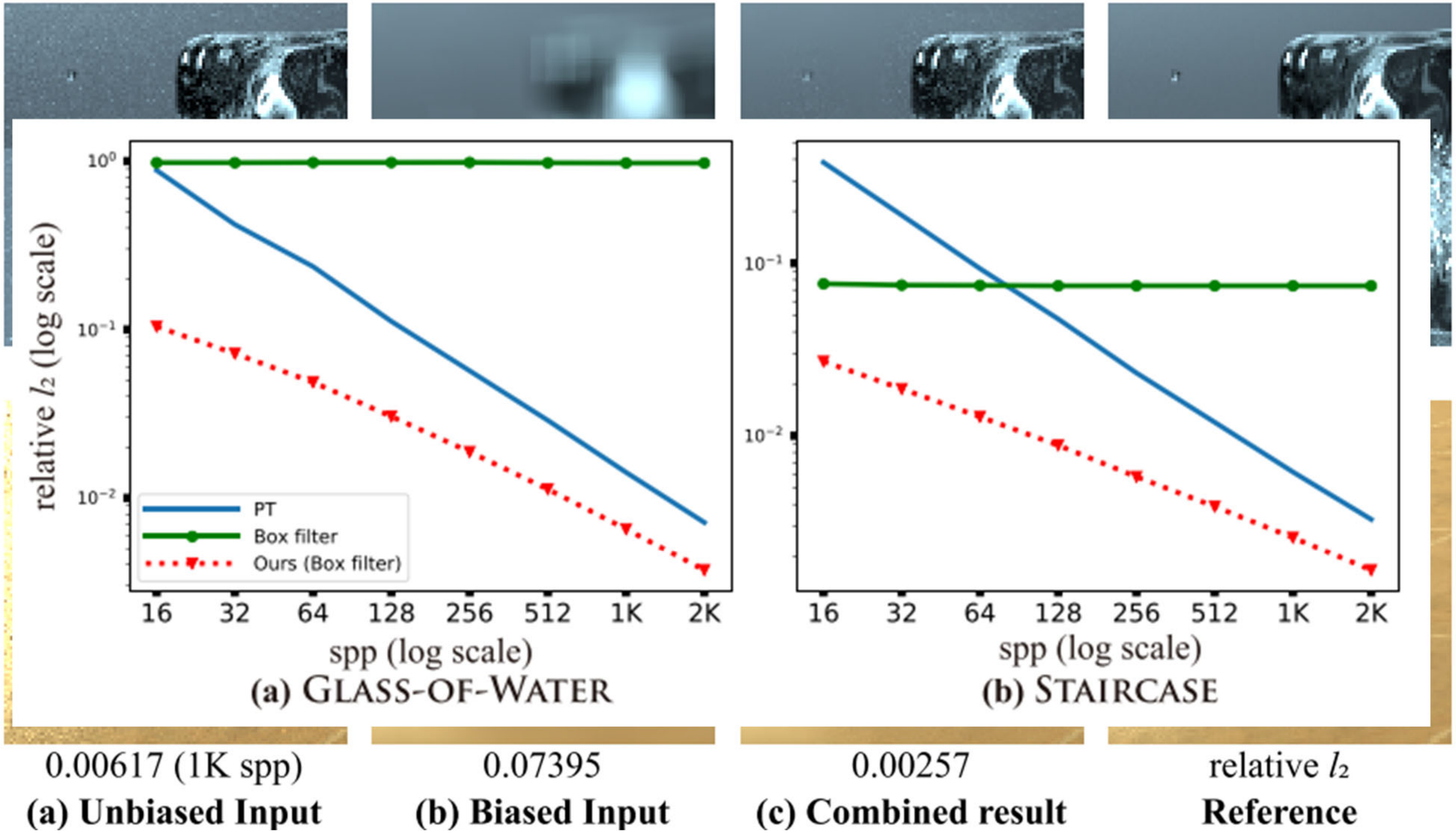
Neural James-Stein Combiner

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

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



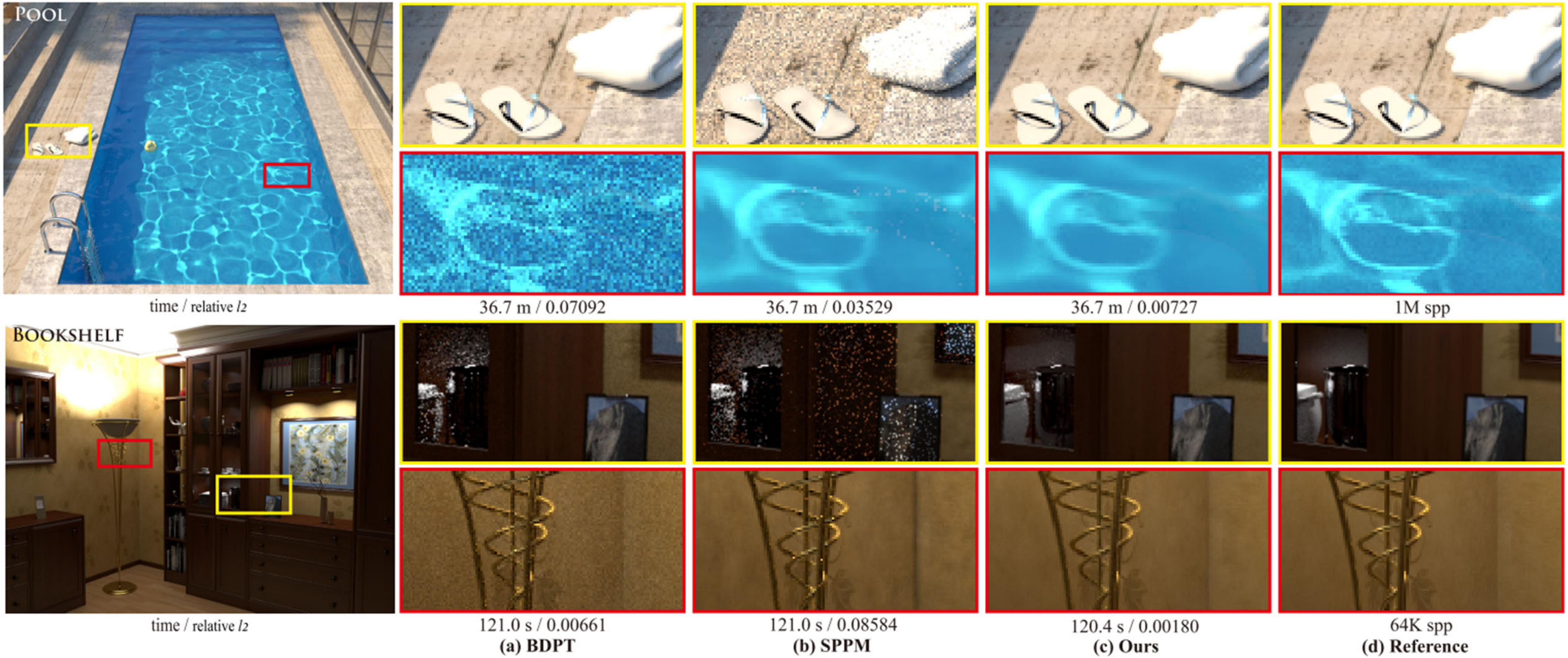
Neural James-Stein Combiner



SGVR Lab KAIST

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

- Image-space MC noise reduction

- Learning-based MC noise reduction
  - Image-space methods
  - Sample- & Path-space methods
  - Post-post processing