Denoising

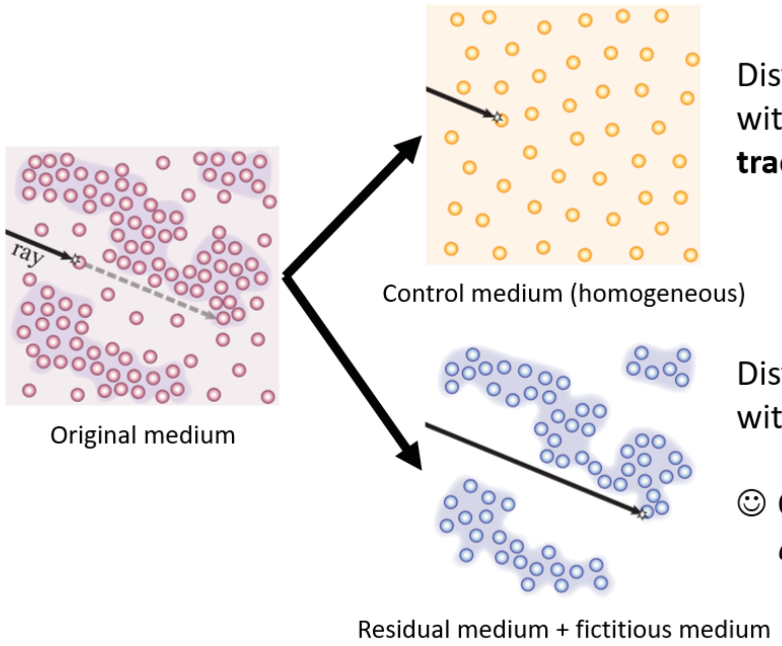
20130501 이철민(Lee CheolMin)

Review

- Spectral and Decomposition Tracking for Rendering Heterogeneous Volumes

1. Decompose original medium into homogeneous and residual

Decomposition tracking

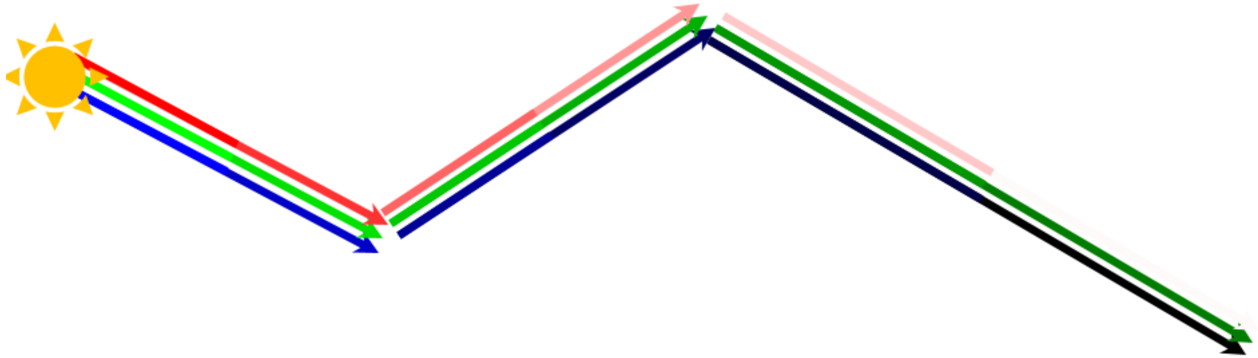


Distance sampling with **closed-form tracking**

Distance sampling with **delta tracking**

☺ Only lookup if $d_r < d_c$

2. Spectral Tracking



Review

- Lighting Grid Hierarchy for Self-illuminating Explosions

How to shade with so many point lights?

1. Building LGH

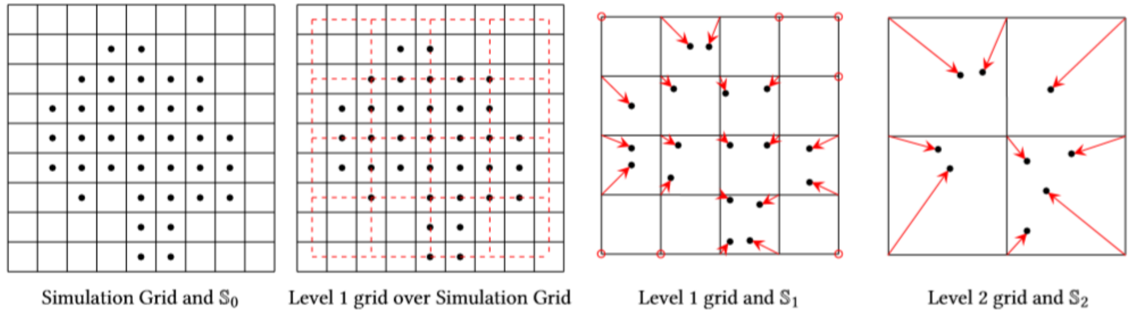
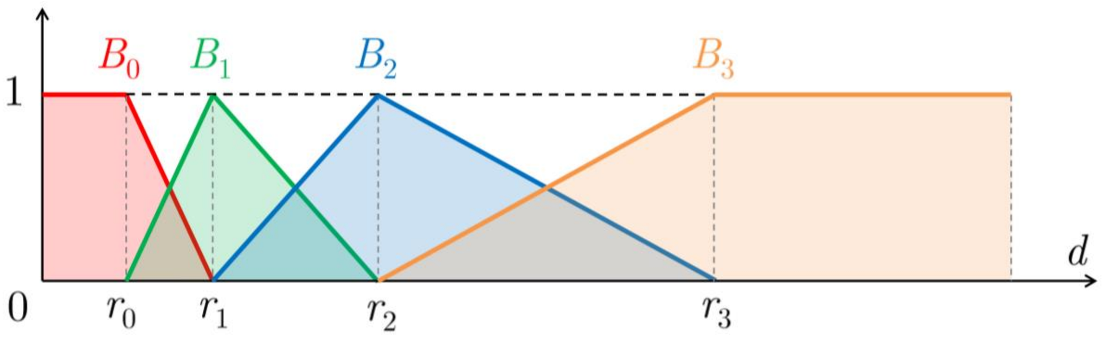


Fig. 2. Our lighting grid hierarchy for explosion rendering. We begin with the explosion simulation grid and generate point lights (shown as black dots) in voxels with high temperature values. We place the highest resolution (level 1) lighting grid, such that vertices of the grid are aligned with voxel centers. For each vertex of the lighting grid at any level, we keep the illumination center, shown as black dots along with offset arrows from their grid vertices.

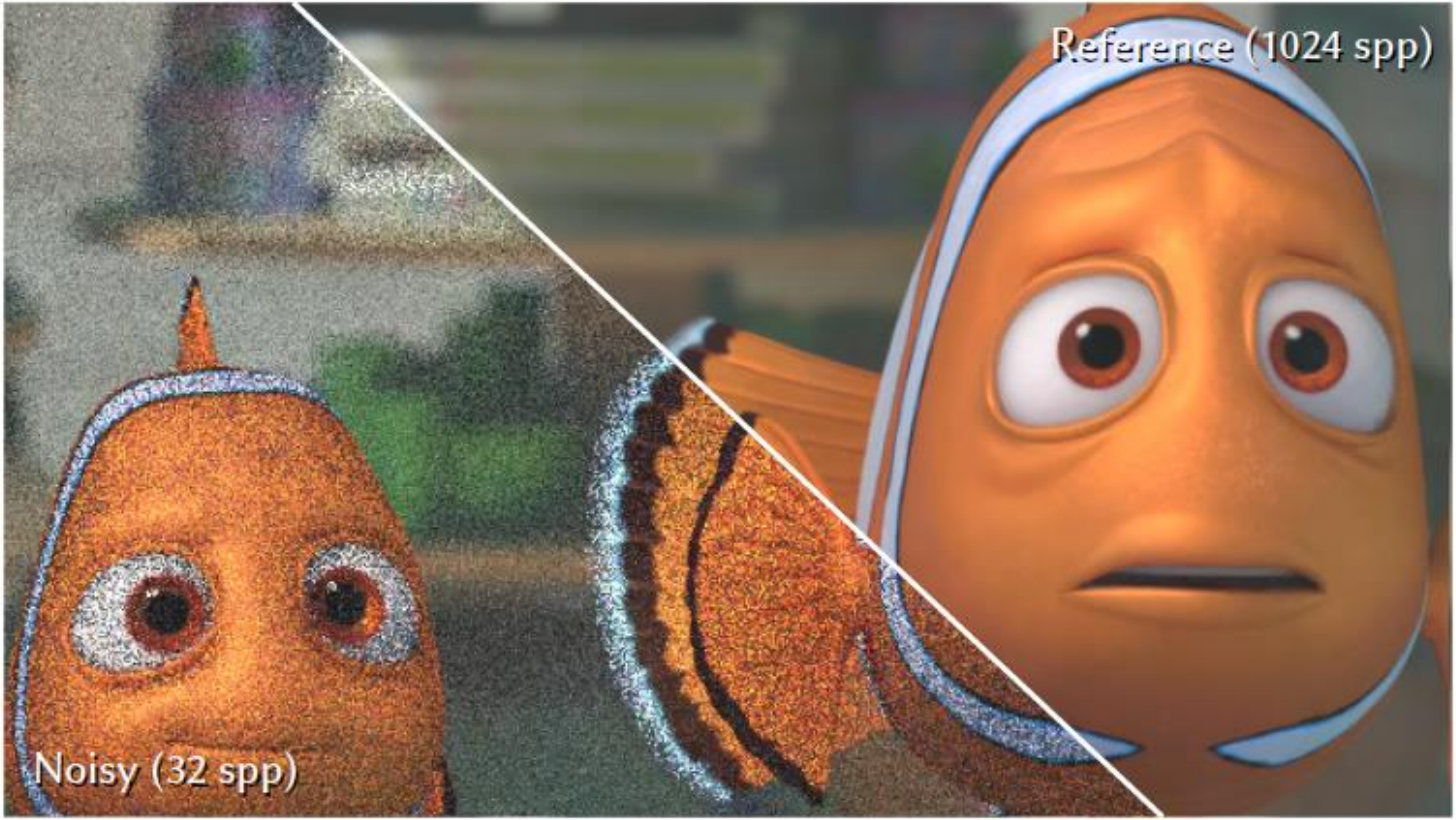
2. Estimating lighting



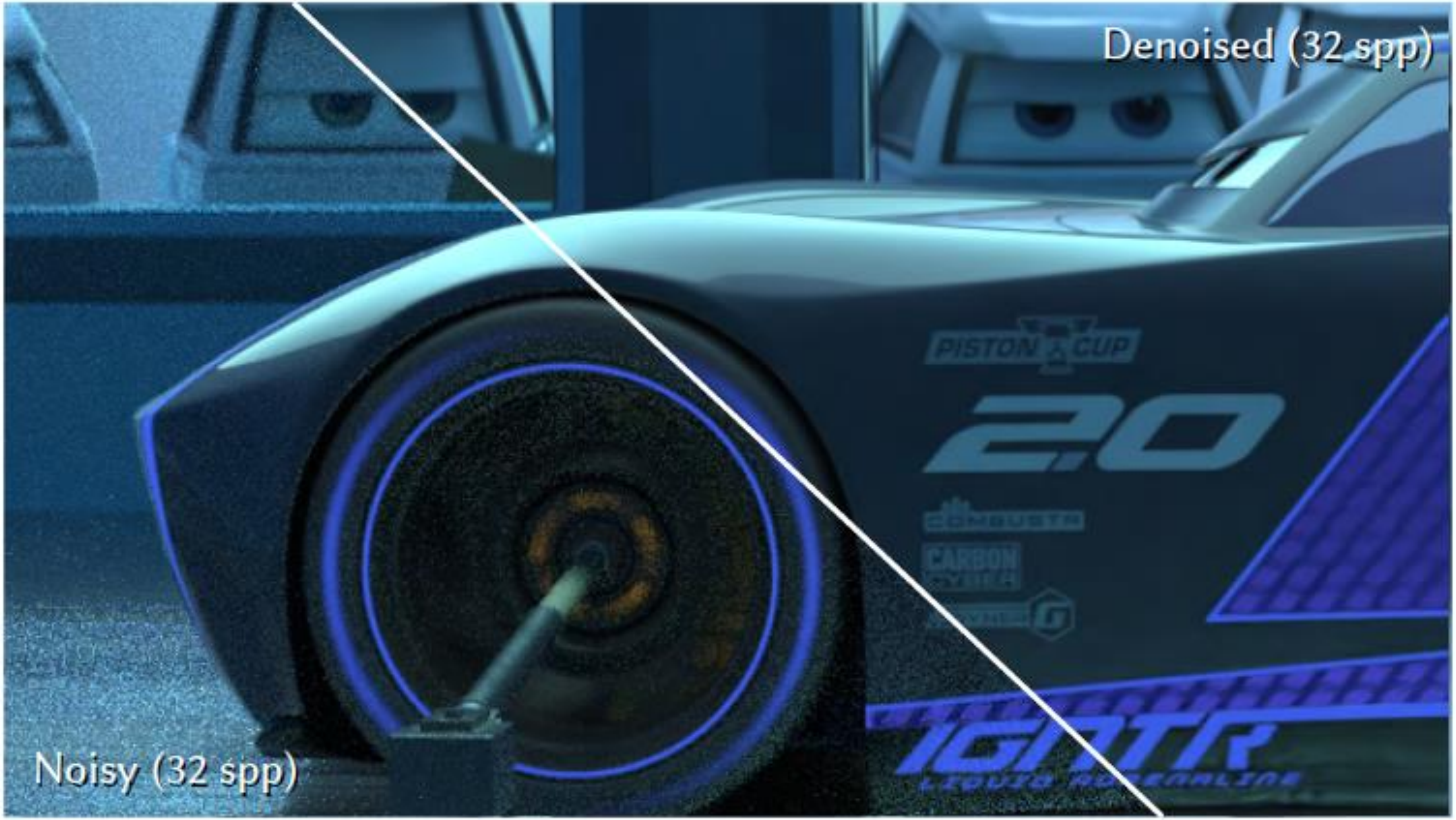
Problem

Why do we need Denoising?

- Noise with Monte Carlo Rendering
- To reduce Noise, more sampling → long time
- Instead, low sampling and denoising → short time



[Moon et. al.] Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings, SIGGRAPH 2017



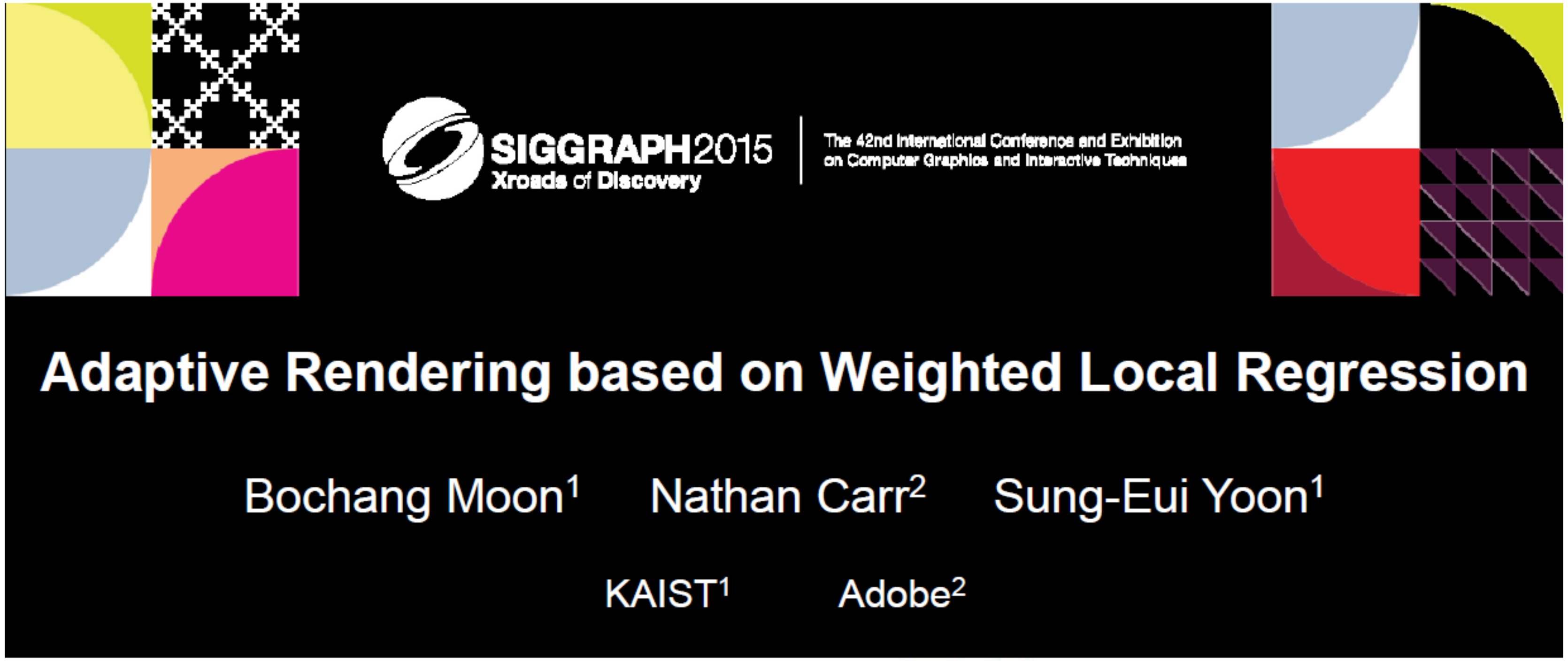
[Moon et. al.] Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings, SIGGRAPH 2017

Adaptive Polynomial Rendering

1. Previous work and Goal
2. Methods
3. Result

Adaptive Polynomial Rendering

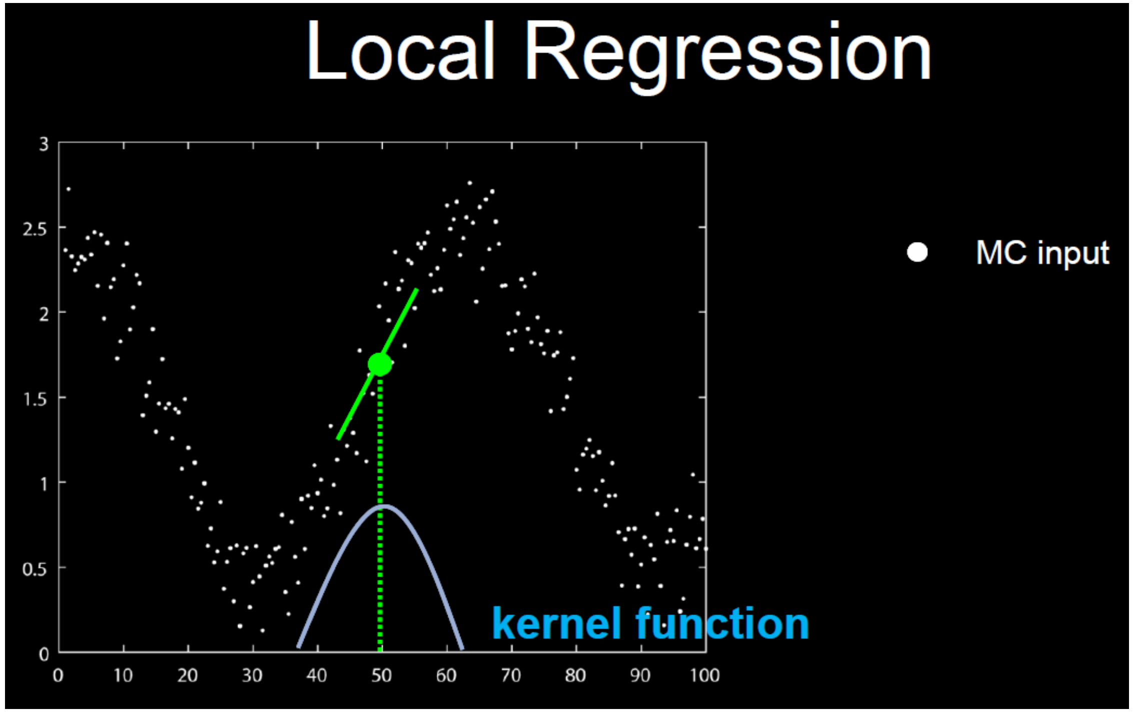
- Follow-up study

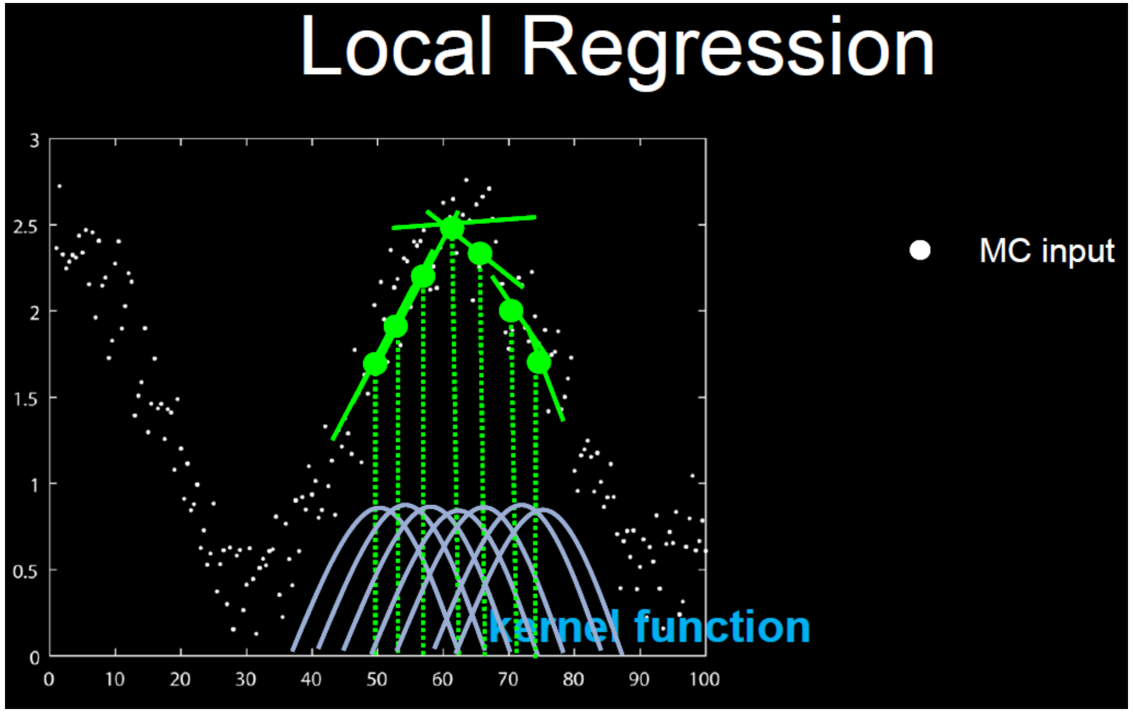


KAIST Adobe

[Moon et. al.] Adaptive Rendering based on Weighted Loss Regression, SIGGRAPH 2015

Previous work and Goal





- Previous work(Learned In class)
  Predict the local value by approximating the **Linear Equation**
- New method
  Predict the local value by approximating the **High dimensional Equation**

기존 연구: 선형 근사를 통해 local value를 추정.

후속 연구: **고차원 함수 근사**를 통해 local value 추정.

[Moon et. al.] Adaptive Rendering based on Weighted Loss Regression, SIGGRAPH 2015

Goal – Find appropriate Order of Polynomial Approximation

Higher order does not guarantee higher performance. Therefore, We need find out the optimal order.

높은 차수가 높은 성능을 보장하지 않는다. 에러가 가장 낮은 최적의 차수가 존재하며 이 차수를 탐색해서 찾아야 한다.

[Moon et. al.] Adaptive Polynomial Rendering, ACM Transactions on Graphics 2016

Method

1. Express our reconstruction bias and variance

2. Propose a robust estimation process for the error terms

In mathematics – PASS!

- Taylor Polynomials
- Least square optimization
- Normal equation
- Reconstruction output

$$y(i) = \mu(i) + \epsilon(i),$$

$$\mu(i) \approx \nabla g(f_c)(f_i - f_c)^T + p(c) + \sum_{1<a<k} \frac{\nabla^a p(c)}{a!}((i-c)^a)^T, \quad (2)$$

$$\sum_{i \in \Omega_c} \left(y(i) - \alpha(f_i - f_c)^T - \beta_0 - \sum_{1 \le a \le k} \beta_a((i-c)^a)^T \right)^2 K_h(i).$$

$$\hat{y}_k(c) = e_1(X_k^T W X_k) X_k^T W y,$$

$$\hat{y}_k = X_k(X_k^T W X_k) X_k^T W y = H(k) y,$$

$$\hat{y}(i) = \sum_{j \in \Omega_i} K_h^j(i) \hat{y}_k^j(i) / \sum_{j \in \Omega_i} K_h^j(i),$$

[Moon et. al.] Adaptive Polynomial Rendering, ACM Transactions on Graphics 2016

In mathematics – PASS!

- Reconstruction Error
- Bias and Variance
- Bias to hat matrix
- Variance approximation

$$\xi_c(k) \equiv \frac{1}{\sum_{i\in\Omega_c} K_h(i)} \sum_{i\in\Omega_c} K_h(i) \left(\hat{y}_k(i) - \mu(i)\right)^2$$

$$E\left(\hat{y}_k(i) - \mu(i)\right)^2 = bias^2(\hat{y}_k(i)) + \sigma^2(\hat{y}_k(i)).$$

$$\begin{aligned} E(\hat{y}_k(i) - \mu(i)) &= \sum_{j\in\Omega_c} H_{ij}(k)E(y(j)) - \mu(i) \\ &\approx \sum_{j\in\Omega_c} H_{ij}(k)\mu(j) - \mu(i), \end{aligned}$$

$$\sigma^2(\hat{y}_k(i)) \approx \sum_{j\in\Omega_c} \left(H_{ij}(k)\right)^2 \sigma^2(y(j)),$$

[Moon et. al.] Adaptive Polynomial Rendering, ACM Transactions on Graphics 2016

Method

1. Express error with bias and variance

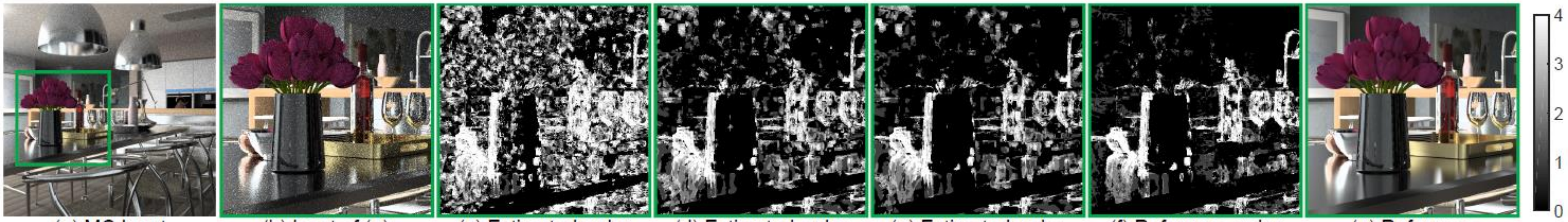
- We cannot know actual error, so we compute error by using bias and variance

$$k_{opt} = \underset{k}{\operatorname{argmin}} \sum_{i \in \Omega_c} K_h(i) \left(\left(E\left(\hat{y}_k(i) - \mu(i) \right) \right)^2 + \sigma^2 \left(\hat{y}_k(i) \right) \right).$$

2. Propose a robust estimation process for the error terms

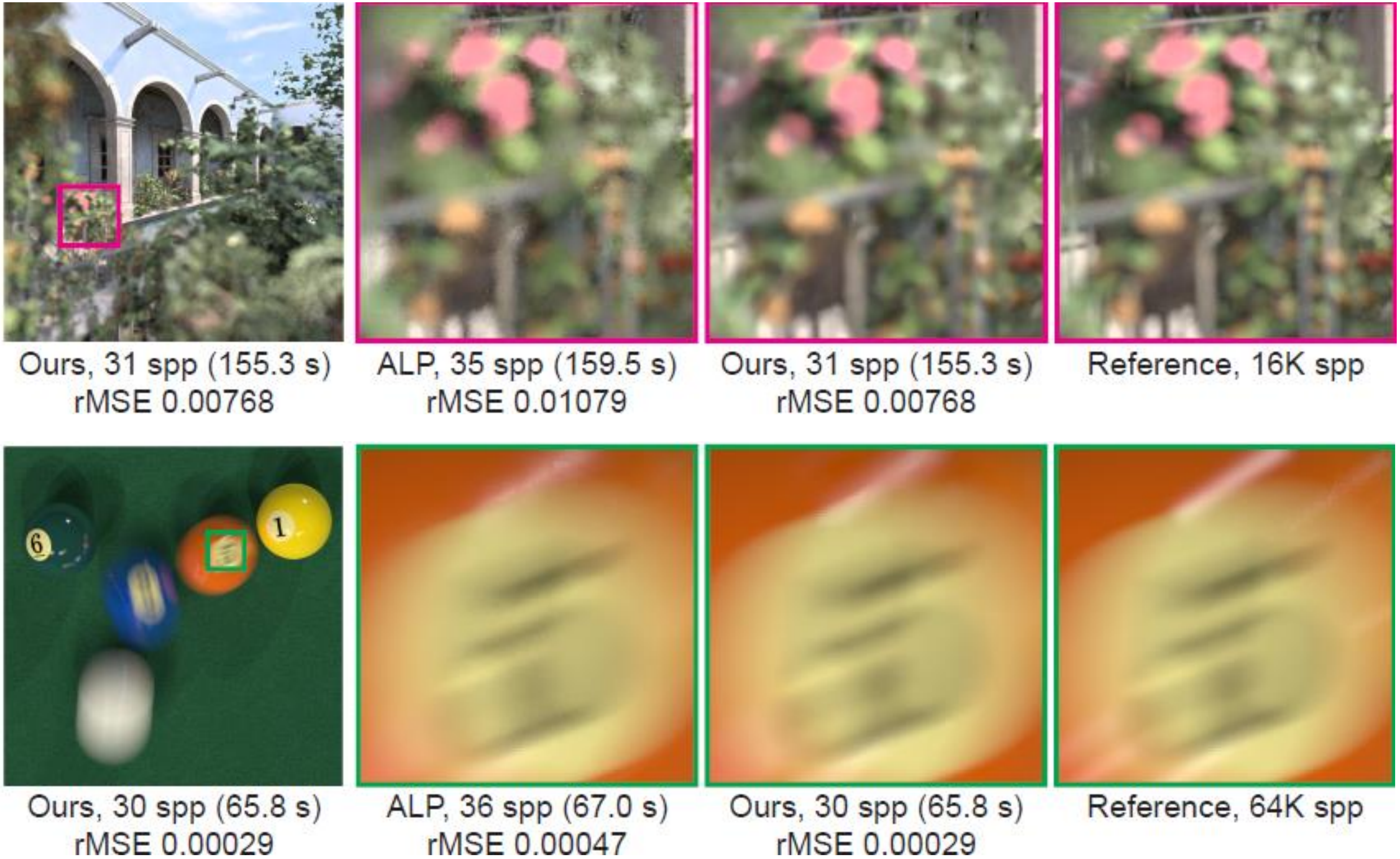
- To compute fast, we compute robust estimation with iteration step.

$$E_t(\hat{y}_k(i) - \mu(i)) \approx \sum_{j \in \Omega_c} H_{ij}(k) \hat{y}_{t-1}(j) - \hat{y}_{t-1}(i), \qquad \sigma_t^2(\hat{y}_k(i)) \approx \sum_{j \in \Omega_c} \left(H_{ij}(k) \right)^2 \hat{\sigma}_{t-1}^2(y(j)),$$



(a) MC Input 64 spp | (b) Inset of (a) | (c) Estimated order iteration 1 | (d) Estimated order iteration 2 | (e) Estimated order iteration 3 | (f) Reference order | (g) Reference 32K spp

[Moon et. al.] Adaptive Polynomial Rendering, ACM Transactions on Graphics 2016



[Moon et. al.] Adaptive Polynomial Rendering, ACM Transactions on Graphics 2016

Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings

Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings

- Denoising with Machine Learning Techinque

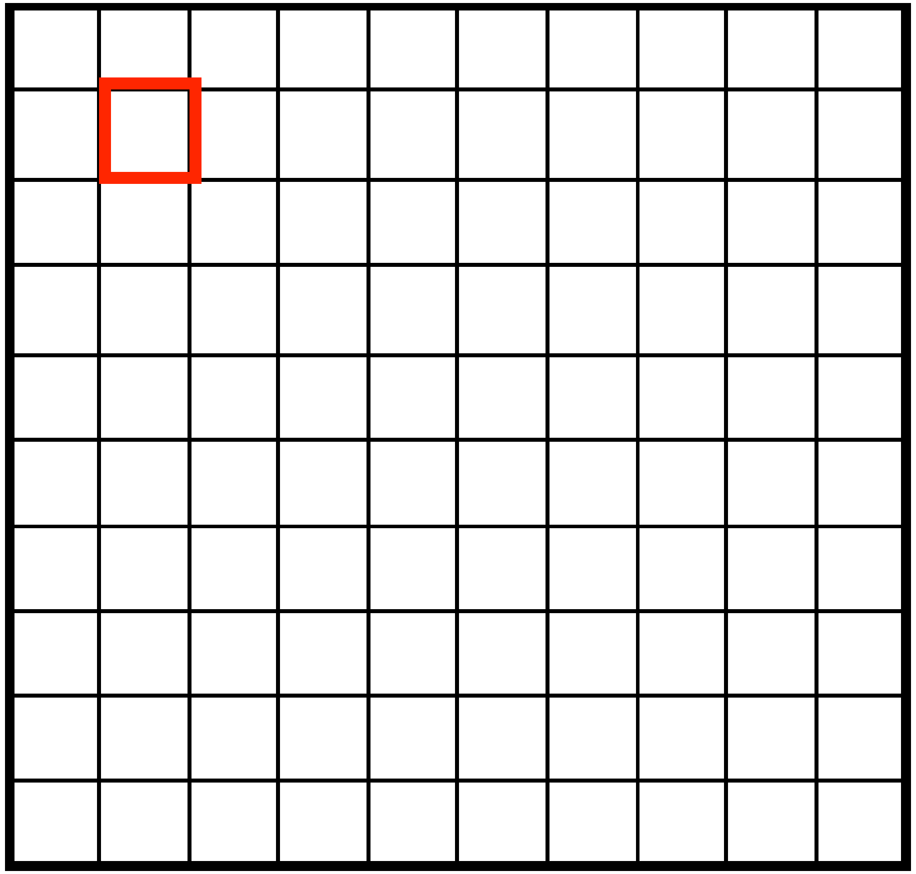
Image Filter

Element-wise Multiplication and Sum

$I[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$



$$f[\cdot,\cdot] = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Image Filter

Element-wise Multiplication and Sum

$f[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$I[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10							

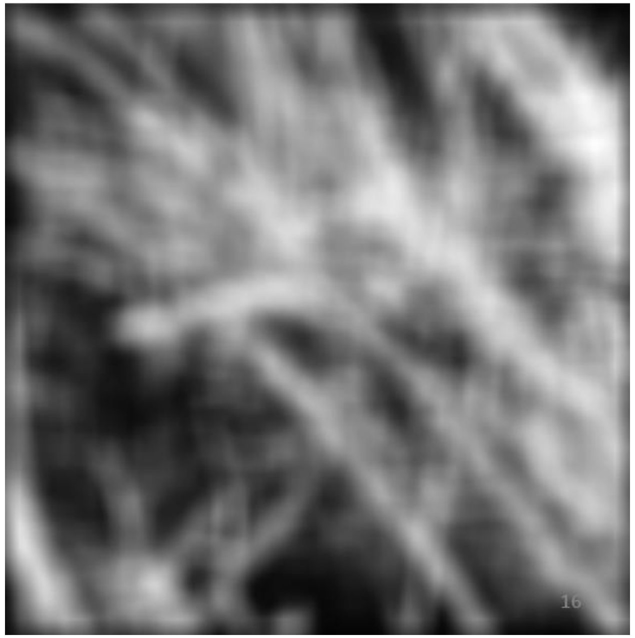
Blurring with kernel(filter)

$*$

$f[\cdot,\cdot]$

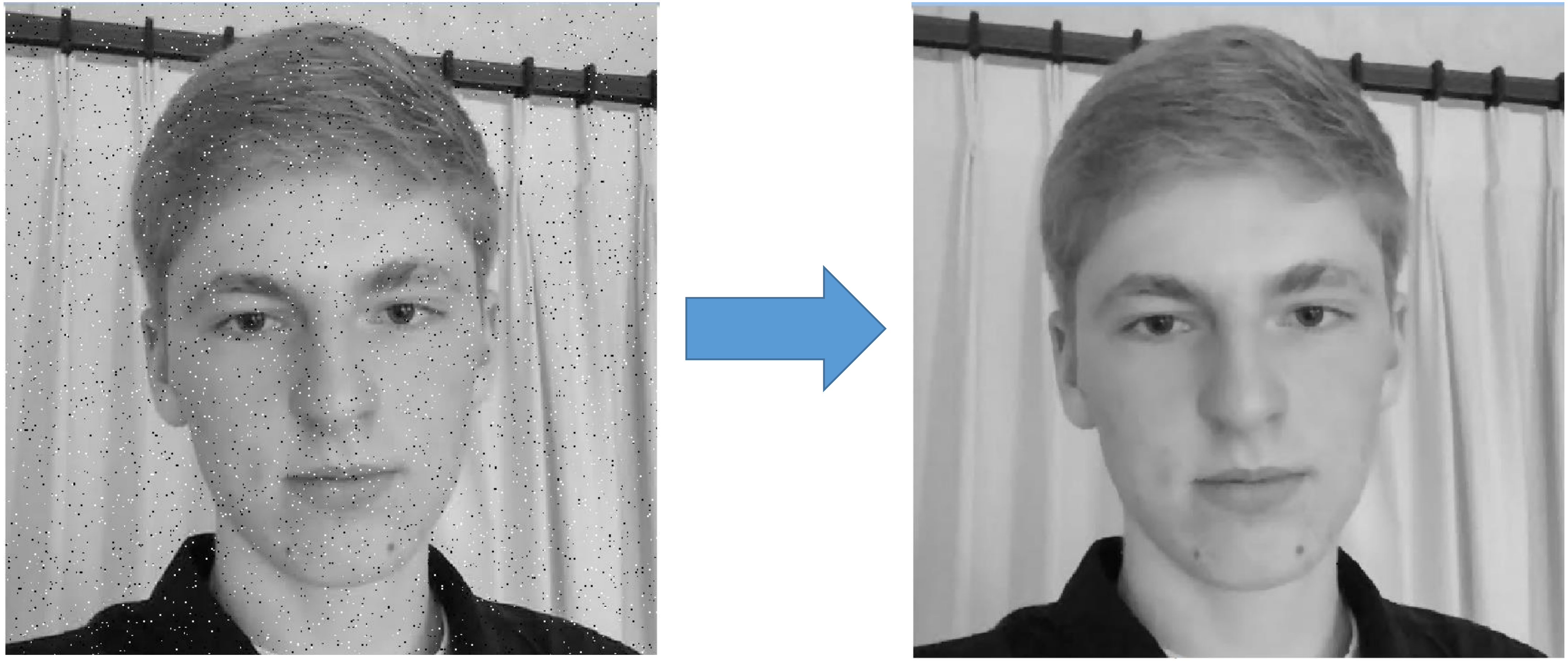
$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

=

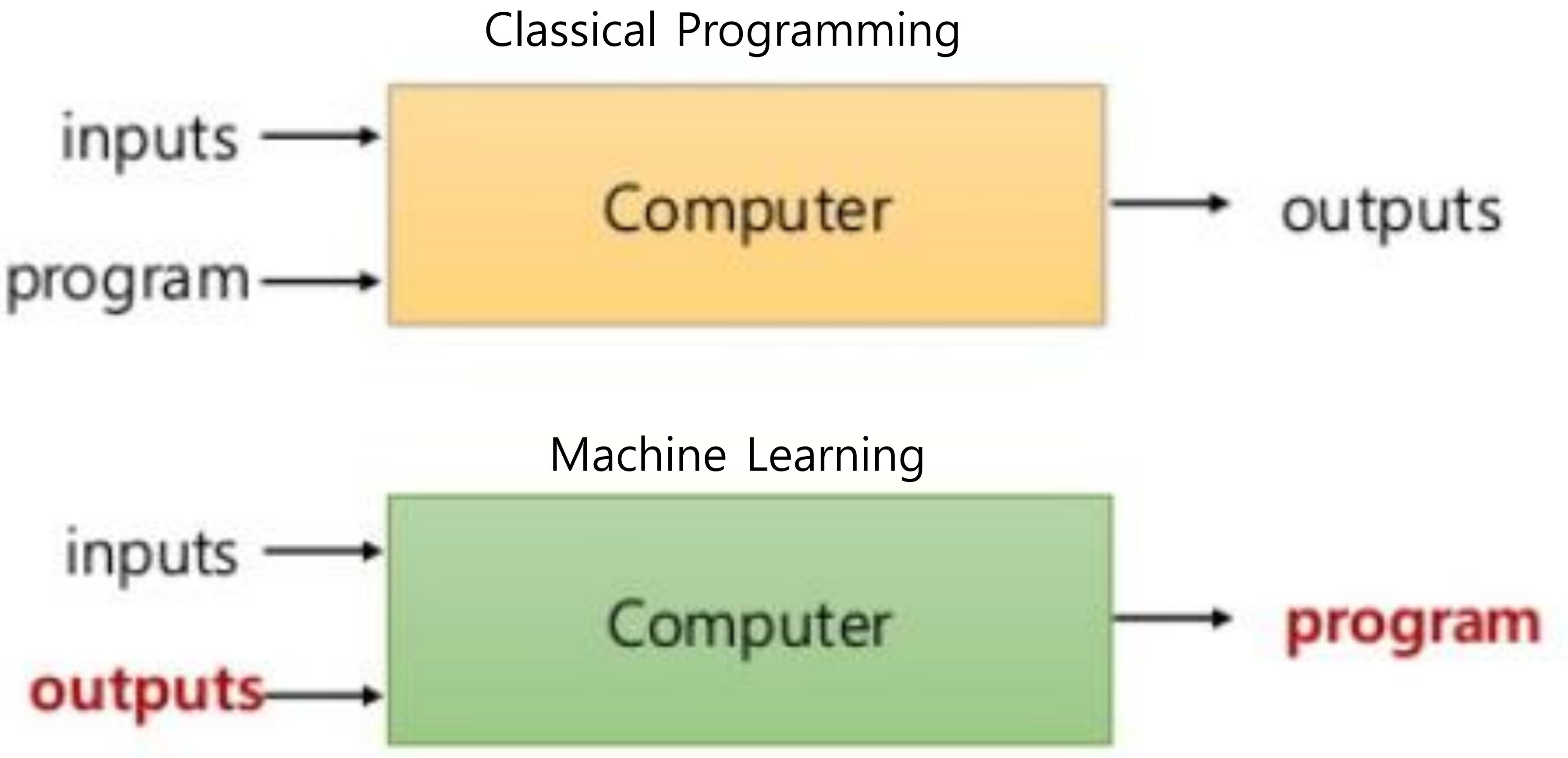


Input * Kernel = Output

Denoising with kernel(filter)



Machine Learning(ML)

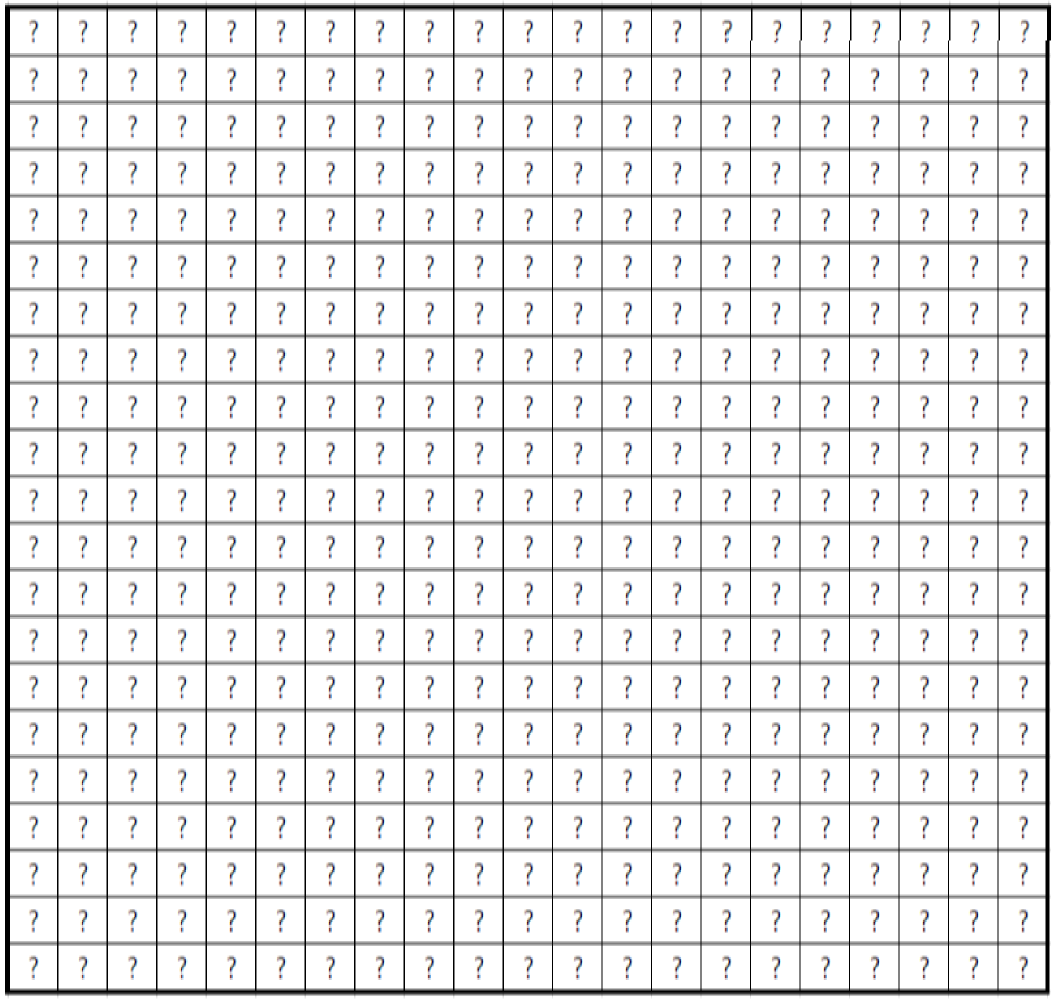


https://www.slideshare.net/JinwonLee9/ss-70446412

Machine Learning(ML)

- Predict denoising Image filter as ML output
  - Image filter = Kernel
  - 21 * 21 size
  - ML method: CNN(Convolutional Neural Network)

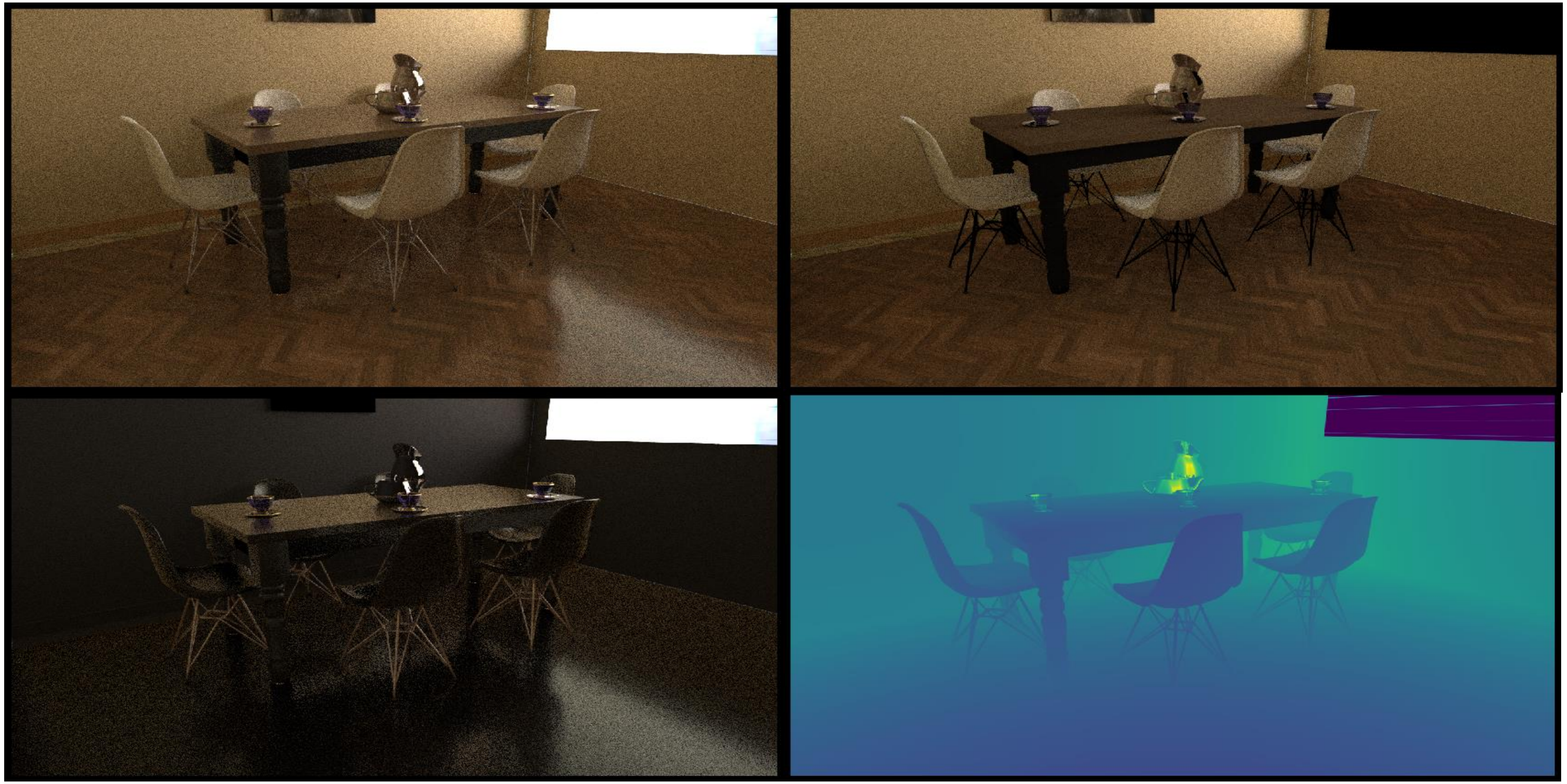
Kernel



Data format(EXR image data)

- Data consists of many channels, not only RGB channels

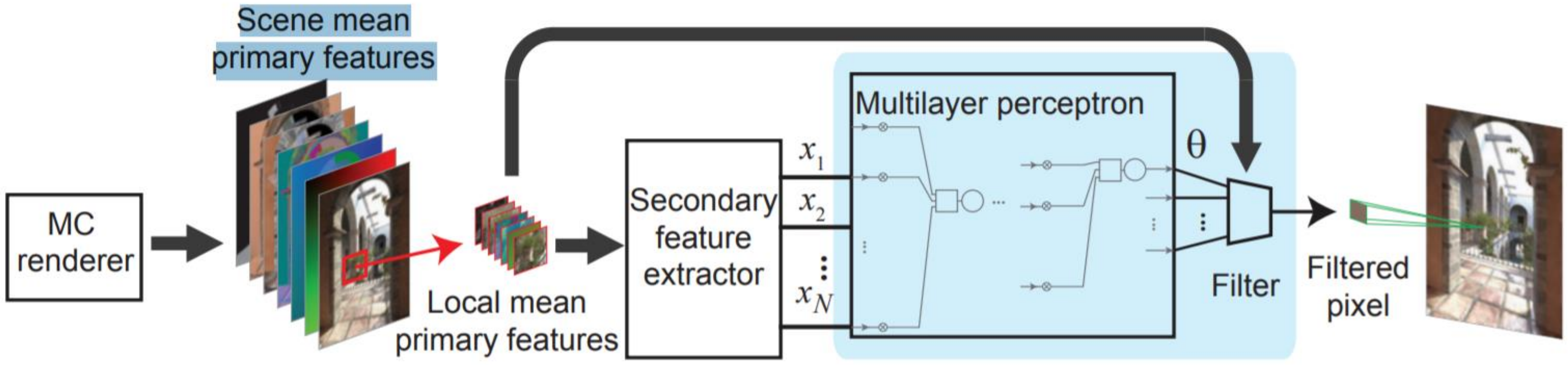
RGB Diffuse



Specular Depth

Method

- Previous Method
  - Accumulated prediction with single network

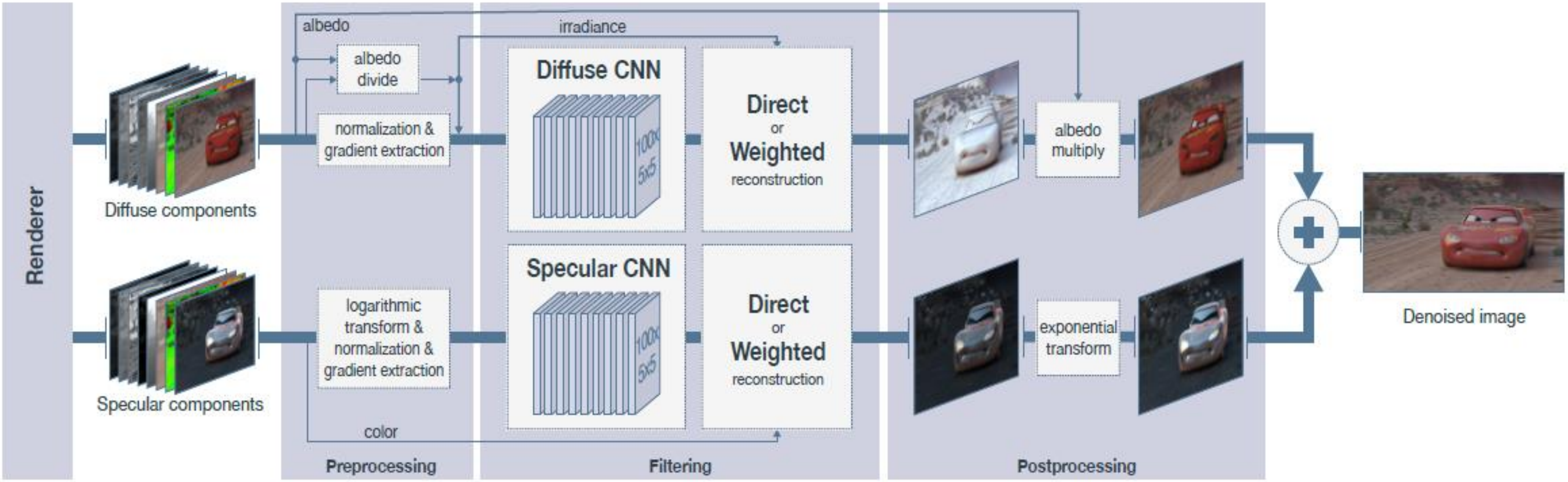


- Proposed Method
  1. Decompose channels into Diffuse Component and Specular Component
  2. Denoising by Kernel-Prediction Convolutional Network (KPCN)

기존 연구: 채널 분리 없이 하나의 모델로 학습
후속 연구: Diffuse와 Specular로 분리하여 두개의 모델로 학습 + 커널 예측 모델

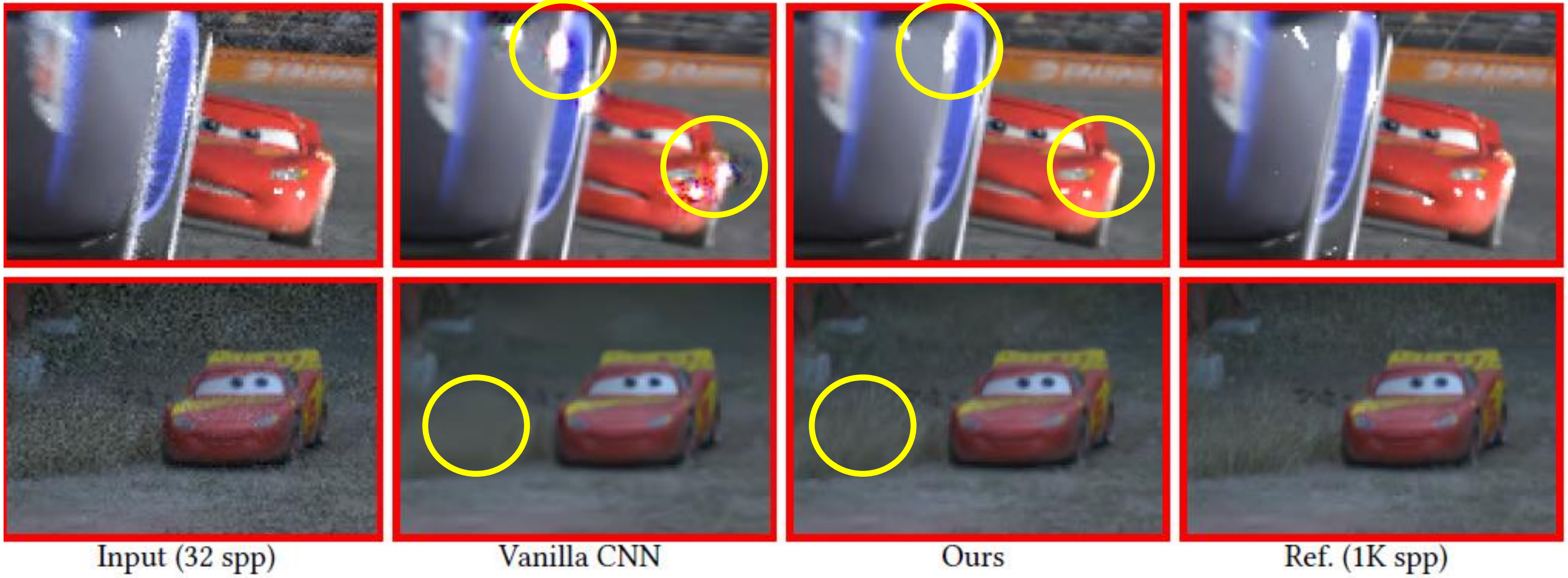
[Kalantari et. al.] A Machine Learning Approach for Filtering Monte Carlo Noise, SIGGRAPH 2015

Decomposition



[Bako et. al.] Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings, SIGGRAPH 2017

Why Decomposition?



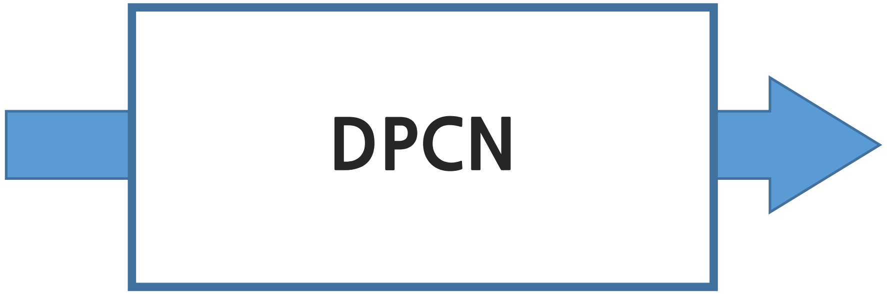
- The various components of the image have different noise characteristics and spatial structure. This leads the single network model into the low quality and overblurring.

채널마다 특성이 달라 noise에 대해 서로 다른 특성을 갖는다. 따라서 모든 채널을 한 네트워크로 학습시키면 overblurring이 일어난다.

[Bako et. al.] Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings, SIGGRAPH 2017

DPCN

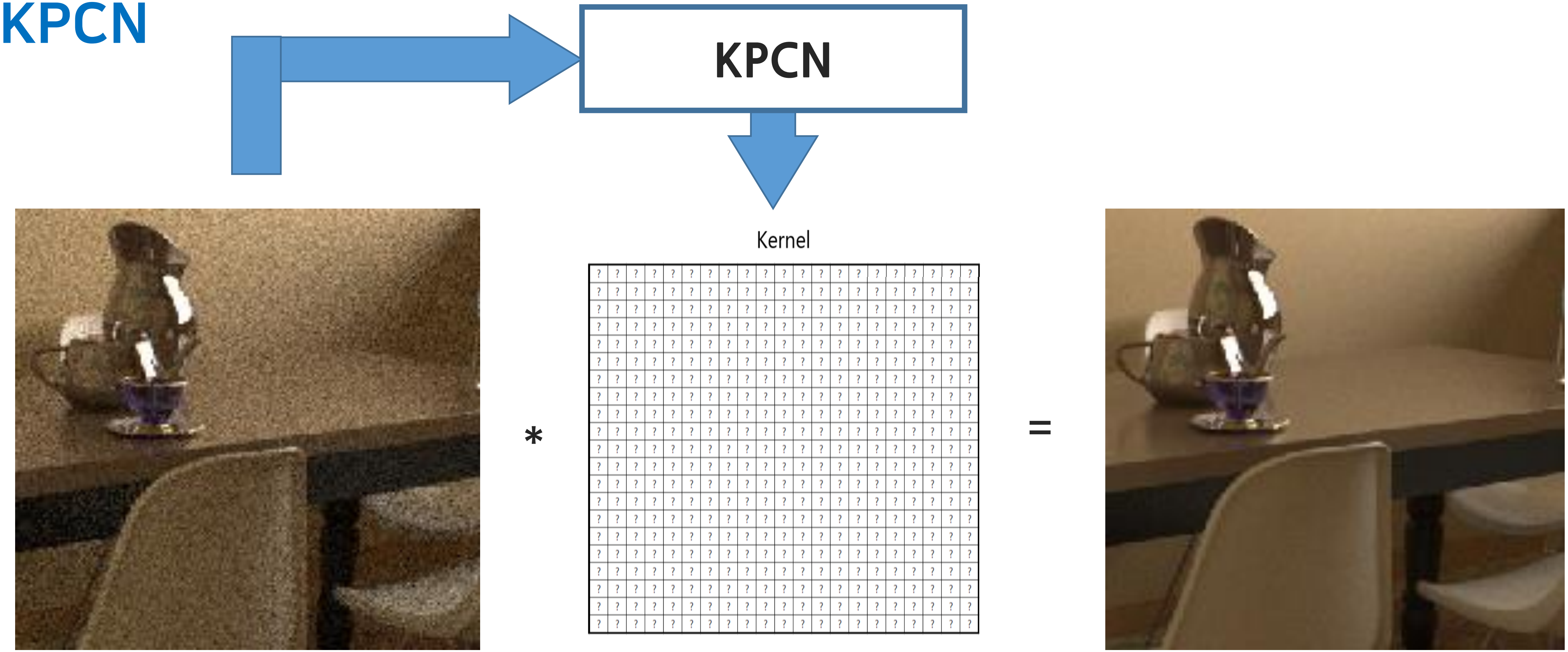
Noisy Image



Clear Image

DPCN predicts the color value directly

KPCN

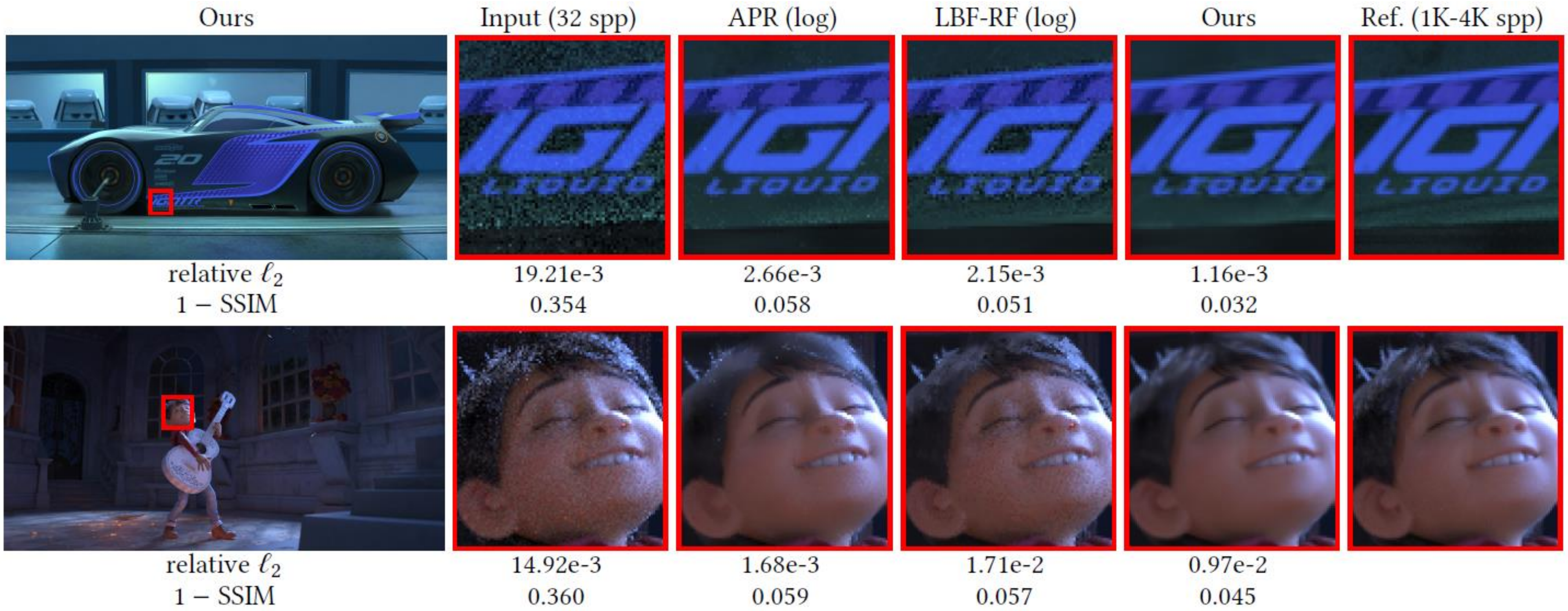


Noisy Image * Kernel = Clear Image

KPCN predicts the Kernel

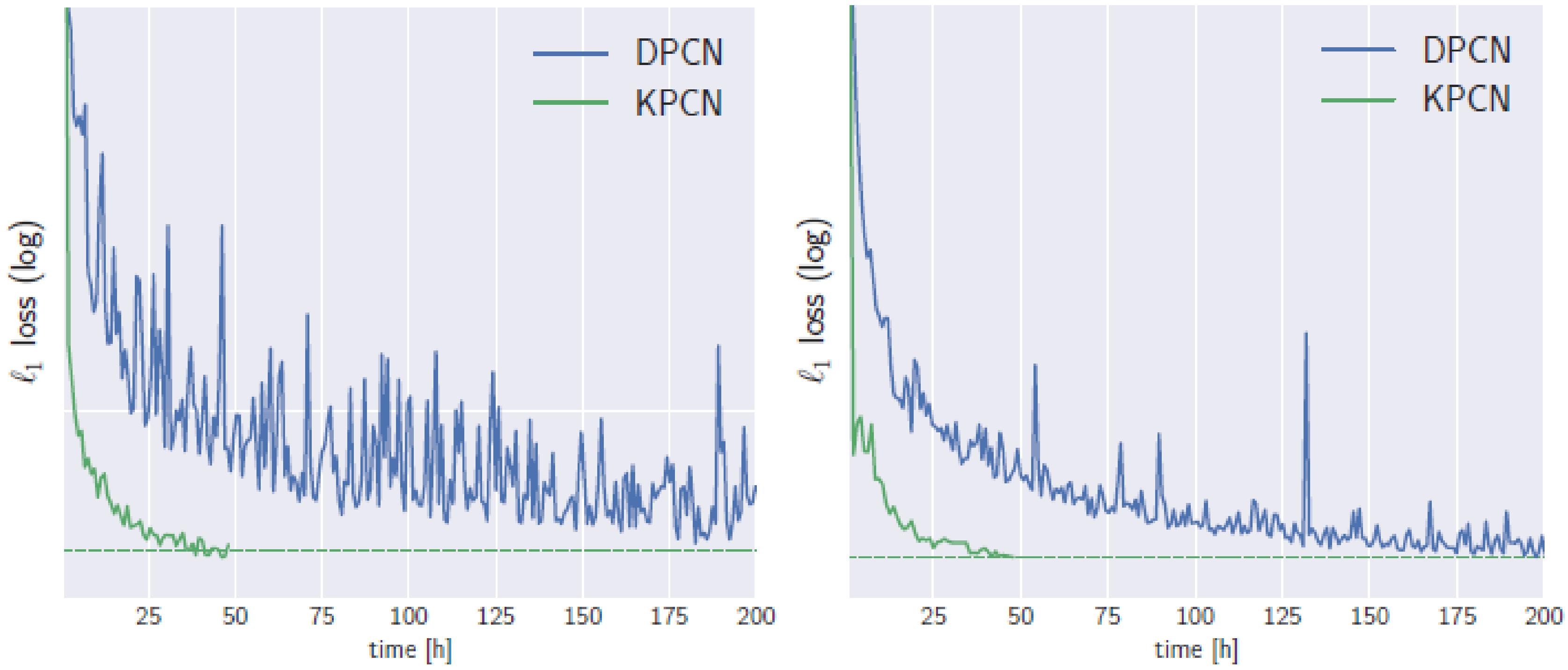
Result

APR : Adaptive Polynomial Regression (just previous one)
LBF-RF : Previous Learning Based Denosing



[Bako et. al.] Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings, SIGGRAPH 2017

Result



(a) Diffuse (b) Specular

Fig. 9. Comparison of optimization speed between the DPCN and KPCN architectures. Although both approaches converge to a similar error on the *Cars 3* validation set, the KPCN system converges 5–6× faster.

[Bako et. al.] Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings, SIGGRAPH 2017

Thanks a lot!

Quiz

1. What is **purpose** of Adaptive Polynomial Rendering?

(1) Compute image with bias and variance

(2) Find appropriate order of local polynomial regression

2. Which is **better** performance in second paper?

(1) Direct-Prediction Convolutional Network (DPCN)

(2) Kernel-Prediction Convolutional Network (KPCN)

References

- Adaptive Polynomial Rendering, Bochang Moon / ACM Transactions on Graphics 2016
- Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings, Steve Bako / SIGGRAPH 2017
- A Machine Learning Approach for Filtering Monte Carlo Noise / SIGGRAPH 2015
- CS580 Computer Graphics Lecture Slide
- CS484 Introduction to Computer Vision Lecture Slide
- CS576 Computer Vision Lecture Slide
- https://www.slideshare.net/JinwonLee9/ss-70446412