GradNet: Unsupervised Deep Screened Poisson Reconstruction for Gradient-Domain Rendering

JIE GUO*, State Key Lab for Novel Software Technology, Nanjing University
MENGTIAN LI*, State Key Lab for Novel Software Technology, Nanjing University
QUEWEI LI, State Key Lab for Novel Software Technology, Nanjing University
YUTING QIANG, State Key Lab for Novel Software Technology, Nanjing University
BINGYANG HU, State Key Lab for Novel Software Technology, Nanjing University
YANWEN GUO†, State Key Lab for Novel Software Technology, Nanjing University
LING-QI YAN†, University of California, Santa Barbara

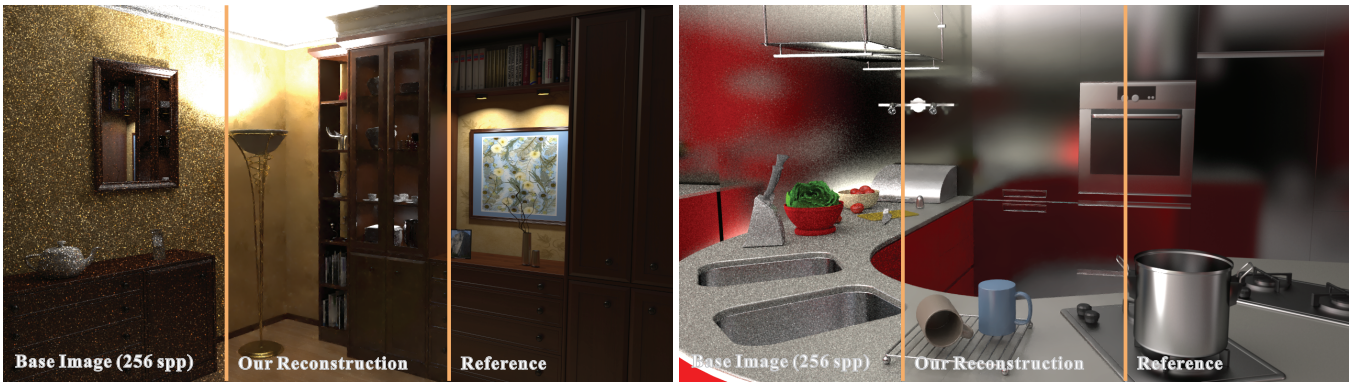


Fig. 1. We propose an unsupervised deep neural network (GradNet) for reconstructing high-quality images from noisy base images and the corresponding image gradients generated by gradient-domain renderers. Even with unlabeled training data, our network can still reproduce noise-free images closely matching the references.

Monte Carlo (MC) methods for light transport simulation are flexible and general but typically suffer from high variance and slow convergence. Gradient-domain rendering alleviates this problem by additionally generating image gradients and reformulating rendering as a screened Poisson image reconstruction problem. To improve the quality and performance of the reconstruction, we propose a novel and practical deep learning based approach in this paper. The core of our approach is a multi-branch auto-encoder, termed GradNet, which end-to-end learns a mapping from a noisy input image and its corresponding image gradients to a high-quality image with low variance. Once trained, our network is fast to evaluate and does not require manual parameter tweaking. Due to the difficulty in preparing ground-truth images for training, we design and train our network in a completely unsupervised manner by learning directly from the input data. This is the first solution incorporating unsupervised deep learning into the gradient-domain rendering framework. The loss function is defined as an energy function including a data fidelity term and a gradient fidelity term. To further reduce the noise of the reconstructed image, the loss function is reinforced by adding a regularizer constructed from selected rendering-specific features. We demonstrate that our method improves the reconstruction quality for a diverse set of scenes, and reconstructing a high-resolution image takes far less than one second on a recent GPU.

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

Additional Key Words and Phrases: Gradient-domain rendering, Deep learning, Unsupervised learning, Image reconstruction

ACM Reference Format:
Jie Guo, Mengtian Li, Quewei Li, Yuting Qiang, Bingyang Hu, Yanwen Guo, and Ling-Qi Yan. 2019. GradNet: Unsupervised Deep Screened Poisson Reconstruction for Gradient-Domain Rendering. *ACM Trans. Graph.* 38, 6, Article 223 (November 2019), 13 pages. https://doi.org/10.1145/3355089.3356538

*Both authors contributed equally to the paper
†Corresponding authors

Authors' addresses: Jie Guo, State Key Lab for Novel Software Technology, Nanjing University, guojie@nju.edu.cn; Mengtian Li, State Key Lab for Novel Software Technology, Nanjing University, lemonsky@smail.nju.edu.cn; Quewei Li, State Key Lab for Novel Software Technology, Nanjing University, liquewei@163.com; Yuting Qiang, State Key Lab for Novel Software Technology, Nanjing University, qiangyuting.new@gmail.com; Bingyang Hu, State Key Lab for Novel Software Technology, Nanjing University, fhymyang@gmail.com; Yanwen Guo, State Key Lab for Novel Software Technology, Nanjing University, ywguo@nju.edu.cn; Ling-Qi Yan, University of California, Santa Barbara, lingqi@cs.ucsb.edu.

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.
0730-0301/2019/11-ART223 $15.00
https://doi.org/10.1145/3355089.3356538

1 INTRODUCTION

In many fields including movie production [Keller et al. 2015] and architecture/product visualization [Křivánek et al. 2018], Monte Carlo (MC) integration based photorealistic image synthesis has found widespread adoption due to its conceptual simplicity, flexibility and generality in simulating a wide range of complex lighting effects. However, images produced by MC renderers with a low sampling rate are plagued with noise. The level of noise decreases slowly with respect to the sampling rate, and it is often prohibitively expensive to produce noise-free images from MC rendering.

Gradient-domain rendering exploits image-space coherence to significantly boost the performance of MC rendering [Bauszat et al. 2017; Hua et al. 2019; Kettunen et al. 2015; Lehtinen et al. 2013; Manzi et al. 2015, 2016a, 2014]. At the core of this technique is the usage of horizontal and vertical image gradients in addition to sole base images and a screened Poisson solver [Bhat et al. 2008] to reconstruct the final images. The key insight is that image gradients estimated by correlated path samples have much lower variance than image colors. After reconstruction in the image space, the output images are expected to contain far less noise than those generated by the primal-domain counterpart.

In gradient-domain rendering, image reconstruction is a crucial and non-trivial step that significantly influences the image quality. Current solutions are mainly based on iterative optimization. Unfortunately, most optimization schemes are sensitive to outliers and easily incur distracting artifacts even at a relatively high sampling rate. Although leveraging rendering-specific features such as normal, depth, and texture in optimization can reduce the artifacts, it will significantly increase the running time and memory consumption [Bitterli et al. 2016; Manzi et al. 2016b].

This paper employs a convolutional neural network (CNN) as a novel screened Poisson solver in lieu of the costly optimization-based techniques. Recently, CNNs have been becoming the common workhorse behind a wide variety of computer vision problems [LeCun et al. 2015], benefited from the increasing power of modern GPUs and the availability of extremely large image datasets. Our network, termed GradNet, tries to learn an end-to-end mapping from noisy images and their gradients to noise-free images, incorporating some auxiliary feature buffers. Although it is possible to design and train our network in a supervised fashion, akin to those used in learning-based MC denoising [Bako et al. 2017; Chaitanya et al. 2017; Kalantari et al. 2015; Vogels et al. 2018], we restrict it to be unsupervised. That is no clean data is supplied for training. To our knowledge, this is the first unsupervised deep learning based solution specially designed to enhance the image quality of MC rendering.

In the absence of high-quality target images for supervision, our GradNet learns directly from the noisy input data. The basic idea is to learn and reconstruct low-frequency contents from noisy color images and high-frequency details from the corresponding gradients. To this end, our GradNet is organized as a multi-branch deep auto-encoder [Hinton and Salakhutdinov 2006] with a joint loss function. A *data-branch* and a *data loss* are designed to extract low-frequency features from noisy color images, while a *gradient-branch* and a *gradient loss* aim to best preserve high-frequency structures from gradient images.

To further improve the image quality, we also leverage rendering-specific features (e.g., albedo, normal, depth) in our network. These features are both fed into our GradNet as input and used in the loss function. We develop a novel *first-order loss* accompanied with a new **G**-*branch*. The first-order loss encourages nearby pixels to lie on a hyper-plane (parameterized by **G** which is updated in the **G**-branch) in the high-dimensional space expanded by the features. We observe that including auxiliary feature buffers causes negligible overhead to the prediction of our GradNet but significantly improves the image quality.

In summary, our main contributions are:

- the first unsupervised deep learning solution to screened Poisson reconstruction in gradient-domain rendering,
- a multi-branch auto-encoder allowing extracting both low-frequency contents and high-frequency details from noisy inputs,
- a novel reconstruction loss function incorporating auxiliary feature buffers, and
- a well-designed training dataset that is easy to obtain and is open to the public.

2 RELATED WORK

2.1 Gradient-Domain Rendering

Gradient-domain rendering methods leverage image gradients in addition to colors to improve the convergence of MC rendering. Since the seminal work of gradient-domain Metropolis light transport [Lehtinen et al. 2013], there are several relevant follow-up works. To achieve effective variance reduction, highly correlated paths are generated to build an efficient estimator of the image gradient. After that, a screened Poisson reconstruction is performed to generate the final image. Manzi et al. [2014] propose an improved sampling strategy that estimates the gradient between two non-adjacent pixels. Kettunen et al. [2015] extend path tracing to the gradient domain and achieve a faster convergence rate than the primal-domain counterpart. To further reduce the variance, gradient-domain rendering has also been integrated with other techniques such as bidirectional path tracing [Manzi et al. 2015], path reusing [Bauszat et al. 2017], photon density estimation [Hua et al. 2017] and vertex connection and merging [Sun et al. 2017], combining the advantages of different techniques. By extending from 2D to 3D, temporal coherence can be utilized in gradient-domain rendering [Manzi et al. 2016a]. Recently, volumetric rendering has also been adapted to gradient-domain rendering [Gruson et al. 2018]. Our work is orthogonal to these techniques which focus on robust gradient sampling. The proposed deep neural network can be a competitive alternative in their reconstruction steps.

2.2 Image-Space Reconstruction

Image-space reconstruction is proved useful in many MC rendering methods as a post-processing step. In conventional primal-domain rendering, it mainly refers to image denoising which is routinely used in production environments. A thorough review of MC denoising can be found in some recent surveys [Sen et al. 2015; Zwicker

et al. 2015]. Traditional denoising methods usually rely on image filtering due to its simplicity and efficiency. Denoisers try to select the best reconstruction filter locally to minimize given objectives. To make the filter more robust to noise in the input, the non-local means (NLM) strategy is quite often used [Bitterli et al. 2016; Boughida and Boubekeur 2017; Buades et al. 2005; Delbracio et al. 2014; Kalantari and Sen 2013; Rousselle et al. 2012]. Auxiliary feature buffers are also fully utilized, serving as less noisy cues [Li et al. 2012; Rousselle et al. 2013; Sen and Darabi 2012]. These methods can be interpreted as zeroth-order linear regressions. First-order regressions based denoisers are becoming popular and have achieved state-of-the-art quality [Bauszat et al. 2011; Bitterli et al. 2016; Moon et al. 2014]. Higher-order models have also been explored [Moon et al. 2016], at the risk of overfitting to the input noise. More recently, machine learning approaches have found remarkable success at the task of MC denoising [Bako et al. 2017; Chaitanya et al. 2017; Kalantari et al. 2015; Vogels et al. 2018]. However, these learning-based methods require vast quantities of ground-truth data for training, which is computationally expensive for MC rendering.

In gradient-domain rendering, image reconstruction can be performed either by solving a 2D screened Poisson equation [Bhat et al. 2008] or by iterative optimization based on control variates [Rousselle et al. 2016]. Previous studies [Hua et al. 2019; Kettunen et al. 2015; Lehtinen et al. 2013] show that reconstruction under L_2 norm is unbiased but it is quite sensitive to outliers. L_1 reconstruction generally produces more visually pleasing results although it is biased. Manzi et al. [2016b] regularize the original screened Poisson equation with local patch constraints based on auxiliary features. This method generally shows better performance than either L_1 or L_2 reconstruction at the cost of long running time and high memory consumption. An ideal feature is derived by Back et al. [2018] that can lead to performance improvement by integrating with adaptive sampling. Ha et al. [2019] propose a least trimmed squares (LTS) to remove gradient outliers and enhance the image quality of reconstruction. Unlike these methods using costly optimization, we suggest introducing a deep neural network in the reconstruction step to boost its performance and robustness.

The only deep learning solution to image reconstruction in gradient-domain rendering is our concurrent work [Kettunen et al. 2019]. That method relies on reference images usually generated with a very high sampling rate in the supervised setting. In contrast, our unsupervised learning solution is free of converged images for training. The customized first-order loss and the multi-branch architecture allow our method to produce high-quality results on par with and even exceeding those produced by supervised learning.

2.3 Deep Learning for Rendering

Deep learning has been extensively used in many computer vision applications [LeCun et al. 2015]. Recently, there is a huge interest in applying deep learning techniques to some rendering-related tasks [Keller et al. 2018]. For instance, some recent experimental approaches try to learn better sampling schemes for MC rendering [Kuznetsov et al. 2018; Müller et al. 2018; Zheng and Zwicker 2019]. Nalbach et al. [2017] show that CNNs can predict some screen-space effects in real-time by learning the shading process from per-pixel attributes. Hermosilla et al. [2018] directly learn the mapping from a 3D scene description to a rendered image. Kallweit et al. [2017] employ a deep neural network for simulating volumetric light transport. CNNs have also been successfully used for denoising MC rendering [Bako et al. 2017; Chaitanya et al. 2017; Vogels et al. 2018]. These methods train CNNs in a fully supervised manner where the target outputs are generated by MC renderers at a very high sampling rate. Since generating ground-truth images is extremely computationally expensive in MC rendering, we resort to unsupervised learning which does not require clean data at training time. We demonstrate in Sec. 6 that even trained with unlabeled data, our network can still faithfully reconstruct high-quality images. Lehtinen et al. [2018] also proposed to suppress MC noise with unsupervised learning. Unlike ours, their method trains the network using L_2 loss (or a rescaled version) only and ignores the gradient information.

3 SCREENED POISSON RECONSTRUCTION

Given an input scene, a gradient-domain renderer typically outputs a coarse base image $\mathbf{I}_\mathrm{b}$ and two additional gradient images $\mathbf{I}_{\mathrm{d}x}$ and $\mathbf{I}_{\mathrm{d}y}$ containing the horizontal and vertical finite differences between neighboring pixels. The gradient images are expected to contain far less noise than the base image as they are obtained from pairs of highly correlated paths. For the final image reconstruction, two types of strategies are widely used in gradient-domain rendering. One is screened Poisson reconstruction [Bhat et al. 2008; Lehtinen et al. 2013], and the other is iterative optimization based on control variates [Rousselle et al. 2016]. In the former strategy, the final image $\hat{\mathbf{I}}$ is reconstructed by a screened Poisson solver, written as

$$\hat{\mathbf{I}} = \arg\min_{\mathbf{I}} \left\{ \left\| \begin{pmatrix} \nabla_x \mathbf{I} \\ \nabla_y \mathbf{I} \end{pmatrix} - \begin{pmatrix} \mathbf{I}_{\mathrm{d}x} \\ \mathbf{I}_{\mathrm{d}y} \end{pmatrix} \right\|_p + \alpha \|(\mathbf{I} - \mathbf{I}_\mathrm{b})\|_p \right\} \tag{1}$$

where ∇_x and ∇_y are the horizontal and vertical finite difference operators, respectively. On the right-hand side of the above formula, the first term ensures the gradient fidelity after reconstruction while the second term is used to maintain the correctness of color, both evaluated using L_p norm. Their relative influence is determined by the parameter α.

In general, p is selected as 1 or 2, corresponding to the L_1 or L_2 reconstruction, respectively. It is well-known that L_2 norm is sensitive to outliers [Hua et al. 2019]. As a consequence, L_2 reconstruction tends to produce visually unappealing results with bright spots, although it is unbiased and convenient to implement [Lehtinen et al. 2013]. In contrast, L_1 reconstruction is more robust and is preferred in practice since it produces numerically and perceptually better results with less noticeable artifacts. Concerning performance, L_2 reconstruction time is negligible compared to the rendering time, while L_1 reconstruction, usually relying on the method of *iteratively reweighted least squares* (IRLS), is slower and takes several seconds for a high-resolution image [Bhat et al. 2010; Lehtinen et al. 2013].

Even with the L_1 norm, artifacts still occur frequently if the base images and the gradients contain too much noise. A widely used strategy to further lower the variance is to impose additional constraints or regularizers, pertinent to the application domain of interest. A regularized version of the screened Poisson solver can be

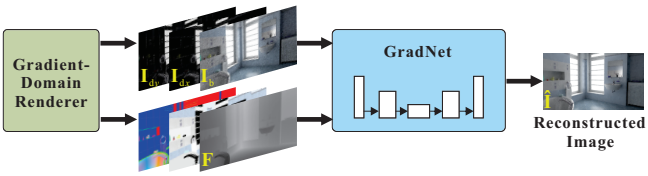


Fig. 2. An overall pipeline of the proposed framework. The key idea is to replace the traditonal optimization in screened Poisson image reconstruction with a deep neural network (GradNet).

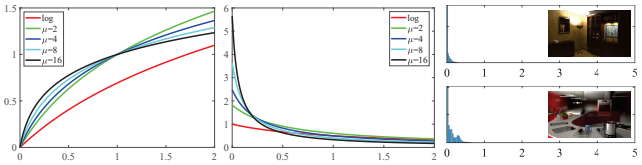


Fig. 3. Left: Plots of the conventional logarithmic transformation (log) and the μ-law transformation with varying μ. Middle: The corresponding derivatives. Right: Intensity histograms of two HDR images.

written as

$$\hat{\mathbf{I}} = \arg\min_{\mathbf{I}} \left\{ \left\| \begin{pmatrix} \nabla_x \mathbf{I} \\ \nabla_y \mathbf{I} \end{pmatrix} - \begin{pmatrix} \mathbf{I}_{\mathrm{d}x} \\ \mathbf{I}_{\mathrm{d}y} \end{pmatrix} \right\|_p + \alpha \|(\mathbf{I} - \mathbf{I}_{\mathrm{b}})\|_p + \lambda \Omega(\mathbf{I}, \mathbf{F}) \right\} \quad (2)$$

in which $\Omega(\mathbf{I}, \mathbf{F})$ is a regularization term typically built upon the feature vector $\mathbf{F}$ including several rendering-specific features. Similarly, solving this problem under L_1 norm usually relies on IRLS which is time-consuming and, worse still, requires too much memory [Manzi et al. 2016b].

4 UNSUPERVISED DEEP SCREENED POISSON RECONSTRUCTION

To further improve the quality of reconstructed images and lower the running time in the screened Poisson image reconstruction, we propose to model this process with a deep neural network termed as "GradNet". An overall pipeline is sketched in Fig. 2. Taking a coarse base image and the gradients as the input, our GradNet, after training on thousands of well-designed data, yields a high-quality image exhibiting low variance and fine details. Likewise, some rendering-specific features serving as less noisy cues are also useful in enhancing the reconstructed results. These feature buffers can be generated inexpensively as a byproduct of MC rendering.

4.1 Problem Formulation

We first formalize our learning-based screened Poisson reconstruction problem. There is little doubt that supervised learning techniques are effective but labor intensive. In MC rendering, due to the difficulty in creating noise-free images, preparing the paired training data for supervised learning is a costly and daunting task. To circumvent this, we turn to design and train an unsupervised learning framework that only uses easier-to-obtain noisy images accompanied with corresponding gradients and some features.

We consider the typical unsupervised learning setup with a training set $\mathcal{D}$ containing K label-free examples $\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(K)}\}$. Each example $\mathbf{x}^{(k)}$ ($k \in [1, K]$) comprises a coarse base image, two gradient images and a multi-channel feature image, i.e.,

$$\mathbf{x}^{(k)} = [\mathbf{I}_{\mathrm{b}}^{(k)}, \mathbf{I}_{\mathrm{d}x}^{(k)}, \mathbf{I}_{\mathrm{d}y}^{(k)}, \mathbf{F}^{(k)}]. \quad (3)$$

Our network training consists in finding a value of parameter vector θ minimizing the following reconstruction error:

$$\epsilon(\theta) = \sum_{k=1}^{K} \mathcal{L}(\mathbf{x}^{(k)}, \Phi_\theta(\mathbf{x}^{(k)})) \quad (4)$$

in which Φ_θ denotes our network parameterized by θ and $\mathcal{L}$ is a loss function. Without converged images serving as target outputs, the loss function, behaving like the energy function in conventional optimization methods, plays an important role in solving our problem. It should be carefully designed such that visually distracting noise are significantly suppressed without weakening important image structures.

Once trained, our network is able to reconstruct high-quality images given new examples, i.e.,

$$\hat{\mathbf{I}} = \Phi_\theta(\mathbf{x}) = \Phi_\theta(\mathbf{I}_{\mathrm{b}}, \mathbf{I}_{\mathrm{d}x}, \mathbf{I}_{\mathrm{d}y}, \mathbf{F}). \quad (5)$$

The ideal network estimation $\hat{\mathbf{I}}$ should be noise-free and preserve most of the scene details.

4.2 Dynamic Range Compression

Unlike reconstruction of natural images, the raw data (e.g., $\mathbf{I}_{\mathrm{b}}$) fed into our network contain high dynamic range (HDR) information. Usually, the HDR regions distribute unevenly in the image space and the dynamic range is unbounded. This causes an obstacle for the existing CNNs initially designed for low dynamic range (LDR) images. Most previous methods handling HDR images map an original input $\mathbf{I}$ to the logarithmic domain by the conventional logarithmic transformation $\tilde{\mathbf{I}} = \log(\mathbf{I}+1)$ [Bako et al. 2017; Kang et al. 2018; Kuznetsov et al. 2018; Vogels et al. 2018]. To accelerate the training of deep networks, we suggest using the following μ-law transformation as inspired by Kalantari et al. [2017]. It performs range compression for an HDR image $\mathbf{I}$ as

$$\mathcal{T}(\mathbf{I}) = \mathrm{sign}(\mathbf{I}) \frac{\log(1 + \mathrm{abs}(\mathbf{I})\mu)}{\log(1 + \mu)} \quad (6)$$

in which μ controls the amount of compression. This type of compressor is differentiable and stable for deep learning systems.

Obviously, the conventional logarithmic transformation is a special case (up to a constant factor) of the μ-law transformation in which $\mu = 1$. We visually compare their differences in Fig. 3. As seen, the derivatives of the μ-law transformation ($\mu > 1$) are relatively large in the low image intensity areas. Since most pixels of a typical HDR image have low intensity as shown in Fig. 3 right, a large derivative may avoid gradient vanishing in training neural networks with back propagation. Currently, we empirically set $\mu = 16$ in our network. Experimental validation is provided in Sec. 6.6.

4.3 Loss Function

One of the difficulties in designing a deep neural network is in establishing a clear and effective objective (e.g., a loss function) for training. This is even more challenging and particularly important for our task since ground-truth images rendered at very high sample counts are unavailable in the unsupervised setting. Without ground truths at hand, the loss function should

- not solely rely on the data fidelity since the source images are often of low quality,
- pay more attention to the gradient fidelity since the gradients contain far less noise, and
- try to make the best of rendering-specific features.

To fulfill these requirements, we employ a joint loss function containing three components: a data loss $\mathcal{L}_{\text{data}}$, a gradient loss $\mathcal{L}_{\text{grad}}$ and a first-order loss $\mathcal{L}_{\text{1st}}$. The overall loss function thus amounts to

$$\mathcal{L}_{\text{all}} = \mathcal{L}_{\text{grad}}(\hat{\mathbf{I}}, \mathbf{I}_{\text{d}x}, \mathbf{I}_{\text{d}y}) + \alpha\mathcal{L}_{\text{data}}(\hat{\mathbf{I}}, \mathbf{I}_{\text{b}}) + \lambda\mathcal{L}_{\text{1st}}(\hat{\mathbf{I}}, \mathbf{F}) \tag{7}$$

in which α and λ are two weights controlling the amount of influence a loss should have on the final image. We use this joint loss function as an energy function, in a similar manner as in [Fan et al. 2018].

4.3.1 Data loss. Though noisy, the input image is still the essential reference to ensure color consistency. In our work, the data loss is evaluated by the per-pixel L_1 distance between the base image $\mathbf{I}_{\text{b}}$ and the reconstructed image $\hat{\mathbf{I}}$. Since both $\mathbf{I}_{\text{b}}$ and $\hat{\mathbf{I}}$ have a high dynamic range, we compute this loss in the logarithmic domain based on the previous μ-law transformation:

$$\mathcal{L}_{\text{data}}(\hat{\mathbf{I}}, \mathbf{I}_{\text{b}}) = \frac{1}{N}\sum_{i=1}^{N} \|\mathcal{T}(\hat{\mathbf{I}}_i) - \mathcal{T}(\mathbf{I}_{\text{b},i})\| \tag{8}$$

where N is the total pixel number and $\|\cdot\|$ denotes L_1 norm unless otherwise stated. It is widely recognized that optimization using L_1 norm can reduce splotchy artifacts from reconstructed images [Chaitanya et al. 2017; Zhao et al. 2017].

4.3.2 Gradient loss. In gradient-domain rendering, image gradients are critical in reproducing high-quality images. In our network, the gradient loss is used to guarantee image smoothness and sharpness of edges. Similar to the data loss, we also evaluate the gradient loss in the logarithmic domain:

$$\begin{aligned}&\mathcal{L}_{\text{grad}}(\hat{\mathbf{I}}, \mathbf{I}_{\text{d}x}, \mathbf{I}_{\text{d}y}) = \\ &\frac{1}{N}\sum_{i=1}^{N}(\|\mathcal{T}(\nabla_x\hat{\mathbf{I}}_i) - \mathcal{T}(\mathbf{I}_{\text{d}x,i})\| + \|\mathcal{T}(\nabla_y\hat{\mathbf{I}}_i) - \mathcal{T}(\mathbf{I}_{\text{d}y,i})\|).\end{aligned} \tag{9}$$

4.3.3 First-order loss. Rendering-specific features can further improve the quality of reconstructed images in MC rendering. In our network, we exploit a rich set of features including albedo (3D), normal (3D) and depth (1D) to construct the first-order loss. Under this configuration, each feature vector is a 7-dimensional vector. Intuitively, pixels with the similar feature vector tend to have the same value. Inspired by the first-order regression models [Bitterli et al. 2016; Moon et al. 2014], we define our first-order loss as

$$\mathcal{L}_{\text{1st}}(\hat{\mathbf{I}}, \mathbf{F}) = \frac{1}{N|\mathcal{N}_i|}\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i} w_{i,j}\|\mathcal{T}(\hat{\mathbf{I}}_j) - \mathcal{T}(\hat{\mathbf{I}}_i) - \mathbf{G}_i^{\top}(\mathbf{F}_j - \mathbf{F}_i)\| \tag{10}$$

where $\mathbf{F}$ is a high-dimensional feature vector and $\mathcal{N}_i$ represents the neighboring pixels around index i. We restrict the neighborhood to the four nearest neighbors, i.e., the immediate left, right, top, and bottom neighbors of pixel i. This actually provides a very small region centered around pixel i. The term $\mathbf{G}_i^{\top}(\mathbf{F}_j - \mathbf{F}_i)$ belongs to the Taylor polynomial of order one that approximates $\hat{\mathbf{I}}_j$ via $\hat{\mathbf{I}}_i$ and its derivative $\mathbf{G}_i$. The weight $w_{i,j}$ is calculated from $\mathbf{I}_{\text{b}}$: $w_{i,j} = \exp[-\|\mathbf{I}_{\text{b},i} - \mathbf{I}_{\text{b},j}\|^2/(2\sigma_{\text{b}}^2)]$ with σ_{b} representing its filtering bandwidth. Currently, we empirically set $\sigma_{\text{b}}^2 = 1/18$. This enable us to reproduce a smooth image with well-preserved details.

In Eq. (10), the derivative $\mathbf{G}$ is unknown and should be jointly optimized with $\hat{\mathbf{I}}$ in the traditional methods [Bitterli et al. 2016; Moon et al. 2014]. In our CNN-based approach, we train an independent branch for $\mathbf{G}$ with the loss function:

$$\mathcal{L}'(\mathbf{G}, \mathbf{I}_{\text{b}}, \mathbf{F}) = \frac{1}{N|\mathcal{N}_i|}\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i} w_{i,j}\|\mathcal{T}(\mathbf{I}_{\text{b},j}) - \mathcal{T}(\mathbf{I}_{\text{b},i}) - \mathbf{G}_i^{\top}(\mathbf{F}_j - \mathbf{F}_i)\|. \tag{11}$$

This loss is quite similar to the first-order loss in Eq. (10) except that $\hat{\mathbf{I}}$ is replaced by $\mathbf{I}_{\text{b}}$. It seems that we may still use $\hat{\mathbf{I}}$ in the above loss, but experimental results show that both the derivative $\mathbf{G}$ and the reconstructed image $\hat{\mathbf{I}}$ will be over-blurred in this case.

4.4 Network Architecture

Now, we describe our network architecture in detail. As shown in Fig. 4, we adopt the basic architecture of a deep encoder-decoder network with several branches. The encoder is split into two branches: a data-branch and a gradient-branch. HDR images are compressed by the μ-law transformation in Eq. (6) before feeding into the network. The data-branch is designed to extract low-frequency contents from noisy base image $\mathbf{I}_{\text{b}}$. The auxiliary feature buffers are also fed into this branch. In contrast, the gradient-branch is expected to extract feature maps from image gradients $\mathbf{I}_{\text{d}x}$ and $\mathbf{I}_{\text{d}y}$ and to enhance the high-frequency scene structures. It seems more straightforward to construct the encoder with a single branch such that both image colors and gradients are fed into this branch. However, experimental results shown in Sec. 6.5 reveal that such a single branch encoder is suboptimal compared with our two-branch version. This is probably because image gradients are quite sparse, and a single branch with shared weights may weaken the effects of image gradients. A separate branch tailored for the gradients tends to preserve the desired features extracted from the gradients as much as possible.

For all convolution operations in both branches, the kernel size is set to 3×3 and the stride is selected as 2. Consequently, the resolution of output feature maps are divided by two while the number of feature channels are doubled. These downsampling steps can extend the receptive fields of our network.

After three convolutional layers, the output feature maps of the two branches are concatenated and fed into several residual blocks [He et al. 2016]. These residual blocks are inserted to accelerate the learning process. Currently, we use 4 residual blocks in GradNet. The deconvolutional layers on the decode side are responsible for upsampling the feature maps and recovering the image details while suppressing noise. The convolution operations in the decoder have the kernel size 4×4 and stride 2. Note that there are no fully

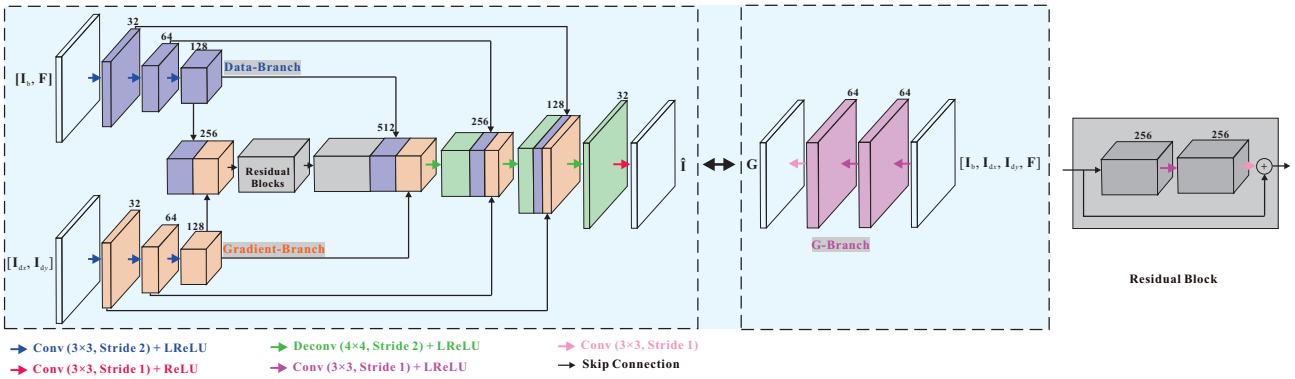


Fig. 4. The network architecture of GradNet. We adapt the U-Net [Ronneberger et al. 2015] and fuse it with a network branch optimized for image gradients. Four residual blocks are inserted to accelerate the learning process. Each residual block contains two convolutional layers, a Leaky ReLU (LReLU) activation unit and a residual connection. Skip connections are used between mirrored layers in the encoder and decoder stacks. We also include a G-branch to update $\mathbf{G}$ which is used in the first-order loss $\mathcal{L}_{1st}$.

connected layers in our network and hence it is only convolutional. This can keep the number of parameters reasonably low.

Similar to the U-Net [Ronneberger et al. 2015], our network also uses skip connections between mirrored layers in the encoder and decoder stacks. Skip connections incorporate local and low-level information from input data into the decoder step-by-step. After several downsampling steps in the encoder, many high-frequency details in earlier layers are lost. However, these information could be potentially used by the decoder to aid reconstructing fine details. Unlike the skip connections used in the U-Net which has only one branch in the encoder, our encoder has two branches and we introduce the following *dual skip connection* based upsampling operation:

$$\mathbf{h}_l^D = \text{Deconv}(\mathbf{h}_{l-1}^D \oplus \mathbf{h}^{ED} \oplus \mathbf{h}^{EG}) \tag{12}$$

where $\mathbf{h}_l^D$ is the feature map generated by the l-th layer of the decoder while $\mathbf{h}^{ED}$ and $\mathbf{h}^{EG}$, with the same spatial resolution with $\mathbf{h}_{l-1}^D$, are the feature maps generated by the data-branch and the gradient-branch, respectively. Deconv is a traditional deconvolution operation and $\oplus$ denotes concatenation along the feature dimension.

A $\mathbf{G}$-branch on the right side of Fig. 4 is added to update the derivative $\mathbf{G}$ which is required in the first-order loss $\mathcal{L}_{1st}$. For this branch, we use a slightly shallow network with only three convolutional layers, because we observe that a deep network may over-blur $\mathbf{G}$, resulting in loss of detail of reconstructed images.

Moreover, we adopt Leaky ReLU (LReLU) [Maas et al. 2013] activations in all the convolutional layers except the last layer of the decoder and the last layer of the $\mathbf{G}$-branch. Unlike the ReLU activation that simply thresholds at zero when the input $x < 0$, the LReLU activation returns ax with a being a small constant (e.g., $a = 0.01$), in the negative region. This avoids zero gradients for negative inputs.

4.5 Post-Processing

The network output is slightly biased due to the usage of the μ-law transformation in HDR compression and decompression. Adding auxiliary feature buffers into our network may also increase the bias. However, we find that a simple post-processing step can reduce the bias. This post-processing step is necessary to alleviate the bias problem regardless of the sampling rate. Specifically, let $\hat{\mathbf{I}}$ be the network output and $\mathbf{k}$ be a pre-defined filtering kernel, the final image is adjusted as

$$\hat{\mathbf{I}} \leftarrow \frac{\mathbf{I}_b * \mathbf{k}}{\hat{\mathbf{I}} * \mathbf{k}} \odot \hat{\mathbf{I}} \tag{13}$$

where $*$ is a convolution operator, $\odot$ indicates the Hadamard product (element-wise product) and the division executes similarly in the element-wise manner. Currently, we choose a simple Gaussian filter with a radius 45 and a standard deviation 15. The visual effect of this post-processing is shown in Fig. 5. As seen, this post-processing step lowers the reconstruction error.

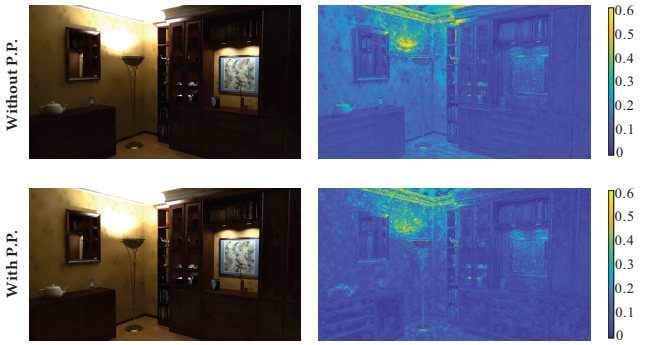


Fig. 5. Visual effect of the post-processing (P.P.) step. Left column: reconstructed images from a base image of 512 samples per pixel. Right column: error maps compared against the reference.

5 IMPLEMENTATION AND TRAINING DETAILS

5.1 Dataset Preparation

The performance of deep learning methods heavily depends on the choice of datasets. Since generating a large quantity of noise-free images using MC rendering is tremendously tedious and time consuming, we organize our training data without ground truths.

This means the training examples used in our network are noisy. Obviously, collecting such a dataset with "poor" images is much easier than that with clean images as supervision.

We generate a dataset consisting of nine base scenes from [Bitterli 2016]. To assure that the dataset covers a sufficient range of color patterns, lighting conditions and geometries, we randomly perturb the base scenes by varying materials, lighting and camera parameters, and finally generate 900 high-resolution (1280×720) images in total. Some randomly changed scenes are demonstrated in the supplemental material. We generate the high-resolution base images and the corresponding gradients with the gradient-domain path tracer [Kettunen et al. 2015] at a sampling rate of 64 spp (samples per pixel), along with three feature buffers (i.e., albedo, normal and depth) as a by-product. The albedo is recorded at the first non-specular surface along the path. The normal buffer ($\mathbf{N}$) is transformed by $(\mathbf{N}+1)/2$ while the depth buffer is linearly scaled to the range $[0, 1]$. In this way, all channels of the feature buffers are normalized to the range $[0, 1]$. Then, we randomly extract 15 patches of size 256×256 from each image. Each patch contains one noisy base image, two image gradients and three feature buffers. Eliminating 846 patches with illegal values [1], 12654 patches are reserved. These patches are further split into 11785 training samples and 869 testing samples to train our network. In the inference phase, images of arbitrary resolutions can be given to the network as input, thanks to the fully convolutional neural network. We plan to release all our training and testing data upon publication.

5.2 Training

Our GradNet is implemented on top of the PyTorch framework [Paszke et al. 2017]. We train it using mini-batch SGD and apply the Adam solver [Kingma and Ba 2015] with moment parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We adjust the learning rate with the basic learning rate of 0.0001 and the power of 0.95 every other epoch. Hence, the learning rate gradually decays based on the epoch. The weights of the network are all initialized using the technique described in He et al. [2015]. Training examples are fed into the network in a mini-batch size of 32. We train the network for 50 epochs, which takes about 4 hours on four NVIDIA GTX 1080Ti GPUs. The auto-encoder and the $\mathbf{G}$-branch are trained alternatively during each epoch.

For the weights in the loss function, we currently set α to a fixed value 1 while setting λ as

$$\lambda = \begin{cases} 0 & \text{if} \quad \text{epoch} \leq 5 \\ \min(0.1 \times 1.1^{(\text{epoch}-5)}, 2.0) & \text{if} \quad \text{epoch} > 5 \end{cases} \tag{14}$$

implying that the first-order loss is introduced after 5 epochs, and its weight increases slowly until it reaches a maximum 2.0. The training schedule for λ is designed such that the auxiliary buffers are not involved in the loss function at an early pre-training stage, avoiding the reconstructed images deviating greatly from the base images. As λ increases, an optimal combination of feature term and data term can be learned, leading to smoother results.

[1] Illegal patches are those patches containing illegal pixels with invalid values, such as NaN and negative values, due to numerical issues in gradient-domain path tracing.

6 RESULTS AND DISCUSSION

To evaluate our method, we compare it to a range of state-of-the-art image reconstruction techniques designed for gradient-domain rendering. We also compare against some image denoising methods for general MC rendering. After that, we thoroughly analyze the various design choices made in our GradNet. The error metric used in comparison is the relative mean square error (RelMSE): $\text{mean}((\hat{\mathbf{I}} - \mathbf{I}_{\text{ref}})^2/((\mathbf{I}_{\text{ref}})^2 + \delta))$ in which $\hat{\mathbf{I}}$ is the reconstructed image, $\mathbf{I}_{\text{ref}}$ is the corresponding reference image generated by path tracing at an extremely high sampling rate and $\delta = 0.01$ is a small constant used to avoid dividing by zero.

6.1 Comparison with Existing Reconstruction Methods

We first compare our deep learning based method to other existing image reconstruction techniques for gradient-domain rendering: L_2 reconstruction, L_1 reconstruction, weighted reconstruction based on control variates (CV) [Rousselle et al. 2016], and the least trimmed squares (LTS) optimization [Ha et al. 2019]. We compare these methods with the same sampling rate because their reconstruction time is negligible compared with the sampling time. The results of five benchmark scenes (Kitchen, Bookshelf, Bathroom, Sponza and Door) are reported in Fig. 6. These scenes cover a variety of scene configurations and are not used in training our network. The input images (base images, gradients and features) are generated by gradient-domain path tracing [Kettunen et al. 2015] implemented on top of the Mitsuba renderer [Jakob 2010] and the reference images are generated by primal-domain path tracing at a sampling rate of 128K spp. From the comparison we see that our method significantly improves L_2 reconstruction and L_1 reconstruction in RelMSE of a factor of 2 to 5. The weighted CV method and the LTS method tend to produce images of better quality than either L_1 or L_2 reconstruction. However, our deep learning based method outperforms these methods both qualitatively and quantitatively. The closeups further validate that our method can significantly reduce the variance of input base images, yielding higher-quality images than those generated by its competitors. We provide a supplemental material containing all images in an interactive HTML viewer.

Fig. 7 shows RelMSE convergence of the above methods in comparison across the five scenes. Here, we plot the RelMSE with respect to the sampling rate. Recall that our model is trained on a dataset with only one sampling rate: 64 spp. Even so, our method still performs well on a wide range of noise levels. It consistently generates numerically improved results compared against the previous techniques.

We also compare our method to the regularized reconstruction method by Manzi et al. [2016a] in Fig. 8. Generally, this regularized reconstruction method quantitatively outperforms other conventional reconstruction methods without leveraging auxiliary feature buffers, as shown in 7. However, it has high time and storage complexities. Using the same sampling rate, our deep learning solution is on par with and even exceeding that method with far less running time (0.16 seconds vs. 1 minute) and memory consumption (3.8GB vs. 7.5GB). The regularized reconstruction method tends to over-blur important scene details such as the shadows in the Door scene and the Kitchen scene. On the other hand, our method also contains

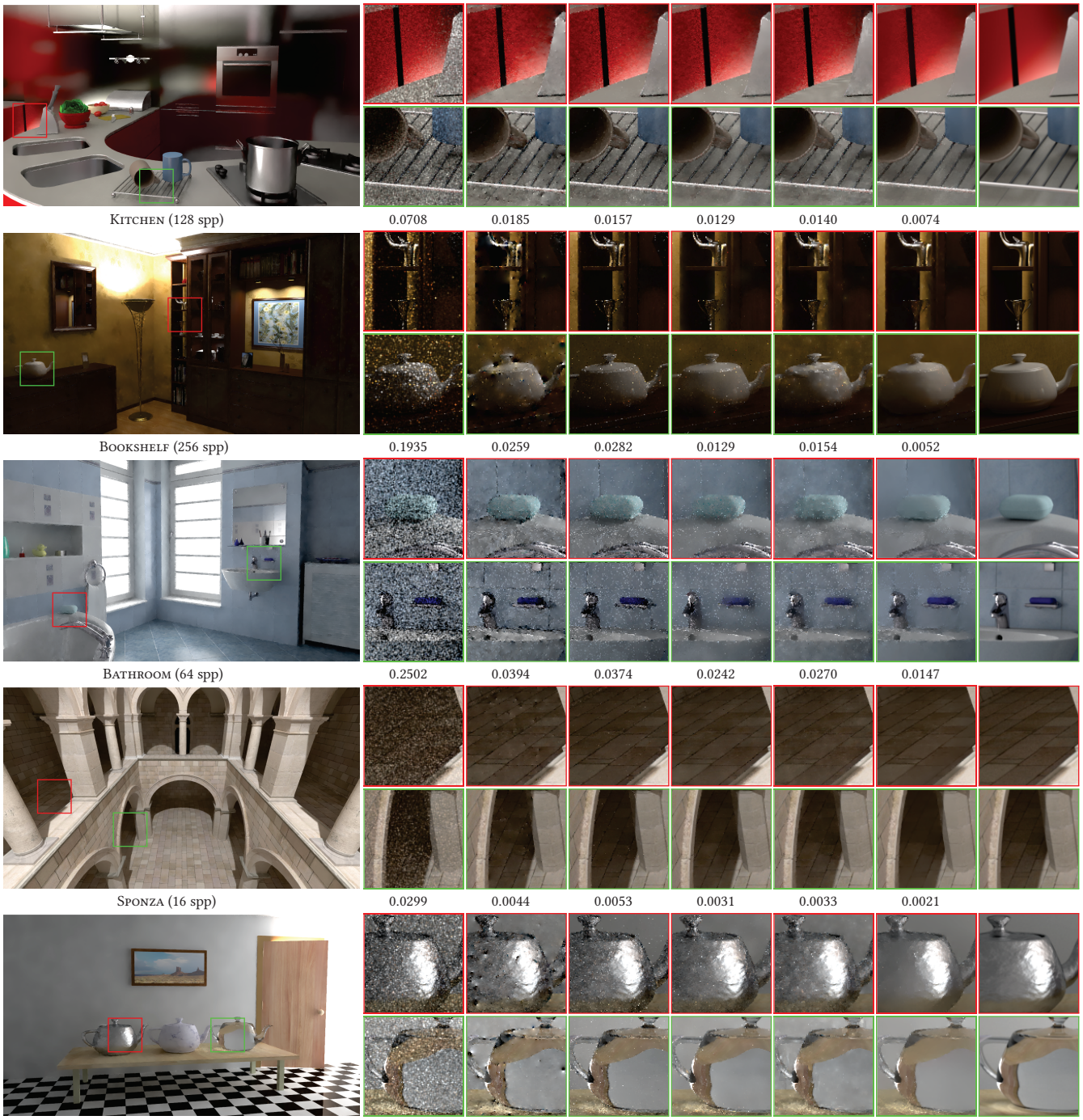
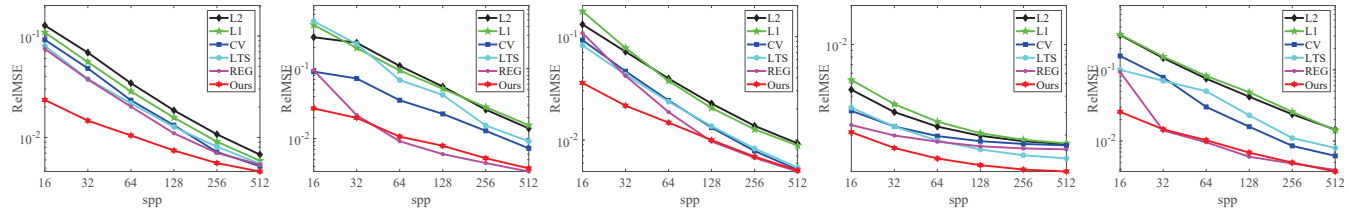


Fig. 6. Image reconstruction results using our GradNet, compared to several existing reconstruction methods: L_2 reconstruction, L_1 reconstruction, weighted reconstruction based on control variates (CV) [Rousselle et al. 2016], and the LTS optimization [Ha et al. 2019]. The numbers at the bottom of each image report the RelMSE of the corresponding reconstruction method.



(a) Kitchen (b) Bookshelf (c) Bathroom (d) Sponza (e) Door

Fig. 7. RelMSE convergence plots of the tested methods as a function of the samples per pixel (spp) on five scenes.

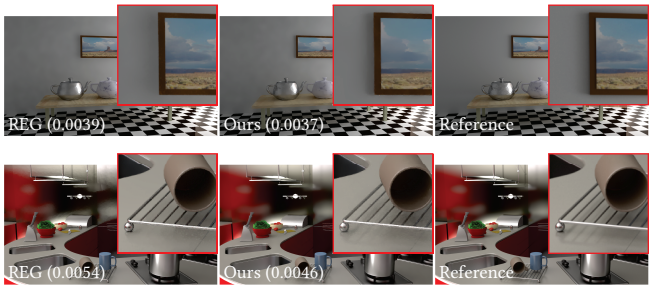
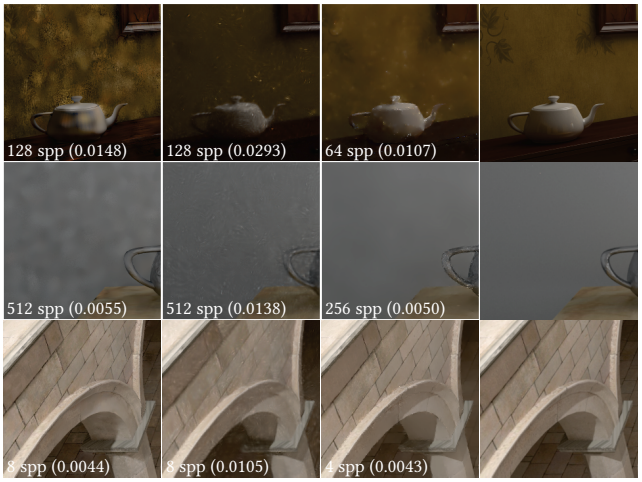


Fig. 8. Comparison between our method and the regularized reconstruction method (REG) by Manzi et al. [2016a]. The sampling rate is 512 spp and the RelMSE is provided in parenthesis.

some over-blurred regions, e.g., the wallpaper in the Bookshelf scene, as shown in the supplemental material. This is because the noise level is high in these regions such that the effect of the albedo has been weakened.

6.2 Comparison with Denoising Methods

To further show the effectiveness of our GradNet, we also compare it to two state-of-the-art image-space denoising methods designed for conventional MC rendering: Nonlinearly Weighted First-order Regression (NFOR) [Bitterli et al. 2016] and Bayesian Collaborative Denoising (BCD) [Boughida and Boubekeur 2017]. Note that the sampling time of gradient-domain path tracing is roughly twice higher than that of its primal-domain counterpart. Therefore, for an almost equal-time (regardless the denoising time) comparison we double the sampling rate for both the NFOR denoiser and the BCD denoiser. As shown in Fig. 9, these two denoisers may produce visually distracting artifacts, while our method generally shows fewer artifacts even if the sampling rate is reduced by half. Moreover, we observe that the NFOR denoiser occasionally smoothes out some shadows as shown in the Sponza scene, probably because the features it relied on are not apparent in this region. Concerning the timing performance, the denoising time of NFOR is typically more than one minute regardless the sampling rate while the denoising time of BCD increases with respect to the sampling rate. In comparison, the reconstruction time of our method is far less than one second and hence is negligible compared to the sampling time. For a comprehensive comparison, please refer to the supplemental material.



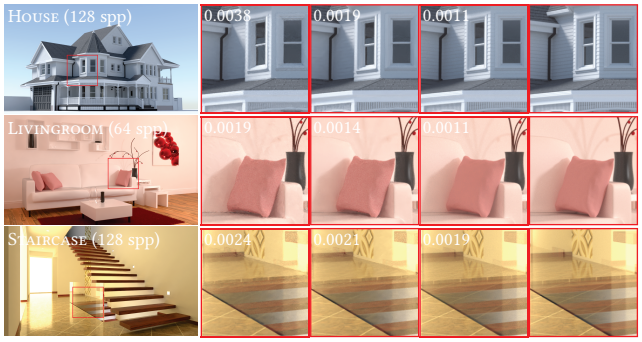
(a) NFOR (b) BCD (c) Ours (d) Reference

Fig. 9. Comparison with two image-space denoising methods: NFOR [Bitterli et al. 2016] and BCD [Boughida and Boubekeur 2017]. The sampling rate and RelMSE (in parenthesis) are provided.

6.3 Comparison with Supervised Learning Solutions

Currently, most learning-based MC denoisers rely heavily on ground-truth images. Compared with these supervised learning solutions, our solution has a significantly lower cost in obtaining unlabeled data. Consequently, a much larger training dataset can be collected using the same time budget as in the supervised learning solutions, leading to better inferring results. To show this, we compare our method to the KPCN of Bako et al. [2017] in Fig. 10 [2]. To ensure roughly the same time budget in collecting the dataset, only 9 ground-truth images rendered with 8192 spp are generated and 3600 training examples are extracted from these 9 images. With such a small dataset, the quality of reconstruction (KPCN/3K) is significantly lower than ours. Even with 4× dataset whose size is roughly the same as ours, the KPCN (KPCN/14K) may still be inferior than ours. Thanks to the special design for the gradients, our method retains most high-frequency details, especially the shadows. Note that the scenes used in Fig. 10 and the training examples of the KPCN are all generated by the Tungsten renderer [Bitterli 2016]. The results

[2]For a fair comparison, we modify the KPCN by incorporating the gradients as an additional auxiliary feature.



(a) Ours (b) KPCN/3K (c) KPCN/14K (d) Ours (e) Reference

Fig. 10. Comparing our method to the KPCN by Bako et al. [2017] with the same time budget in collecting the dataset (KPCN/3K) and the same size of training dataset (KPCN/14K), respectively. Our method outperforms the KPCN both visually and numerically (RelMSE).

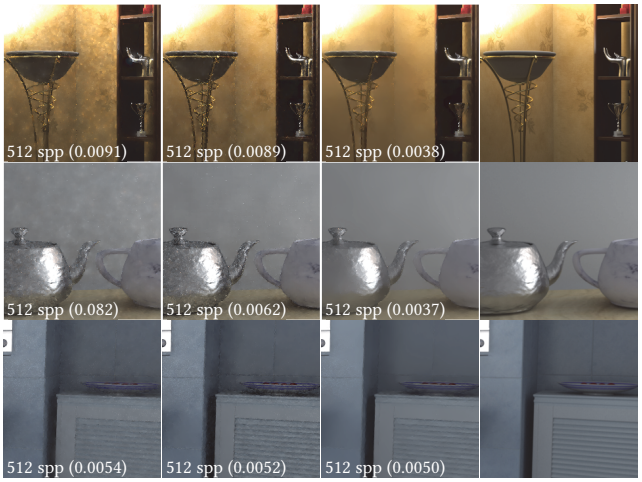
Fig. 11. Impact of the training datasets in our method. We compare our method trained with full dataset (Full) against those trained with 1/4 dataset (Quarter) and 1/2 dataset (Half), respectively. The sampling rate is 256 spp for each scene and the RelMSE is shown below.

of the previous five scenes rendered by the Mitsuba renderer are provided in the supplemental material.

We also test the different sizes of training datasets in our method. In Fig. 11, as the size of the training dataset reduces by half and more, the RelMSE decreases steadily while the visual quality changes imperceptibly. Therefore, even with a small training dataset, our network still works well. Nevertheless, expanding the training dataset will improve the performance of our GradNet.

6.4 Impact of Losses

We conduct several experiments to evaluate the impact of each loss, especially the gradient loss $\mathcal{L}_{\text{grad}}$ and the first-order loss $\mathcal{L}_{\text{1st}}$. In the following, we consider the loss combinations: $\mathcal{L}_{\text{d+g}} = \mathcal{L}_{\text{grad}} + \alpha\mathcal{L}_{\text{data}}$, $\mathcal{L}_{\text{d+1}} = \alpha\mathcal{L}_{\text{data}} + \lambda\mathcal{L}_{\text{1st}}$, and our joint loss function $\mathcal{L}_{\text{all}} = \mathcal{L}_{\text{grad}} + \alpha\mathcal{L}_{\text{data}} + \lambda\mathcal{L}_{\text{1st}}$.



(a) LTS (b) $\mathcal{L}_{\text{d+g}}$ (c) $\mathcal{L}_{\text{all}}$ (d) Reference

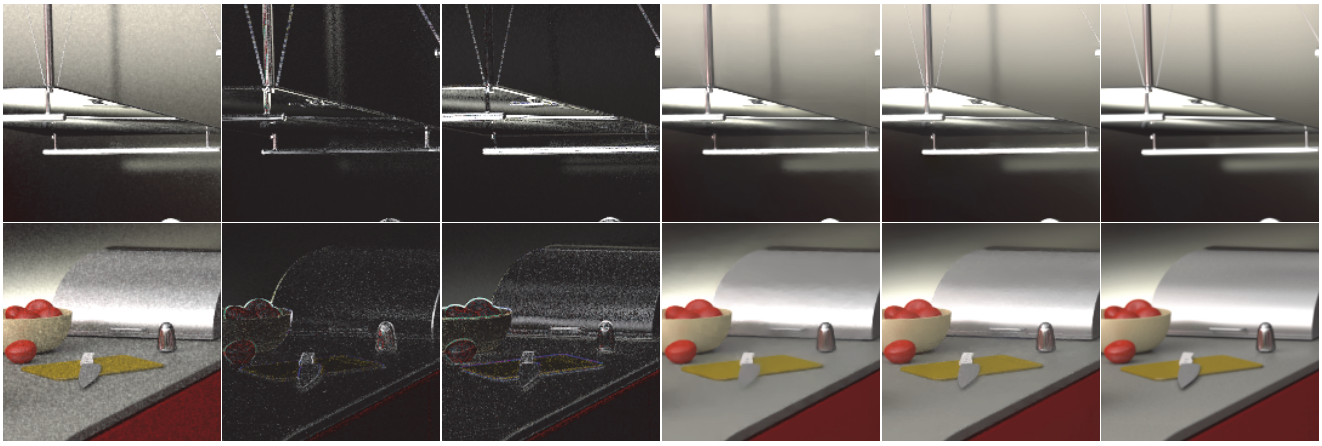
Fig. 12. Visual comparison between our method without ($\mathcal{L}_{\text{d+g}}$) and with ($\mathcal{L}_{\text{all}}$) the first-order loss. The numbers in parenthesis report the RelMSE.

Fig. 13. Impact of the gradient loss on the Door scene. As the sampling rate increases, our method with $\mathcal{L}_{\text{all}}$ (bottom row) consistently surpasses that with $\mathcal{L}_{\text{d+1}}$ (top row) both visually and in terms of RelMSE (in parenthesis).

Fig. 12 compares our method without and with the first-order loss. The same training configuration is applied to each case. As expected, without the first-order loss, the reconstruction (using $\mathcal{L}_{\text{d+g}}$) fails to remove some noticeable outliers and the results would be quite noisy even at a high sampling rate (e.g., 512 spp). The situation worsens as the sampling rate decreases. However, even without feature buffers, our method still outperforms the state-of-the-art LTS method that also does not rely on any feature. Note that the LTS method only aims to remove strong outlier gradients, and is ineffective to inlier gradients [Ha et al. 2019]. In addition, the LTS method may fail given a small number of samples.

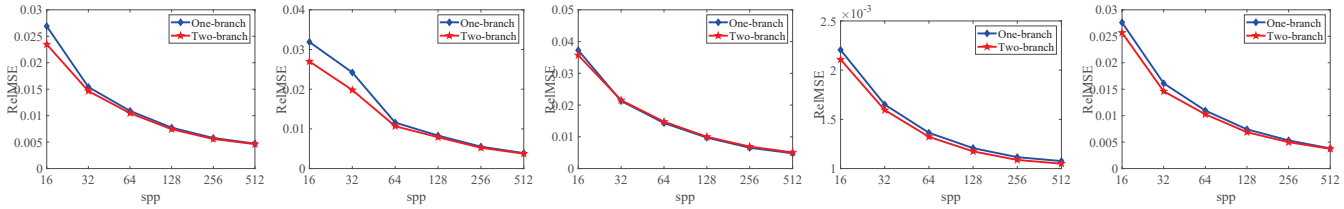
Fig. 13 and Fig. 14 compare our method without [3] and with the gradient loss. The gradient can be viewed as a special feature of the scene. Unlike other features used in our network, the gradient can preserve glossy reflection, shadows and caustics that can hardly be

[3]The gradients are absent in both the network architecture and the loss function.



(a) $\mathbf{I}_{\mathrm{b}}$ (1024 spp) (b) $\mathbf{I}_{\mathrm{d}x}$ (c) $\mathbf{I}_{\mathrm{d}y}$ (d) $\mathcal{L}_{\mathrm{d+1}}$ (e) $\mathcal{L}_{\mathrm{all}}$ (f) Reference

Fig. 14. Visual comparison between our method without ($\mathcal{L}_{\mathrm{d+1}}$) and with the gradient loss ($\mathcal{L}_{\mathrm{all}}$) on the KITCHEN scene. Without the gradient serving as a special feature, some important scene structures such as highlights and shadows can hardly be captured by our auxiliary feature buffers.



(a) KITCHEN (b) BOOKSHELF (c) BATHROOM (d) SPONZA (e) DOOR

Fig. 15. RelMSE convergence comparison between the one-branch style (blue curves) and the two-branch style (red curves) on the encoder side.

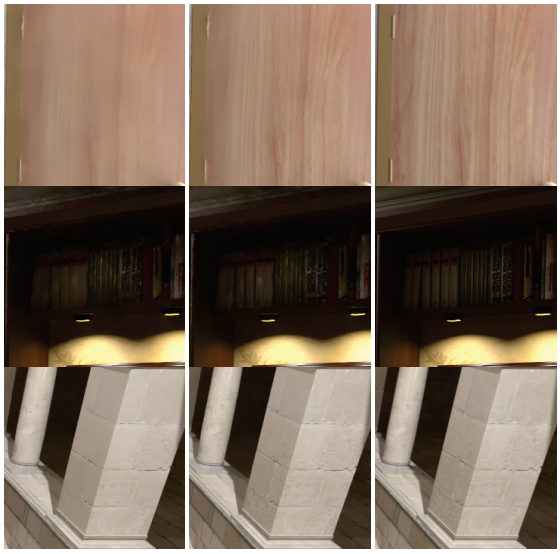
captured by geometric features and textures. As seen in the first row of Fig. 14, some metal wires are smoothed out since geometric features and textures on these wires are inapparent. In contrast, the gradients are quite high in these regions due to strong glossy reflection and hence are more appropriate for recovering these tiny structures. The second row shows that shadows (under the knife) are another type of scene information that can be well-preserved by the gradient loss. Fig. 13 shows that our method with $\mathcal{L}_{\mathrm{all}}$ consistently reproduces high-quality images at different sampling rates, while the lack of the gradient loss will result in large variance when the sampling rate is low and overly blur as the sampling rate increases. Note that in this group of comparison, the maximum value of λ is set to 1.0 for the case of $\mathcal{L}_{\mathrm{d+1}}$. A slightly reduced value of λ can compensate the absence of the gradient loss and make the results more clear. Even so, some important structures mentioned above still disappear.

6.5 Impact of Branches

Our GradNet leverages two separate branches to encode the dense color images (and the features) and the sparse gradients, respectively. In Fig. 15 we prove that such a design choice is generally better than that of a single branch on the encoder side. For a fair comparison, we train these two deep networks with the same training set and hyper-parameters. Moreover, for the one-branch network, we double the channel number of feature maps in the encoder to ensure as much as possible that these two styles of networks have the same number of trainable parameters. Obviously, the two-branch style shows clear superiority over the one-branch style especially when the sampling rate is low. As the sampling rate increases, the gap of accuracy between these two styles decreases gently. We further observe that the two-branch style performs particularly well on very noisy inputs such as the BOOKSHELF scene and the DOOR scene. Visual comparison in Fig. 16 shows that the one-branch encoder occasionally smoothes out the scene details (e.g., the wood grains, the book edges and the wrinkles) as the extracted high-frequency features are weakened if sparse and low-energy gradients are mixed in one input branch with the dense and high-energy image colors.

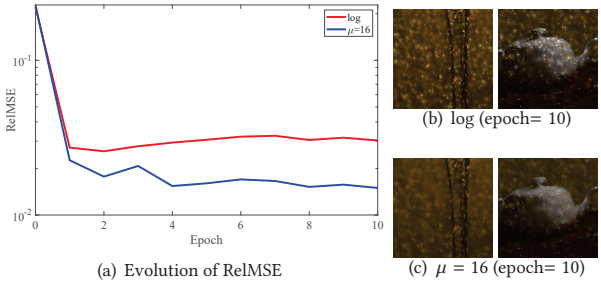
6.6 Impact of the μ-Law Transformation

In our current implementation, we compress the HDR inputs with the μ-law transformation defined in Eq. (6). We observe that this transformation generally converges faster that the conventional logarithmic transformation. This is evidenced in Fig. 17 where we show the evolution of RelMSE over the number of epochs for these



(a) One-branch (b) Two-branch (c) Reference

Fig. 16. Visual comparison between the one-branch style and the two-branch style on the decoder side. The two-branch style tends to preserve more image details than the one-branch style.



(a) Evolution of RelMSE (b) log (epoch= 10) (c) μ = 16 (epoch= 10)

Fig. 17. Convergence rate comparison between the μ-law transformation ($\mu = 16$) and the conventional logarithmic transformation (log). For both methods we do not include the first-order loss in training because the features are LDR and they may influence the accuracy.

two strategies. We compare the evolution of the RelMSE instead of the training (testing) losses, because the losses range differently for different HDR compression methods. The plots verify that the μ-law transformation ($\mu = 16$) consistently wins the conventional logarithmic transformation in terms of RelMSE. The visual comparison in the closeups further shows the benefit of the μ-law transformation. Note that we set $\mu = 16$ currently, and find it is a good trade-off between the convergence rate and the bias. The results tend to bias more if a larger μ is used.

6.7 Runtime Performance

We evaluate the runtime performance at an input resolution of 1280×720 on a PC with a 4.2 GHz Intel Core i7 processor and an NVIDIA GTX 1080Ti GPU. Our model takes 0.16s on average to reconstruct an image irrespective of the sampling rate and the noise level. In comparison, the conventional L_1 reconstruction typically takes 0.67s per image on the same GPU. Moreover, the memory cost of our model in prediction is less than 4GB.

7 CONCLUSION AND FUTURE WORK

We have proposed GradNet, the first unsupervised deep learning architecture that can be trained end-to-end to reconstruct high-quality images from noisy inputs in gradient-domain rendering. No clear data is needed in training, and our network still works well and outperforms state-of-the-art reconstruction methods both quantitatively and qualitatively. It also enjoys a fast speed by taking advantage of recent advancements in deep learning. The key of the proposed deep network is a multi-branch auto-encoder equipped with a joint loss function incorporating auxiliary feature buffers. By carefully designing the network architecture and the loss function, both low-frequency contents and high-frequency details of the noisy inputs are well-preserved while suppressing the disturbing MC noise as much as possible. Without requiring prohibitive computing time for generating ground-truth images, our training dataset is relatively easy to build and expand. We believe that deep learning systems in the future will mostly be of the unsupervised variety and we hope our work can promote the usage of unsupervised deep learning in rendering.

Several interesting future works would further boost the performance and robustness of the proposed method. First, it would be possible to add an adversarial loss [Goodfellow et al. 2014] in our GradNet to further enhance important local structures of reconstructed images. Second, our method is also likely to be extended to the temporal domain by introducing temporal finite differences [Manzi et al. 2016a] and adding a temporal loss term in our joint loss function. Finally, it is also interesting to combine our technology with adaptive sampling to improve the overall performance of rendering.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments. This work was supported in part by the National Key Research and Development Program of China (2018YFB1004901) and NSFC (61502223 and 61972194).

REFERENCES

Jonghee Back, Sung-Eui Yoon, and Bochang Moon. 2018. Feature Generation for Adaptive Gradient-Domain Path Tracing. *Computer Graphics Forum* 37, 7 (2018), 65–74.

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 (July 2017), 97:1–97:14.

Pablo Bauszat, Martin Eisemann, and Marcus Magnor. 2011. Guided Image Filtering for Interactive High-quality Global Illumination. *Computer Graphics Forum* (2011).

Pablo Bauszat, Victor Petitjean, and Elmar Eisemann. 2017. Gradient-domain Path Reusing. *ACM Trans. Graph.* 36, 6 (Nov. 2017), 229:1–229:9.

Pravin Bhat, Brian Curless, Michael Cohen, and C. Lawrence Zitnick. 2008. Fourier Analysis of the 2D Screened Poisson Equation for Gradient Domain Problems. In *Computer Vision – ECCV 2008*. Springer Berlin Heidelberg, 114–128.

Pravin Bhat, C. Lawrence Zitnick, Michael Cohen, and Brian Curless. 2010. GradientShop: A Gradient-domain Optimization Framework for Image and Video Filtering. *ACM Trans. Graph.* 29, 2 (April 2010), 10:1–10:14.

Benedikt Bitterli. 2016. Rendering resources. https://benedikt-bitterli.me/resources/.

Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, Jose A. Iglesias-Guitian, David Adler, Kenny Mitchell, Wojciech Jarosz, and Jan Novak. 2016. Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings. *Computer Graphics Forum* (2016).

Malik Boughida and Tamy Boubekeur. 2017. Bayesian Collaborative Denoising for Monte Carlo Rendering. *Computer Graphics Forum (Proc. EGSR 2017)* 36, 4 (2017), 137–153.

A. Buades, B. Coll, and J. Morel. 2005. A Review of Image Denoising Algorithms, with a New One. *Multiscale Modeling & Simulation* 4, 2 (2005), 490–530.

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 (July 2017), 98:1–98:12.

Mauricio Delbracio, Pablo Musé, Antoni Buades, Julien Chauvier, Nicholas Phelps, and Jean-Michel Morel. 2014. Boosting Monte Carlo Rendering by Ray Histogram Fusion. *ACM Trans. Graph.* 33, 1 (Feb. 2014), 8:1–8:15. https://doi.org/10.1145/2532708

Qingnan Fan, Jiaolong Yang, David Wipf, Baoquan Chen, and Xin Tong. 2018. Image Smoothing via Unsupervised Learning. *ACM Trans. Graph.* 37, 6 (Dec. 2018), 259:1–259:14.

Ian J. Goodfellow, Jean Pouget-abadie, Mehdi Mirza, Bing Xu, David Warde-farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems (NIPS)*.

Adrien Gruson, Binh-Son Hua, Nicolas Vibert, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2018. Gradient-domain Volumetric Photon Density Estimation. *ACM Transactions on Graphics* (2018).

Saerom Ha, Sojin Oh, Jonghee Back, Sung-Eui Yoon, and Bochang Moon. 2019. Gradient Outlier Removal for Gradient-Domain Path Tracing. In *Eurographics 2019*.

K. He, X. Zhang, S. Ren, and J. Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*. 1026–1034.

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

Pedro Hermosilla, Sebastian Maisch, Tobias Ritschel, and Timo Ropinski. 2018. Deep-learning the Latent Space of Light Transport. *CoRR* abs/1811.04756 (2018). arXiv:1811.04756 http://arxiv.org/abs/1811.04756

G. E. Hinton and R. R. Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *Science* 313, 5786 (2006), 504–507. https://doi.org/10.1126/science.1127647

Binh-Son Hua, Adrien Gruson, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2017. Gradient-Domain Photon Density Estimation. *Eurographics* (2017).

Binh-Son Hua, Adrien Gruson, Victor Petitjean, Matthias Zwicker, Derek Nowrouzezahrai, Elmar Eisemann, and Toshiya Hachisuka. 2019. A Survey on Gradient-Domain Rendering. *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)* 38, 2 (2019).

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

Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. 2015. A Machine Learning Approach for Filtering Monte Carlo Noise. *ACM Trans. Graph.* 34, 4 (July 2015), 122:1–122:12.

Nima Khademi Kalantari and Ravi Ramamoorthi. 2017. Deep High Dynamic Range Imaging of Dynamic Scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)* 36, 4 (2017).

Nima Khademi Kalantari and Pradeep Sen. 2013. Removing the Noise in Monte Carlo Rendering with General Image Denoising Algorithms. *Computer Graphics Forum* 32, 2pt1 (2013), 93–102.

Simon Kallweit, Thomas Müller, Brian Mcwilliams, Markus Gross, and Jan Novák. 2017. Deep Scattering: Rendering Atmospheric Clouds with Radiance-predicting Neural Networks. *ACM Trans. Graph.* 36, 6 (Nov. 2017), 231:1–231:11.

Kaizhang Kang, Zimin Chen, Jiaping Wang, Kun Zhou, and Hongzhi Wu. 2018. Efficient Reflectance Capture Using an Autoencoder. *ACM Trans. Graph.* 37, 4 (July 2018), 127:1–127:10.

A. Keller, L. Fascione, M. Fajardo, I. Georgiev, P. Christensen, J. Hanika, C. Eisenacher, and G. Nichols. 2015. The Path Tracing Revolution in the Movie Industry. In *ACM SIGGRAPH 2015 Courses (SIGGRAPH '15)*. 24:1–24:7.

Alexander Keller, Jaroslav Křivánek, Jan Novák, Anton Kaplanyan, and Marco Salvi. 2018. Machine Learning and Rendering. In *ACM SIGGRAPH 2018 Courses*. 19:1–19:2.

Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. 2019. Deep Convolutional Reconstruction for Gradient-domain Rendering. *ACM Trans. Graph.* 38, 4 (July 2019), 126:1–126:12.

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-domain Path Tracing. *ACM Trans. Graph.* 34, 4 (July 2015), 123:1–123:13.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2015).

Alexandr Kuznetsov, Nima Khademi Kalantari, and Ravi Ramamoorthi. 2018. Deep Adaptive Sampling for Low Sample Count Rendering. *Computer Graphics Forum* 37 (2018), 35–44.

Jaroslav Křivánek, Christophe Chevallier, Vladimir Koylazov, Ondřej Karlíik, Henrik Wann Jensen, and Thomas Ludwig. 2018. Realistic Rendering in Architecture and Product Visualization. In *ACM SIGGRAPH 2018 Courses (SIGGRAPH '18)*. Article 10, 5 pages.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521 (2015), 436. https://doi.org/10.1038/nature14539

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-domain Metropolis Light Transport. *ACM Trans. Graph.* 32, 4 (July 2013), 95:1–95:12.

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 (Proceedings of Machine Learning Research)*, Vol. 80. PMLR, 2965–2974.

Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. 2012. SURE-based Optimization for Adaptive Sampling and Reconstruction. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2012)* 31, 6 (November 2012), 186:1–186:9.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

Marco Manzi, Markus Kettunen, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-Domain Bidirectional Path Tracing. In *Eurographics Symposium on Rendering*, Jaakko Lehtinen and Derek Nowrouzezahrai (Eds.). The Eurographics Association.

Marco Manzi, Markus Kettunen, Frédo Durand, Matthias Zwicker, and Jaakko Lehtinen. 2016a. Temporal Gradient-domain Path Tracing. *ACM Trans. Graph.* 35, 6 (Nov. 2016), 246:1–246:9.

Marco Manzi, Fabrice Rousselle, Markus Kettunen, Jaakko Lehtinen, and Matthias Zwicker. 2014. Improved Sampling for Gradient-domain Metropolis Light Transport. *ACM Trans. Graph.* 33, 6 (Nov. 2014), 178:1–178:12.

Marco Manzi, Delio Vicini, and Matthias Zwicker. 2016b. Regularizing Image Reconstruction for Gradient-Domain Rendering with Feature Patches. *Computer Graphics Forum* 35, 2 (2016), 263–273.

Bochang Moon, Nathan Carr, and Sung-Eui Yoon. 2014. Adaptive Rendering Based on Weighted Local Regression. *ACM Trans. Graph.* 33, 5 (Sept. 2014), 170:1–170:14.

Bochang Moon, Steven McDonagh, Kenny Mitchell, and Markus Gross. 2016. Adaptive Polynomial Rendering. *ACM Trans. Graph.* 35, 4 (July 2016), 40:1–40:10.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2018. Neural Importance Sampling. *CoRR* abs/1808.03856 (2018). arXiv:1808.03856 http://arxiv.org/abs/1808.03856

Oliver Nalbach, Elena Arabadzhiyska, Dushyant Mehta, Hans-Peter Seidel, and Tobias Ritschel. 2017. Deep Shading: Convolutional Neural Networks for Screen-Space Shading. 36, 4 (2017).

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. 234–241.

Fabrice Rousselle, Wojciech Jarosz, and Jan Novák. 2016. Image-space Control Variates for Rendering. *ACM Trans. Graph.* 35, 6 (Nov. 2016), 169:1–169:12.

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2012. Adaptive Rendering with Non-local Means Filtering. *ACM Trans. Graph.* 31, 6 (Nov. 2012), 195:1–195:11.

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

Pradeep Sen and Soheil Darabi. 2012. On Filtering the Noise from the Random Parameters in Monte Carlo Rendering. *ACM Trans. Graph.* 31, 3 (May 2012), 18:1–18:15.

Pradeep Sen, Matthias Zwicker, Fabrice Rousselle, Sung-Eui Yoon, and Nima Khademi Kalantari. 2015. Denoising Your Monte Carlo Renders: Recent Advances in Image-space Adaptive Sampling and Reconstruction. In *ACM SIGGRAPH 2015 Courses (SIGGRAPH '15)*. 11:1–11:255.

Weilun Sun, Xin Sun, Nathan A. Carr, Derek Nowrouzezahrai, and Ravi Ramamoorthi. 2017. Gradient-Domain Vertex Connection and Merging. In *Eurographics Symposium on Rendering*. The Eurographics Association.

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 Trans. Graph.* 37, 4 (July 2018), 124:1–124:15.

H. Zhao, O. Gallo, I. Frosio, and J. Kautz. 2017. Loss Functions for Image Restoration With Neural Networks. *IEEE Transactions on Computational Imaging* 3, 1 (2017), 47–57.

Quan Zheng and Matthias Zwicker. 2019. Learning to importance sample in primary sample space. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library.

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and Sung-Eui Yoon. 2015. Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)* 34, 2 (may 2015), 667–681.