Aalto University

OPEN ACCESS

This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Kettunen, Markus; Harkonen, Erik; Lehtinen, Jaakko

Deep Convolutional Reconstruction For Gradient-Domain Rendering

Published in:
ACM Transactions on Graphics

DOI:
10.1145/3306346.3323038

Published: 01/07/2019

Document Version
Publisher's PDF, also known as Version of record

Please cite the original version:
Kettunen, M., Harkonen, E., & Lehtinen, J. (2019). Deep Convolutional Reconstruction For Gradient-Domain Rendering. *ACM Transactions on Graphics*, *38*(4), Article 126. https://doi.org/10.1145/3306346.3323038

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

Deep Convolutional Reconstruction For Gradient-Domain Rendering

MARKUS KETTUNEN, Aalto University
ERIK HÄRKÖNEN, Aalto University
JAAKKO LEHTINEN, Aalto University and Nvidia

Fig. 1. Comparison of the primal-domain denoisers NFOR [Bitterli et al. 2016] and KPCN [Bako et al. 2017] to our gradient-domain reconstruction NGPT from very noisy equal-time inputs (8 samples for ours and 20 for others). Generally outperforming the comparison methods, our results show that gradient sampling is useful also in the context of non-linear neural image reconstruction, often resolving e.g. shadows better than techniques that do not make use of gradients.

It has been shown that rendering in the gradient domain, i.e., estimating finite difference gradients of image intensity using correlated samples, and combining them with direct estimates of pixel intensities by solving a screened Poisson problem, often offers fundamental benefits over merely sampling pixel intensities. The reasons can be traced to the frequency content of the light transport integrand and its interplay with the gradient operator. However, while they often yield state of the art performance among algorithms that are based on Monte Carlo sampling alone, gradient-domain rendering algorithms have, until now, not generally been competitive with techniques that combine Monte Carlo sampling with post-hoc noise removal using sophisticated non-linear filtering.

Drawing on the power of modern convolutional neural networks, we propose a novel reconstruction method for gradient-domain rendering. Our technique replaces the screened Poisson solver of previous gradient-domain techniques with a novel dense variant of the U-Net autoencoder, additionally taking auxiliary feature buffers as inputs. We optimize our network to minimize a perceptual image distance metric calibrated to the human visual system. Our results significantly improve the quality obtained from gradient-domain path tracing, allowing it to overtake state-of-the-art comparison techniques that denoise traditional Monte Carlo samplings. In particular, we observe that the correlated gradient samples — that offer information about the smoothness of the integrand unavailable in standard Monte Carlo sampling — notably improve image quality compared to an equally powerful neural model that does not make use of gradient samples.

Authors' addresses: Markus Kettunen, Aalto University, markus.kettunen@aalto.fi; Erik Härkönen, Aalto University, erik.harkonen@aalto.fi; Jaakko Lehtinen, Aalto University and Nvidia, jaakko.lehtinen@aalto.fi.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Association for Computing Machinery.
0730-0301/2019/7-ART126 $15.00
https://doi.org/10.1145/3306346.3323038

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

Additional Key Words and Phrases: gradient-domain rendering, gradient-domain reconstruction, screened poisson, ray tracing

ACM Reference Format:
Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. 2019. Deep Convolutional Reconstruction For Gradient-Domain Rendering. *ACM Trans. Graph.* 38, 4, Article 126 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3323038

1 INTRODUCTION

Realistic image synthesis seeks to produce realistic virtual photographs by computationally solving the Rendering Equation [Kajiya 1986], often by randomly sampling paths that carry light from the light sources to the sensor. Rendering with too few samples leaves the image with visually distracting noise. Unsurprisingly, practical applications constantly struggle with striking a balance between the complexity of content (slower, more noise) and available computational resources.

Since many Monte Carlo samples are required for a high quality image, this leaves four main approaches for making rendering faster: (1) making samples faster to evaluate (e.g. GPU rendering, ray tracing hardware, optimized low-level algorithms), (2) sharing contributions between nearby paths (e.g. photon mapping), (3) being clever in choosing the light paths to sample (e.g. Bidirectional Path Tracing, adaptive importance samplers), and, finally, (4) denoising or reconstruction, attempting to produce a better picture out of the samples by relying on various smoothness assumptions or analytic models of the transport phenomena being modeled.

Despite a long history, and continuous research progress in all of these areas, significant problems remain. Only naturally, the quality obtained by "more pure" techniques that rely on few assumptions or heuristics tends to lag behind those that assume more. For instance,

in cases where the diffuse albedo or surface normal — cheap, easy-to-estimate helper variables [McCool 1999] — are good predictors of the smoothness of the final image, making use of this heuristic will, naturally, yield better pictures; however, when this is not the case, but an algorithm assumes it anyway, the reconstruction will be overly smooth and blurry. It is generally difficult to reason about the validity of their assumptions in novel scenes. As a concrete example, the presence of (particularly indirect) shadows cannot be reliably detected from easily-available auxiliary variables, which leads many denoising algorithms to struggle with shadow fidelity.

The recently-introduced gradient-domain family of rendering algorithms is a combination of the final two approaches [Kettunen et al. 2015; Lehtinen et al. 2013; Manzi et al. 2016a]. In addition to estimating pixel intensities, they also compute Monte Carlo estimates of image gradients using correlated samples, and combine these two kinds of samples into the final image by solving a screened Poisson problem. It can be shown that the combined gradient sampling followed by integration exploits the frequency distribution of the transport integrand in a way that often results in higher-quality results within an equal sampling budget [Kettunen et al. 2015; Lehtinen et al. 2013]. Depending on the choice of norm in which the screened Poisson equation is solved, the techniques remain unbiased (L_2) or consistent (L_1), while still exploiting smoothness for better reconstruction, but without relying on heuristics based on auxiliary helper variables.

In this work, we seek to combine the power of the natural smoothness information provided by gradient samples with the high performance of modern denoising techniques that draw on auxiliary features, and aim to do this in a way that adapts to the myriad of different light transport configurations. Determining which input signal to trust to correlate with coherence is a highly context-dependent task. We solve it with a convolutional neural network that is trained to minimize a perceptually-motivated loss function. Our network is a novel hybrid of Densely Connected Convolutional Networks (DenseNet) [Huang et al. 2016] and U-Net [Ronneberger et al. 2015], an autoencoder with skip connections. Our model is relatively simple to implement given an existing gradient-domain path tracer. In particular, we do not require splitting path contributions to separate diffuse and specular channels, as is the case with the Kernel-Predicting Convolutional Network (KPCN) of Bako et al. [2017].

Our results surpass the performance of KPCN and the Nonlinearly Weighted First-order Regression NFOR of Bitterli et al. [2016], two state-of-the-art denoisers, on held-out test scenes. By directed tests, we show that gradient samples clearly improve the results compared to an otherwise equal deep model fed with equal-time primal-only images. Our reconstruction also significantly improves the quality of reconstructed shadows despite working with a much lower sample count in equal time comparisons (Figure 1).

2 RELATED WORK

We will briefly summarize the most important related work in the following. We refer the reader to Zwicker et al. [2015] for a discussion on denoising methods, and to O'Shea and Nash [2015] for an introduction to convolutional neural networks.

2.1 Denoising

The non-local means filter (NL-means) [Buades et al. 2005] denoises pixel colors by averaging over all pixels in an image whose local neighborhoods look similar. Rousselle et al. [2012] extend NL-means to filter Monte Carlo renderings, and use repeated reconstructions to guide sample placement. Rousselle et al. [2013] combine NL-means filtering with feature buffers using Stein's Unbiased Risk Estimate (SURE). Bitterli et al. [2016] evaluate regression weights from NL-means prefiltered feature buffers, but run a first-order weighted regression with only the color data. Moon et al. [2016] run a local polynomial regression with heuristics for selecting the optimal polynomial degree for each pixel.

Kalantari et al. [2015] use a multilayer perceptron to compute optimal filter parameters for cross-bilateral and NL-means filters from primal color and auxiliary feature buffers. The work closest to ours is the Kernel-Predicting Convolutional Network (KPCN) of Bako et al. [2017], a convolutional neural network that predicts a separate smoothing filter kernel for each pixel. They split inputs into diffuse and specular light transport, and process them in separate pipelines. Vogels et al. [2018] extend this work to animations, and build a modular system for supporting multiple samplers with the same denoising network. Chaitanya et al. [2017] design a feature-based convolutional neural network which uses the multi-resolution U-Net network structure as its base, enhanced with recurrent connections for use in animation. Their method is tailored to extremely low sample counts.

2.2 Gradient-Domain Rendering

Lehtinen et al. [2013] introduced gradient-domain rendering by showing that it is beneficial, in the Metropolis Light Transport context, to estimate finite differences between adjacent pixels by sampling and subtracting light paths in highly correlated pairs. They reconstruct the final image from the typically more noisy colors and less noisy gradients by solving a screened Poisson equation. They show that the L_2 solution of the screened Poisson equation is unbiased, but recommend using the slightly biased but consistent L_1 version as it typically produces more visually pleasing results. Intuitively, the high performance of gradient-domain renderers, where applicable, can be explained by examining the screened Poisson problem: its solution shares information spatially over non-trivially large image patches.

Kettunen et al. [2015] adapt Path Tracing to Gradient-Domain Rendering. Subsequently, many other rendering methods have been adapted for the gradient domain: Gradient-Domain [Manzi et al. 2015] Bidirectional Path Tracing [Veach and Guibas 1995], Gradient-Domain [Bauszat et al. 2017] Path Reusing [Bekaert et al. 2002], Gradient-Domain [Hua et al. 2017] Stochastic Progressive Photon Mapping [Hachisuka and Jensen 2009], Gradient-Domain [Sun et al. 2017] Vertex Connection and Merging [Georgiev et al. 2012], and rendering of homogenous volumes with Gradient-Domain adaptations [Gruson et al. 2018] of Beam Radiance Estimates [Jarosz et al. 2008], Progressive Photon Beams [Jarosz et al. 2011] and Photon Planes [Bitterli and Jarosz 2017]. To enforce and exploit temporal coherence, Manzi et al. [2016a] extend gradient computation to the

time domain by mapping random seeds of light paths between successive animation frames, and solve a temporal extension of the screened Poisson equation.

Somewhat related to our work, Manzi et al. [2016b] regularize the screened Poisson reconstruction with the soft constraint that the solution should be expressible locally as a linear combination of the truncated SVD of auxiliary feature buffers. Rousselle et al. [2016] design a new reconstruction method for Gradient-Domain Rendering by formulating the reconstruction with control variates. Back et al. [2018] formulate an ideal feature for local regression based denoising, and show how to approximate it with gradient-domain rendering. This effectively joins gradient-domain rendering with Adaptive Polynomial Rendering [Moon et al. 2016].

Our work is the first to combine gradient-domain sampling with powerful deep models.

2.3 Neural Networks

Convolutional neural networks are powerful parametric non-linear models that can be applied to many image restoration tasks, e.g., denoising, by tuning their performance over a training dataset of noisy images and corresponding noise-free target images in an attempt to match the model's predictions to the clean targets, given noisy inputs.

In particular, the U-Net network architecture [Ronneberger et al. 2015] is a simple and powerful modification of the classical autoencoder, i.e., an encoder-decoder model, that has found widespread use in image restoration [Mao et al. 2016]. Execution first proceeds towards smaller resolutions, processing with a number of convolutional layers at each resolution, before downsampling; after a bottleneck layer, the process is mirrored. U-Net's main innovation is the addition of "skip connections" that link faraway parts of the encoder with the decoder, significantly easing the task of producing pixel-accurate results and facilitating training.

Huang et al. [2016] introduced the DenseNet architecture where all layers get the results of all previous layers as inputs. This facilitates reuse of computation and improves gradient flow, leading to improved accuracy and easier training of deep networks. Previous work in medical imaging has partially combined the DenseNet and U-Net architectures [Guan et al. 2018; Li et al. 2017; Yan et al. 2018]. We present a different, more connected hybrid architecture in the following section, and defer discussion until then.

2.4 Perceptual Losses

Measuring image similarity in a way that corresponds to human judgment is a hard, long-standing problem. Recently, Zhang et al. [2018] formalized the accumulating anecdotal knowledge that the hidden variables of deep convolutional image classifier networks form good perceptual spaces. They introduced the Learned Perceptual Image Patch Similarity (LPIPS), which, given two images, runs them through an image classification network such as VGG [Simonyan and Zisserman 2014] or SqueezeNet [Iandola et al. 2016], and returns a weighted L_2 distance of the network activations. In concurrent work, Kettunen et al. [2019, see supplemental] observe that LPIPS is not robust against optimization, and suggest an ensembled modification (E-LPIPS) that retains the predictive power of LPIPS, but is also a robust optimization target. Their key insight is that applying simple randomized geometry and color transformations to the images makes the feature space much more robust. We make use of E-LPIPS in training our models. Using perceptual losses in image restoration tasks is in itself not a novel approach (e.g. [Johnson et al. 2016]).

3 NEURAL RECONSTRUCTION

Gradient-domain rendering algorithms produce, in addition to a Monte Carlo estimate I_p of the pixel intensities (p for "primal"), estimates of the finite difference pixel gradient $(I_{\mathrm{d}x}, I_{\mathrm{d}y})$ using pairs of correlated path samples. After sampling, the final image is reconstructed by solving

$$\underset{I}{\operatorname{argmin}} \left\| I - I_p \right\| + \alpha \left\| \nabla I - (I_{\mathrm{d}x}, I_{\mathrm{d}y}) \right\|, \tag{1}$$

where ∇ denotes a finite difference gradient operator over the pixels and α is a parameter that can be derived from the relative variances of primal and gradient samples. Solving the problem in the L_2 norm results in an unbiased image, but the outlier-suppressing L_1 norm is preferred in practice. Both norms have efficient solvers.

To incorporate auxiliary feature information, e.g., depth, normal, albedo, without designing (potentially complex) smoothness priors to weight the terms in the equations by hand, and also to support optimizing the result in terms of complex, non-linear perceptual image distance metrics, we take a different route and cast the reconstruction into a direct regression problem solved by a convolutional neural network (CNN)

$$I = \mathrm{CNN}(I_p, I_{\mathrm{d}x}, I_{\mathrm{d}y}, F_1, F_2, \ldots; \boldsymbol{\theta}) \tag{2}$$

where the additional inputs $F_1 \ldots$ are auxiliary feature buffers. (Section 3.3 details the precise inputs given to our model.) Following earlier deep image restoration models, the parameters $\boldsymbol{\theta}$ are obtained by minimizing the "empirical risk"

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; \mathbb{E}_i \left\{ \mathcal{L} \left(\mathrm{CNN}(I_p^i, \ldots; \boldsymbol{\theta}) - I^i \right) \right\} \tag{3}$$

over a training set comprising of noisy primal and gradient inputs, as well as the corresponding noise-free targets I^i, using a variant of stochastic gradient descent. The loss function $\mathcal{L}(\cdot)$ measures the difference between predictions and targets. Note that while we lose the analytic tractability of Equation 1, key to prior fast solvers, and are forced to training by ahead-of-time numerical optimization, this also leaves the door open for more complex loss functions that do not admit analytic solutions.

This overall strategy leaves two questions:

(1) the choice of network architecture; and
(2) the choice of loss function.

Both directly affect the characteristics of the results: a good loss will result in pleasing pictures, and the model needs to be powerful enough to find good solutions while remaining trainable. We now treat each in turn.

3.1 Network Architecture

We base our model on the popular U-Net architecture [Ronneberger et al. 2015], an hourglass-shaped deep autoencoder with non-local "skip connections" that link corresponding resolutions in the encoder

and decoder. We further present a novel combination of U-Net with DenseNets [Huang et al. 2016].

The diagram in Figure 2 describes our network structure. Overall, it is similar to U-Net: computation proceeds from high resolutions to low resolutions and back, with skip connections linking corresponding layers. Each resolution consists of chained convolutional layers followed by non-linearities, and the latter half of each resolution receives data from the first half and from the lower resolution. We use a total of four different resolutions, up to 1/8 the size of the original image along both axes. The figure describes the sizes of convolution kernels as well as numbers of feature maps.

Each resolution of the neural network is also a single densely connected block from DenseNet, and each of its layers receives as input not only the previous layer's activations, but a concatenation of, loosely speaking, all previous layers of the same resolution, and also the final results of the lower resolution. While dense blocks have been used in U-Net architectures before, we take this idea to its logical conclusion: each resolution is, in its entirety, a single dense block, with the decoder end getting additional input from the lower resolutions. As detailed in Section 4, this improves results somewhat over the standard U-Net.

3.2 The Loss Function and Dynamic Range

3.2.1 Dynamic range. A key problem in neural network based reconstruction in Monte Carlo rendering is that magnitudes of input values can vary widely; indeed, the dynamic range of the inputs is unbounded. Particularly tail effects such as caustic paths regularly result in extremely bright outliers. This is in stark contrast with the well known behavior that neural networks tend to work best when the range of the inputs is small and uniform. A common strategy (see for example Bako et al. [2017] and Vogels et al. [2018]) is thus to map the network inputs to logarithmic domain by $y = \log(1 + x)$, apply the network in this domain, and invert the mapping at the end of the network by $x = \exp(y) - 1$. We follow this approach.

The immediate challenge in applying this to gradient-domain is that the gradients contain negative numbers, and consequently the logarithmic mapping does not work directly. We observe, however, that $\log(1 + x)$ is very well behaved also near the origin (where $\log(1 + x) \approx x$), and the natural extension for gradients is the odd reflection:

$$y = \operatorname{sgn}(x) \log(1 + |x|), \tag{4}$$

which is a monotonous function with inverse

$$x = \operatorname{sgn}(y) \left(\exp(|y|) - 1\right). \tag{5}$$

This extension not only adds support for negative inputs, but it can also be used to learn negative targets, for example if using pixel filters with negative lobes.

3.2.2 Perceptual loss. While the goal for an efficient rendering method is normally to produce images with as small residual error as possible as judged by a human observer, it is widely known that pixel-wise norms, such as L_1 and L_2, do not correspond to image similarity as perceived by humans. We make use of the human-calibrated LPIPS perceptual image distance metric that compares images in the L_2 sense in the non-linear feature space of pre-trained image classifier CNNs [Zhang et al. 2018]. More precisely, we employ a self-ensembled variant (E-LPIPS) of it [Kettunen et al. 2019, see supplemental] that has been shown to be a more robust optimization target. In the results section, we show comparisons to the L_1 loss.

While the logarithmic decompression after the network yields high-dynamic range outputs as desired, evaluating the perceptual E-LPIPS loss yields one more complication: it expects inputs to be in the range $[0, 1]$. We solve this by applying Reinhard tone mapping [Reinhard et al. 2002] before evaluating the loss:

$$R(x) = \frac{x}{1 + \operatorname{mean}(x)}. \tag{6}$$

E-LPIPS also expects the images to be in non-linear sRGB, so the images need to be gamma corrected with the sRGB conversion formula before evaluation:

$$\operatorname{sRGB}(c) = \begin{cases} 12.92\, c & c \le 0.0031308 \\ 1.055\, c^{\frac{1}{2.4}} - 0.055 & c > 0.0031308. \end{cases} \tag{7}$$

Approximation with e.g. $\operatorname{sRGB}(c) \approx c^{1/2.2}$ is not possible due to the singularity in the derivative (at zero) which would make training unstable.

3.2.3 Regularization by L_1 loss. Human judgment is much less sensitive to error in tone than in shape, and the LPIPS family of losses is somewhat insensitive to brightness and tone. We still want our reconstruction to accurately predict tones, so we complement our training loss with a small L_1 component applied to a Reinhard-tonemapped result image, as well as its gradients. Since gradients may be negative, we use the odd reflection: $\bar{R}(x) = x/(1+\operatorname{mean}(|x|))$. The tone mapping avoids instability that is observed using HDR inputs. Combining all the components, the final loss function is

$$\begin{aligned} \mathcal{L}(x, y) = {} & d_{\text{E-LPIPS}}(\operatorname{sRGB}(R(x)), \operatorname{sRGB}(R(y))) \\ & + \beta \cdot \|R(x) - R(y)\|_1 \\ & + \gamma \cdot \left\|(\bar{R}(\nabla x) - \bar{R}(\nabla y)\right\|_1. \end{aligned} \tag{8}$$

We get good results with $\beta = \gamma = 0.01$.

3.3 Network Details

3.3.1 Type of convolution. We use strided 2×2 convolution pooling and 2×2 convolution transpose unpooling. The input transformation $\log(1 + x)$ is undone at the very end of the network by applying $\exp(x) - 1$ to the final RGB layer.

3.3.2 Nonlinearity. We use leaky ReLUs [Maas 2013] with factor 0.01 after all convolutional layers, except for the final one whose purpose is to produce a three-channel image.

3.3.3 Parameterization and optimization schedule. We use Weight Normalization [Salimans and Kingma 2016] with He initialization [He et al. 2015] for weights, and optimize with Adam [Kingma and Ba 2014] with parameters $\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$. The learning rate is first ramped geometrically up to 0.0005 during the first 10 000 minibatches of 20 images, and it slowly starts to decrease after the first 50 000 minibatches, halving approximately every 11 hours following curve $L(t) = L_0 2^{-t}$.

Since the E-LPIPS loss is stochastic, we evaluate it thrice for each predicted image during training. While this makes each training sample somewhat more expensive, we find that this significantly

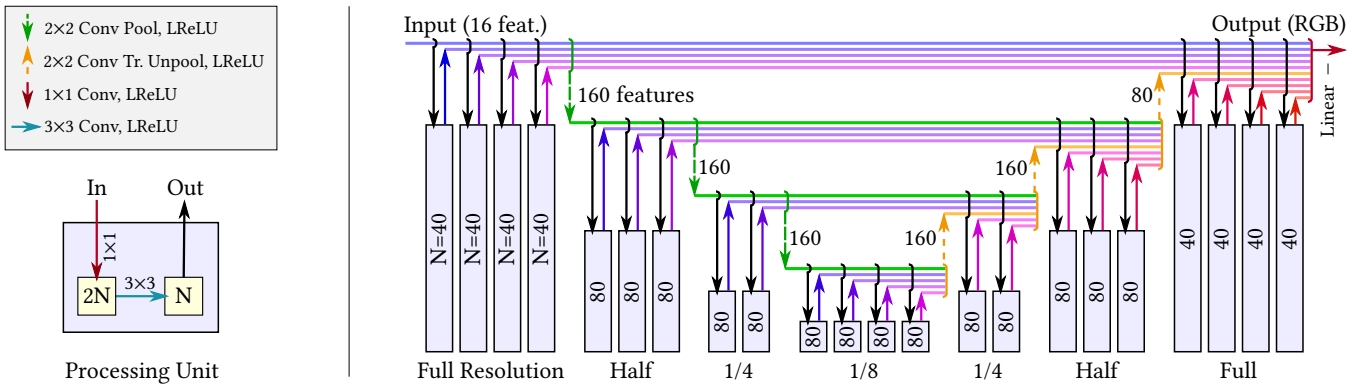

Fig. 2. Network architecture. We process the colors, gradients and other features in multiple resolutions. We repeatedly give the concatenation of all previous results of the current resolution to a processing unit (left) which contracts the large number of features into a smaller number with a 1×1 convolution and applies a 3×3 convolution further contracting the number of features. This result will again be given as input to all future layers of this resolution. At the half-way of each resolution, all previous results of that resolution are concatenated and pooled with 2×2 strided convolution, and the process repeats. At the end of each resolution, we unpool the concatenated results with a 2×2 convolution transpose. Leaky ReLUs are applied after each convolution except the 1×1 convolution at the very end of the network. The final result is passed through $\exp(x) - 1$ to undo the initial $\log(1 + x)$ mapping of the inputs.

reduces wallclock time-to-convergence. This is in line with earlier findings [Kettunen et al. 2019].

The optimization schedule and the other hyperparameters (e.g. sizes and numbers of resolutions) were chosen to approximately minimize the E-LPIPS loss after training for 72 hours. This is orthogonal to using the E-LPIPS loss for training.

3.3.4 Data augmentation. We augment our dataset by randomly flipping, mirroring, and transposing each of the 128×128 input-target crops that we use for training. We also mirror-pad the images randomly such that the total amount of padding in both x and y directions is 16 pixels. This essentially makes the network see differently offset versions of the images. (Note that while it is similar to the internal workings of the E-LPIPS loss, training data augmentation serves a different purpose.)

We also randomly permute the color channels, and sample non-negative random color multipliers from the plane $mean(R, G, B) = 1$. We apply these to the color, gradient and albedo buffers. We also sample a log-normal brightness multiplier which we apply only to the color and gradient buffers.

3.3.5 Input data format. The inputs to our method are the standard color and gradient buffers produced by a gradient-domain renderer, along with albedos, normals and depths. These are the features we find the most useful. We did not find feature variances to produce a significant improvement and dropped them for simplicity. We transform normals to the camera's perspective, and normalize depths to the range $[0, 1]$. Radiance quantities are compressed as detailed in Section 3.2.

During testing, we normalize all depth buffers to range $[0, 1]$. Since our training images consist of small crops, normalizing each of them to $[0, 1]$ would make the network expect a very unrealistic depth distribution. We instead normalize them to range $[0, s]$ where s is sampled uniformly from $[0.1, 1.1]$. Going slightly over 1 during training is a non-issue, but it makes the network encounter values close to one slightly more often.

4 RESULTS AND DISCUSSION

In our evaluation, we seek to study the usefulness of gradient samples in the context of modern denoising techniques, neural or otherwise. For this we choose two state-of-the-art comparison techniques from different algorithm families. The first is the Kernel-Predicting Convolutional Network (KPCN) of Bako et al. [2017], a general-purpose supervised learning technique shown to yield good denoising performance for moderately low sample counts. The second is the NFOR [Bitterli et al. 2016] algorithm that does not make use of machine learning techniques and instead relies on an *a priori* model. Furthermore, we include the L_1 screened Poisson solver used in previous gradient-domain renderers [Lehtinen et al. 2013] as a baseline, although it is working with strictly less information as it does not utilize auxiliary features. We compensate the overhead of gradient sampling for the non-gradient based methods by giving them input buffers with 2.5 times the samples.

4.1 Test Scenes

Our test scenes include the almost diffuse-only Sponza, chosen to establish an easy baseline where all algorithms perform well; the harder Bookshelf, Kitchen, and Cutting Board that all feature glossy and specular transport, with Cutting Board additionally featuring depth-of-field; and the most difficult Bedroom that yields high gradient variance due to transport through transparent curtains.

Bookshelf, Kitchen and Dining Room all test generalization within the convex hull of the training set. The training scenes are all more complicated than the test set's Sponza. In contrast, Cutting Board and Running Man test generalization outside of the convex hull of the training data: depth of field and motion blur effects this strong are very rarely, if ever, present in the training data. The huge gradient variance seen in the curtains of Bedroom is also not present in the training data.

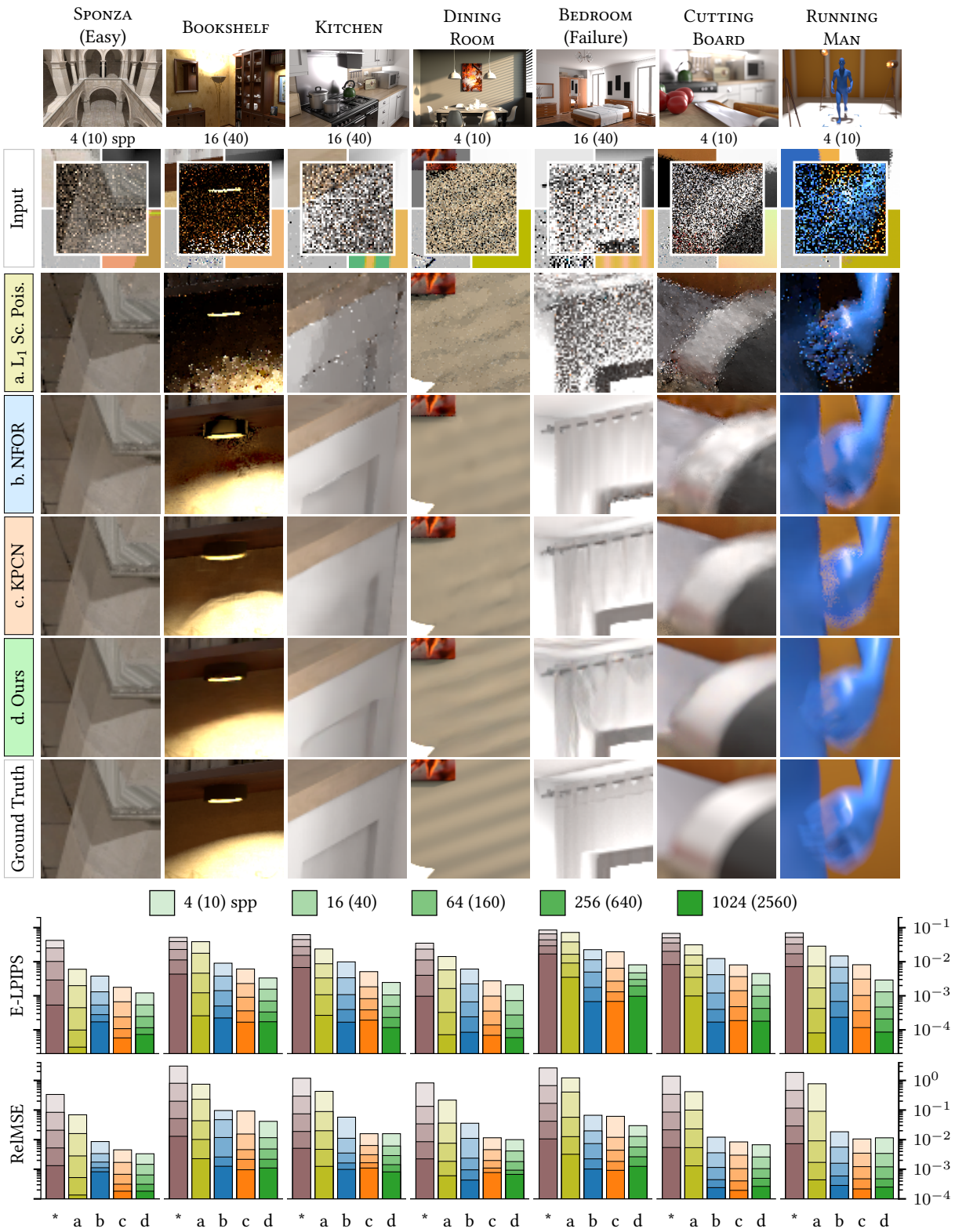

Fig. 3. Equal-time comparisons between Path Tracing (*), the baseline L_1 screened Poisson reconstruction [Lehtinen et al. 2013] (a), a state-of-the-art feature-based denoiser NFOR [Bitterli et al. 2016] (b), kernel-predicting deep convolutional denoising [Bako et al. 2017, KPCN] (c), and the proposed method (d). Path Tracing, NFOR and KPCN are given the sample counts in parentheses.

4.2 Metrics

We measure result quality by two metrics, relative mean square error (RelMSE) [Rousselle et al. 2011] and E-LPIPS [Kettunen et al. 2019]. RelMSE measures squared error of pixels relative to their brightnesses and is often considered a better match to human judgment than L_1 and L_2. To avoid measurement bias for E-LPIPS, we use the VGG version of E-LPIPS for measuring, and train with the SqueezeNet version. We Reinhard tone map our images (Equation 6) for E-LPIPS measurement.

4.3 Overall Performance and Analysis

We first compare our method to L_1 screened Poisson, NFOR and KPCN across a variety of test scenes that were not used in training the deep models. We focus particularly on the low sample regime studied by Bako et al. [2017], where previous gradient-domain rendering algorithms that do not employ feature buffers struggle. KPCN and NFOR are given 2.5 times the samples to equalize the time required for generating the inputs. Figure 3 shows insets of low sample count results and numeric evaluation up to 1024 samples per pixel for the gradient-based methods (ours and L_1 Poisson).

We generally notice our results to be overall of higher subjective quality compared to previous methods, which is also supported by the numerical error measurements.

We also observe steady improvement as the sample count increases (Figure 4). We attribute this to training the model using inputs with many different sample counts and scenes of varying difficulty – the model has to both estimate the noise level and remove it. While no strict consistency guarantees can be made for deep nonlinear models like ours, it would be easy to strictly ensure consistency by simply blending between the reconstruction result and a screened Poisson solution with increasing sample counts.

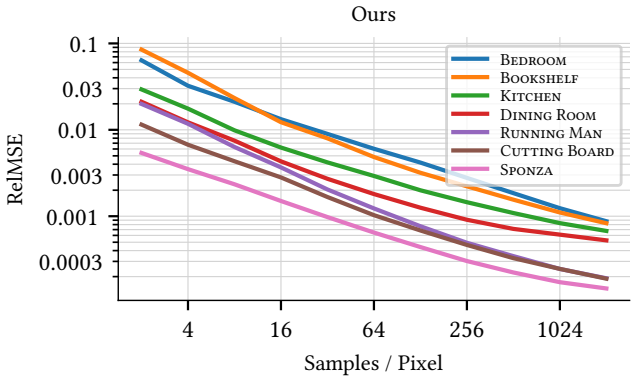

Fig. 4. We study the convergence of our method up to 2048 gradient samples per pixel. Despite our training set mostly targeting the low sample count regime, we observe steady convergence up until very high sample counts, with a slight slowdown in the Dining Room scene after 512 samples per pixel. The network has never seen input images with over 1024 gradient samples.

The supplemental material contains all images in an HTML viewer that facilitates detailed study and comparison.

Sponza features relatively easy light transport, visible as significantly less noisy inputs, and all methods perform relatively well. Our method yields the best results in the low-sample regime, but visible differences fade relatively quickly with increasing samples.

The Bookshelf, Kitchen and Dining Room scenes are more typical scenes with both specular and glossy transport and more geometric detail. Bookshelf and Kitchen also feature many small light sources, and Dining Room is lit through a window with blinds. The shadows and glossy reflections in Dining Room feature interesting complexity. Our method clearly improves over the other methods in these scenes, even with large sample counts.

The Cutting Board scene is a view of Kitchen with a strong depth of field effect. Running Man features an animated character with strong motion blur. This scene looks relatively simple at a first glance, but metallic light stands, lampshades, and much of the light entering the scene through a small hole in the ceiling make it quite challenging. Again, our method comes out on top in these scenes.

In Bedroom, all light enters the scene through large windows covered by translucent curtains which are extremely difficult for the gradient sampler. Indeed, in parts of the image the relative variance of the gradient and primal samples increases to such a level that numerical performance drops below KPCN at higher sample counts, and our dense U-Net trained *without* gradients works overall the best. Interestingly, the error in our model trained with the E-LPIPS loss is visible as hallucinated detail where none should exist. This is also visible in the numerical results, and Figure 5 studies this case in detail. As with previous gradient-domain rendering algorithms, high gradient variance relative to primal samples can be detrimental to result quality. Including more such scenes in the training set may be useful.

Apart from the pathological case in Bedroom, we observe that our method generally improves over KPCN, particularly in the low-sample regime. In some scenes our advantage decreases slightly at higher sample counts. The authors of KPCN originally trained specialized networks for each sample count. We train KPCN and our networks for variable amount of noise, and our training set is concentrated toward smaller sample counts.

Besides our results generally showing fewer artifacts, we also notice a general trend: our method tends to capture shadows more accurately, as seen for example in the scenes Bookshelf, Kitchen and Dining Room. We attribute this to the smoothness information implicitly carried by gradient samples. Figure 6 presents close-up demonstrations. We invite the reader to make use of the HTML viewer for more detailed inspection.

4.4 Ablations and Directed Tests

4.4.1 Are gradients useful? While we have demonstrated best performance overall across the comparison methods, architectural differences between our network and KPCN warrant a question: What is the role of gradient-domain sampling in the results? For this, we train a CNN precisely of the same architecture as ours with the only difference that it does not take gradient inputs; the loss, training procedure, etc., all remain unchanged. The supplemental image viewer contains their images as well, labeled "Ours (w/o grads)". Like KPCN and NFOR, the method is given inputs with 2.5 times

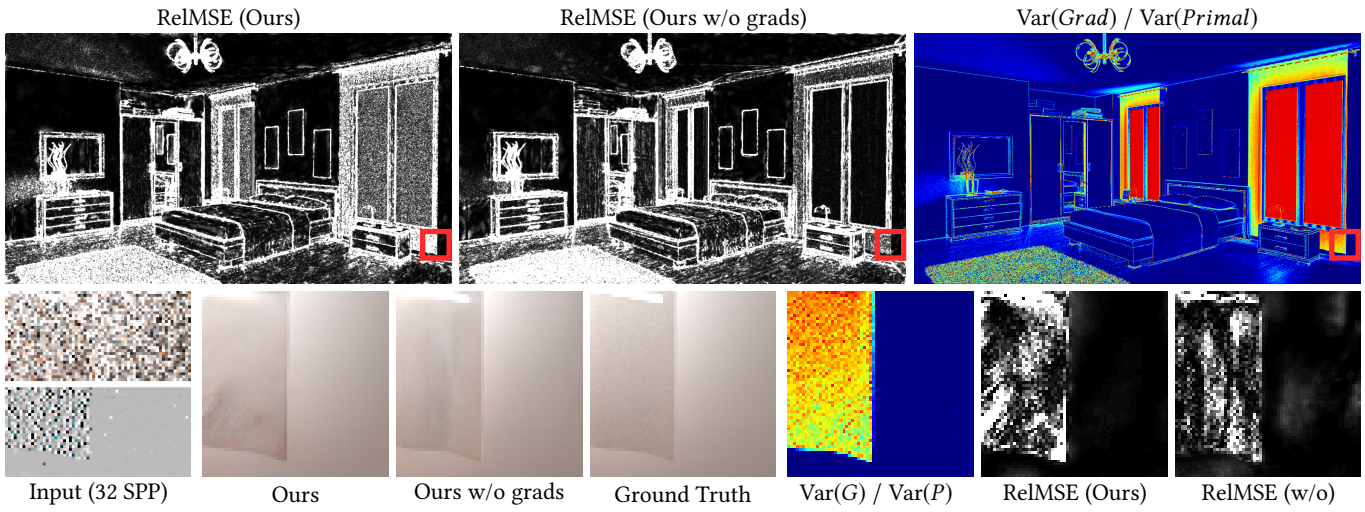

Fig. 5. The top left image shows the RelMSE distribution of our reconstruction in the Bedroom scene which is challenging for our method. Comparing this to our network trained without gradients (top middle), we observe that the areas where gradients are not beneficial correlate highly with areas of high gradient variance relative to primal variance (top right), as originally shown for screened Poisson by Kettunen et al. [2015]. The inset below shows this in practice: The gradient reconstruction "Ours" shows higher error than "Ours w/o grads" where the relative gradient variance is high, and vice versa, as can also be seen in the RelMSE views.

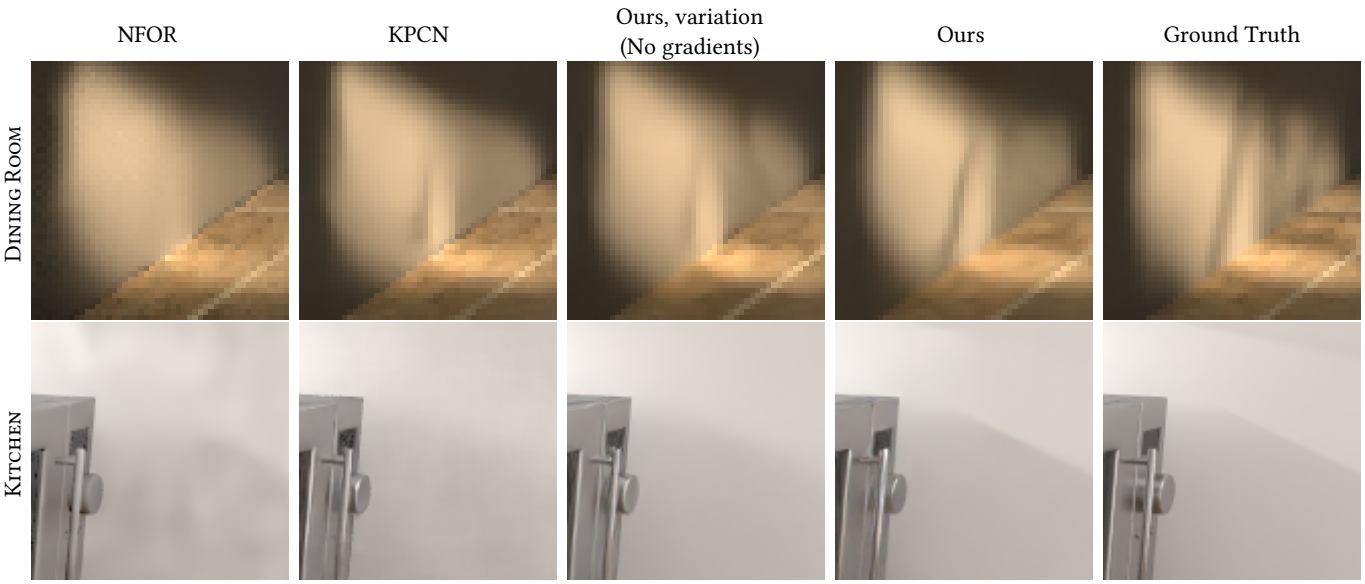

Fig. 6. Equal-time visual comparison of shadows between NFOR, KPCN, our network trained without gradients and our method. Since shadows cannot be well predicted from the traditional feature buffers, gradient-free denoisers (three first columns) tend to have hard time reconstructing them. Shadow edges and penumbra are better captured in gradient samples, and our gradient-domain reconstruction typically reconstructs shadows more accurately.

as many samples to compensate for the reduced sampling cost and maintain equal time.

As demonstrated in Figure 7, the gradient-domain version performs consistently better than the non-gradient version with two exceptions.

The curtains in Bedroom are hard for gradient-domain path tracing so that scene is best rendered without gradients. Cutting Board, on the other hand, is a scene with very strong depth of field, and while our method without gradients improves over our method with gradients, the difference is rather small.

The gradients in the Bedroom scene are noisy due to the gradient shift implementation only partially supporting how the rest of the renderer handles translucency in the curtains, which results in increased noise but not bias (Figure 5). While simple to fix, we use

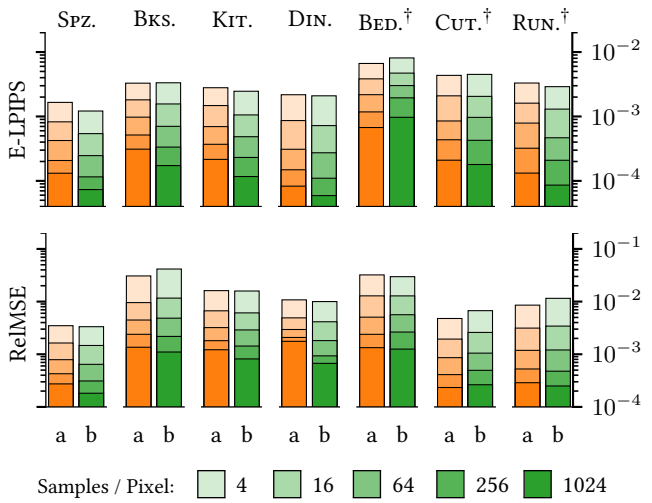

Fig. 7. Our network trained with gradients (b, green) typically improves over our network trained without gradients (a, orange) for equal-time inputs (Sponza — Dining Room, Running Man). †: The Bedroom scene is hard for the gradient sampler while Cutting Board and Running Man contain very strong blur effects. See text in Section 4.4.1.

this to study the effect of high relative variance of gradients which could also result e.g. from an abundance of sub-pixel scale geometric features, or a complex material not obeying the assumptions behind the shift mapping.

Overall, the benefits obtained from sampling gradients in addition to pixel radiance appear to carry over to non-linear neural reconstruction: the smoothness information carried by the finite difference samples most often helps the algorithm in our test cases. Unfortunately, the complex non-linear nature of the network makes analysis in the style of earlier gradient-domain rendering work [Kettunen et al. 2015] difficult.

4.4.2 Is the perceptual loss useful? In light of earlier work on perceptual error metrics [Kettunen et al. 2019], we expect training our network with the E-LPIPS loss instead of the traditional L_1 loss to yield an improvement in perceptual visual quality, most likely at a slight cost to per-pixel color accuracy. We also expect the reconstructions to generally be slightly less blurry and more detailed, as training with L_1 makes the network try to predict pixel-wise medians of potential ground truth images. We train a network with the L_1 loss applied to the Reinhard-tonemapped image (see Section 3.2.3), and find this to generally be the case. Figure 8 shows two representative close-ups: the E-LPIPS loss yields more structurally sound images.

From our test set we observe two exceptions. If the result is actually supposed to be very blurry, for example due to a strong defocus effect, an E-LPIPS trained network might still hallucinate spurious sharp detail, whereas when an L_1 trained network sees an ambiguous situation, it will instead hallucinate blur. Figure 9 shows an example. Careful control of the distribution effects in the training set may alleviate the issues, but this remains future work.

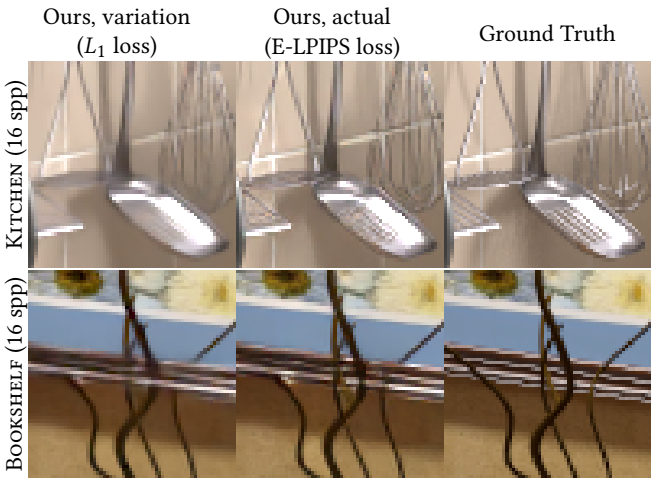

Fig. 8. A network trained with the L_1 loss produces a slightly over-blurred image (left column). As E-LPIPS captures the structure of natural images better, its prediction (middle column) is more natural.

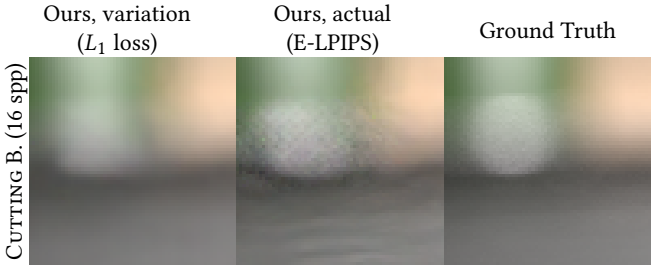

Fig. 9. A scene with very strong depth of field or motion blur effects will have very noisy inputs, yet the output is supposed to be smooth. A network trained with the L_1 loss will naturally resolve the input noise into a strong blur (left) while a network trained with E-LPIPS is more prone to seeing spurious detail in the noise (middle). In this case the L_1 network's over-blurring may be more satisfactory.

We study the above effects numerically in Figure 10 – training with the E-LPIPS loss often produces slightly improved numerical results, but not so when very strong blur effects are present. The improved clarity seems to generalize also to very high sample counts – by visual inspection – even though the numeric results seem to slightly favor the L_1 network in that range. We recommend the reader to judge the high sample count regime with the viewer in the supplemental material.

4.4.3 Dense U-Net. Our network structure is a novel hybrid of DenseNet and U-Net. We find our network to perform consistently better than a traditional U-Net of the same capacity (Figure 11).

4.4.4 Temporal stability. While our algorithm is designed for still images, and orthogonal techniques exist for exploiting temporal coherence in neural denoising and gradient-domain reconstruction [Chaitanya et al. 2017; Manzi et al. 2016a], the temporal behavior of our algorithm is still a relevant question.

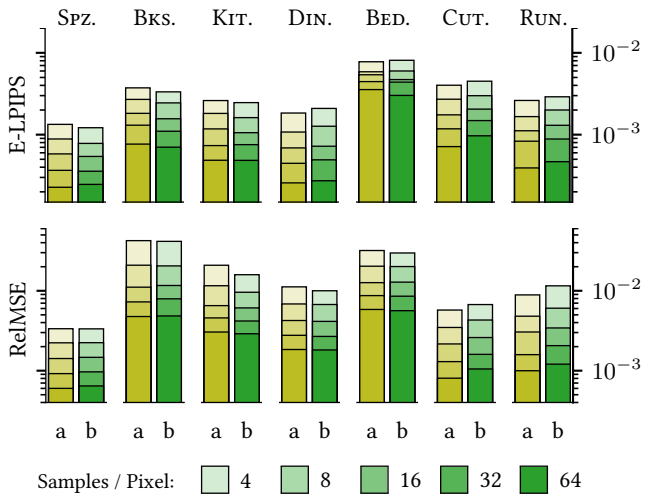

Fig. 10. The improved image structure and clarity gained from training with the E-LPIPS loss (b, green), as opposed to the L_1 loss (a, yellow), is often visible in the numbers in the low–medium sample counts in the typical scenes (Sponza — Dining Room), but training with the L_1 loss often handles blur effects better (Cutting Board, Running Man). Multiply the sample counts by 2.5 to get their Path Tracing equivalents. See the supplemental material for high sample count results.

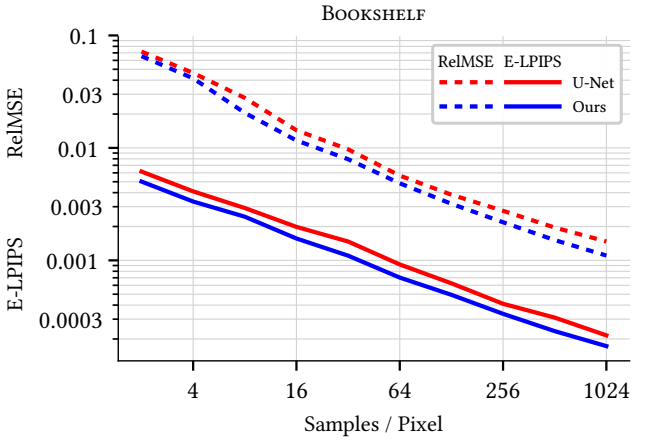

Fig. 11. Our Dense U-Net architecture (blue) consistently improves over the traditional U-Net structure with an equal number of model parameters (red), as indicated here by the perceptual E-LPIPS loss (solid lines) and RelMSE (dashed lines). The results are similar for all of our test scenes.

We conducted a simple study of temporal stability by rendering videos for two sample counts with a static camera, with different random seeds for each frame (Figure 13). The videos are available in the supplemental material. As expected, we observe some high-frequency temporal flicker, but not more than KPCN, and we believe that earlier reprojection techniques can be applied to alleviate these issues.

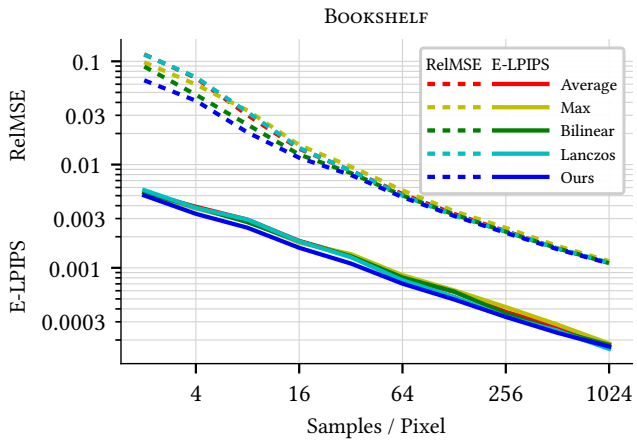

Fig. 12. Our network uses strided 2×2 convolutions for downsampling and 2×2 transposed convolution for upsampling. Other options include for example average or max pooling followed by nearest neighbor upsampling, or using bilinear or Lanczos resampling. We find that convolution pooling followed by convolution transpose unpooling (blue) usually performs best in our case, although only with a small margin. We show the E-LPIPS (solid lines) and RelMSE (dashed lines) results for the Bookshelf scene as an example of the common case.

4.4.5 Choice of pooling. We also empirically validate our decision of using strided 2×2 convolution pooling and corresponding convolution transpose unpooling. See Figure 12.

4.5 Training Set and Generalization

The total training set consists of 2500 pieces of 128×128 crops, each in five different sample counts, totaling 12 500. We supplement this training set with basic augmentation. The relationship between training and test scenes is discussed in the beginning of Section 4.

Our training set consists of ten scenes. For each scene we predefine a box guaranteed to be free of occlusions, and sample 250 random camera positions and normally distributed velocity vectors to simulate motion blur. We set the camera to look towards one of several predefined targets, with a randomly chosen aperture size to simulate depth of field. From each of these points of view we sample a crop of size 128×128 pixels, and render it with six different sample counts: one target image with 8K samples, and five input images with sample counts sampled log-uniformly from range $[2, 1024]$, emphasizing the smaller sample counts where the image quality varies faster. The training set is rendered with the Gradient-Domain Path Tracing implementation from Kettunen et al. [2015] built on top of Mitsuba renderer [Jakob 2010].

When still designing our method, we used leave-one-out cross validation for our test results. As we did not find the reconstruction quality to be overly sensitive to the scenes used in training, we now use a fixed training/test split instead.

See the supplemental material for examples from our training set.

4.6 Implementation Details

4.6.1 Software and hardware, training. We implement our algorithm in Tensorflow. We train both our network and KPCN [Bako et al.

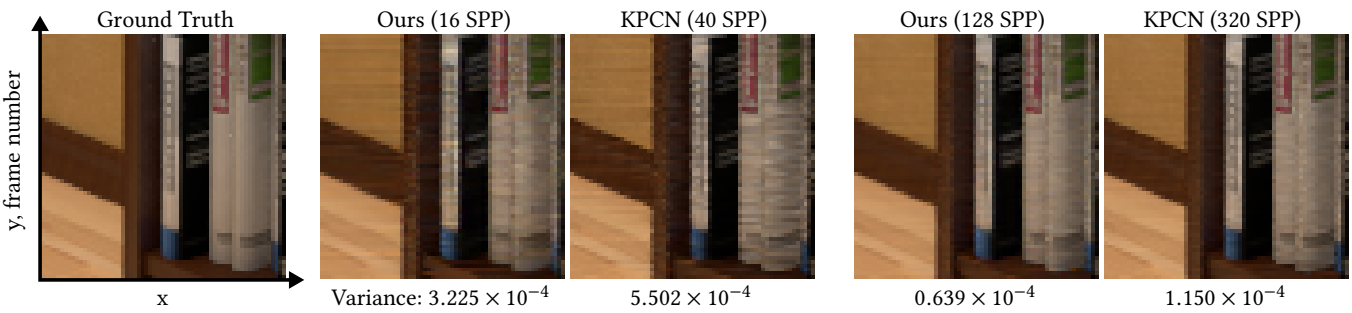

Fig. 13. Equal-time rolling shutter comparison between our method and KPCN at two different rendering times in the Bookshelf scene. The rolling shutter images, constructed by taking every image row from a different animation frame, show that both methods will show some flickering in animation for both rendering times. The sample variances of the pixels over 24 independent renders suggest that our method might not be more temporally unstable than KPCN. See the supplemental material for comparison videos.

2017] on an NVIDIA Tesla V100 GPU using the same training set. We train all networks for three full days which reaches around 12 million shown training examples for our network. After this, we do not observe significant qualitative improvements in results for either model. (See Section 3.3 for the optimization schedule.) Following the authors' original procedure, we pre-train KPCN for one day with separate diffuse and specular targets, and then tune for two more days with the usual non-separated targets.

4.6.2 Running time. All result images are 1280×720 pixels in size. Our network runs in 0.3 seconds per image, while KPCN takes 1.7 seconds per image. The publicly available NFOR implementation processes our images in approx. two minutes on an Intel i7 3770K CPU.

4.6.3 Model size. Our network has 4.4M trainable parameters, whereas KPCN has 5.8M (31% more). Our model size was chosen so that no further improvement was observed. For KPCN we use the default parameters suggested by the authors.

4.6.4 Input data format. Our method, KPCN and NFOR all use the same feature buffers (albedo, depth and normals). KPCN and NFOR additionally require the sample variances for all buffers. NFOR renders the image in two equal-sized sample buckets, while KPCN requires the separation of light transport into diffuse and specular components. Ours method requires neither, but to compensate for the cost of the gradients we give NFOR and KPCN 2.5× as many samples. This equalizes input rendering time between all methods.

5 CONCLUSIONS

We have shown that Gradient-Domain Path Tracing, combined with a powerful deep model replacing the screened Poisson solver of prior work, consistently and visibly improves over the current state of the art in Monte Carlo rendering in equal time. Comparing our reconstruction model with and without gradients, we generally find gradient sampling beneficial, despite their increased cost per-sample. This is supported by both visual inspection and numerical error measurements. This indicates that the smoothness information carried by gradient samples offers benefits also in the non-linear, feature-driven reconstruction domain.

An additional power of deep neural models is the ability to optimize metrics that match human judgment of visual similarity better than traditional per-pixel norms or their simple extensions such as MS-SSIM. We optimize our network to minimize a variant of E-LPIPS by light hyper-parameter search and training, and generally observe improved image quality and less blurring. Unfortunately, the non-linear nature of the E-LPIPS metric means that we cannot employ the recently-proposed "noise2noise" training technique that does not require converged target images [Lehtinen et al. 2018] but can instead be trained on noisy data alone. Seeking ways to overcome this limitation remains interesting future work, as does combining our technique with orthogonal methods for ensuring temporal consistency for rendering animations. Furthemore, we are keen to investigate the potential usefulness of kernel prediction [Bako et al. 2017; Vogels et al. 2018] in gradient-domain reconstruction instead of our "direct prediction" approach.

ACKNOWLEDGMENTS

We thank Sampo Rask, Marco Dabrovic, Blendswap users Jay-Artist, Wig42, SlykDrako, Mareck, nacimus and NovaZeeke for the scenes used in this work, and Benedikt Bitterli and Tiziano Portenier for converting many of them to Mitsuba. This work was created using the computational resources provided by the Aalto Science-IT project, and was partially supported by Academy of Finland (grant 277833).

REFERENCES

Jonghee Back, Sung-Eui Yoon, and Bochang Moon. 2018. Feature Generation for Adaptive Gradient-Domain Path Tracing. *Computer Graphics Forum* (2018). https://doi.org/10.1111/cgf.13548

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 Transactions on Graphics (Proceedings of SIGGRAPH 2017)* 36, 4, Article 97 (2017), 97:1–97:14 pages. https://doi.org/10.1145/3072959.3073708

Pablo Bauszat, Victor Petitjean, and Elmar Eisemann. 2017. Gradient-domain Path Reusing. *ACM Trans. Graph.* 36, 6, Article 229 (Nov. 2017), 9 pages. https://doi.org/10.1145/3130800.3130886

Philippe Bekaert, Mateu Sbert, and John Halton. 2002. Accelerating Path Tracing by Reusing Paths. In *Proceedings of the 13th Eurographics Workshop on Rendering (EGRW '02)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 125–134. http://dl.acm.org/citation.cfm?id=581896.581914

Benedikt Bitterli and Wojciech Jarosz. 2017. Beyond Points and Beams: Higher-Dimensional Photon Samples for Volumetric Light Transport. *ACM Transactions*

on Graphics (Proceedings of SIGGRAPH) 36, 4 (July 2017). https://doi.org/10.1145/3072959.3073698

Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, José A. Iglesias-Guitián, David Adler, Kenny Mitchell, Wojciech Jarosz, and Jan Novák. 2016. Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings. In *Proc. Eurographics Symposium on Rendering (EGSR) 2016.*

Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. 2005. A review of image denoising algorithms, with a new one. *SIAM Journal on Multiscale Modeling and Simulation: A SIAM Interdisciplinary Journal* 4, 2 (2005), 490–530. https://hal.archives-ouvertes.fr/hal-00271141

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. https://doi.org/10.1145/3072959.3073601

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light Transport Simulation with Vertex Connection and Merging. *ACM Trans. Graph.* 31, 6, Article 192 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366211

Adrien Gruson, Binh-Son Hua, Nicolas Vibert, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2018. Gradient-domain Volumetric Photon Density Estimation. *ACM Trans. Graph.* 37, 4, Article 82 (July 2018), 13 pages. https://doi.org/10.1145/3197517.3201363

Steven Guan, Amir A. Khan, Siddhartha Sikdar, and Parag V. Chitnis. 2018. Fully Dense UNet for 2D Sparse Photoacoustic Tomography Artifact Removal. *CoRR* abs/1808.10848 (2018). arXiv:1808.10848 http://arxiv.org/abs/1808.10848

Toshiya Hachisuka and Henrik Wann Jensen. 2009. Stochastic Progressive Photon Mapping. In *ACM SIGGRAPH Asia 2009 Papers (SIGGRAPH Asia '09)*. ACM, New York, NY, USA, Article 141, 8 pages. https://doi.org/10.1145/1661412.1618487

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *CoRR* abs/1502.01852 (2015). arXiv:1502.01852 http://arxiv.org/abs/1502.01852

Binh-Son Hua, Adrien Gruson, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2017. Gradient-Domain Photon Density Estimation. *Comput. Graph. Forum* 36, 2 (May 2017), 31–38. https://doi.org/10.1111/cgf.13104

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. 2016. Densely Connected Convolutional Networks. *CoRR* abs/1608.06993 (2016). arXiv:1608.06993 http://arxiv.org/abs/1608.06993

Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *CoRR* abs/1602.07360 (2016). arXiv:1602.07360 http://arxiv.org/abs/1602.07360

Wenzel Jakob. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

Wojciech Jarosz, Derek Nowrouzezahrai, Robert Thomas, Peter-Pike Sloan, and Matthias Zwicker. 2011. Progressive Photon Beams. In *Proceedings of the 2011 SIGGRAPH Asia Conference (SA '11)*. ACM, New York, NY, USA, Article 181, 12 pages. https://doi.org/10.1145/2024156.2024215

Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. 2008. The Beam Radiance Estimate for Volumetric Photon Mapping. *Comput. Graph. Forum* 27 (04 2008), 557–566. https://doi.org/10.1111/j.1467-8659.2008.01153.x

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*. Springer, 694–711.

James T. Kajiya. 1986. The Rendering Equation. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986), 143–150. https://doi.org/10.1145/15886.15902

Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. 2015. A Machine Learning Approach for Filtering Monte Carlo Noise. *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2015)* 34, 4 (2015).

Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. 2019. Robust Perceptual Image Similarity via Self-Ensembled CNNs. Manuscript in preparation.

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-domain Path Tracing. *ACM Trans. Graph.* 34, 4, Article 123 (July 2015), 13 pages. https://doi.org/10.1145/2766997

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-domain Metropolis Light Transport. *ACM Trans. Graph.* 32, 4, Article 95 (July 2013), 12 pages. https://doi.org/10.1145/2461912.2461943

Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. 2018. Noise2Noise: Learning Image Restoration without Clean Data. In *Proceedings of the 35th International Conference on Machine Learning, PMLR*, Vol. 80.

Xiaomeng Li, Hao Chen, Xiaojuan Qi, Qi Dou, Chi-Wing Fu, and Pheng-Ann Heng. 2017. H-DenseUNet: Hybrid Densely Connected UNet for Liver and Liver Tumor Segmentation from CT Volumes. *CoRR* abs/1709.07330 (2017). arXiv:1709.07330 http://arxiv.org/abs/1709.07330

Andrew L. Maas. 2013. Rectifier Nonlinearities Improve Neural Network Acoustic Models.

Marco Manzi, Markus Kettunen, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-Domain Bidirectional Path Tracing. In *Proc. Eurographics Symposium on Rendering*.

Marco Manzi, Markus Kettunen, Frédo Durand, Matthias Zwicker, and Jaakko Lehtinen. 2016a. Temporal Gradient-domain Path Tracing. *ACM Trans. Graph.* 35, 6, Article 246 (Nov. 2016), 9 pages. https://doi.org/10.1145/2980179.2980256

Marco Manzi, Delio Vicini, and Matthias Zwicker. 2016b. Regularizing Image Reconstruction for Gradient-Domain Rendering with Feature Patches. In *Computer graphics forum*, Vol. 35. Wiley Online Library, 263–273.

Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. 2016. Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections. In *Proc. NIPS*.

Michael D. McCool. 1999. Anisotropic Diffusion for Monte Carlo Noise Reduction. *ACM Trans. Graph.* 18, 2 (April 1999), 171–194. https://doi.org/10.1145/318009.318015

Bochang Moon, Steven McDonagh, Kenny Mitchell, and Markus Gross. 2016. Adaptive Polynomial Rendering. *ACM Trans. Graph.* 35, 4, Article 40 (July 2016), 10 pages. https://doi.org/10.1145/2897824.2925936

Keiron O'Shea and Ryan Nash. 2015. An Introduction to Convolutional Neural Networks. *CoRR* abs/1511.08458 (2015). arXiv:1511.08458 http://arxiv.org/abs/1511.08458

Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. 2002. Photographic Tone Reproduction for Digital Images. *ACM Trans. Graph.* 21, 3 (July 2002), 267–276. https://doi.org/10.1145/566654.566575

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR* abs/1505.04597 (2015). arXiv:1505.04597 http://arxiv.org/abs/1505.04597

Fabrice Rousselle, Wojciech Jarosz, and Jan Novák. 2016. Image-space Control Variates for Rendering. *ACM Trans. Graph.* 35, 6, Article 169 (Nov. 2016), 12 pages. https://doi.org/10.1145/2980179.2982443

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2011. Adaptive Sampling and Reconstruction Using Greedy Error Minimization. *ACM Trans. Graph.* 30, 6, Article 159 (Dec. 2011), 12 pages. https://doi.org/10.1145/2070781.2024193

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2012. Adaptive Rendering with Non-local Means Filtering. *ACM Trans. Graph.* 31, 6, Article 195 (Nov. 2012), 11 pages. https://doi.org/10.1145/2366145.2366214

Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. 2013. Robust Denoising using Feature and Color Information. *Computer Graphics Forum* 32, 7 (2013), 121–130.

Tim Salimans and Diederik P. Kingma. 2016. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. *CoRR* abs/1602.07868 (2016). arXiv:1602.07868 http://arxiv.org/abs/1602.07868

Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014). arXiv:1409.1556 http://arxiv.org/abs/1409.1556

Weilun Sun, Xin Sun, Nathan A. Carr, Derek Nowrouzezahrai, and Ravi Ramamoorthi. 2017. Gradient-Domain Vertex Connection and Merging. In *Eurographics Symposium on Rendering - Experimental Ideas & Implementations*, Matthias Zwicker and Pedro Sander (Eds.). The Eurographics Association. https://doi.org/10.2312/sre.20171197

Eric Veach and Leonidas J. Guibas. 1995. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. ACM, New York, NY, USA, 419–428. https://doi.org/10.1145/218380.218498

Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. 2018. Denoising with Kernel Prediction and Asymmetric Loss Functions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018)* 37, 4, Article 124 (2018), 124:1–124:15 pages. https://doi.org/10.1145/3197517.3201388

Siming Yan, Feng Shi, Yuhua Chen, Damini Dey, Sang-Eun Lee, Hyuk-Jae Chang, Debiao Li, and Yibin Xie. 2018. Calcium Removal From Cardiac CT Images Using Deep Convolutional Neural Network. *CoRR* abs/1803.00399 (2018). arXiv:1803.00399 http://arxiv.org/abs/1803.00399

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. *CoRR* abs/1801.03924 (2018). arXiv:1801.03924 http://arxiv.org/abs/1801.03924

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and Sungeui E. Yoon. 2015. Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum* (2015). https://doi.org/10.1111/cgf.12592