Eurographics 2015
The 36th Annual Conference of the
European Association for Computer Graphics

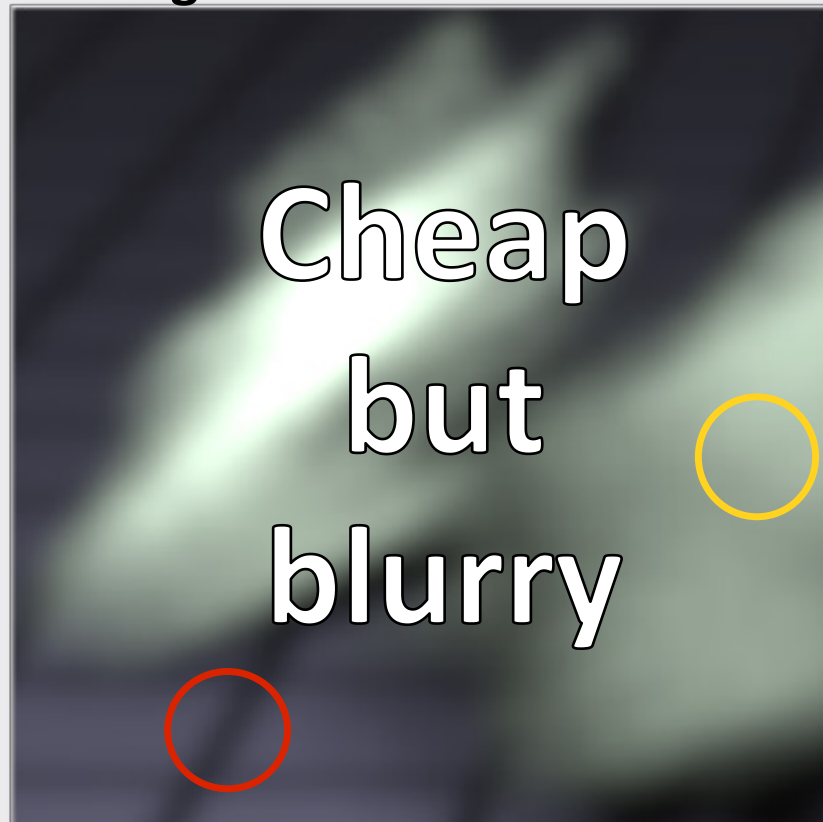
Adaptive Rendering
a posteriori methods

Fabrice Rousselle
fabrice.rousselle@disneyresearch.com

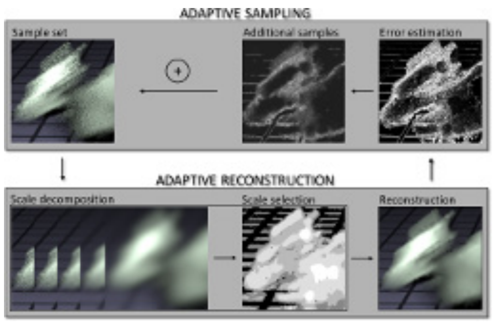
Adaptive rendering : two strategies

Higher sampling rate

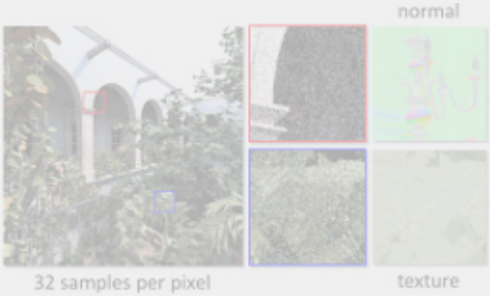
Larger reconstruction filter



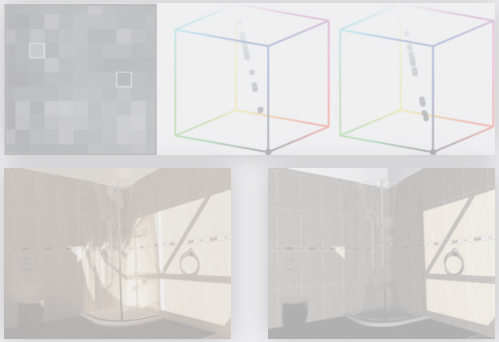
Plan



Adaptive Framework

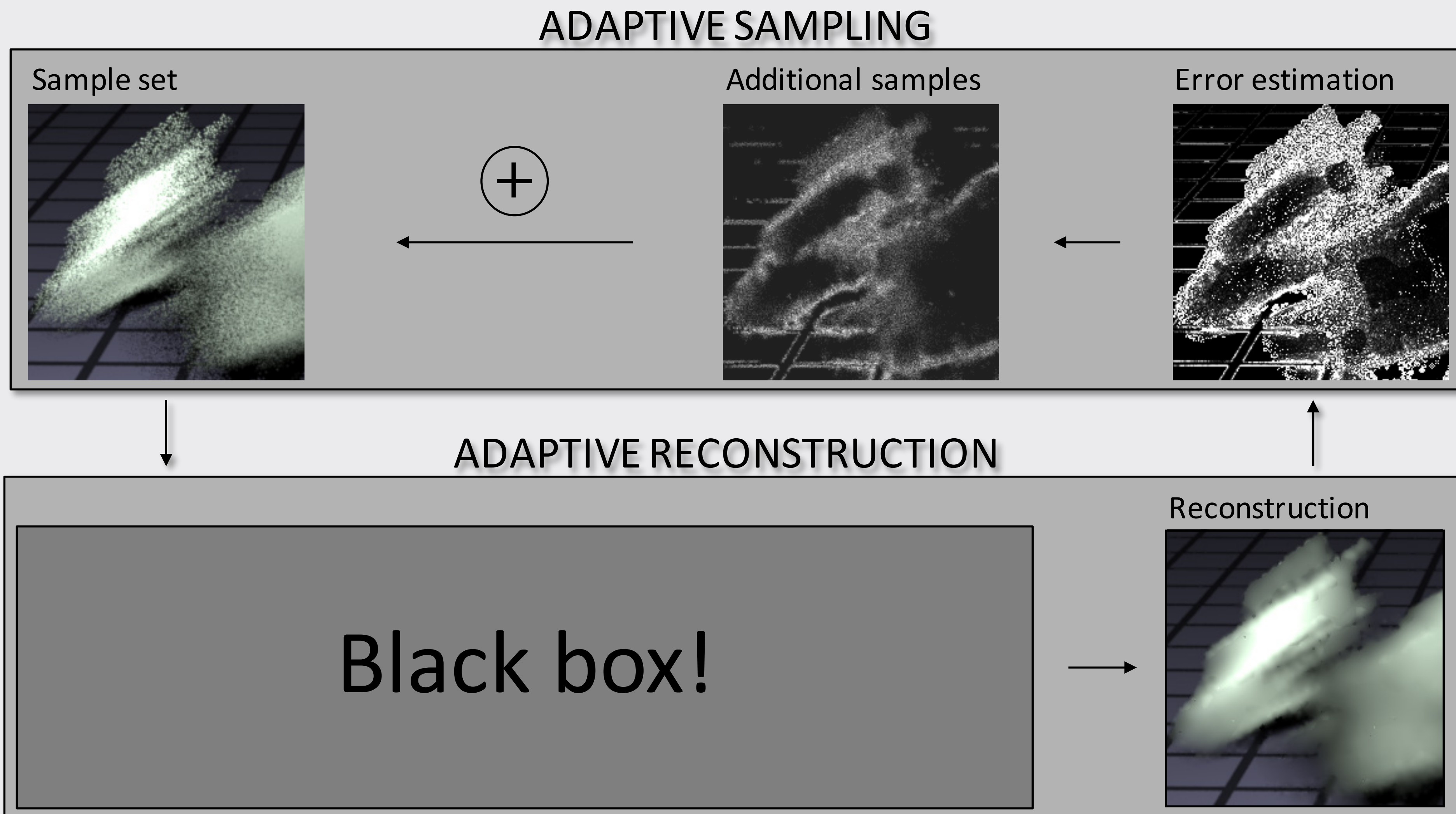


Joint NL-Means

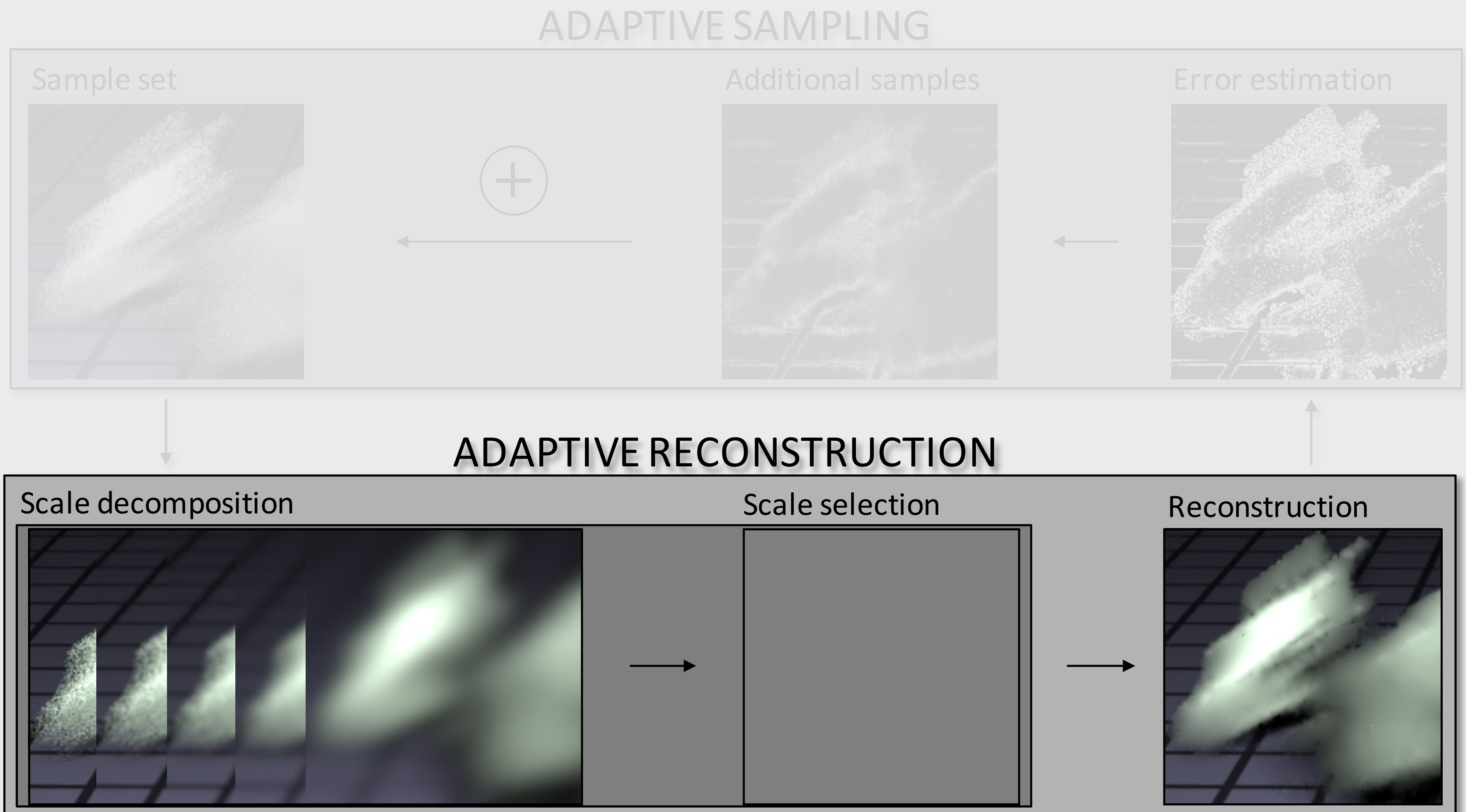


Other Implementations

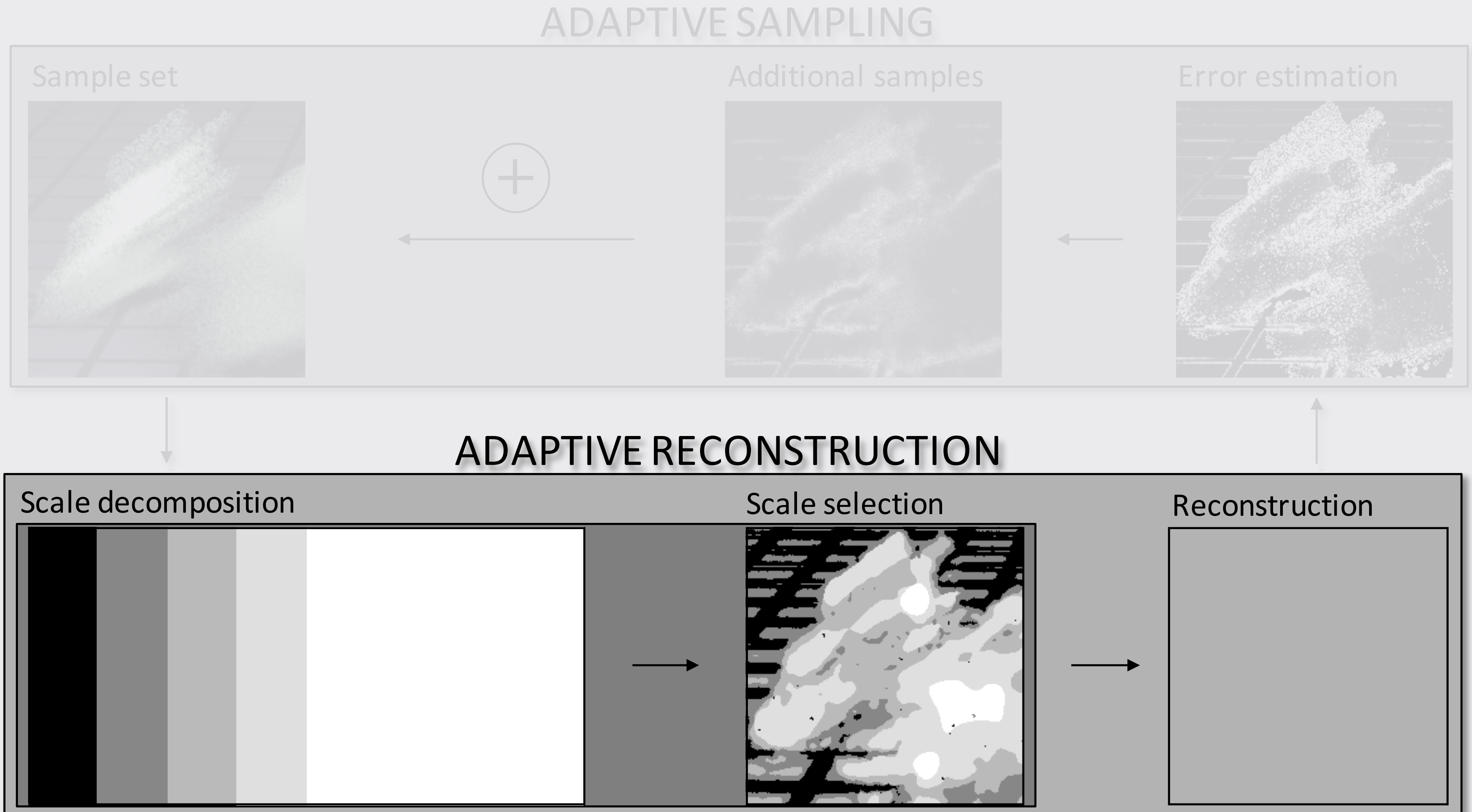
Framework overview



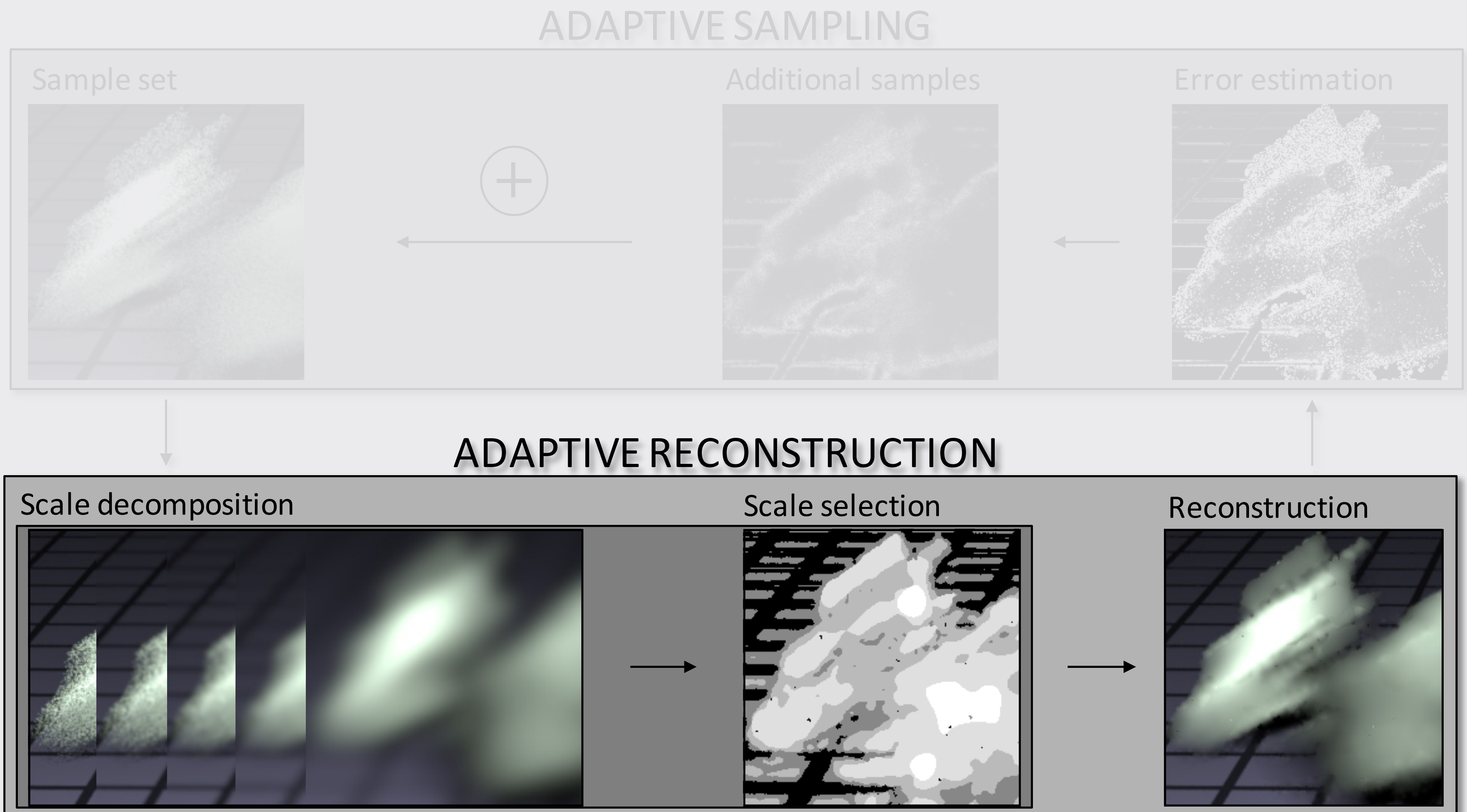
Framework overview



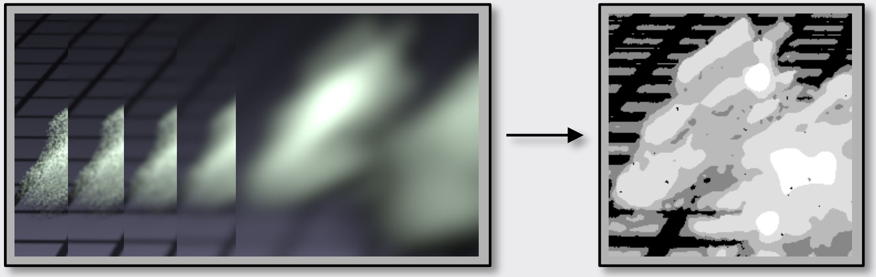
Framework overview

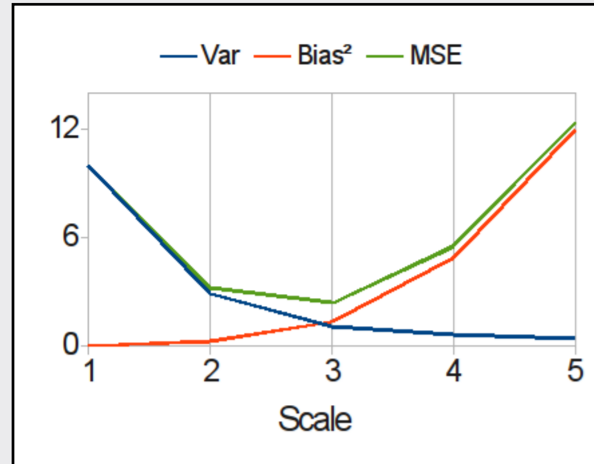


Framework overview



Framework overview – scale selection



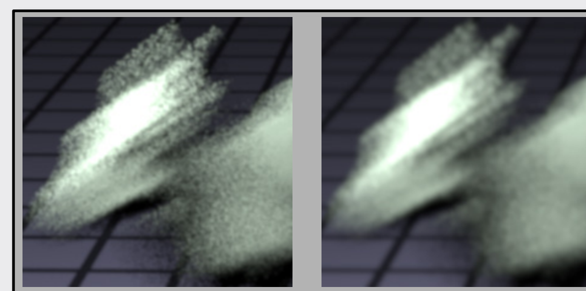


Goal: minimize Mean Squared Error

$$\mathrm{MSE} = \mathrm{Var} + \mathrm{Bias}^2$$

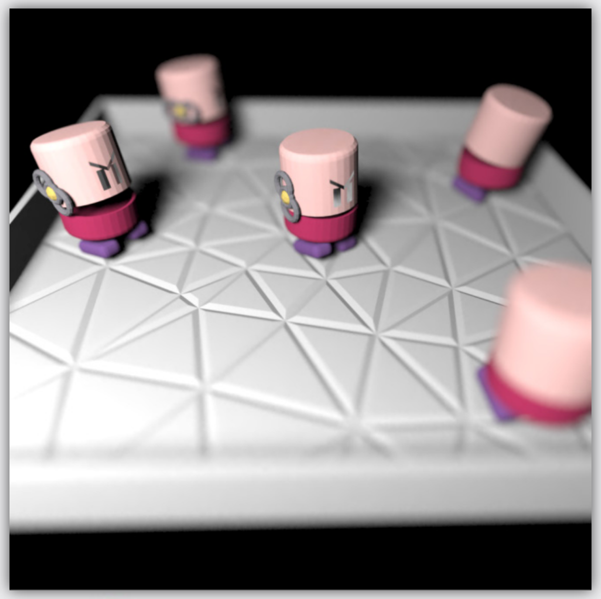
Binary decision

$$\mathrm{MSE}(\mathrm{fine} \rightarrow \mathrm{coarse}) > 0$$

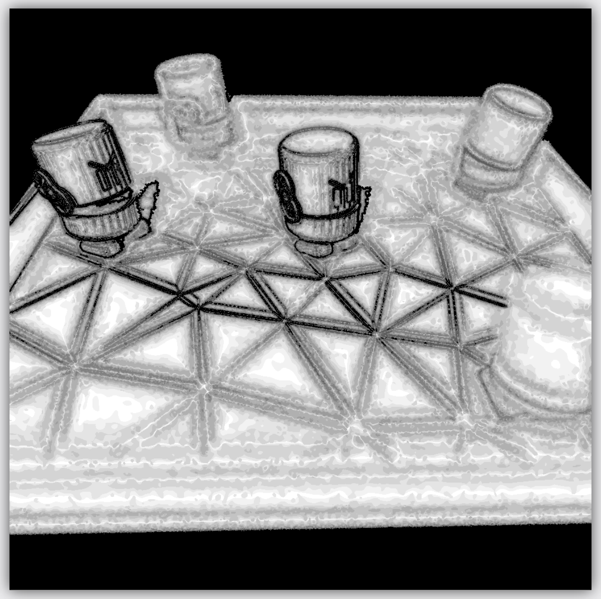


fine ⟶ coarse

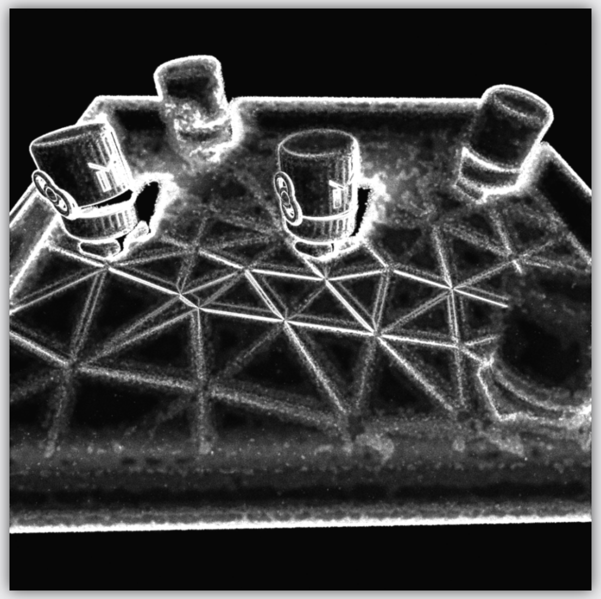
Framework overview – results



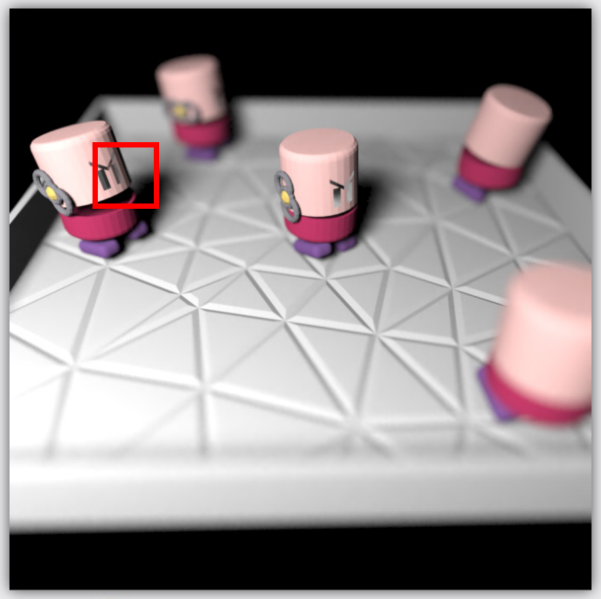
Scale selection



Sampling rate



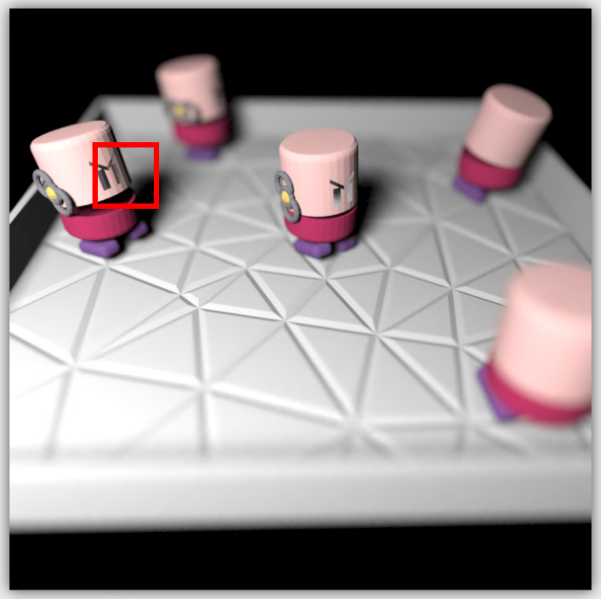
Framework overview – results



GEM – 32 spp, 68s

RND – 50 spp, 69s

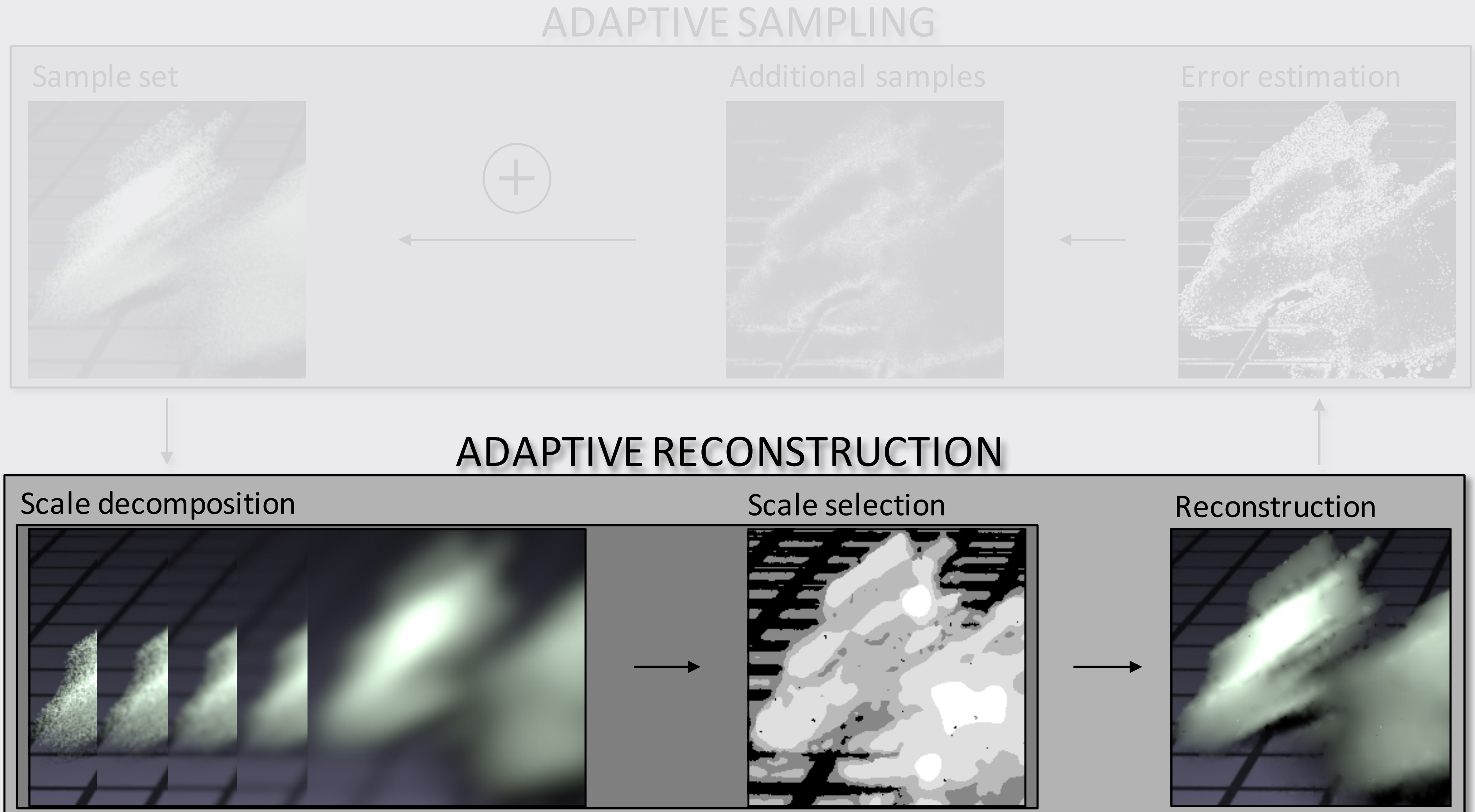
Framework overview – results



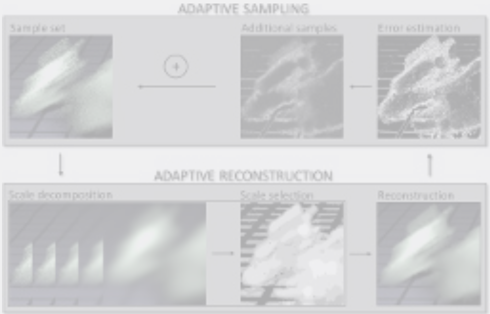
GEM – 32 spp, 68s

Ground truth – 4000 spp

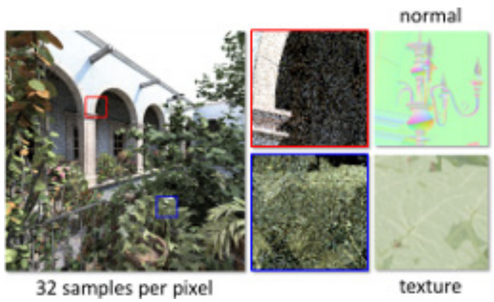
Framework overview



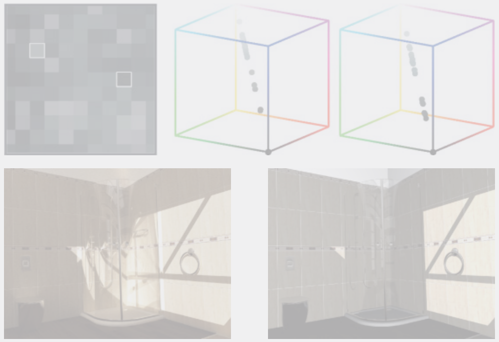
Plan



Adaptive Framework



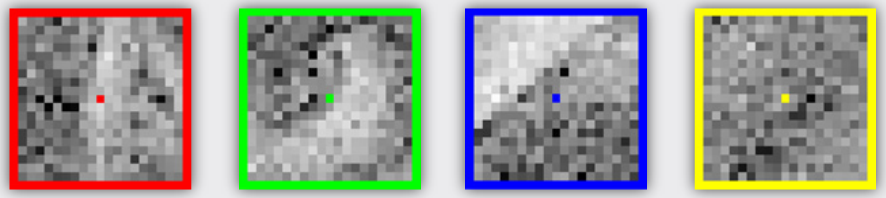
Joint NL-Means



Other Implementations

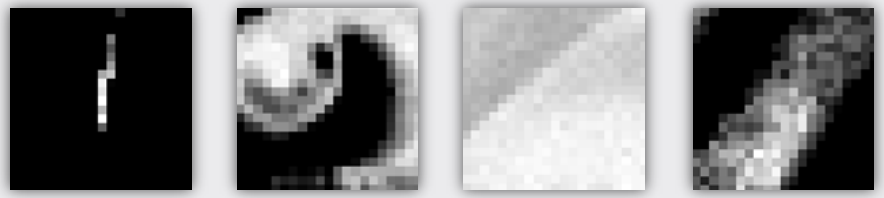
Image-based denoising

Noisy



Reference

Ideal weights



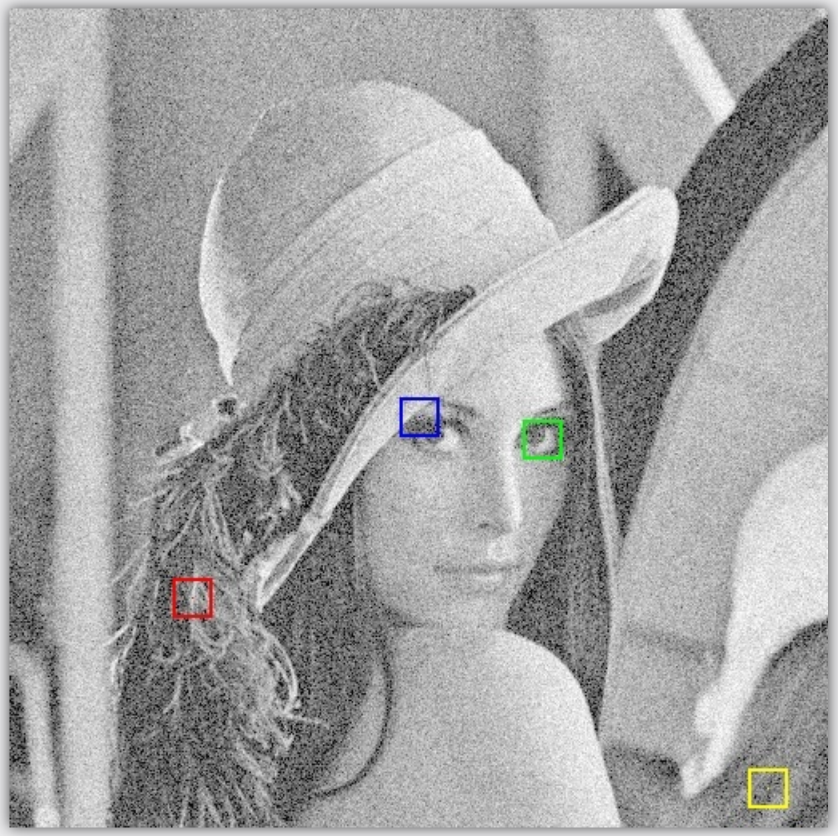
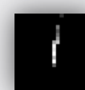
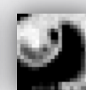


Image-based denoising

- Use a **flexible** and **robust** filter
  - Flexible: irregular weights  
  - Robust: few denoising artifacts

Noisy data

Bilateral Filter

Non-Local Means Filter

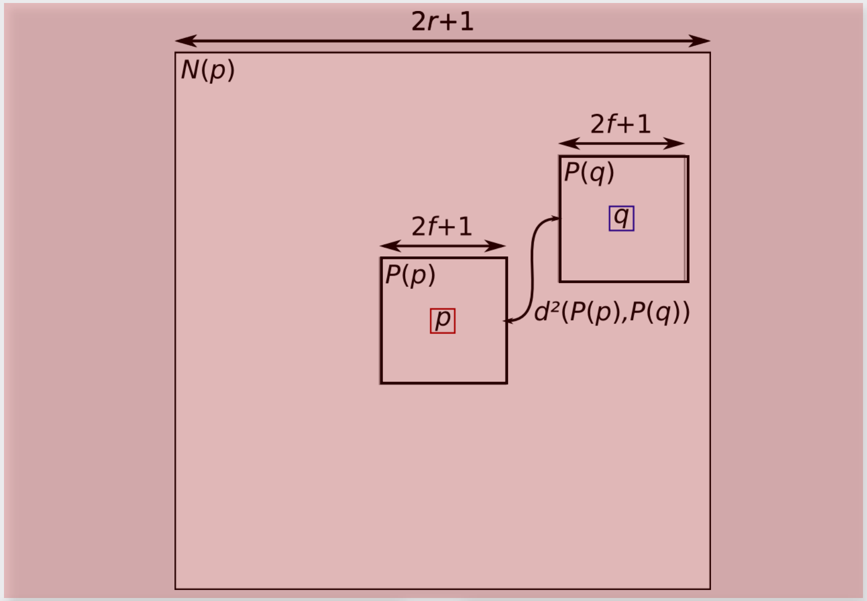
Buades et al. 2006

NL-Means Filter

- Per pixel weight *w*(*p*,*q*)
- Generalized distance *d*
  - spatial (in pixels, exact)
  - range (difference in value *u*, noisy)

$$d^2(p,q) = \frac{(u(p) - u(q))^2 - 2\sigma^2}{\epsilon + k^2 2\sigma^2}$$

- Weights
  - decrease exponentially with the distance



bilateral filter

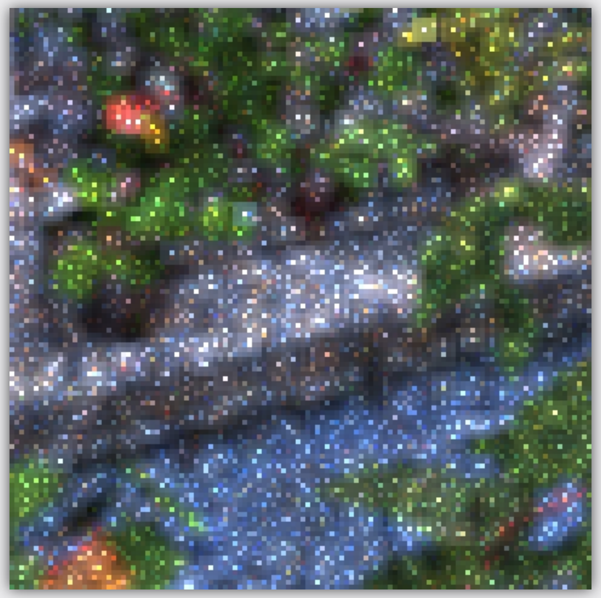
NL-Means

128 samples per pixel

NL-Means – Non-uniform variance

128 samples per pixel

Variance



NL-Means – Non-uniform variance

- Uniform variance

$$d^2(p,q) = \frac{(u(p)-u(q))^2 - \boxed{2\sigma^2}}{\epsilon + 2k^2\boxed{\sigma^2}}$$

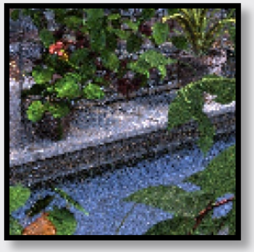
- Non-uniform variance

$$d^2(p,q) = \frac{(u(p)-u(q))^2 - \boxed{(\mathrm{Var}[p] + \min(\mathrm{Var}[q], \mathrm{Var}[p]))}}{\epsilon + 2k^2\boxed{\mathrm{Var}[p]}}$$

NL-Means – Variance estimation

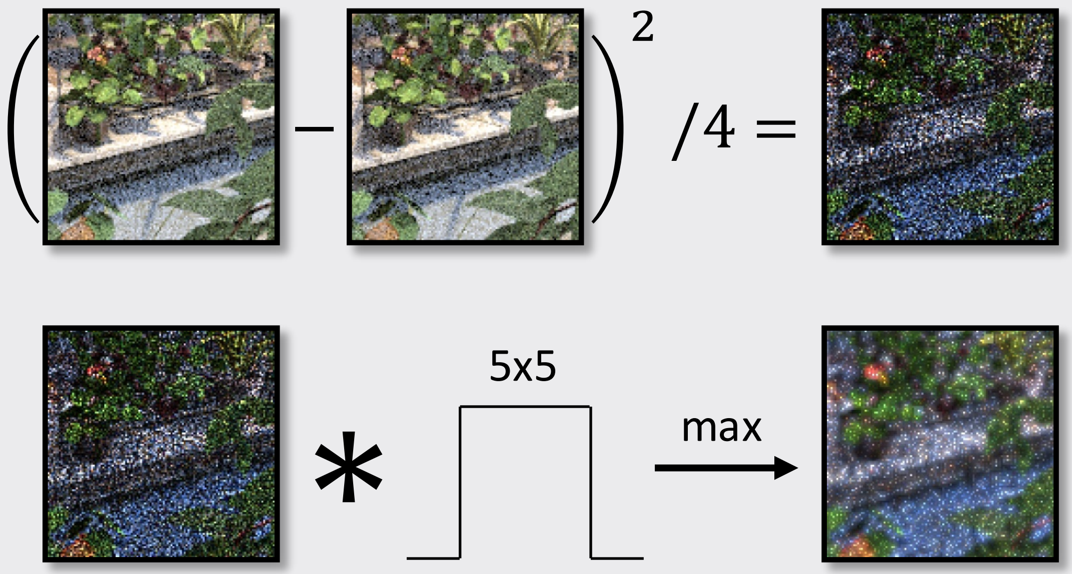
- Independent samples: sample mean variance

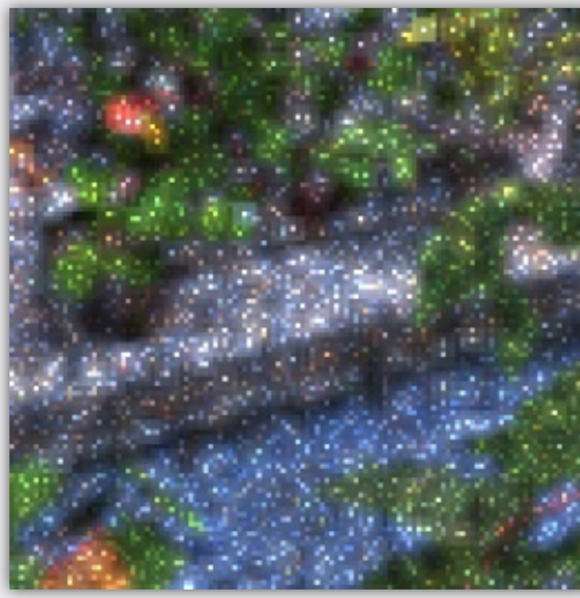
$$\mathrm{Var}[p] = \left(\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right) / n$$



NL-Means Filter – Variance Estimation

- Correlated samples: use buffer difference
  - Stratified sampling, low-discrepancy sampling, etc.
  - Generate two images using independent seeds

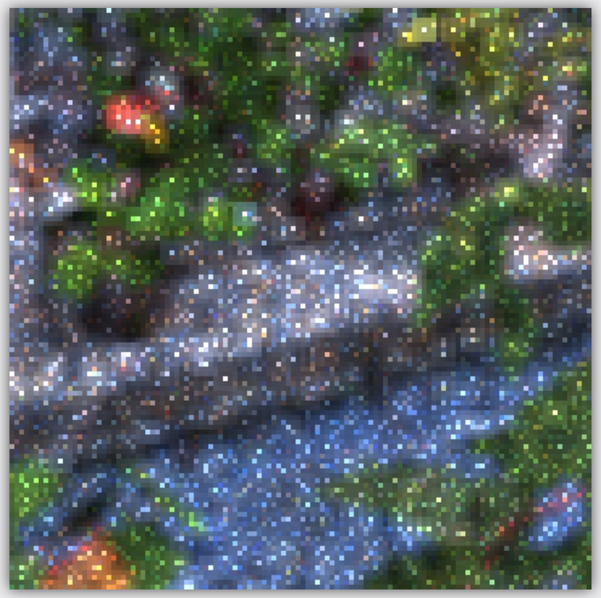




NL-Means

128 samples per pixel

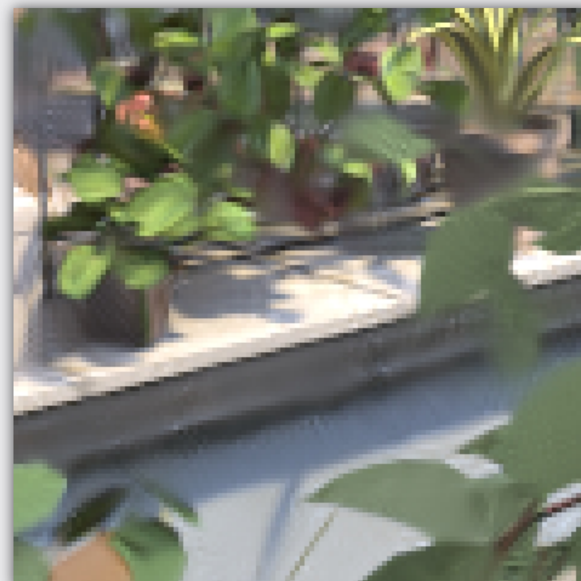
Variance



NL-Means

128 samples per pixel

Denoised



NL-Means

128 samples per pixel

Denoised

NL-Means – Leveraging Scene Info

Normal buffer

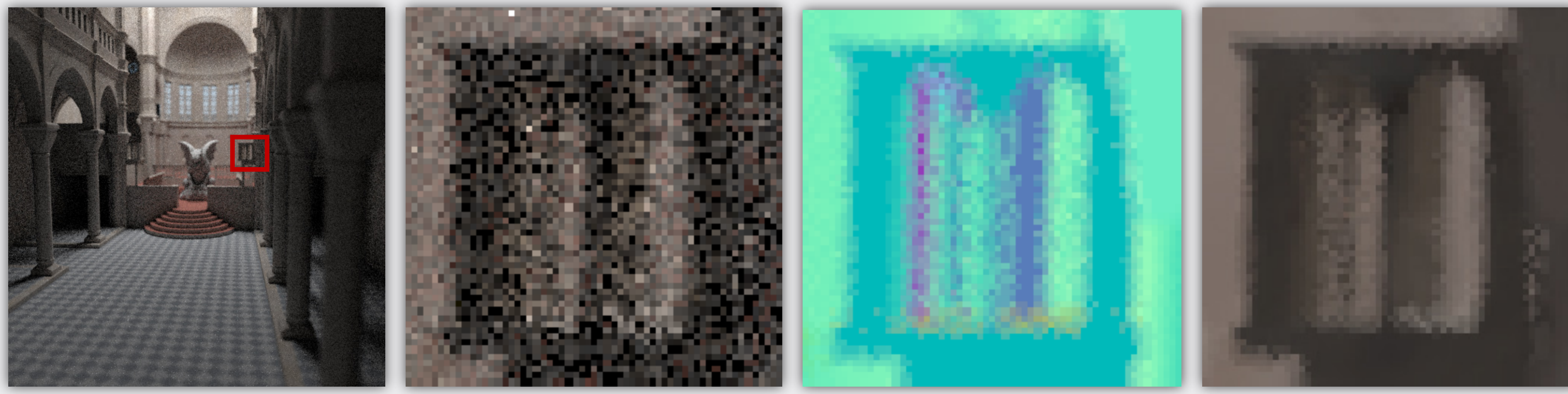
Albedo buffer

NL-Means – Leveraging Scene Info

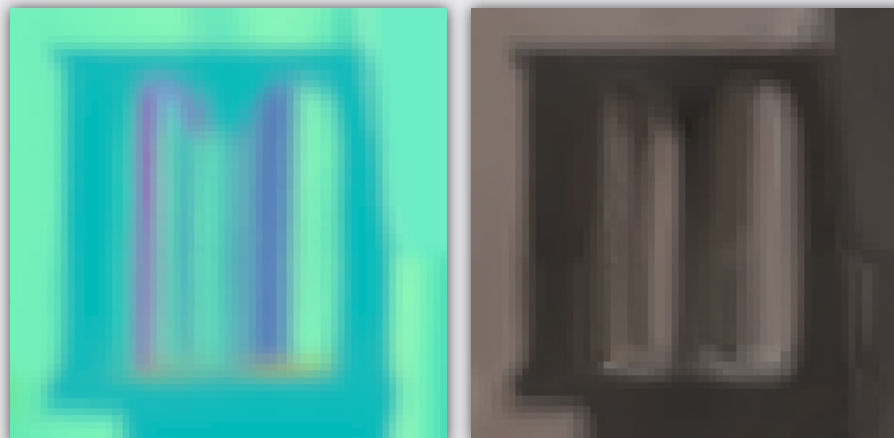
Joint filtering

Standard filtering

NL-Means Filter – Leveraging Scene Information



Unfiltered normal

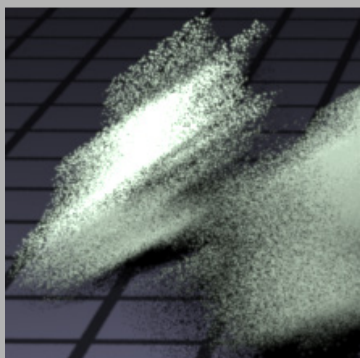


Filtered normal

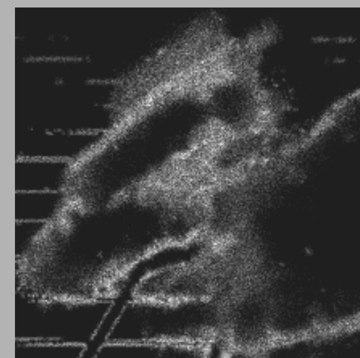
Framework overview

ADAPTIVE SAMPLING

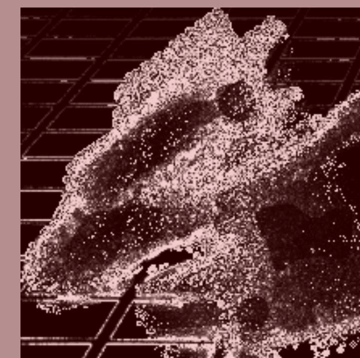
Sample set



Additional samples



Error estimation



ADAPTIVE RECONSTRUCTION

Black box!

Reconstruction

MSE estimation using SURE

SURE-based Optimization for Adaptive Sampling and Reconstruction

Li et al., ACM SIGGRAPH Asia 2012

MSE estimation using SURE

- Stein's Unbiased Risk Estimate (SURE)
  - $\mathrm{MSE}(p) = \left(\hat{u}(p) - \boxed{f(p)}\right)^2$, where $f(p)$ is the converged pixel value
  - $\left(\hat{u}(p) - \boxed{u(p)}\right)^2$, where $u(p)$ is the noisy pixel value

MSE estimation using SURE

- Stein's Unbiased Risk Estimate (SURE)
  - $\mathrm{MSE}(p) \;= \left(\hat{u}(p) - f(p)\right)^2$, where $f(p)$ is the converged pixel value
  - $$\mathrm{SURE}(p) = \left(\hat{u}(p) - u(p)\right)^2 + \underbrace{\boxed{2\frac{\delta\hat{u}(p)}{\delta u(p)}\mathrm{Var}[u(p)] - \mathrm{Var}[u(p)]}}_{0,\text{ if } \mathrm{Var}[u(p)] = 0}$$

MSE estimation using SURE

$$\mathrm{SURE}(p) = \underbrace{\left(\hat{u}(p) - u(p)\right)^2}_{0} + \underbrace{2\,\frac{\delta\hat{u}(p)}{\delta u(p)}\mathrm{Var}[u(p)] - \mathrm{Var}[u(p)]}_{\mathrm{Var}[u(p)]}$$

- Identity filter

$$\hat{u}(p) = u(p),\ \frac{\delta\hat{u}(p)}{\delta u(p)} = 1$$

$$\mathrm{SURE}(p) = \mathrm{Var}[u(p)]$$

MSE estimation using SURE

$$\mathrm{SURE}(p) = \left(\hat{u}(p) - u(p)\right)^2 + \underbrace{2\,\frac{\delta\hat{u}(p)}{\delta u(p)}\mathrm{Var}[u(p)] - \mathrm{Var}[u(p)]}_{-\mathrm{Var}[u(p)]}$$

- Identity filter

$$\hat{u}(p) = u(p),\ \frac{\delta\hat{u}(p)}{\delta u(p)} = 1$$

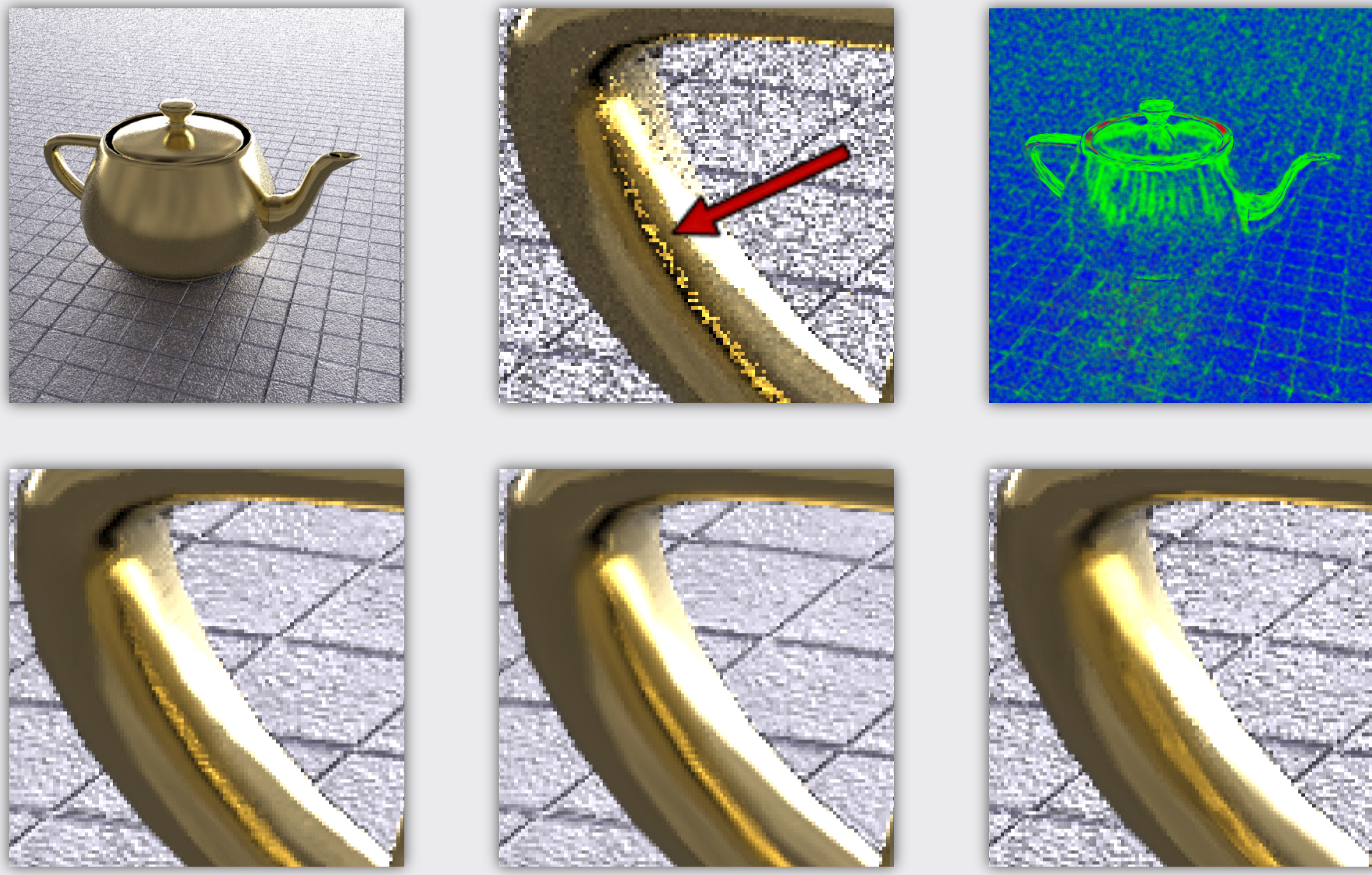
$$\mathrm{SURE}(p) = \mathrm{Var}[u(p)]$$

- Constant filter

$$\hat{u}(p) = C,\ \frac{\delta\hat{u}(p)}{\delta u(p)} = 0$$

$$\mathrm{SURE}(p) = \left(\hat{u}(p) - u(p)\right)^2 - \mathrm{Var}[u(p)]$$

Multiscale NL-Means using SURE

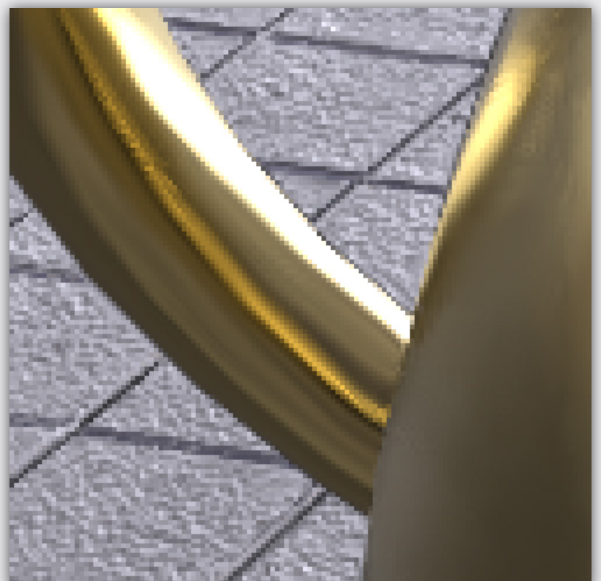


Most sensitive to **color details**

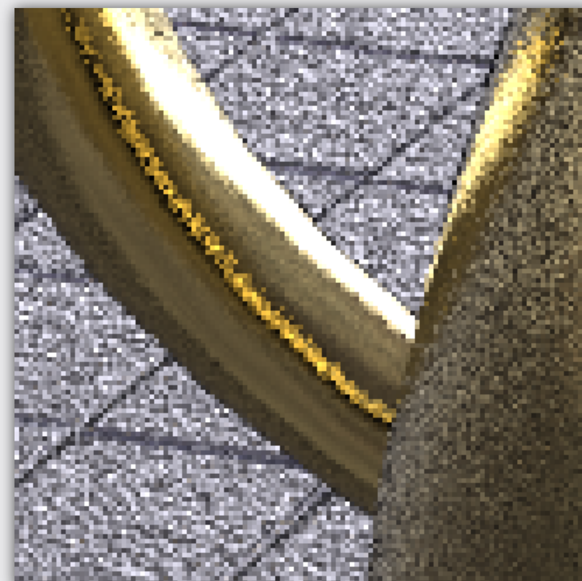
Most sensitive to **features details**

Multiscale NL-Means – results

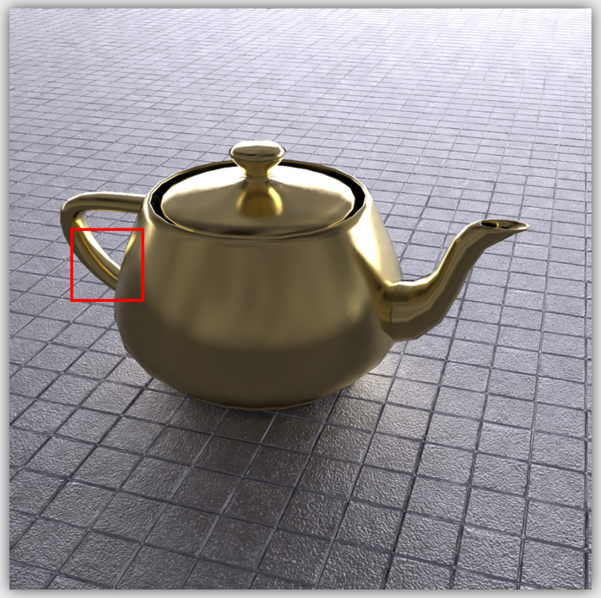
Denoising – 16 spp, 19s



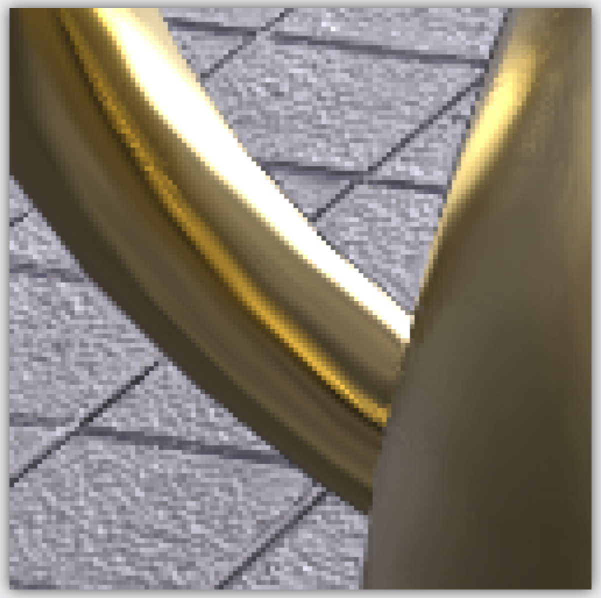
LDS – 33s, 19s



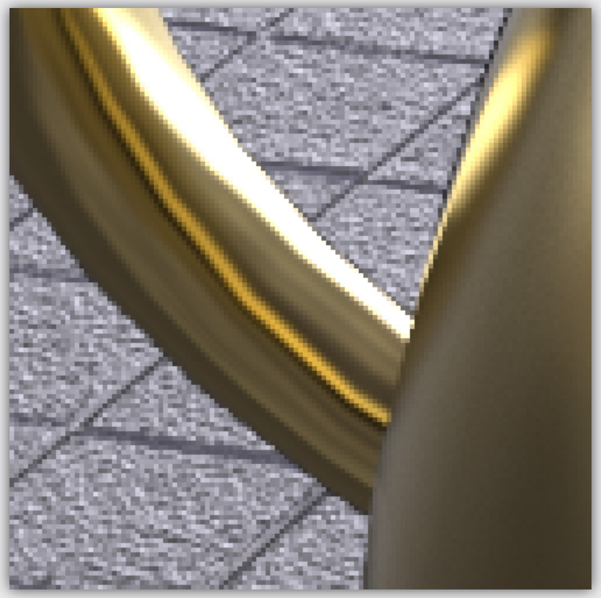
Multiscale NL-Means – results



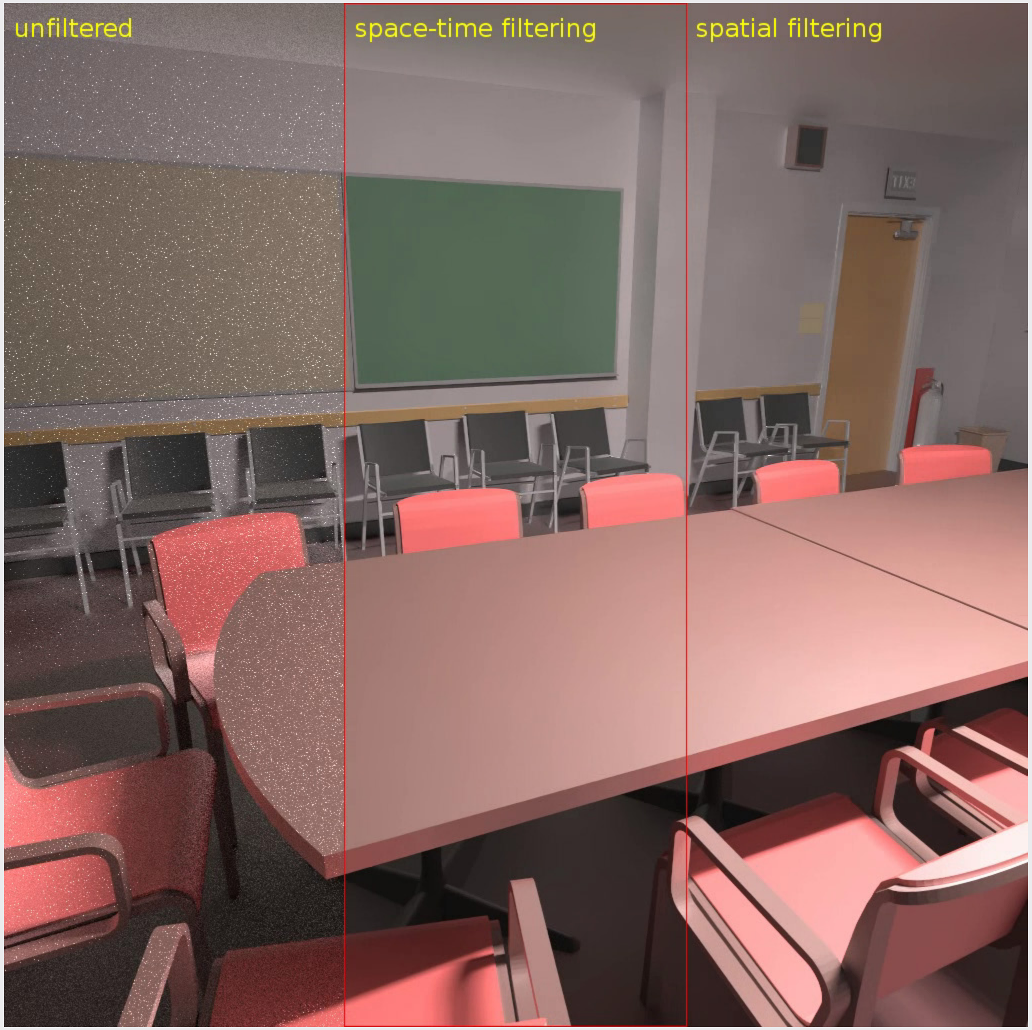
Denoising – 16 spp, 19s



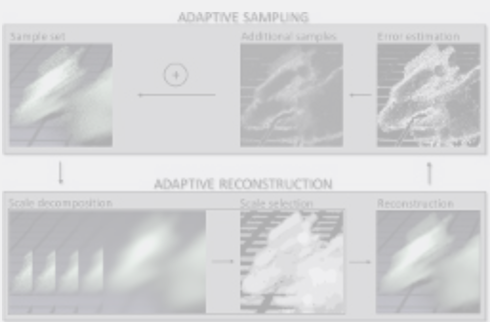
Ground truth – 4000 spp



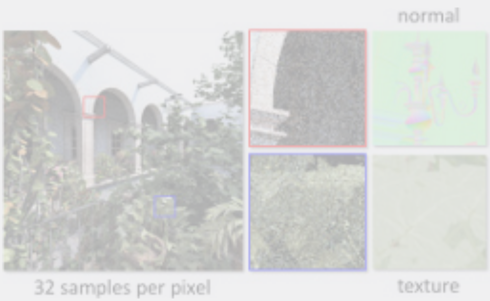
Multiscale NL-Means – results



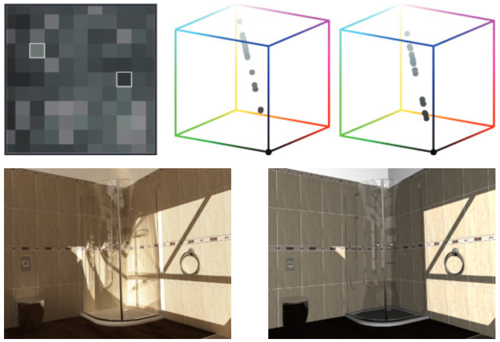
Plan



Adaptive Framework



Joint NL-Means

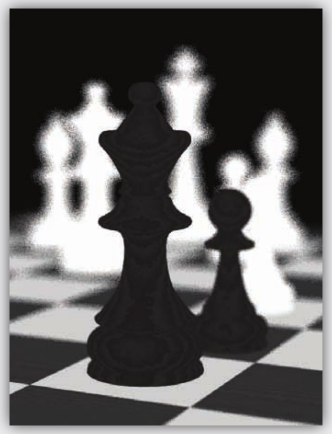


Other Implementations

Robustness

Pixel color

Normal buffer



Albedo buffer

Position buffer

Robustness

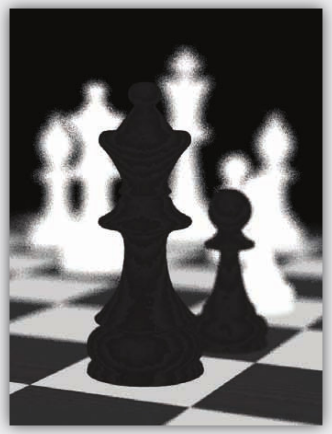
On Filtering the Noise from the Random Parameters in Monte Carlo Rendering

Sen and Darabi, TOG 2014

- Problem: what differences should be preserved?
- Solution: use mutual dependency to asses how relevant a buffer is

Pixel color

Normal buffer



Albedo buffer

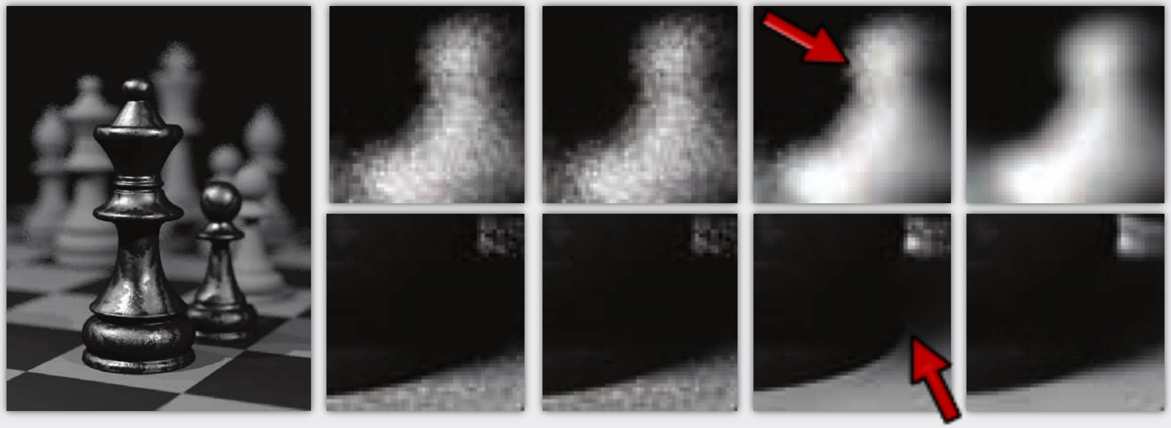
Position buffer

Robustness

On Filtering the Noise from the Random Parameters in Monte Carlo Rendering

Sen and Darabi, TOG 2014

- Problem: how to weight the color and features contributions?
- Solution: use mutual dependency with color and random parameters



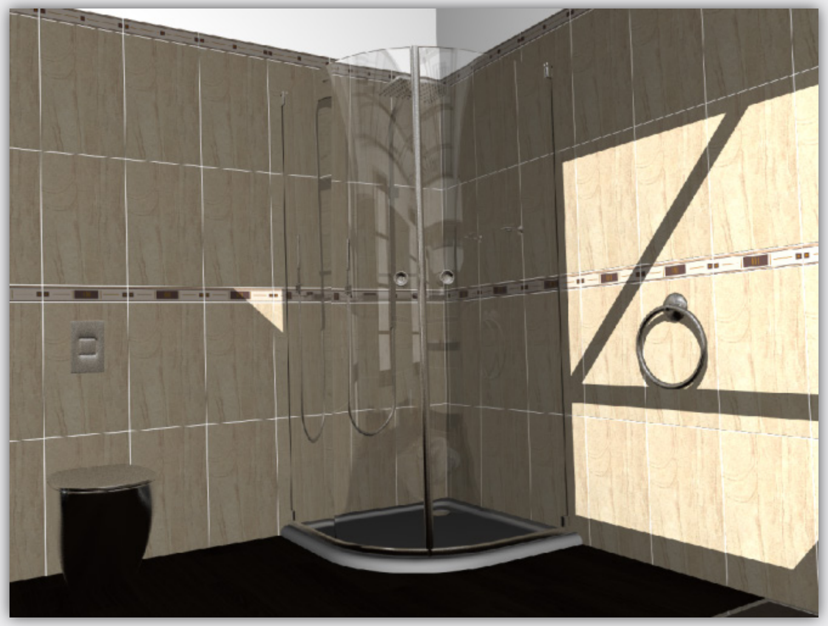
Input | Full color No feature | Full feature No color | Balanced

Robustness – new features

Robust Image Denoising using a Virtual Flash Image

Moon et al., CGF 2013

Input image

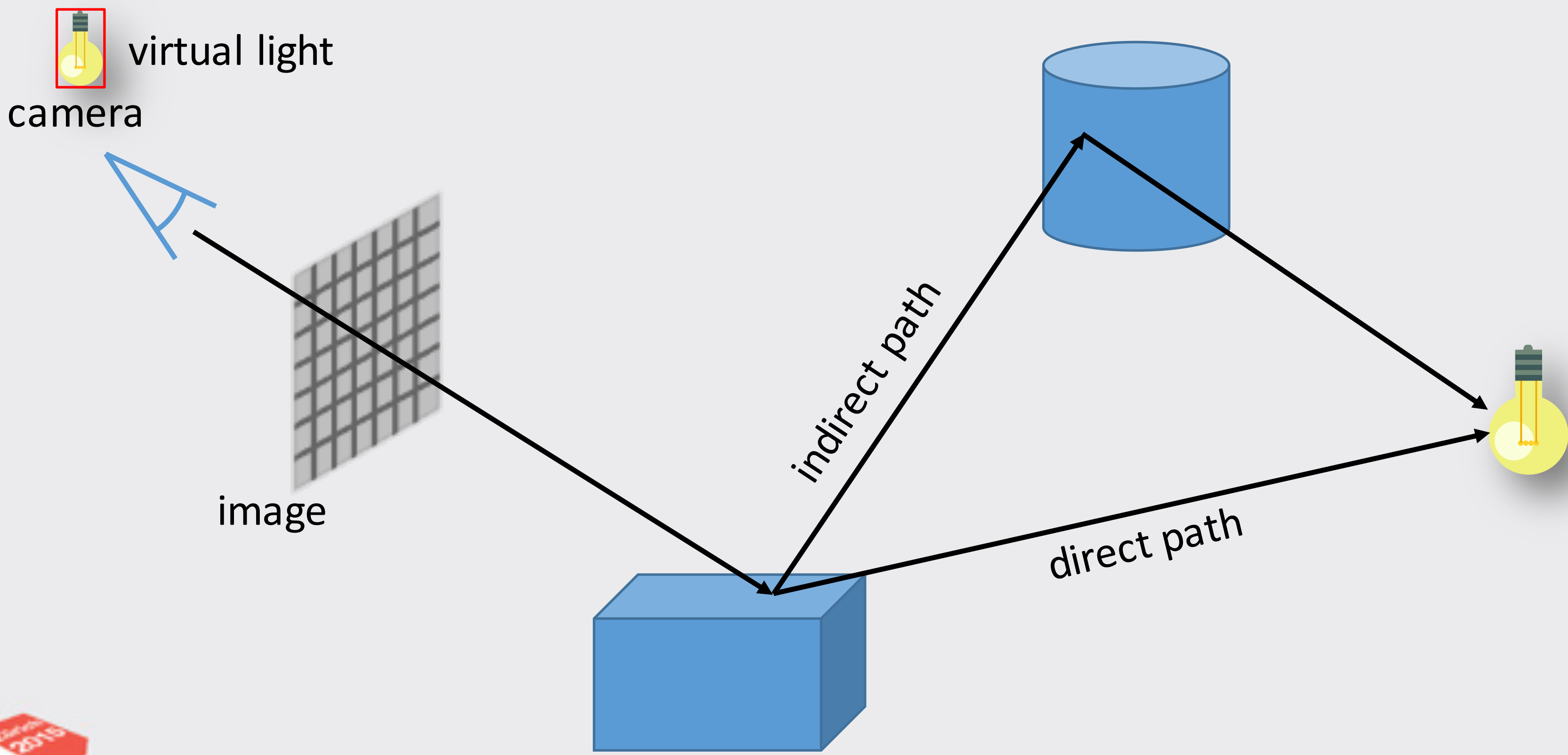


Virtual flash image

Robustness – new features

Robust Image Denoising using a Virtual Flash Image

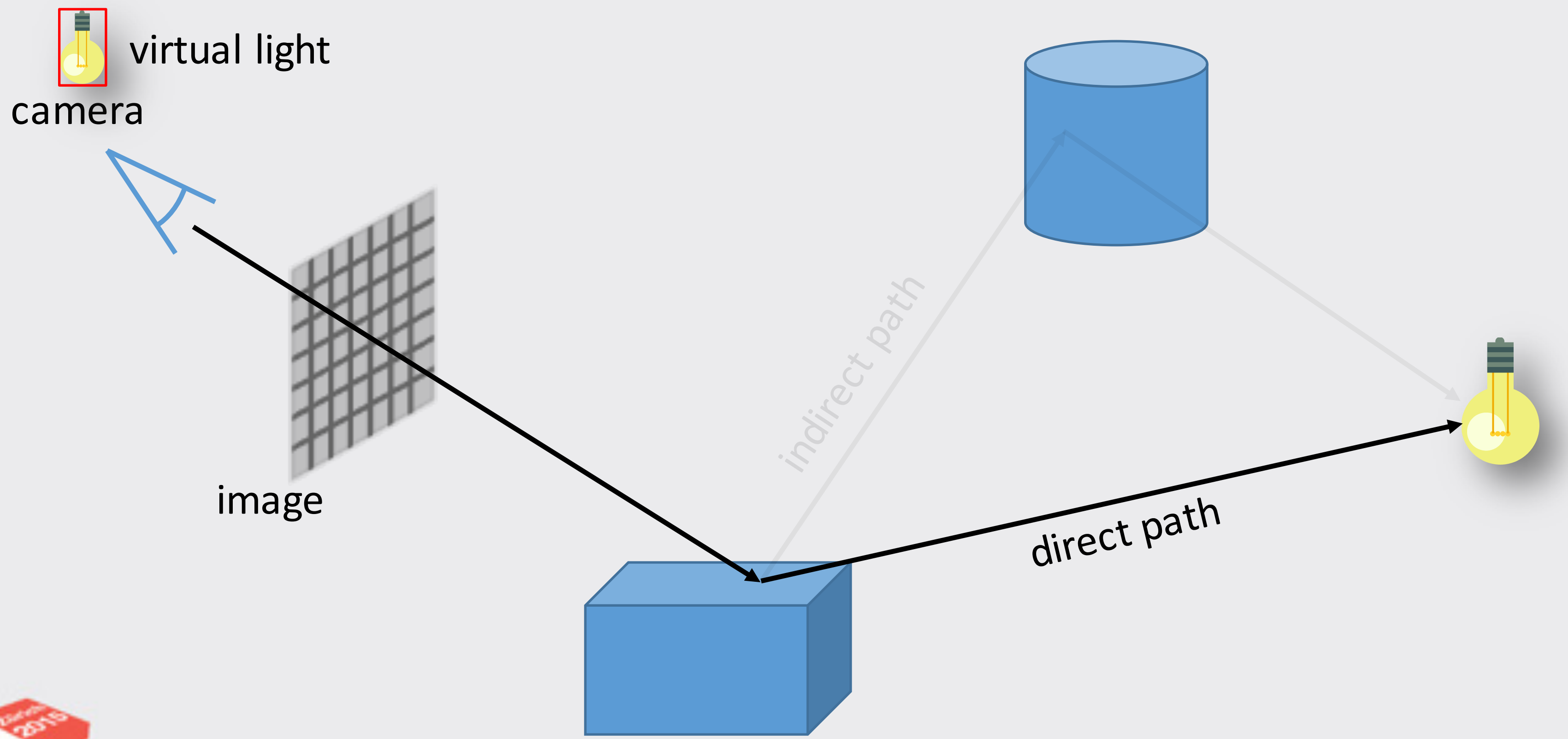
Moon et al., CGF 2013



Robustness – new features

Robust Image Denoising using a Virtual Flash Image

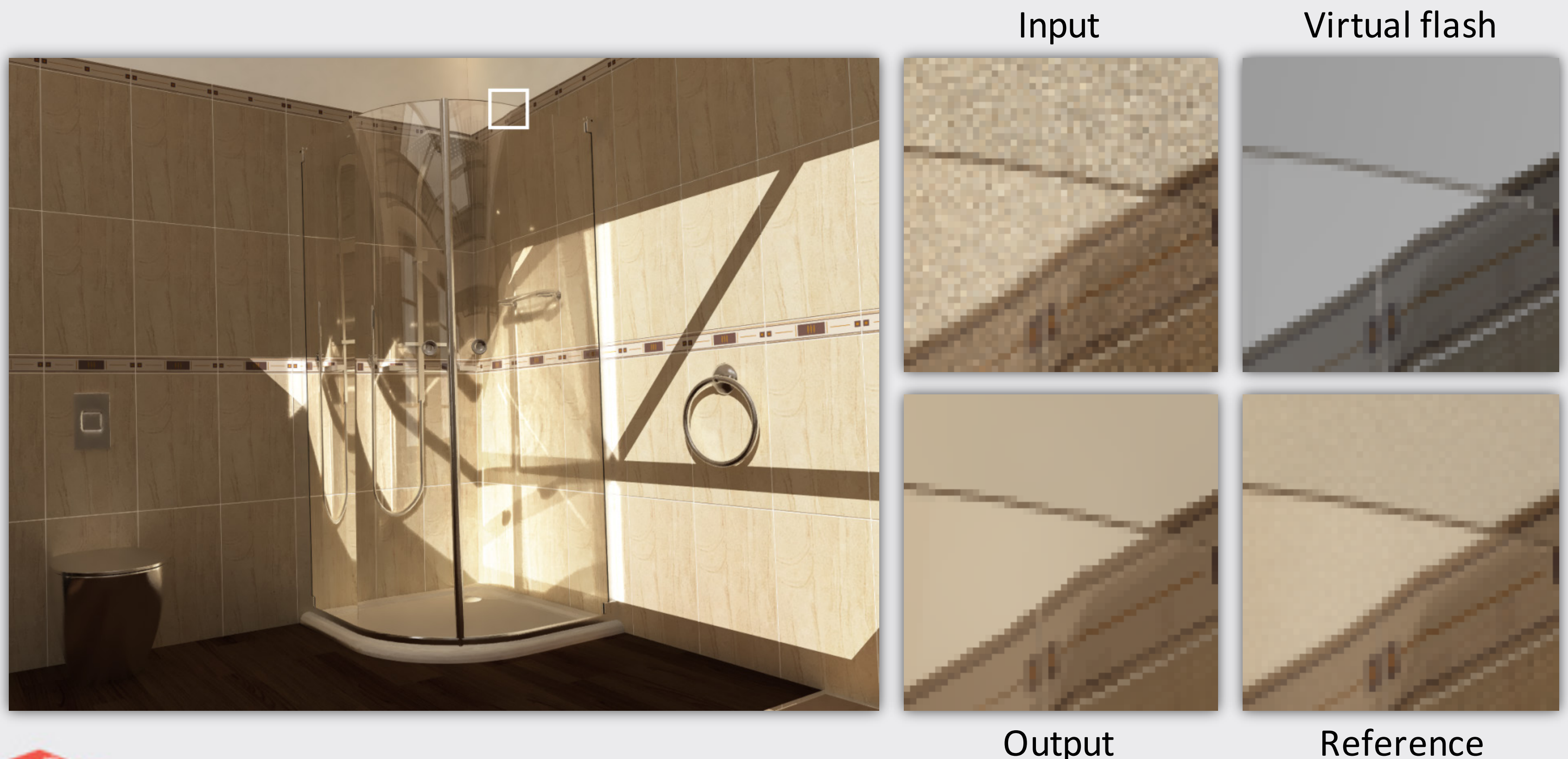
Moon et al., CGF 2013



Robustness – new features

Robust Image Denoising using a Virtual Flash Image

Moon et al., CGF 2013



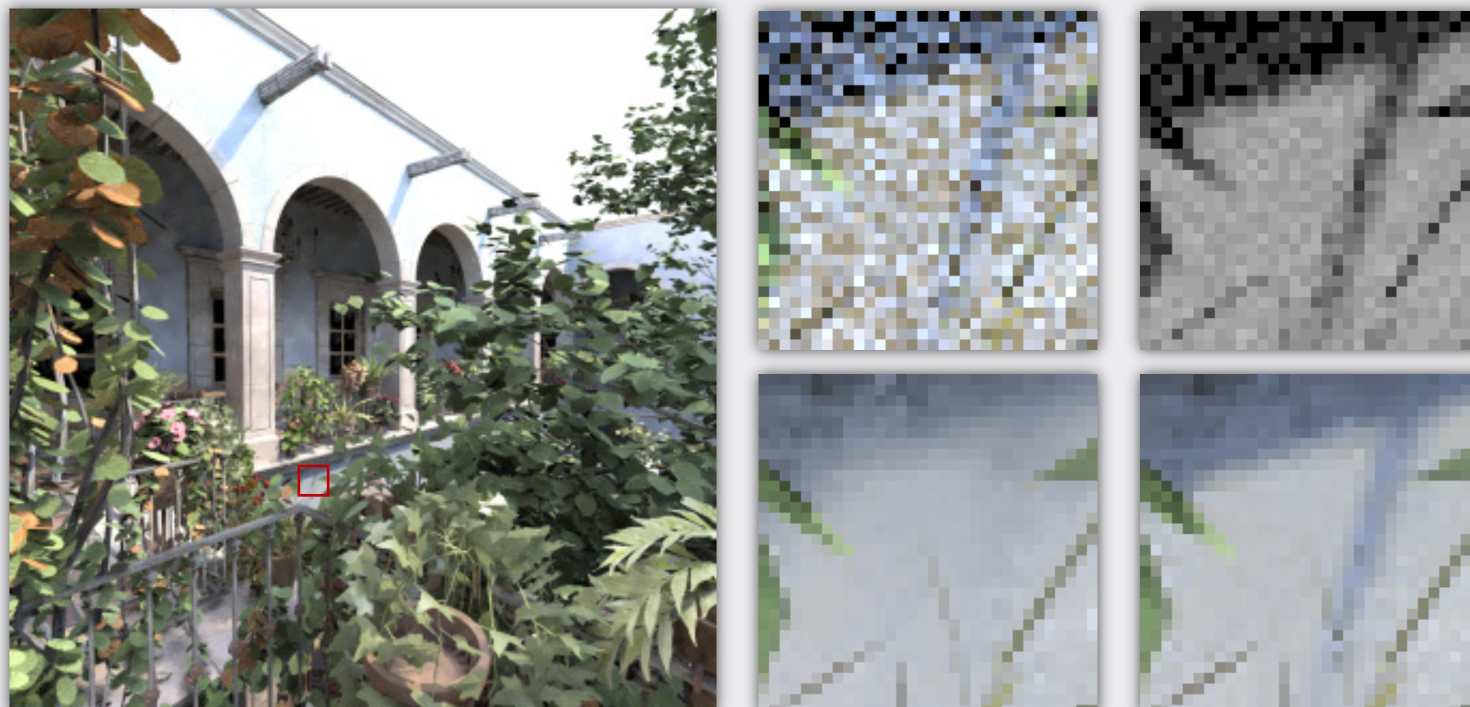
Robustness – new features

Robust Denoising using Features and Color Information

Rousselle et al., Pacific Graphics 2013

Input

Direct visibility



Without visibility

With visibility

Robustness – new features

Boosting Monte Carlo Rendering by Ray Histogram Fusion

Delbracio et al., TOG 2014

Robustness – new features

Boosting Monte Carlo Rendering by Ray Histogram Fusion

Delbracio et al., TOG 2014

176 spp

NL-Means – 168spp

RHF – 128 spp

Reference

Robustness – new features

Boosting Monte Carlo Rendering by Ray Histogram Fusion

Delbracio et al., TOG 2014

176 spp

NL-Means – 168spp

Reference

Robustness – new features

Boosting Monte Carlo Rendering by Ray Histogram Fusion

Delbracio et al., TOG 2014

176 spp

RHF – 128spp

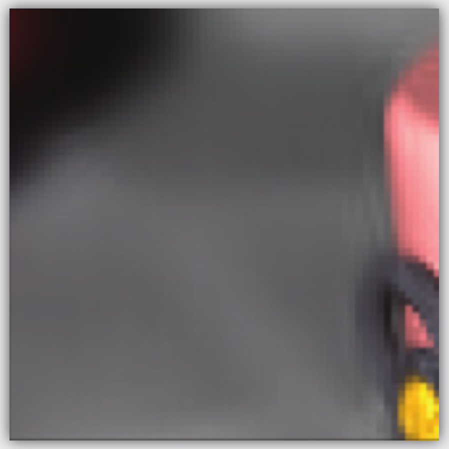
Reference

Robustness – new features

Boosting Monte Carlo Rendering by Ray Histogram Fusion

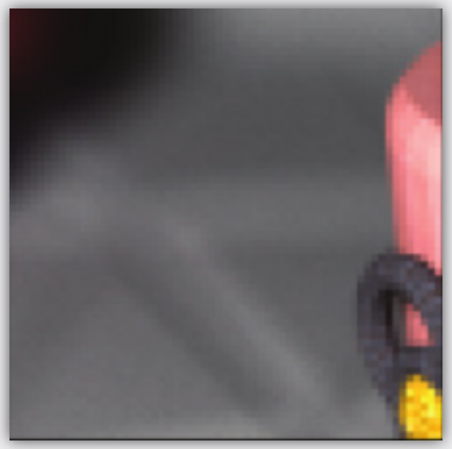
Delbracio et al., TOG 2014

LD – 353 spp, 89s



NLM – 316 spp, 89s

RHF – 256 spp, 89s



Reference

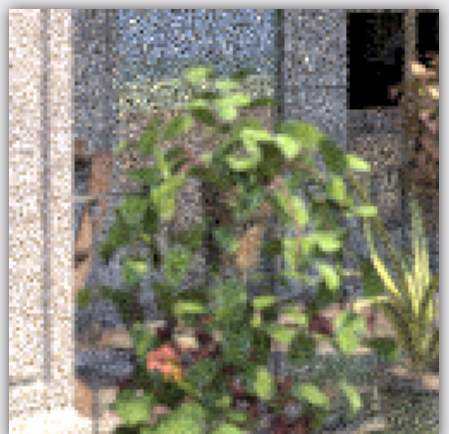
Other denoising schemes

Other denoising schemes

Adaptive Rendering based on Weighted Local Regression

Moon et al., TOG 2014

$y = f(\mathbf{x}) + \epsilon$ $\mathbf{x}$ = (Px, Py :)



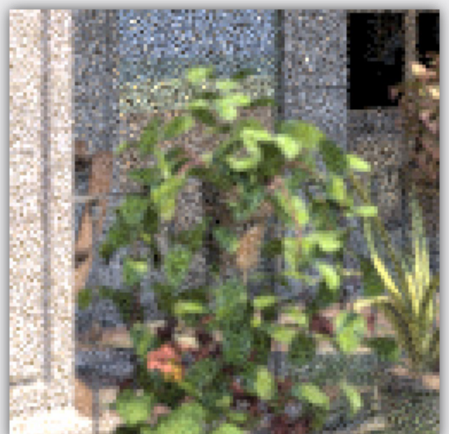
Pixel color

Other denoising schemes

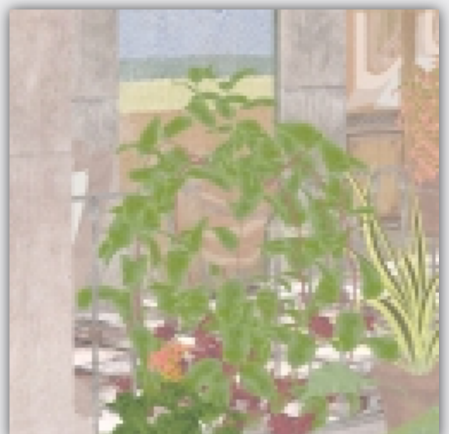
Adaptive Rendering based on Weighted Local Regression

Moon et al., TOG 2014

$y = f(\mathbf{x}) + \epsilon$ $\mathbf{x}$ = (Px, Py, Ar, Ag, Ab)



Pixel color



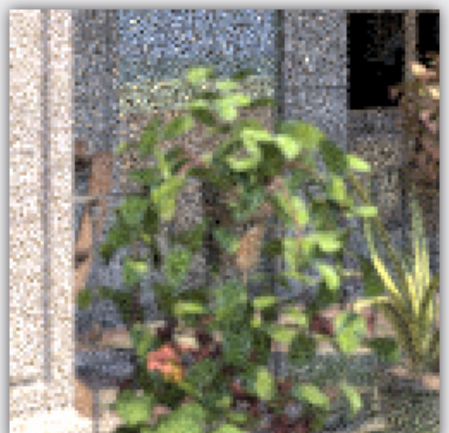
Albedo buffer

Other denoising schemes

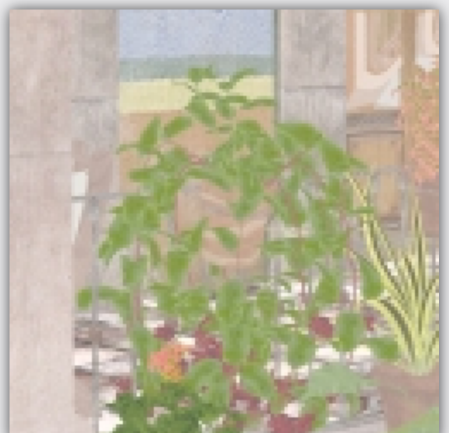
Adaptive Rendering based on Weighted Local Regression

Moon et al., TOG 2014

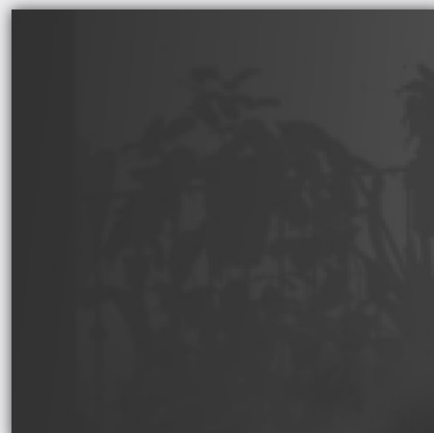
$y = f(\mathbf{x}) + \epsilon$ $\mathbf{x}$ = (Px, Py, Ar, Ag, Ab, D :)



Pixel color



Albedo buffer



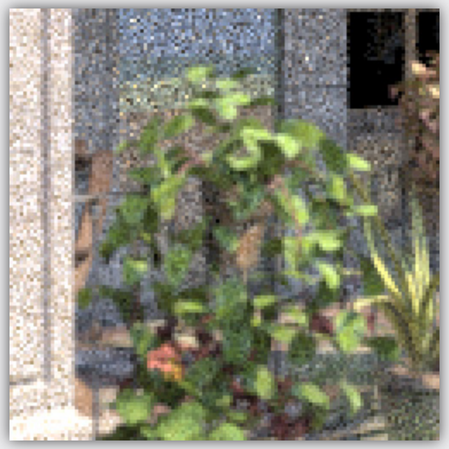
Depth buffer

Other denoising schemes

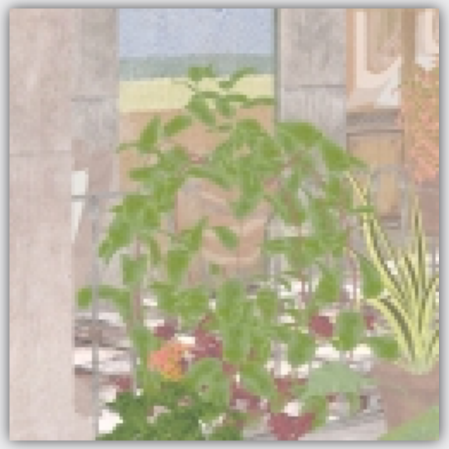
Adaptive Rendering based on Weighted Local Regression

Moon et al., TOG 2014

$y = f(\mathbf{x}) + \epsilon$ $\mathbf{x}$ = (Px, Py, Ar, Ag, Ab, D, Nx, Ny, Nz)



Pixel color



Albedo buffer

Depth buffer

Normal buffer

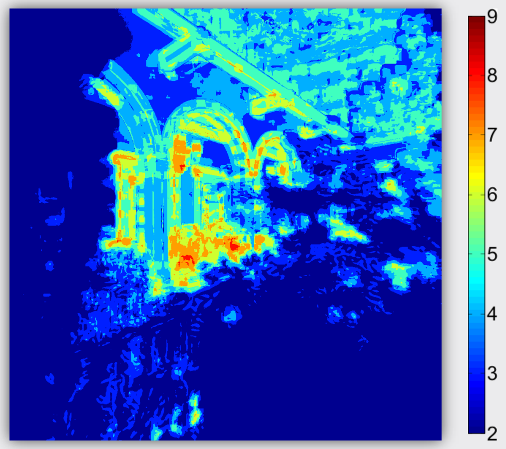
Other denoising schemes

Adaptive Rendering based on Weighted Local Regression

Moon et al., TOG 2014

$y = f(\mathbf{x}) + \epsilon$ $\mathbf{x}$ = (Px, Py, Ar, Ag, Ab, D, Nx, Ny, Nz) $\Rightarrow \mathrm{TSVD} \Rightarrow \mathbf{z}$

Input

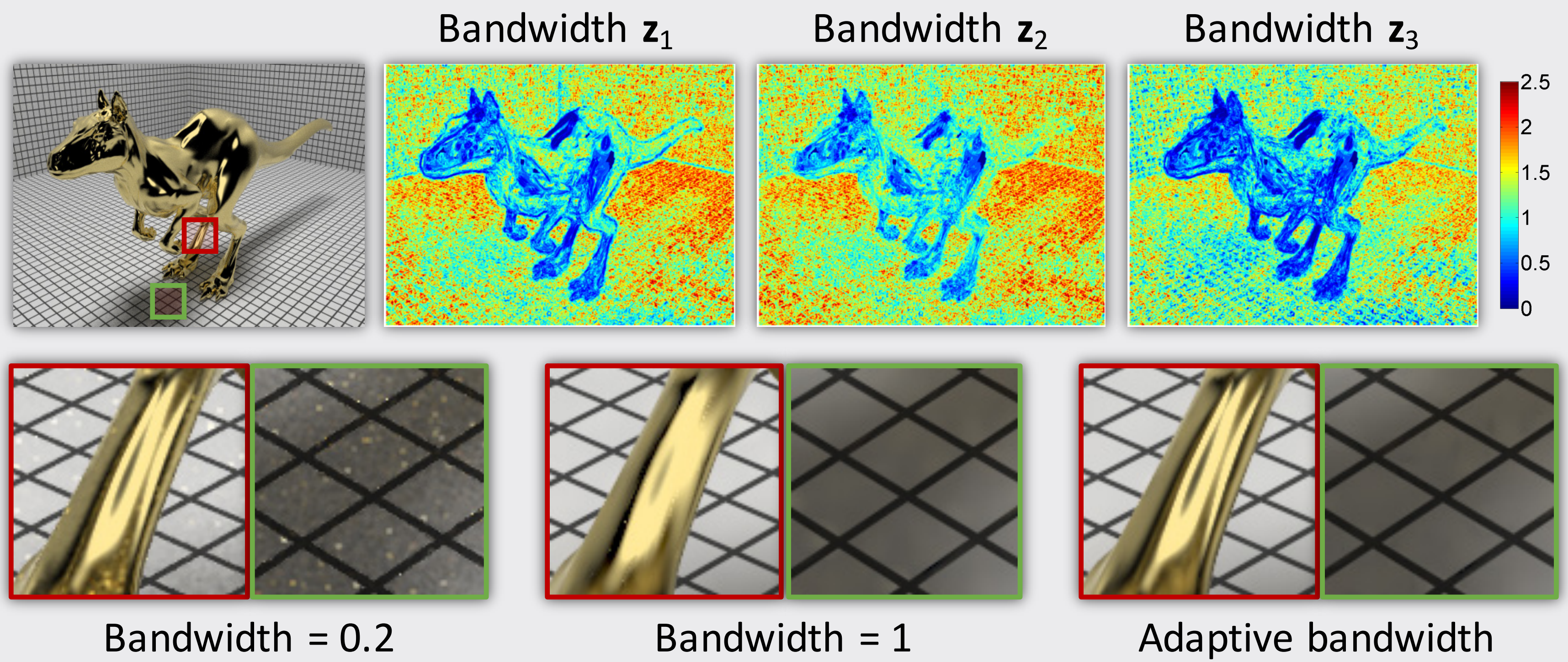


Dimensions after truncation

Other denoising schemes

Adaptive Rendering based on Weighted Local Regression

Moon et al., TOG 2014



Other denoising schemes

Adaptive Rendering based on Weighted Local Regression

Moon et al., TOG 2014

LD – 128 spp   WLR – 115 spp   Reference

Other denoising schemes

Removing the Noise in Monte Carlo Renderings with General Image Denoising Algorithms

Kalantari and Sen, Eurographics 2013

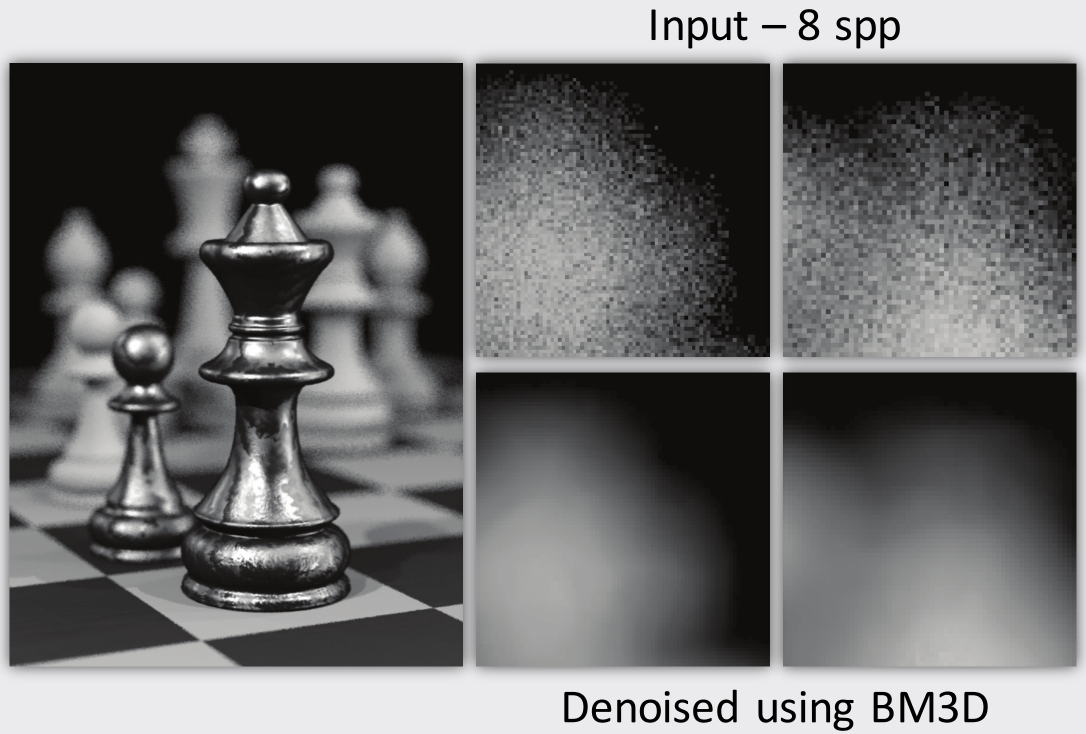
- Problem: general filters assume uniform variance
- Idea: multi-scale filtering over a range of variances

σ^2=0.0 σ^2 = 28.73 σ^2 =43.16 σ^2 = 57.46 σ^2 = 64.44 σ^2 = 118.2

Other denoising schemes

Removing the Noise in Monte Carlo Renderings with General Image Denoising Algorithms

Kalantari and Sen, Eurographics 2013

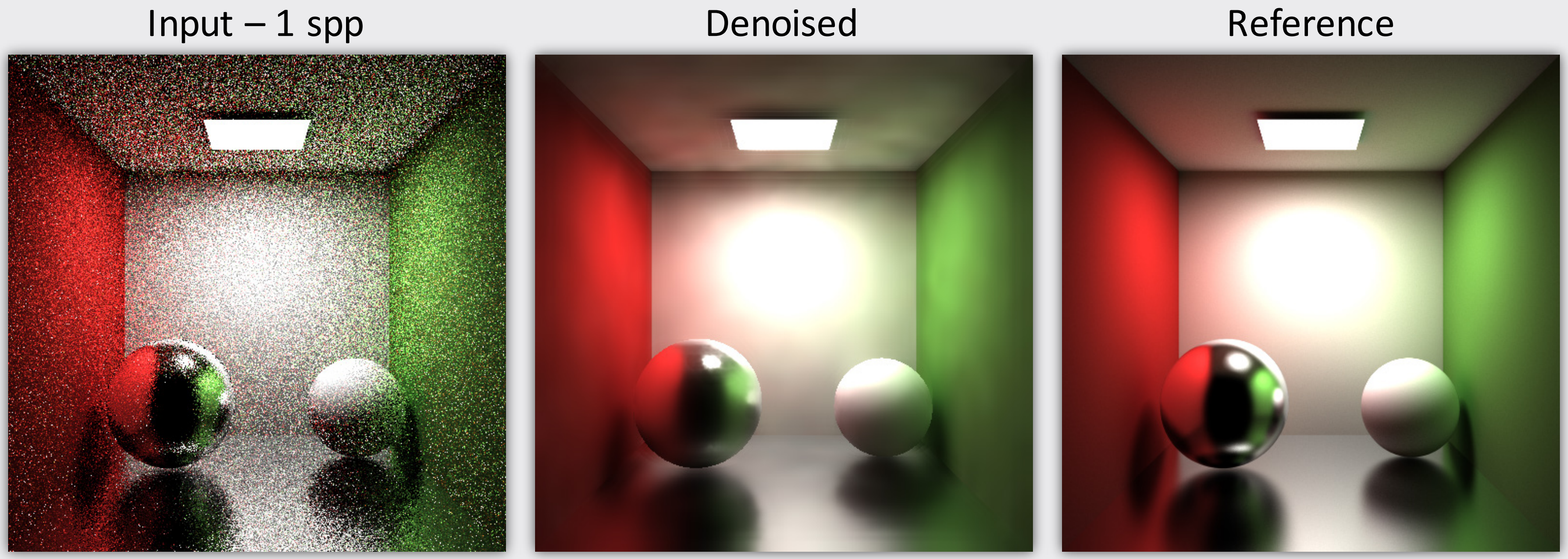


Interactive applications

Interactive applications

Edge-Avoiding A-Trous Wavelet Transform for Fast Global Illumination Filtering

Dammertz et al., HPG 2010

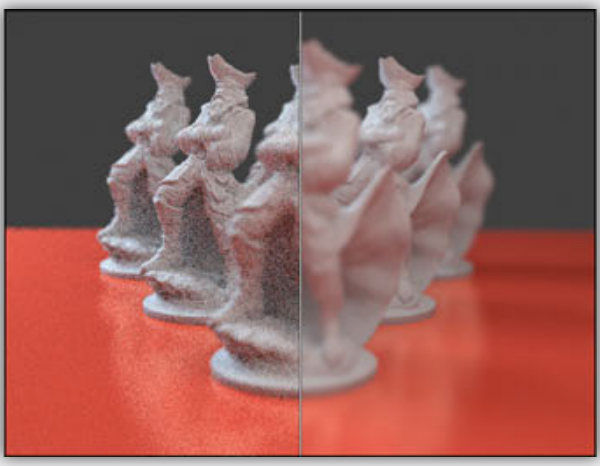


Filtering time: 5.6ms

Interactive applications

Sample-Based Manifold Filtering for Interactive Global Illumination and Depth of Field

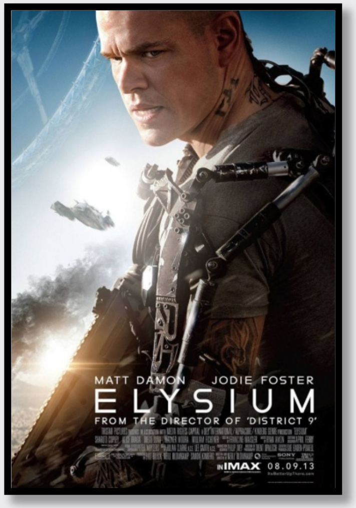
Bauszat et al., CGF 2014



Conclusion

- Image-space adaptive rendering
  - Very effective
  - Low computational overhead
  - Preserve the generality of Monte Carlo rendering
- Applications
  - Facilitate the deployment of Monte Carlo rendering in production
  - Key to enable interactive/realtime Monte Carlo rendering

Production use



Plan

LD | SBF | RFC | WLR | Reference

Interactive applications

Specular Lobe-Aware Filtering and Upsampling for Interactive Indirect Illumination

Tokuyoshi, CGF 2015

Normal-aware | Lobe-aware | Ground truth

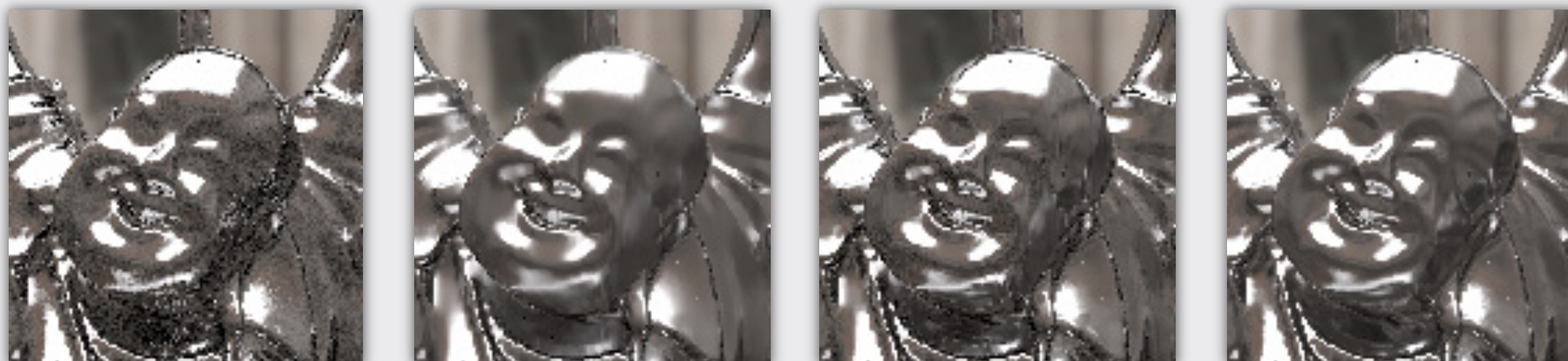
MSE: 0.0069 | MSE: 0.0069

Interactive applications

Specular Lobe-Aware Filtering and Upsampling for Interactive Indirect Illumination

Tokuyoshi, CGF 2015

Normal-aware | Lobe-aware | Ground truth



MSE: 0.0549 | MSE: 0.0158

Interactive applications

Specular Lobe-Aware Filtering and Upsampling for Interactive Indirect Illumination

Tokuyoshi, CGF 2015

Normal-aware

MSE: 0.0549

Interactive applications

Specular Lobe-Aware Filtering and Upsampling for Interactive Indirect Illumination

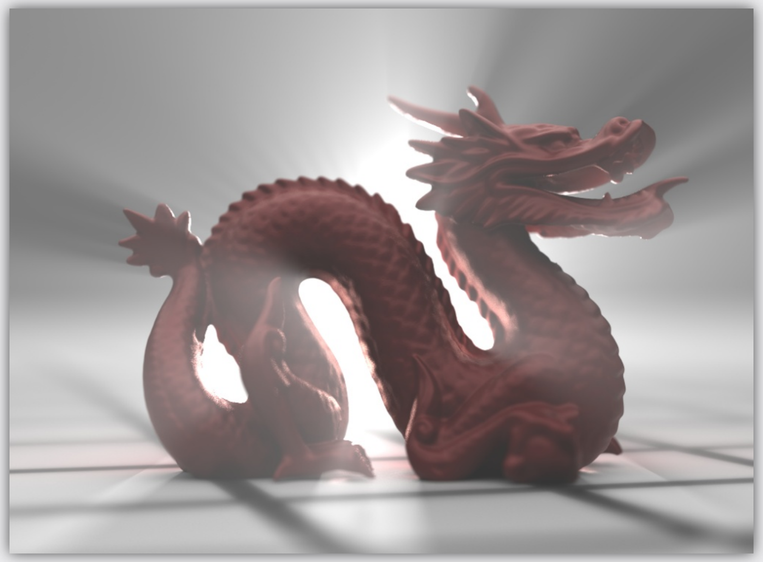
Tokuyoshi, CGF 2015

Lobe-aware

MSE: 0.0158

Introduction

32000 samples per pixel, 12h

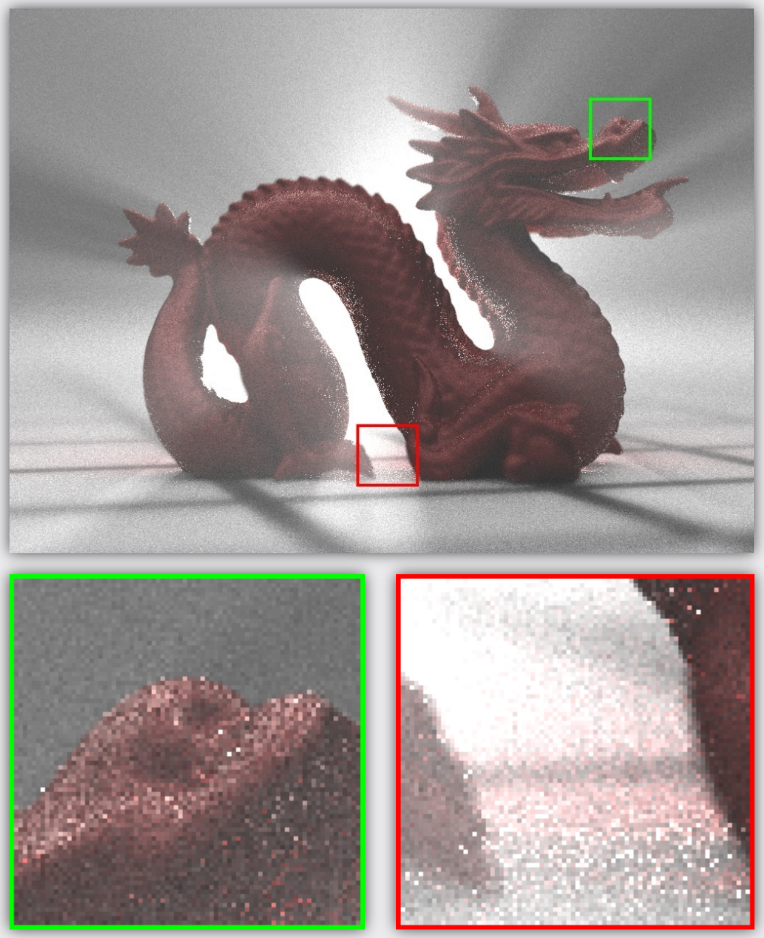


- Monte Carlo path tracing
  - Physically based
  - Very general
  - Guaranteed convergence (except pathological cases)
- Disadvantages
  - Noise, slow convergence

Introduction

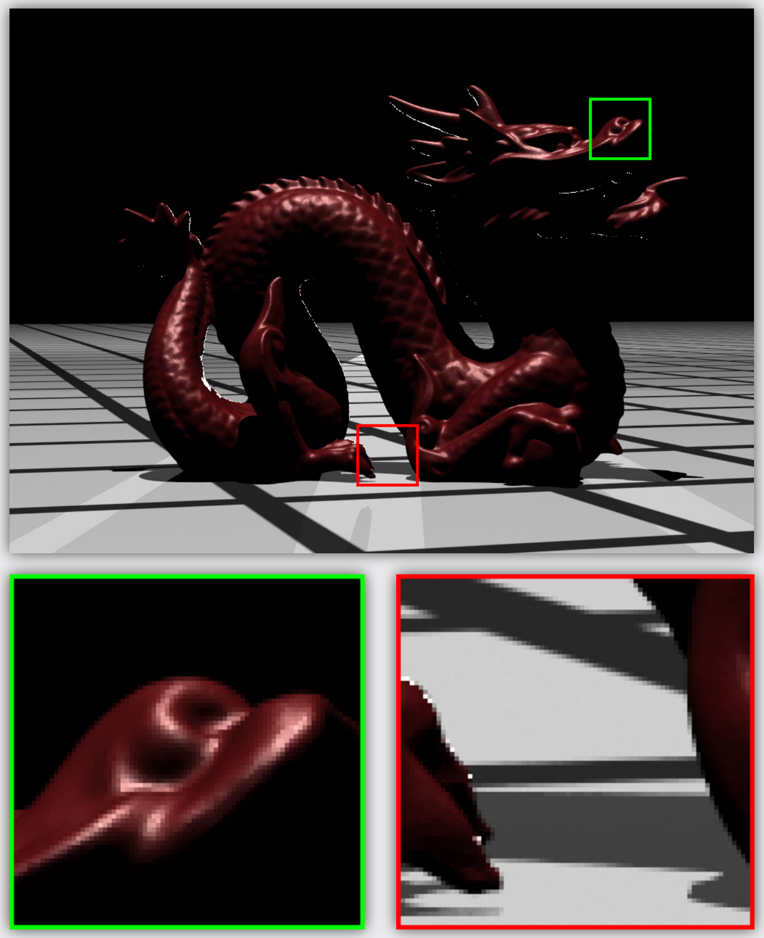
- Monte Carlo path tracing
  - Physically based
  - Very general
  - Guaranteed convergence (except pathological cases)
- Disadvantages
  - Noise, slow convergence

32 samples per pixel, 42s



Introduction – curse of dimensionality

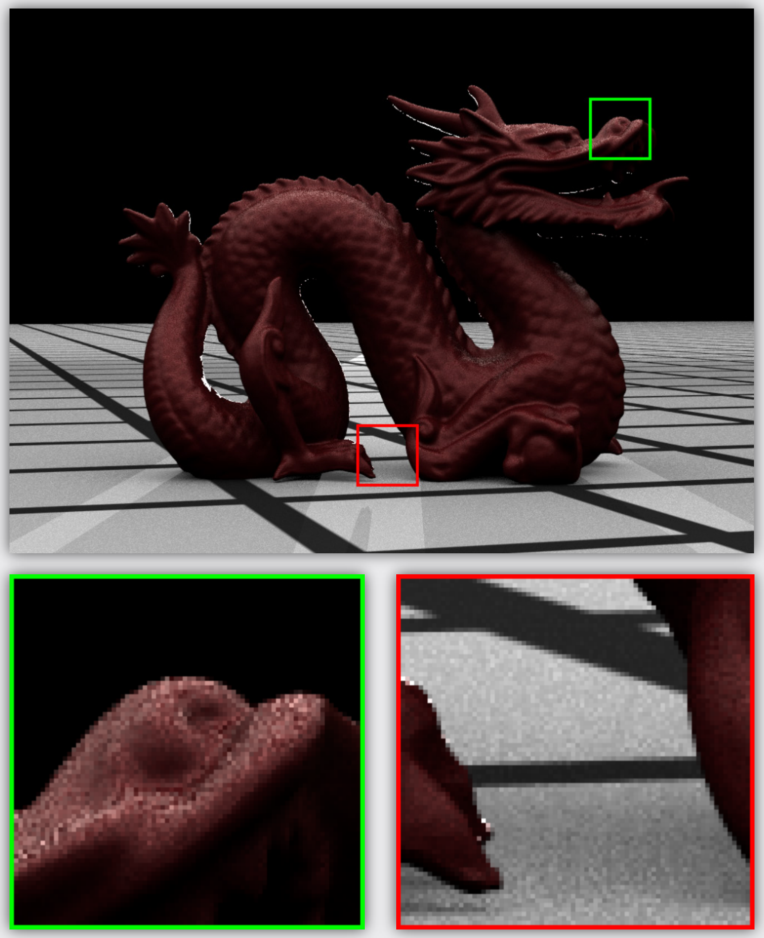
→ Anti-aliasing – 2D



Introduction – curse of dimensionality

→ Anti-aliasing – 2D

→ Area-lighting – 2D

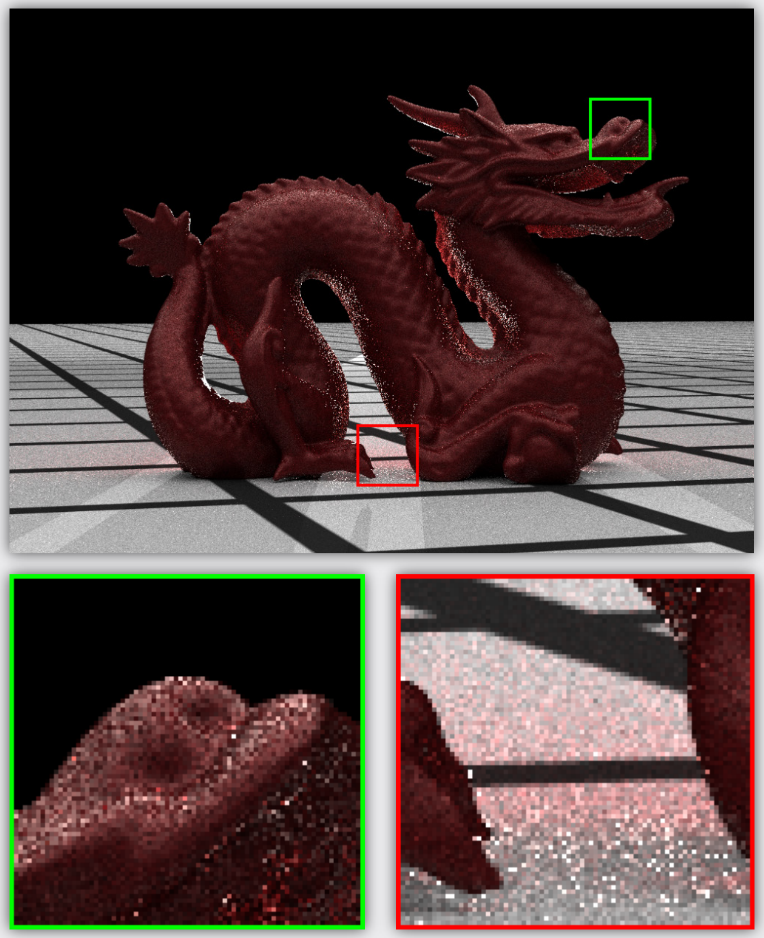


Introduction – curse of dimensionality

→ Anti-aliasing – 2D

→ Area-lighting – 2D

→ Single-bounce indirect illumination – 2D



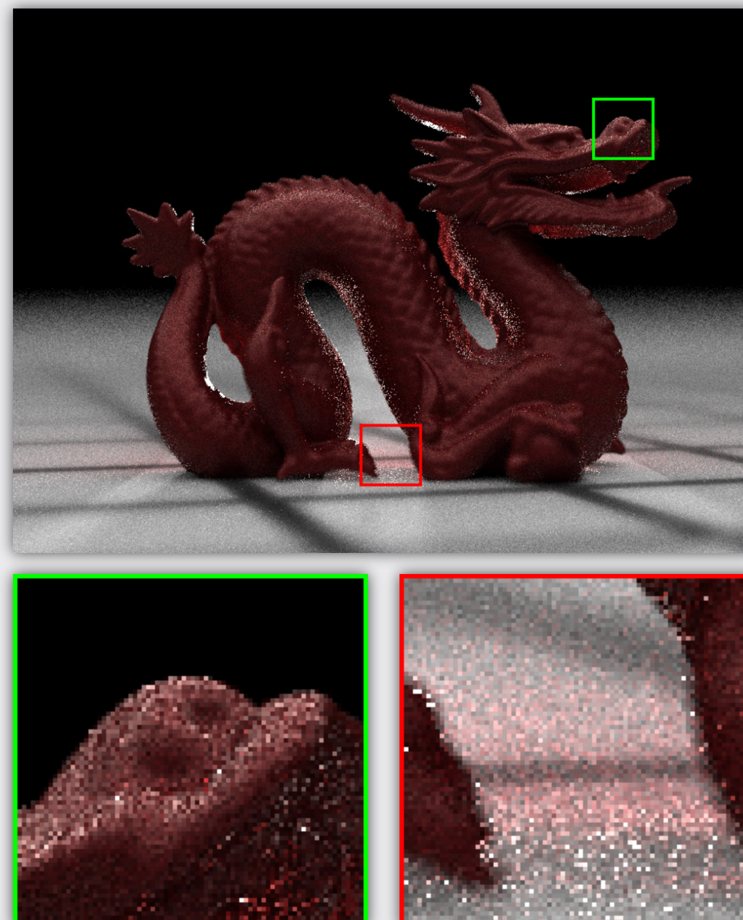
Introduction – curse of dimensionality

→ Anti-aliasing – 2D

→ Area-lighting – 2D

→ Single-bounce indirect illumination – 2D

→ Depth-of-field – 2D



Introduction – curse of dimensionality

→ Anti-aliasing – 2D
→ Area-lighting – 2D
→ Single-bounce indirect illumination – 2D
→ Depth-of-field – 2D
→ Single-scattering participating media – 1D

Total: 9D