CS5630 Physically Based Realistic Rendering

Steve Marschner

Spring 2026

21 Denoising Monte Carlo renderings

Outline

Need for denoising

Basic noise removal methods

- bilateral filter
- nonlocal means filter

Making use of extra information

- arbitrary output value (AOV) buffers
- joint bilateral and joint nonlocal means filters
- denoising components separately

Learned denoisers

- kernel prediction
- direct output prediction
- kernel splatting

Reducing noise

Monte Carlo integration has variance

Many ways to reduce

- more samples
- better importance sampling
- better renderering algorithms
- etc., etc.

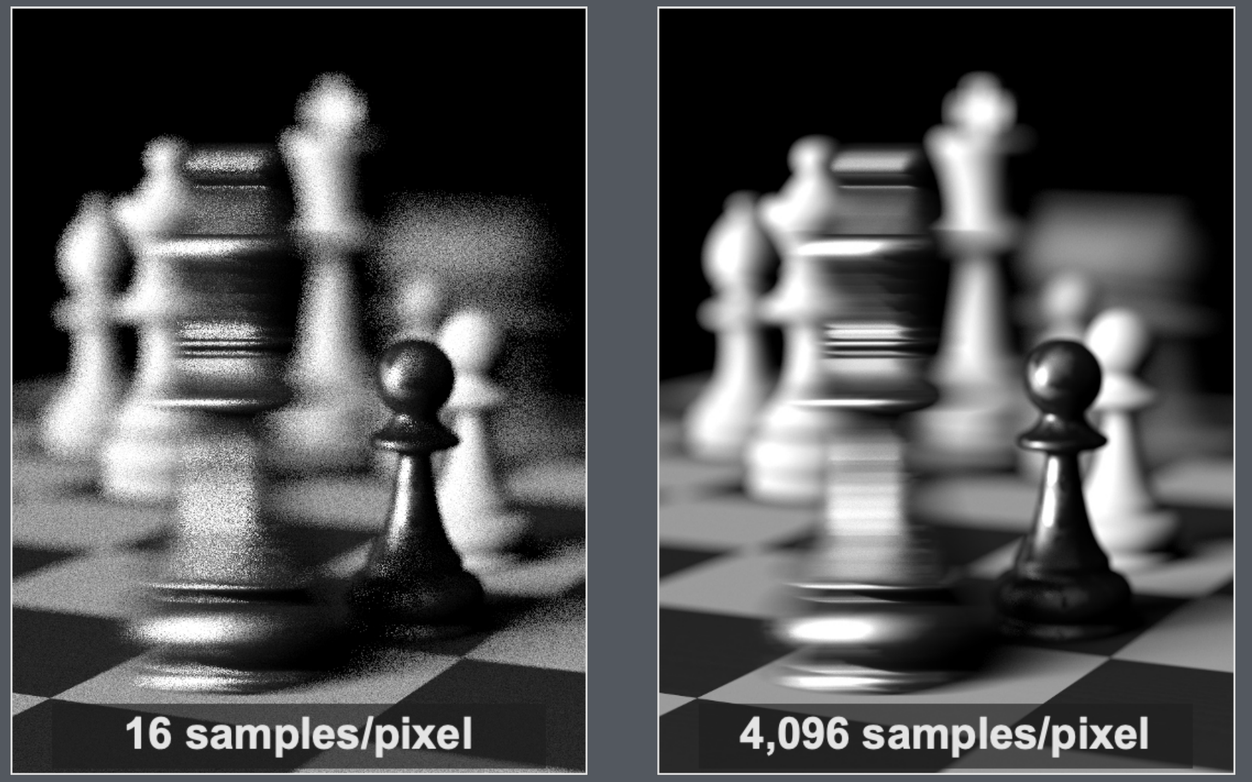


image: Pradeep Sen

This is great, but there is still noise!

- is the noisy image really our best guess at the real image?
- can we get closer to the real image (visually and numerically) if we tolerate some bias?
- this is the goal of post-processing **denoising filters** or **denoisers**

Bilateral filter

Recall bilateral filter

- basic plan: weighed sum

denoised color | weight | noisy color | total weight

$$\tilde{C}(p) = \frac{1}{W_p} \sum_{q \in N(p)} w(p,q) C(q) \text{ where } W_p = \sum_{q \in N(p)} w(p,q)$$

neighborhood

- weghts computed based on both $d_s(p,q) = \|p - q\|$ and $d_r(p,q) = \|C(p) - C(q)\|$

$$w(p,q) = e^{-d^2(p,q)} \text{ where } d^2(p,q) = k_s^2 d_s^2(p,q) + k_r^2 d_r^2(p,q)$$

generalized distance

- some let $k_s = 0$ and let the box shaped neighborhood define the spatial exent

original noisy image

bilateral filter

images: Fabrice Rousselle

Nonlocal means filter

Generalization of bilateral filter

- instead of comparing the pixels p and q, compare *patches* centered at p and q

patch distance | range distance for indiviual pixels

$$d_p^2(p,q) = \frac{1}{N} \sum_{n \in P(0)} d_r^2(p+n, q+n) \text{ where } N = (2f+1)^2$$

  - here $P(0)$ is a patch of size $(2f+1) \times (2f+1)$ centered at zero

- then as before $w(p,q) = e^{-d^2(p,q)}$ and $d^2(p,q) = k_s^2 d_s^2(p,q) + k_r^2 d_p^2(p,q)$

- to reduce noise in weights: average weights over several patch pairs with same offset:

$$W(p,q) = \frac{1}{N} \sum_{n \in P(0)} w(p+n, q+n)$$

original noisy image

bilateral filter

nonlocal means filter $f = 3$

images: Fabrice Rousselle

Nonlocal means refinements

Correcting for variance

- patch differences are due to both variance and actual differences
- want to weight based on patch difference relative to uncertainty
- so: subtract off the variance, normalize by the variance

$$w(p,q) = \exp\left[-\frac{d_p^2(p,q) - 2\sigma^2}{k^2 2\sigma^2}\right]$$

variance of C(p) and C(q)

tuning constant

- estimate and use spatially varying Monte Carlo variance in this correction

128 samples per pixel

Denoised

figure: Fabrice Rousselle

Cross bilateral filter

In rendering more information is available than with measured images

- visible shading point
- visible surface normal
- surface diffuse color
- …

It’s easy to write this information into extra channels in the image

Cross bilateral filter

Simple adjustment: range distance includes both pixel color and features

- patch-to-patch color distance d_p same as before
- feature distance is similar but depends on features rather than colors

$$d_f^2(p,q) = \sum_i k_j^2 (F_j(p) - F_j(q))^2$$

$$d^2(p,q) = k_s^2 d_s^2(p,q) + k_r^2 d_p^2(p,q) + d_f^2(p,q)$$

Machine learning for denoising

Next steps in denoising all involved machine learning

- since this is essentially a business of tuning heuristics, we might expect that…

Will look at some research papers from late 2010s

- A Machine Learning Approach… [Kalantari et al. 2015 (UCSB)]
- Kernel-Predicting Convolutional Networks… [Bako et al. 2017 (Disney Research)]
- …Recurrent Denoising Autoencoder [Chakravarty et al. 2017 (NVIDIA)]
- …Kernel-Splatting Network [Gharbi et al. 2019 (MIT)]

A Machine Learning Approach…

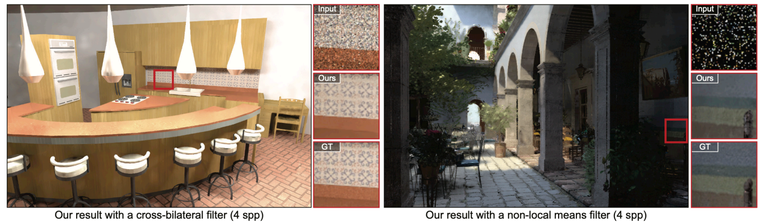
Built on previous bilateral and NLM filters

Learns parameters for filters

Considerable quality/robustness gains

A Machine Learning Approach for Filtering Monte Carlo Noise

Nima Khademi Kalantari Steve Bako Pradeep Sen
University of California, Santa Barbara



Our result with a cross-bilateral filter (4 spp) Our result with a non-local means filter (4 spp)

Figure 1: *We propose a machine learning approach to filter Monte Carlo rendering noise as a post-process. In our method, we use a set of scenes with a variety of distributed effects to train a neural network to output correct filter parameters. We then use the trained network to drive a filter to denoise a new MC rendered image. We show the result of our approach with a cross-bilateral filter for the* KITCHEN *scene (1200 × 800) on the left and with a non-local means filter for the* SAN MIGUEL HALLWAY *scene (800 × 1200) on the right. Both of these scenes are path-traced and contain severe noise at 4 samples per pixel (spp). However, our trained network is able to estimate the appropriate filter parameters and effectively reduce the noise in only a few seconds. Note, the tonemapping of the insets has been adjusted for best visibility. Scene credits:* KITCHEN *– Jo Ann Elliott;* SAN MIGUEL HALLWAY *– Guillermo M. Leal Llaguno.*

Abstract

The most successful approaches for filtering Monte Carlo noise use feature-based filters (e.g., cross-bilateral and cross non-local means filters) that exploit additional scene features such as world positions and shading normals. However, their main challenge is finding the optimal weights for each feature in the filter to reduce noise but preserve scene detail. In this paper, we observe there is a complex relationship between the noisy scene data and the ideal filter parameters, and propose to *learn* this relationship using a nonlinear regression model. To do this, we use a multilayer perceptron neural network and combine it with a matching filter during both training and testing. To use our framework, we first train it in an offline process on a set of noisy images of scenes with a variety of distributed effects. Then at run-time, the trained network can be used to drive the filter parameters for new scenes to produce filtered images that approximate the ground truth. We demonstrate that our trained network can generate filtered images in only a few seconds that are superior to previous approaches on a wide range of distributed effects such as depth of field, motion blur, area lighting, glossy reflections, and global illumination.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

Keywords: Monte Carlo rendering, neural networks

1 Introduction

Producing photorealistic images from a scene model requires computing a complex multidimensional integral of the scene function at every pixel of the image. For example, generating effects like depth of field and motion blur requires integrating over domains such as lens position and time. Monte Carlo (MC) rendering systems approximate this integral by tracing light rays (samples) in the multidimensional space to evaluate the scene function. Although an approximation to this integral can be quickly evaluated with just a few samples, the inaccuracy of this estimate relative to the true value appears as unacceptable noise in the resulting image. Since the variance of the MC estimator decreases linearly with the number of samples, many samples are required to get a reliable estimate of the integral. The high cost of computing additional rays results in lengthy render times that negatively affect the applicability of MC renderers in modern film production.

One way to mitigate this problem is to quickly render a noisy image with a few samples and then filter it as a post-process to generate an acceptable, noise-free result. This approach has been the subject of extensive research in recent years [Dammertz et al. 2010; Bauszat et al. 2011; Rousselle et al. 2012; Sen and Darabi 2012; Li et al. 2012; Rousselle et al. 2013; Moon et al. 2014]. The more successful methods typically use feature-based filters (e.g., cross-bilateral or cross non-local means filters) to leverage additional scene features such as world positions that help guide the filtering process. Since these features are highly correlated with scene detail, using them in the filtering process greatly improves the quality of the results.

Some approaches have used this information to handle specific distributed effects such as global illumination [Dammertz et al. 2010; Bauszat et al. 2011] and depth of field [Chen et al. 2011]. However, the major challenge is how to exploit this additional information to denoise *general* distributed effects, which requires setting the filter weights for all features (called *filter parameters* hereafter) so that noise is removed while scene detail is preserved. To do this, Sen and Darabi [2011; 2012] proposed to use the functional dependencies between scene features and random parameters calculated us-

A Machine Learning Approach…

Start with a filter like the ones we saw earlier

denoised color

average of color samples

$$\hat{\mathbf{c}}_i = \frac{\sum_{j \in \mathcal{N}(i)} d_{i,j} \bar{\mathbf{c}}_j}{\sum_{j \in \mathcal{N}(i)} d_{i,j}},$$

weight for pixel j contributing to pixel i

difference in pixel positions

difference in pixel colors

$$d_{i,j} = \exp\left[-\frac{\|\bar{\mathbf{p}}_i - \bar{\mathbf{p}}_j\|^2}{2\alpha_i^2}\right] \times \exp\left[-\frac{D(\bar{\mathbf{c}}_i, \bar{\mathbf{c}}_j)}{2\beta_i^2}\right] \times \prod_{k=1}^{K} \exp\left[-\frac{D_k(\bar{\mathbf{f}}_{i,k}, \bar{\mathbf{f}}_{j,k})}{2\gamma_{k,i}^2}\right],$$

difference in value of feature k

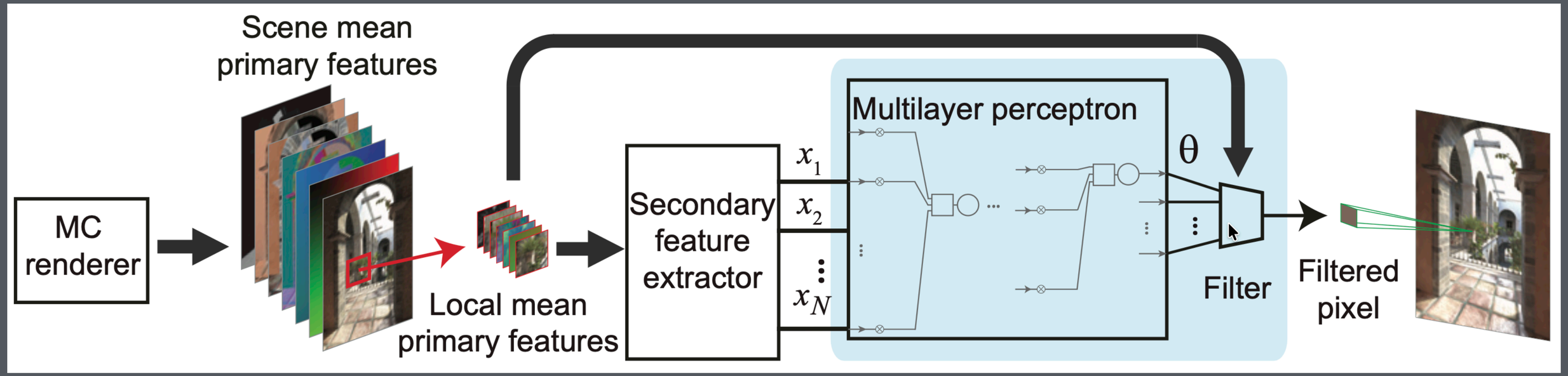
- comes with parameters α, β, γ_1, …, γ_K, which can be different for each i

Learnable model for optimal weights

Start with same features as before (color, normal, position, …)

Fixed calculations compute secondary features

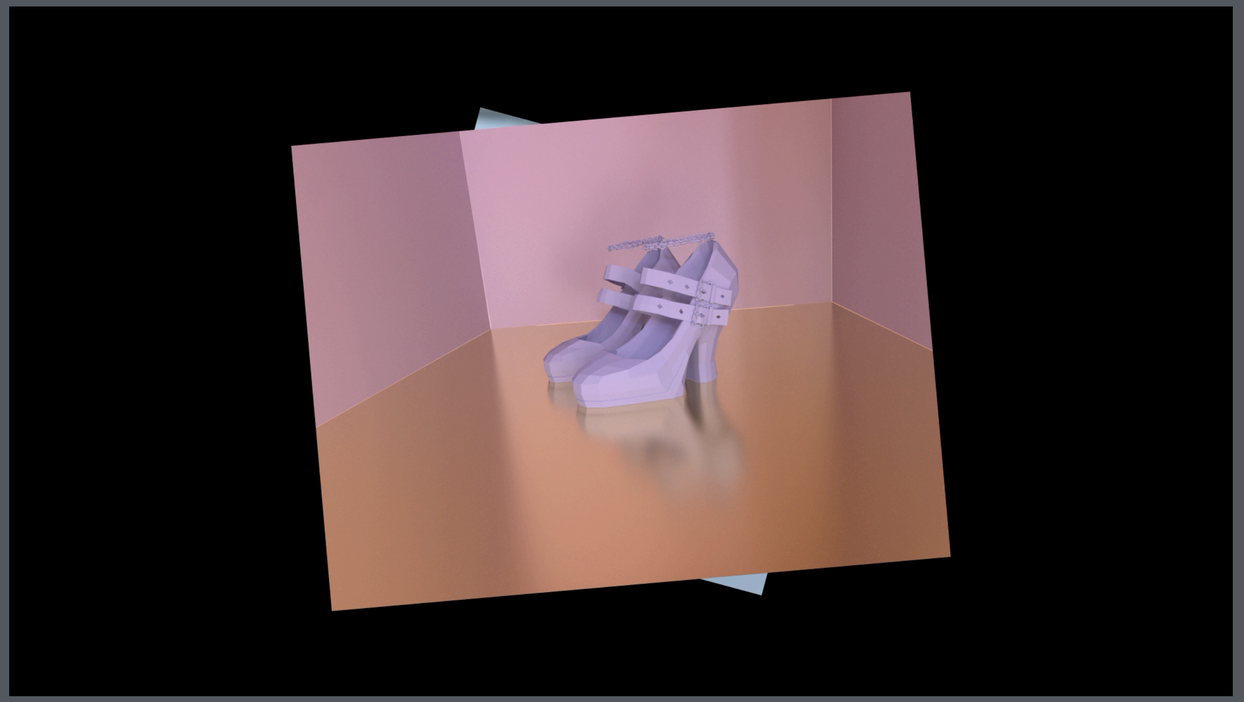
- {mean, variance} of {samples of pixel, pixels of patch}; feature gradients; other stats
- one-hidden-layer MLP maps 36 features to 6 weights
- NLM filter uses those weights to compute filtered pixel from primary features

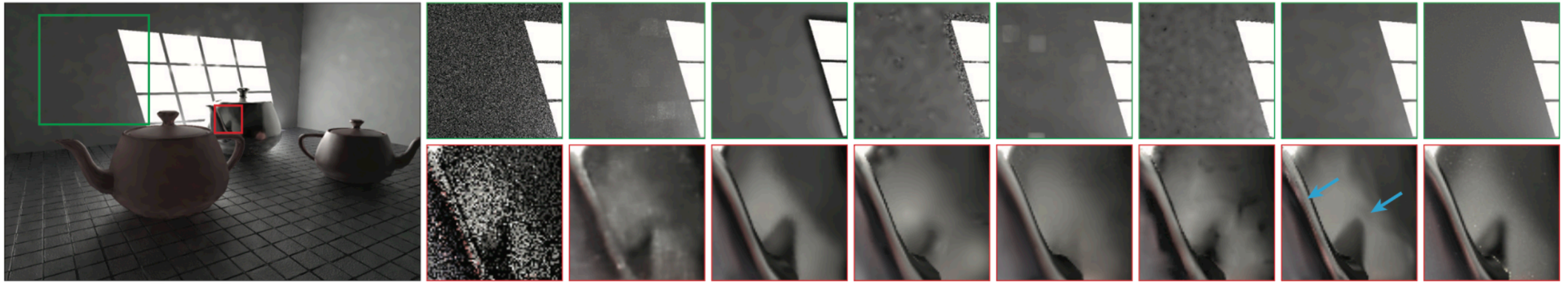


Training

New dataset of (noisy, clean) image pairs

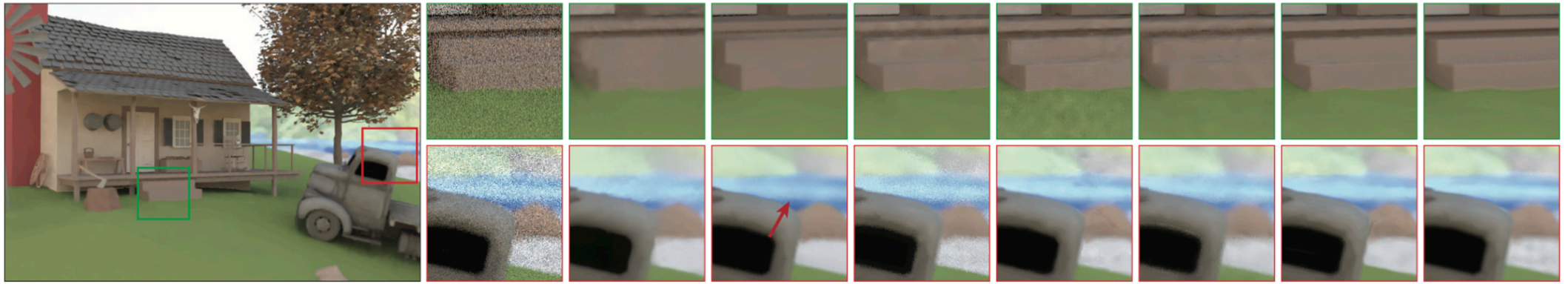
- moderate sized data — 20 scenes, 500 images
- noisy images fed into model
- loss is based on (model output image) – (clean reference image)





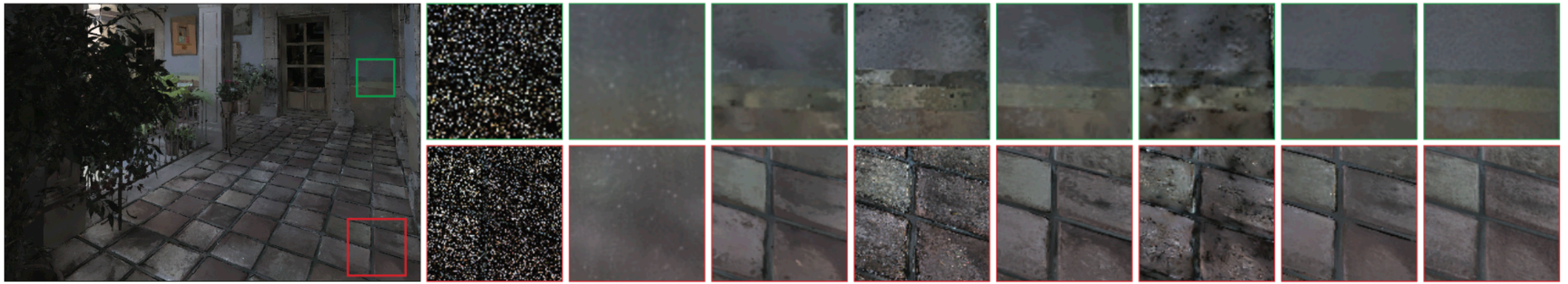
TEAPOT ROOM

	16 spp (27.3 s)	16 spp (37.3 s)	16 spp (1785.1 s)	16 spp (83.8 s)	16 spp (62.8 s)	16 spp (40.4 s)	16 spp (36 s)	96K spp
RelMSE:	1011.22×10^{-2}	38.47×10^{-2}	3.30×10^{-2}	15.78×10^{-2}	2.11×10^{-2}	2.99×10^{-2}	1.92×10^{-2}	
SSIM:	0.404	0.817	0.894	0.876	0.935	0.862	0.933	



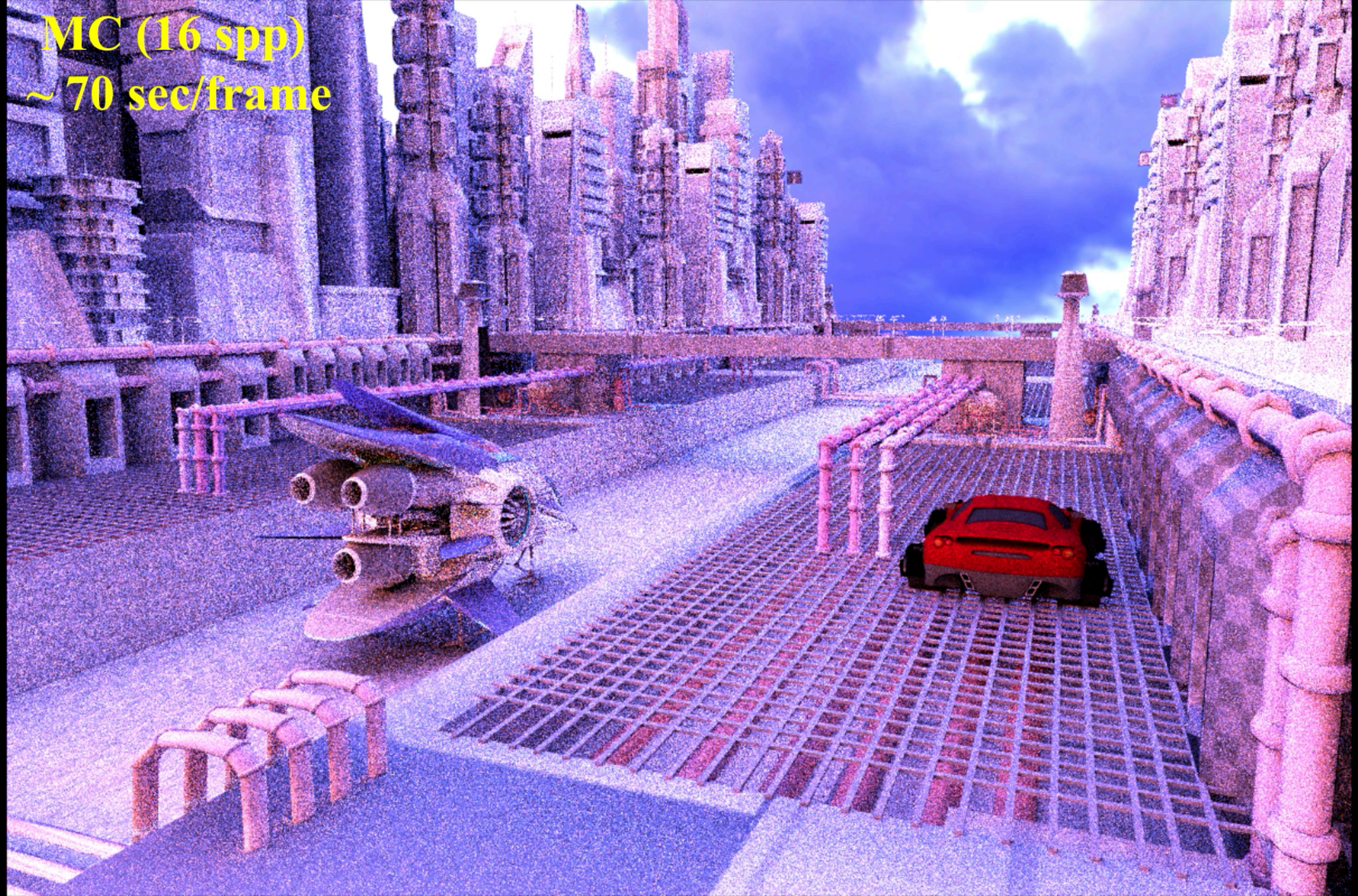
CABIN

	8 spp (33.7 s)	8 spp (48.7 s)	8 spp (818.4 s)	8 spp (137.9 s)	8 spp (71.9 s)	8 spp (51.6 s)	8 spp (41.9 s)	2K spp
RelMSE:	11.84×10^{-2}	1.75×10^{-2}	2.40×10^{-2}	1.28×10^{-2}	1.01×10^{-2}	1.55×10^{-2}	1.18×10^{-2}	
SSIM:	0.806	0.958	0.913	0.965	0.960	0.955	0.969	



SAN MIGUEL BALCONY

	8 spp (125.3 s)	8 spp (147.6 s)	8 spp (815.5 s)	8 spp (299.3 s)	8 spp (161.4 s)	8 spp (154.2 s)	8 spp (133.7 s)	64K spp
RelMSE:	121.2×10^{-2}	3.07×10^{-2}	1.72×10^{-2}	8.16×10^{-2}	1.61×10^{-2}	4.05×10^{-2}	1.92×10^{-2}	
SSIM:	0.458	0.754	0.892	0.841	0.883	0.781	0.899	


MC (16 spp)
~ 70 sec/frame

Kernel-Predicting Convolutional Networks

Observes performance of previous learned method is limited by fixed-function NLM filter

Proposes model that outputs filter weights directly

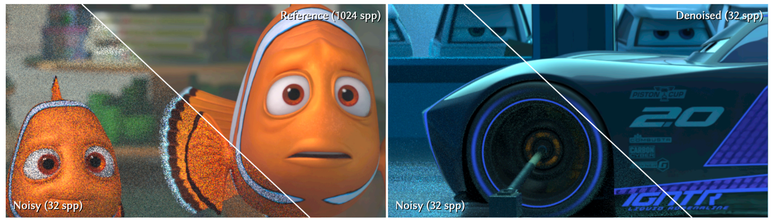
- requires a larger neural network

Larger training dataset

- 600 frames from *Finding Dory*
- tested on other films, good generalization

Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings

STEVE BAKO*, University of California, Santa Barbara
THIJS VOGELS*, ETH Zürich & Disney Research
BRIAN MCWILLIAMS, Disney Research
MARK MEYER, Pixar Animation Studios
JAN NOVÁK, Disney Research
ALEX HARVILL, Pixar Animation Studios
PRADEEP SEN, University of California, Santa Barbara
TONY DEROSE, Pixar Animation Studios
FABRICE ROUSSELLE, Disney Research



TRAINING TEST

Fig. 1. We introduce a deep learning approach for denoising Monte Carlo-rendered images that produces high-quality results suitable for production. We train a convolutional neural network to learn the complex relationship between noisy and reference data across a large set of frames with varying distributed effects from the film *Finding Dory* (left). The trained network can then be applied to denoise new images from other films with significantly different style and content, such as *Cars 3* (right), with production-quality results.

Regression-based algorithms have shown to be good at denoising Monte Carlo (MC) renderings by leveraging its inexpensive by-products (e.g., feature buffers). However, when using higher-order models to handle complex cases, these techniques often overfit to noise in the input. For this reason, supervised learning methods have been proposed that train on a large collection of reference examples, but they use explicit filters that limit their denoising ability. To address these problems, we propose a novel, supervised learning approach that allows the filtering kernel to be more complex and general by leveraging a deep convolutional neural network (CNN) architecture. In one embodiment of our framework, the CNN directly predicts the final denoised pixel value as a highly non-linear combination of the input features. In a second approach, we introduce a novel, kernel-prediction network which uses the CNN to estimate the local weighting kernels used to compute each denoised pixel from its neighbors. We train and evaluate our networks on production data and observe improvements over state-of-the-art MC denoisers, showing that our methods generalize well to a variety of scenes. We conclude by analyzing various components of our architecture and identify areas of further research in deep learning for MC denoising.

CCS Concepts: • **Computing methodologies → Computer graphics**; *Rendering*; Ray tracing;

Additional Key Words and Phrases: Monte Carlo rendering, Monte Carlo denoising, global illumination

ACM Reference format:
Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony DeRose, and Fabrice Rousselle. 2017. Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings. *ACM Trans. Graph.* 36, 4, Article 97 (July 2017), 14 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073708

*Joint first authors

1 INTRODUCTION

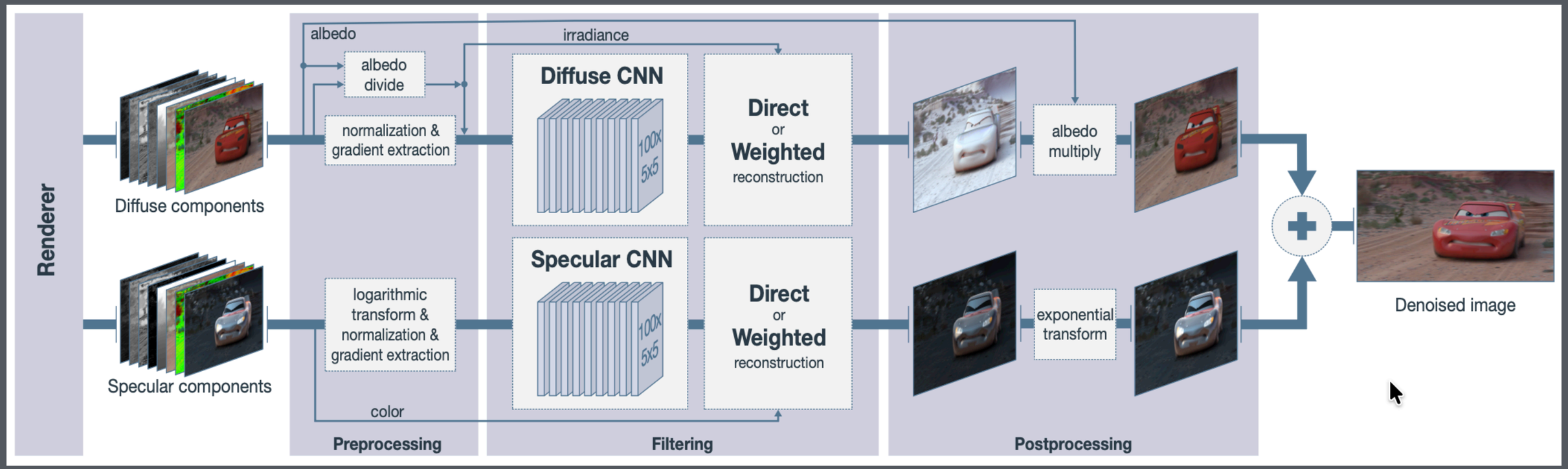
In recent years, physically-based image synthesis has become widespread in feature animation and visual effects [Keller et al. 2015].

Separate models for diffuse and specular components

- diffuse: divide out albedo and multiply back at the end, to better preserve textures
- specular: log-transform pixel colors and exp at the end, to squash dynamic range
- both: normalize and provide image-space gradients as features

Deep convolutional neural nets for each branch

- output is a $k \times k$ grid of weights, normalized using softmax, for positive and normalized weights
- 8 hidden layers, 100 channels, 5x5 convolutions

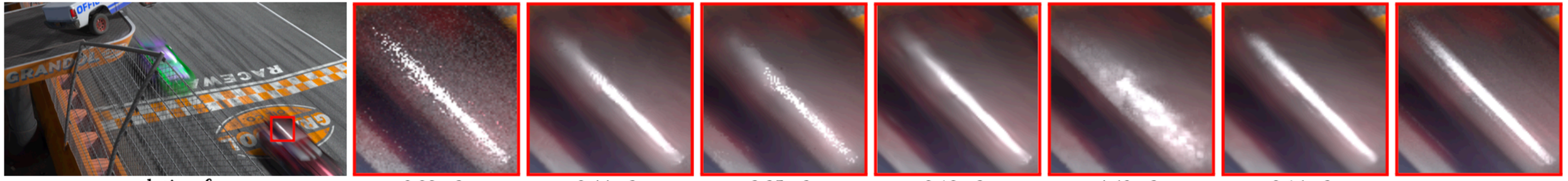


Training

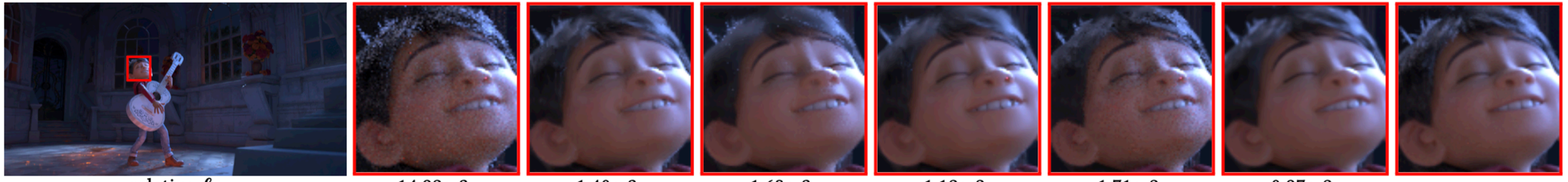
Dataset: 600 representative frames from *Finding Dory*

- refs: 1024spp; inputs: 128spp and 32spp
- features are {RGB mean, gray variance} of {diffuse color, specular color, albedo, depth}
- training is on 65x65 patches, 400 per frame -> 240,000 patches
- loss is L1
- first phase: diffuse, specular trained separately, supervised right at network output
- second phase: fine-tune whole system, supervised on final image

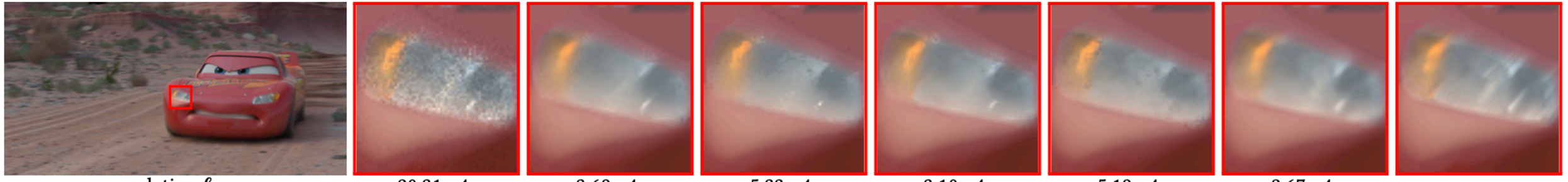
relative ℓ_2	18.88e-3	1.54e-3	1.95e-3	1.24e-3	2.67e-3	0.93e-3
$1 - \text{SSIM}$	0.271	0.026	0.028	0.019	0.038	0.016



relative ℓ_2	9.28e-3	2.44e-3	3.35e-3	2.12e-3	4.69e-3	2.16e-3
$1 - \text{SSIM}$	0.090	0.023	0.030	0.019	0.027	0.019



relative ℓ_2	14.92e-3	1.40e-3	1.68e-3	1.12e-2	1.71e-2	0.97e-2
$1 - \text{SSIM}$	0.360	0.058	0.059	0.046	0.057	0.045



relative ℓ_2	20.31e-4	3.69e-4	5.33e-4	3.10e-4	5.19e-4	2.67e-4
$1 - \text{SSIM}$	0.069	0.011	0.016	0.009	0.015	0.008

...Recurrent Denoising Autoencoder

Targeting real time performance

Uses color samples and aux features

Predicts pixels directly

- a full 128x128 block at a time

Trained on flythroughs of 3 scenes

Inference is pretty fast

- about 15fps at 720p

Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder

CHAKRAVARTY R. ALLA CHAITANYA, NVIDIA, University of Montreal and McGill University
ANTON S. KAPLANYAN, NVIDIA
CHRISTOPH SCHIED, NVIDIA and Karlsruhe Institute of Technology
MARCO SALVI, NVIDIA
AARON LEFOHN, NVIDIA
DEREK NOWROUZEZAHRAI, McGill University
TIMO AILA, NVIDIA

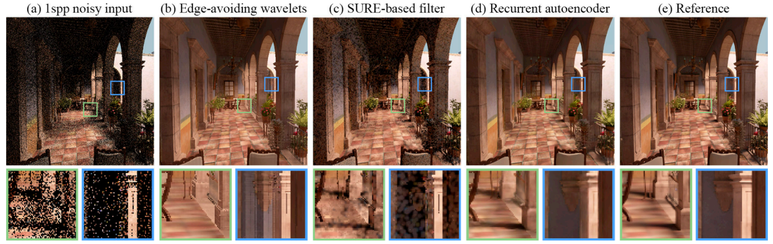


Fig. 1. Left to right: (a) noisy image generated using path-traced global illumination with one indirect inter-reflection and 1 sample/pixel; (b) edge-avoiding wavelet filter [Dammertz et al. 2010] (10.3ms at 720p, SSIM: 0.7737); (c) SURE-based filter [Li et al. 2012] (74.2ms, SSIM: 0.5960); (d) our recurrent denoising autoencoder (54.9ms, SSIM: 0.8438); (e) reference path-traced image with 4096 samples/pixel.

We describe a machine learning technique for reconstructing image sequences rendered using Monte Carlo methods. Our primary focus is on reconstruction of global illumination with extremely low sampling budgets at interactive rates. Motivated by recent advances in image restoration with deep convolutional networks, we propose a variant of these networks better suited to the class of noise present in Monte Carlo rendering. We allow for much larger pixel neighborhoods to be taken into account, while also improving execution speed by an order of magnitude. Our primary contribution is the addition of recurrent connections to the network in order to drastically improve temporal stability for sequences of sparsely sampled input images. Our method also has the desirable property of automatically modeling relationships based on auxiliary per-pixel input channels, such as depth and normals. We show significantly higher quality results compared to existing methods that run at comparable speeds, and furthermore argue a clear path for making our method run at realtime rates in the near future.

CCS Concepts: • **Computing methodologies** → **Ray tracing**; *Neural networks*; *Image processing*;

Additional Key Words and Phrases: Monte Carlo denoising, image reconstruction, interactive global illumination, machine learning

ACM Reference format:
Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder. *ACM Trans. Graph.* 36, 4, Article 98 (July 2017), 12 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073601

1 INTRODUCTION

Ray and path tracing have recently emerged as the rendering algorithms of choice for visual effects [Keller et al. 2015]. This has encouraged the development of filtering and reconstruction techniques to reduce the noise inherent in Monte Carlo renderings [Zwicker et al. 2015], but the focus on film-quality results provides hundreds to thousands of samples per pixel prior to filtering.

Meanwhile, games have also recently migrated towards physically-based shading from more empirical models [Hill et al. 2015], but much of the potential increase in realism from this transition hinges on the possibility of sampling light transport paths more flexibly than rasterization allows. Unfortunately, even the fastest ray tracers can only trace a few rays per pixel at 1080p and 30Hz. While this number doubles every few years, the trend is (at least partially)

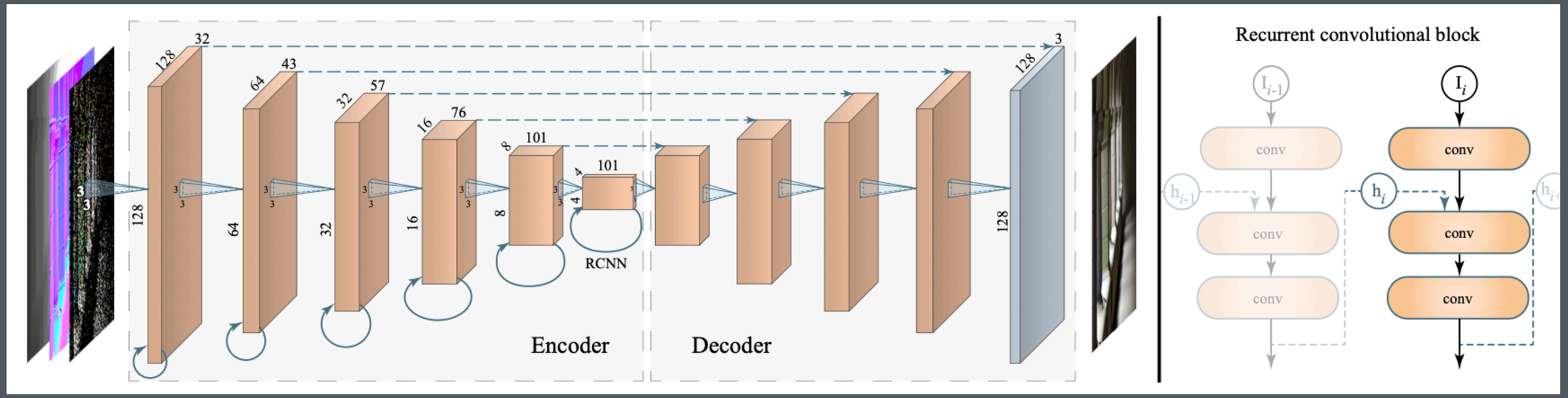
Architecture

U-Net like architecture (encoder-decoder with skip connections)

- 3 convolutions per layer in encoder, 2 in decoder

Recurrent connections from each frame to the next

- enables averaging over time



Training

Data is from 3 scenes

- several flythroughs of each
- crops of 128x128 pixels by 7 frames
- 10 noisy 1-sample images per reference frame

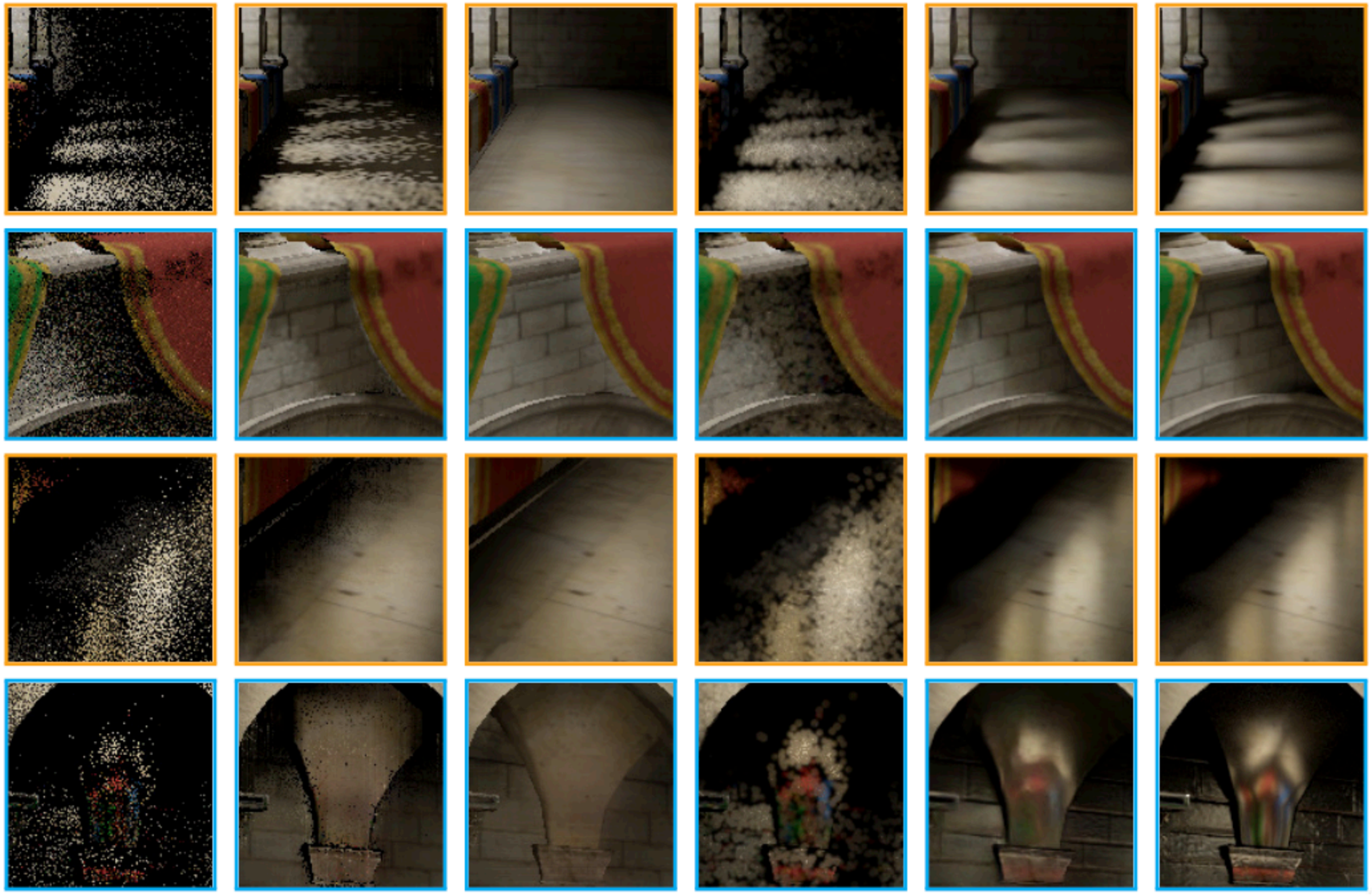
Fig. 6. We train the network using fly-throughs of these 3 scenes.

Loss functions

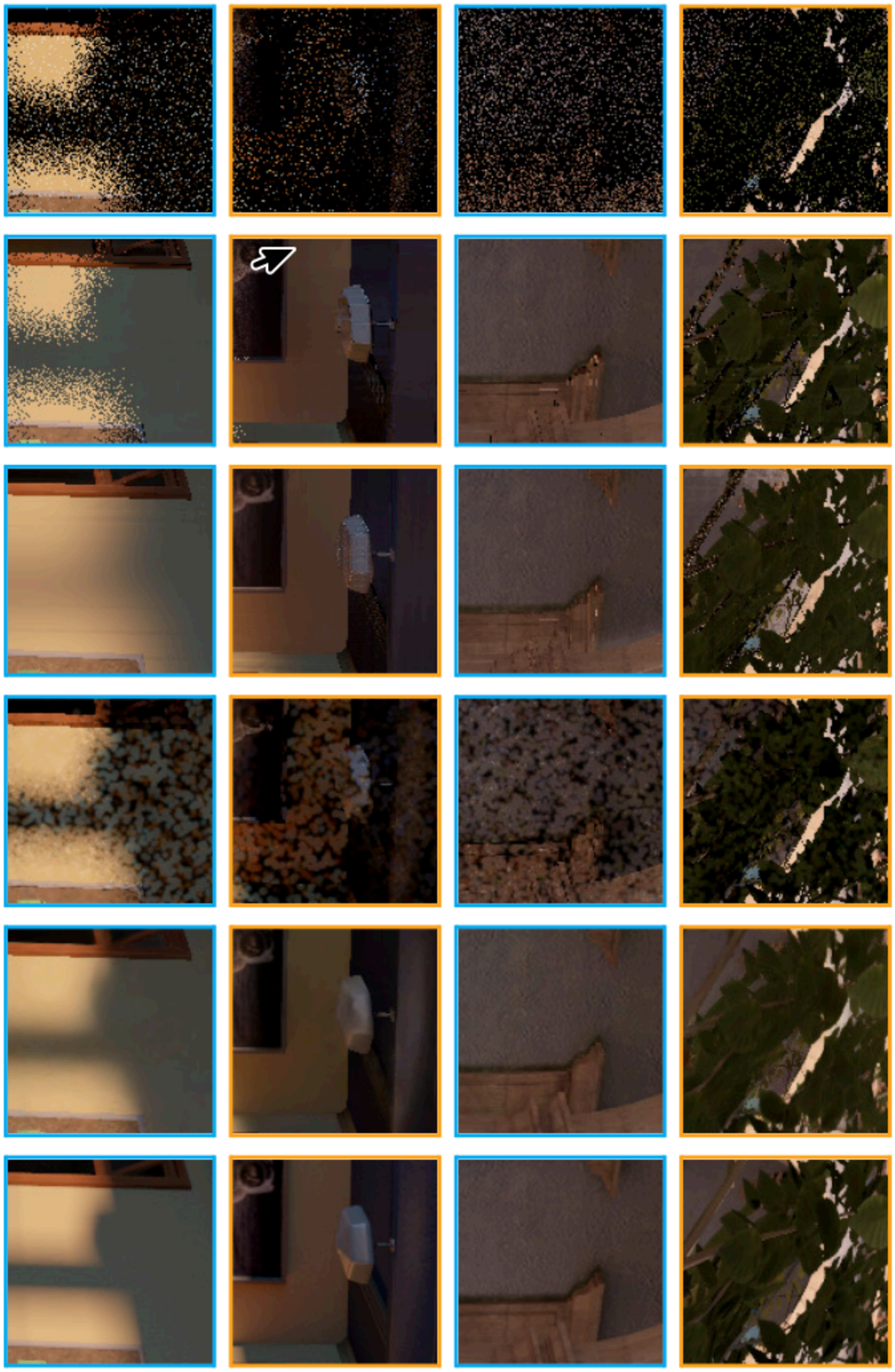
- L1 loss on image pixels
- L1 loss on image-space gradients (with some smoothing)
- L1 loss on temporal gradients (just differences of pixels across frames)
- weights ramp up: higher for later frames to help train RNN blocks

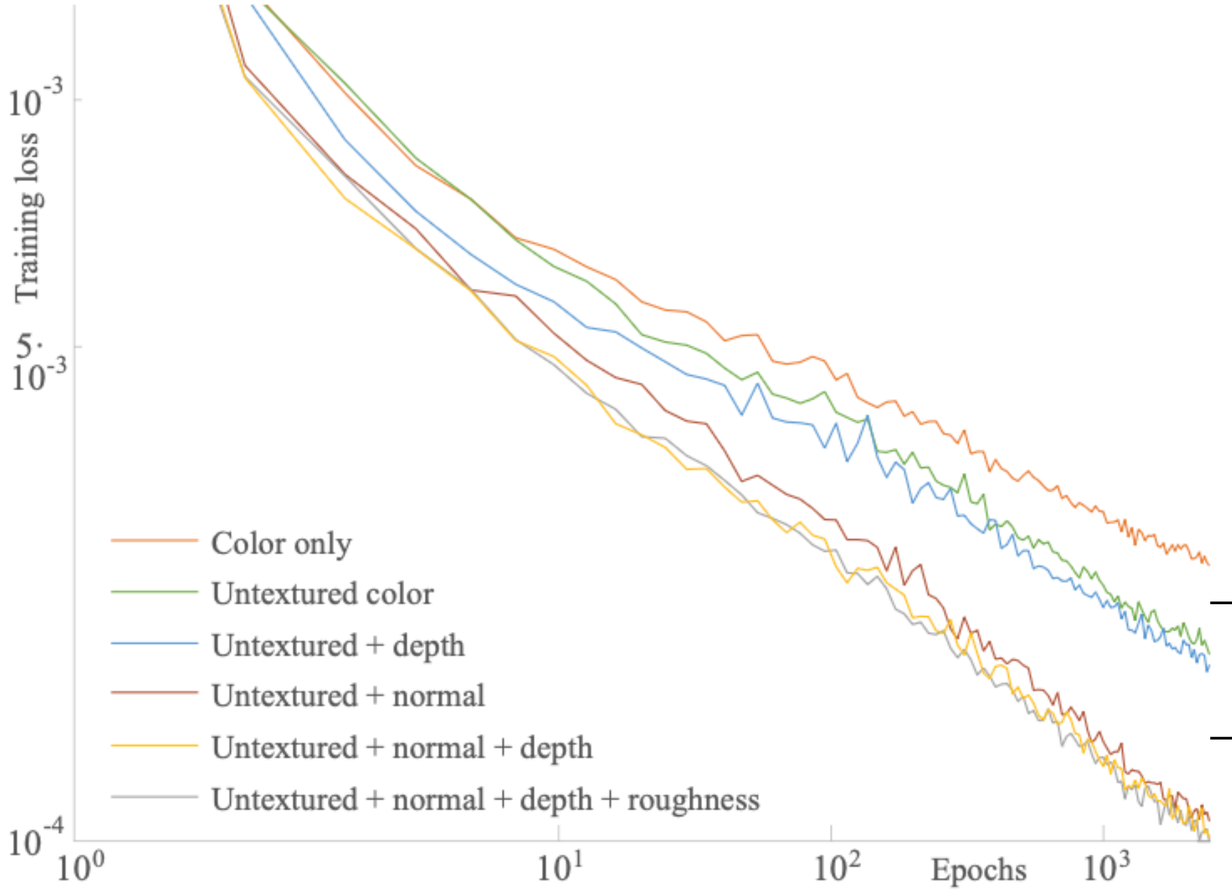
Sponza

GlossySponza



CLASSROOM SANMIGUEL





factoring out albedo helps training

adding normal helps training

Fig. 3. The convergence of average training loss as a function of epochs for our network trained with and without auxiliary features.

...using a Kernel-Splatting Network

Transposes the weight-prediction approach

- change computation from a gather to a scatter

Generates weights per sample

- rather than weights per output pixel
- sample counts might vary in adaptive rendering
- permutation invariance is desired

High quality results

- but slow inference times

Sample-based Monte Carlo Denoising using a Kernel-Splatting Network

MICHAËL GHARBI, Adobe and MIT CSAIL
TZU-MAO LI, MIT CSAIL
MIIKA AITTALA, MIT CSAIL
JAAKKO LEHTINEN, Aalto University and NVIDIA
FRÉDO DURAND, MIT CSAIL

8spp input | [Bako 2017] rMSE = 0.056 (14.6s) | our output rMSE = 0.005 (10s) | ground truth 8192spp

Fig. 1. State-of-the-art pixel-based Monte Carlo denoising algorithms (right) struggle with very noisy inputs rendered with a low sample count (left). Our method (middle) works with the *samples* directly, it uses a *splatting* approach, and is trained using deep learning. This makes it possible to appropriately handle various components of the illumination (indirect lighting, specular reflection, motion blur, depth of field, etc) more effectively.

Denoising has proven to be useful to efficiently generate high-quality Monte Carlo renderings. Traditional pixel-based denoisers exploit summary statistics of a pixel's sample distributions, which discards much of the samples' information and limits their denoising power. On the other hand, sample-based techniques tend to be slow and have difficulties handling general transport scenarios. We present the first convolutional network that can learn to denoise Monte Carlo renderings directly from the samples. Learning the mapping between samples and images creates new challenges for the network architecture design: the order of the samples is arbitrary, and they should be treated in a permutation invariant manner. To address these challenges, we develop a novel kernel-predicting architecture that *splats* individual samples onto nearby pixels. Splatting is a natural solution to situations such as motion blur, depth-of-field and many light transport paths, where it is easier to predict which pixels a sample contributes to, rather than a *gather* approach that needs to figure out, for each pixel, which samples (or nearby pixels) are relevant. Compared to previous state-of-the-art methods, ours is robust to the severe noise of low-sample count images (e.g. 8 samples per pixel) and yields higher-quality results both visually and numerically. Our approach retains the generality and efficiency of pixel-space methods while enjoying the expressiveness and accuracy of the more complex sample-based approaches.

Authors' addresses: Michaël Gharbi, Adobe and MIT CSAIL, mgharbi@adobe.com; Tzu-Mao Li, MIT CSAIL, tzumao@mit.edu; Miika Aittala, MIT CSAIL, miika@mit.edu; Jaakko Lehtinen, Aalto University and NVIDIA, jaakko.lehtinen@aalto.fi; Frédo Durand, MIT CSAIL, fredo@mit.edu.

0730-0301/2019/7-ART125 $15.00
https://doi.org/10.1145/3306346.3322954

CCS Concepts: • **Computing methodologies** → **Neural networks**; **Ray tracing**; *Image processing*.

Additional Key Words and Phrases: Monte Carlo denoising, deep learning, data-driven methods, convolutional neural networks

ACM Reference Format:
Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. 2019. Sample-based Monte Carlo Denoising using a Kernel-Splatting Network. *ACM Trans. Graph.* 38, 4, Article 125 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3322954

1 INTRODUCTION

Because Monte Carlo methods rely on stochastic point samples of an intricate integrand, they often suffer from noise. This motivates Monte Carlo denoising techniques, which broadly fall into two categories. *Sample-based* techniques keep track of the individual samples, while *pixel-based* methods work directly on the rendered image. Most methods operate in pixel space (e.g. [Bako et al. 2017; Bitterli et al. 2016]). In addition to the noisy input radiance, they usually exploit first and second order statistics of auxiliary buffers (like depth, normal, albedo, etc) [McCool 1999]. We argue that, in many lighting configurations, simple per-pixel aggregates can underrepresent the complexity of the local light transport phenomena, in particular because the distribution is often multimodal.

We present a new Monte Carlo denoising technique that leverages the power of deep learning in the following key manners compared to previous denoising methods:

- Rather than work on pixel-based representations, our input is the raw set of Monte Carlo samples, which we argue allows our method to appropriately handle information of different nature. In particular, depth of field and motion blur generate

Gathering vs. scattering

Similar averaging over neighborhood

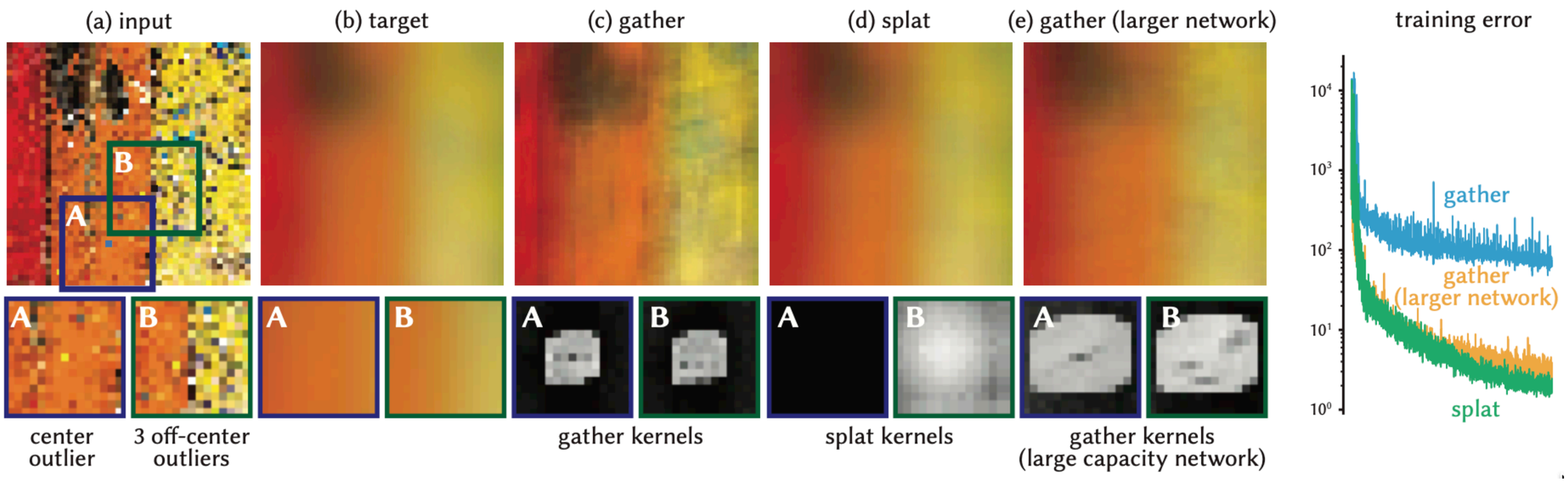
- u,v are output pixel; x,y are input pixel; s is sample index

$$I_{uv} = \frac{\sum_{x,y,s} K_{xyuvs} L_{xys}}{\sum_{x,y,s} K_{xyuvs}},$$

- previous methods thought in terms of $K_{xys|uv}$ — weights to compute pixel u,v from the input
- this method thinks in terms of $K_{uv|xys}$ — weights to contribute sample x,y,s to the output

Advantages of being per-sample

- highly blurred effects like motion blur, out-of-focus blur become easier
- outlier sample can just not splat itself


(a) input
(b) target
(c) gather
(d) splat
(e) gather (larger network)
training error
A
B
center outlier
3 off-center outliers
gather kernels
splat kernels
gather kernels (large capacity network)
gather
gather (larger network)
splat
10^4
10^3
10^2
10^1
10^0

Training

Dataset: random piles of objects

Loss: simply on output image

- L2 loss
- computed on tonemapped images

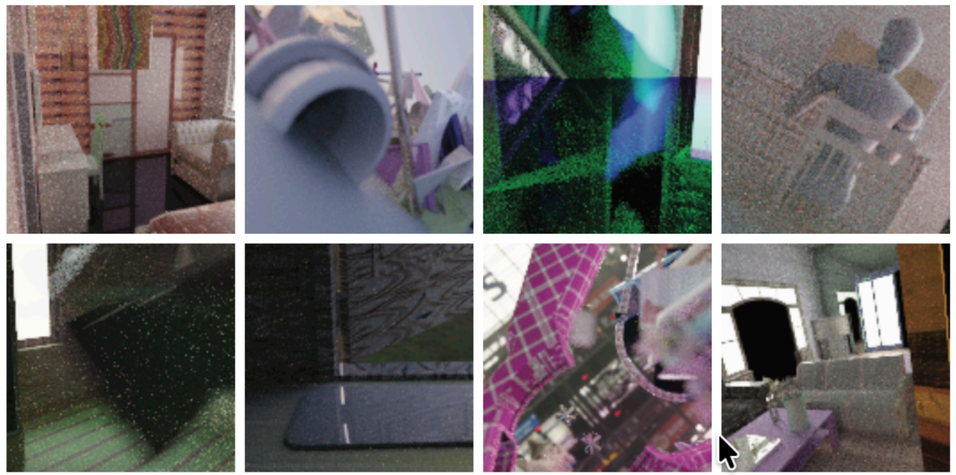
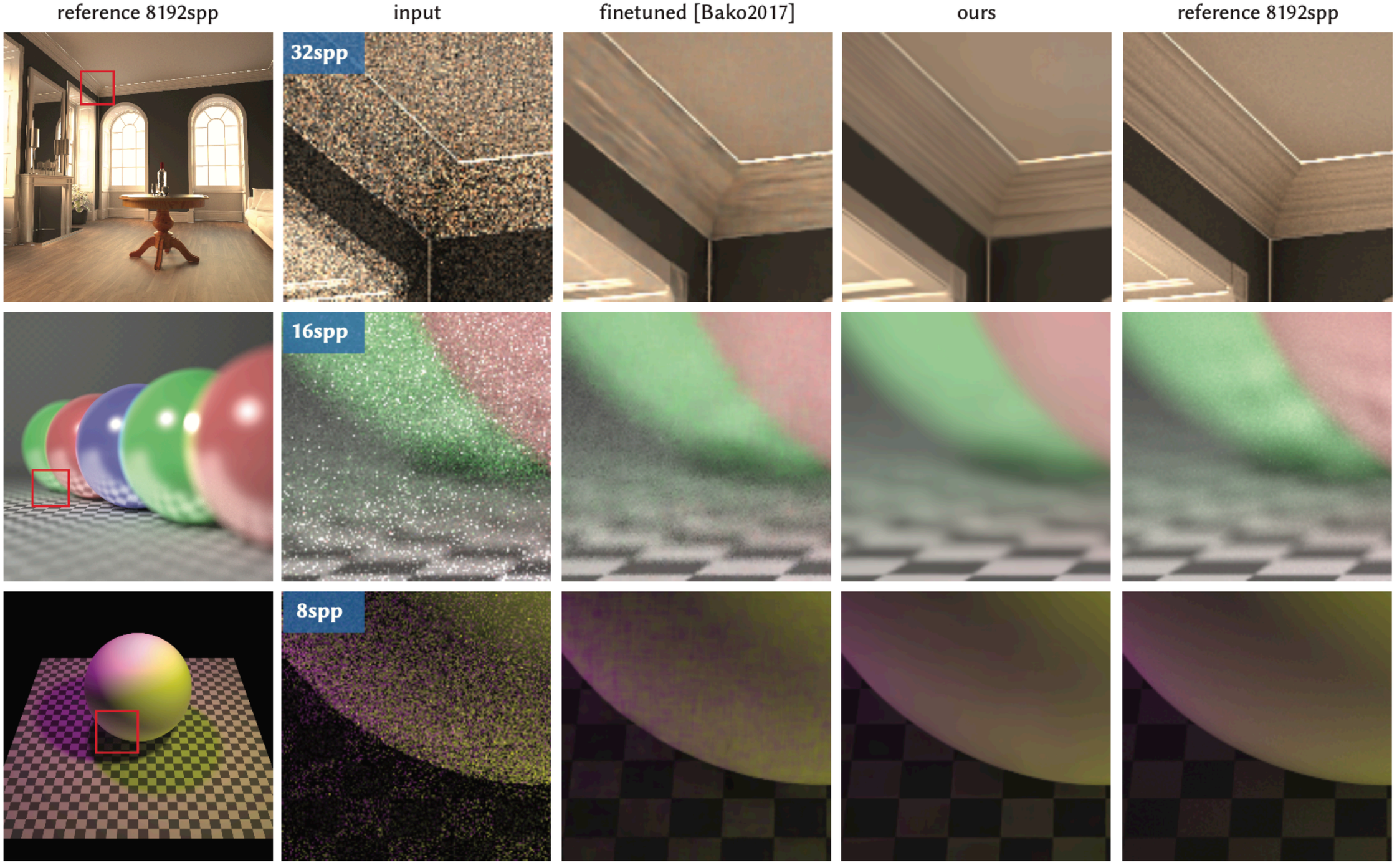
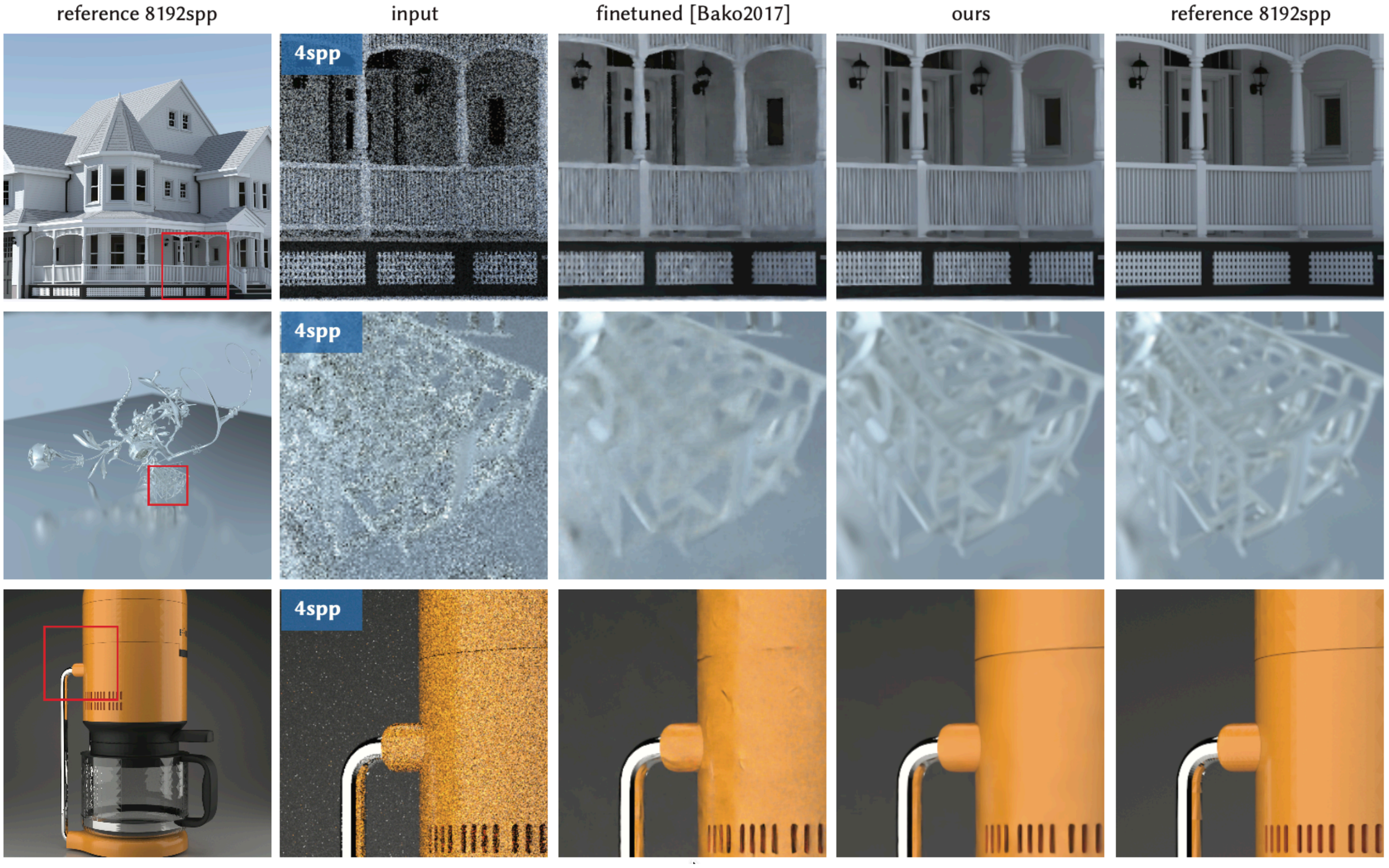


Fig. 7. A selection of the random indoor and outdoor scenes we generate for training.


reference 8192spp
input
finetuned [Bako2017]
ours
reference 8192spp
32spp
16spp
8spp


reference 8192spp
input
finetuned [Bako2017]
ours
reference 8192spp
4spp
4spp
4spp

Architecture

Samples embedded to 128-dim vectors independently

All samples communicate “context” via a UNet operating on per-pixel averages

Samples transformed independently again, using context info

Final independent-sample network predicts splatting kernels