DOI: 10.1111/cgf.14337
Eurographics Symposium on Rendering 2021
A. Bousseau and M. McGuire
(Guest Editors)

Volume 40 (2021), Number 4

Deep Compositional Denoising for High-quality Monte Carlo Rendering

Xianyao Zhang[†1,2], Marco Manzi[2], Thijs Vogels[3], Henrik Dahlberg[4], Markus Gross[1,2] and Marios Papas[†2]

[1]ETH Zürich, Switzerland
[2]DisneyResearch|Studios, Switzerland
[3]EPFL, Switzerland
[4]Industrial Light & Magic, United Kingdom

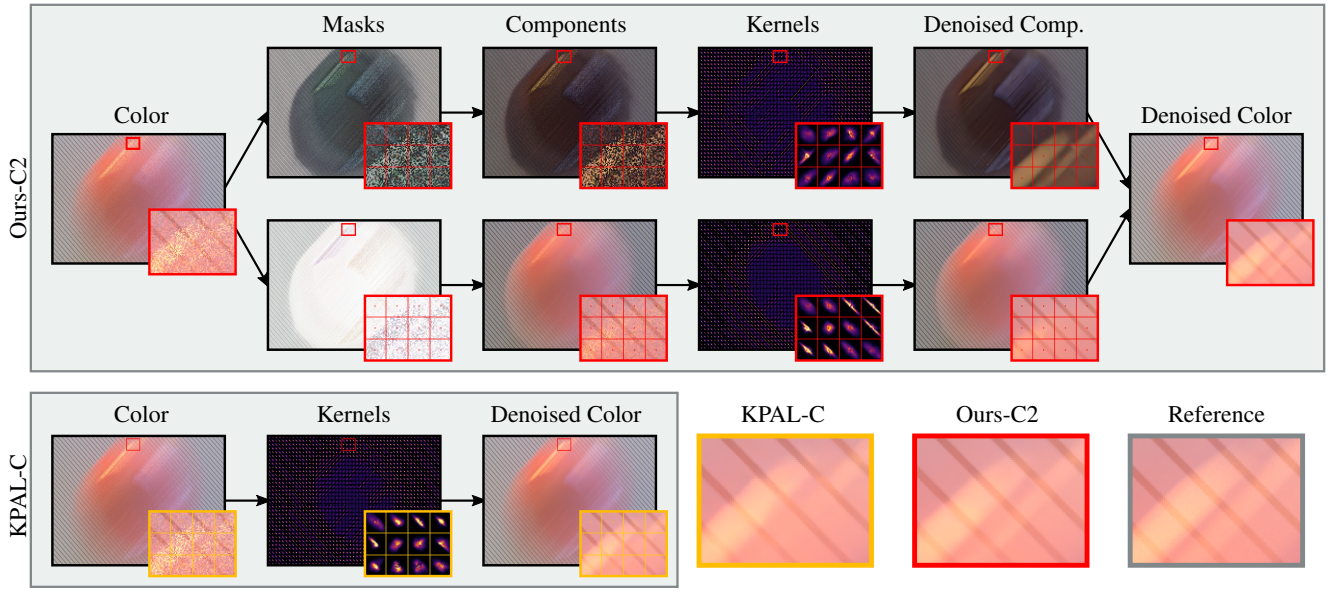


Figure 1: ***(Top)*** *Our proposed approach first predicts from a noisy color image masks that are used to decompose the color image into additive components. For each component, a denoiser independently predicts per-pixel filter kernels, visualized at each tile's central pixel. Each component is separately denoised by its corresponding kernels, and the final denoised image is produced by summing all components. For clarity, we demonstrate our approach only with a two-way decomposition ("Ours-C2") and large single-scale kernels. More powerful decomposition and reconstruction approaches are described in the text.* ***(Bottom)*** *We contrast our approach with the existing KPAL-C [VRM* 18] that does not decompose the signal before denoising. Without decomposition, the kernels cannot adapt as well to each underlying signal's structure, leading to detail loss compared to Ours-C2.*

Abstract

We propose a deep-learning method for automatically decomposing noisy Monte Carlo renderings into components that kernel-predicting denoisers can denoise more effectively. In our model, a neural decomposition module learns to predict noisy components and corresponding feature maps, which are consecutively reconstructed by a denoising module. The components are predicted based on statistics aggregated at the pixel level by the renderer. Denoising these components individually allows the use of per-component kernels that adapt to each component's noisy signal characteristics. Experimentally, we show that the proposed decomposition module consistently improves the denoising quality of current state-of-the-art kernel-predicting denoisers on large-scale academic and production datasets.

CCS Concepts

• ***Computing methodologies*** → ***Ray tracing; Neural networks;***

† xianyao.zhang@inf.ethz.ch, marios.papas@disneyresearch.com

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

1. Introduction

Animated movies and visual effects rely on Monte Carlo rendering [Kaj86, PJH16] to simulate light transport [FHF*17, JWB*19]. While Monte Carlo algorithms are ubiquitous due to their generality, they typically incur high computational costs. A Monte Carlo renderer estimates pixel colors using numerical integration over sampled light paths, and errors manifest as pixel-wise noise in rendered images. Achieving high-quality renderings with low error can require sampling a computationally prohibitively expensive number of light paths.

A crucial ingredient in reducing the number of samples and making Monte Carlo rendering affordable for movie production is *denoising* [ZJL*15, JWB*19]. In this context, denoising is a post process that turns noisy images from a renderer into clean images. For high-quality renderings, which is the focus of this work, the computation overhead of denoising is negligible compared to the computational cost of rendering. Recently, deep neural network have been adopted for this task and dramatically improved the denoising quality. Our work builds upon such work that predicts per-pixel filter-kernels to denoise a noisy image [BVM*17, VRM*18].

We combine a pixel-based *neural decomposition module* with a kernel-predicting denoiser. The decomposition module is a neural network that decomposes a noisy rendered image into additive components that are denoised separately. The intuition behind this approach is to learn to decompose a signal into components that are simpler to denoise than the full image. Figure 1 illustrates this in a 2-way decomposition where two orthogonally aligned signals with different noise characteristics overlap. Separating the two signals into components and denoising them separately allows the use of specialized kernels that adapt to the individual sub-signals. This often leads to better results than when one set of kernels have to reconstruct the whole input signal.

In addition to decomposing noisy color images, our decomposition module can also be used to further decompose *user-defined* components [ZRJ*15] that are commonly outputted by renderers instead of the full color to improve denoising quality [RMZ13, BVM*17, VRM*18, XZW*19]. Further, our decomposition module can either be trained jointly with a kernel-predicting denoiser or trained separately to work with a pre-trained denoiser. In all aforementioned cases we observe quality benefits compared to the corresponding baseline methods that do not use our neural decomposition module. Our proposed approach comes at the cost of multiple denoising and decomposition passes, but this added cost is minuscule compared to the resulting computation savings for final-quality renderings. It is worth stressing that a benefit of our approach is that it does not not require any modifications on the data of existing pixel-based denoising methods [VRM*18]. This property makes our method very practical. It can serve as a drop-in replacement for current production denoisers, *i.e.*, without changing current data generation and data loading pipelines for final-quality renderings.

In this work, we focus on high-quality single-frame denoising without incorporating temporal information from neighboring frames. It is important to note that single-frame denoisers are helpful in production, especially in scenarios where temporal information is unreliable, *e.g.*, in the presence of fast complex motion, or even unavailable, *e.g.*, during look development or lighting. We leave promising temporal extensions of our method as future work.

We evaluate our method on large-scale academic and production datasets and show consistent improvements over state-of-the-art neural pixel-based denoising methods, yielding significant savings in sampling budget required to reach the same quality. Compared to sample-based neural decomposition [MH20] our method performs similarly for preview-quality denoising and scales better for high-quality scenarios in terms of performance and accuracy.

2. Related work

Non-neural denoising. Traditional kernel-based denoisers compute a pixel's color as a weighted average of the noisy pixel estimates of the surrounding pixels. Ideal denoising weights (kernels) should yield maximal variance reduction with minimal bias increase in the final image. Some algorithms produce kernels based on heuristics for pixel similarity [RKZ12, RMZ13, MJL*13, BCM05]. Later works start from heuristics-based kernels and fit higher-order regression models on image patches to increase expressivity [BRM*16, MCY14]. In both cases, auxiliary feature buffers such as surface normals, surface albedo, and depth [RMZ13] are often used to reveal the relationships between noisy pixels. We refer the reader to the report by Zwicker et al. [ZJL*15] for a comprehensive discussion on non-neural denoising and reconstruction methods.

User-defined decompositions. To improve their performance, denoisers may operate on different components of the color image separately. This can help to retain fine details in the image that would get lost when operating on the full image directly. Such components can be directly produced by the renderer. Two of the most popular choices are to separate according to the first non-delta interaction into *specular* and *diffuse* components, or separating the *direct* and *indirect* illumination parts. Path-space decomposition [ZRJ*15] methods aggregate paths with more complex common user-defined prefixes into separate image components. Our proposed learned decomposition is not intended to replace any readily available user-defined decompositions like diffuse–specular and direct–indirect. Rather, our approach should be used in combination with them by further decomposing these components for additional quality benefits.

Deep pixel-based denoising. After their success in natural image denoising [MSY16, ZZC*17, MBC*18, LZZ*18, XPG*19], deep learning methods have become a popular choice for Monte Carlo denoising. Kalantari et al. [KBS15] use a multi-layer perceptron to learn optimal parameters of traditional denoising methods like bilateral filtering [TM98] and non-local means [BCM05]. Bako et al. [BVM*17] introduce the *kernel-predicting* convolutional network (KPCN), which employs per-pixel kernels predicted by a neural network to filter noisy pixel estimates. A later extension to KPCN [VRM*18] incorporates temporal information, a more efficient multi-scale architecture, and the asymmetric loss, which allows user control of the denoising strength. The denoiser module in our work is based on the KPCN architecture [BVM*17], along with the multi-scale efficiency optimizations proposed by Vogels et

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

al. [VRM*18], but we do not employ the asymmetric loss in this work.

Direct-predicting neural denoisers directly output the color values of each pixel in the resulting image without requiring kernel applications. These denoisers typically run faster than kernel-predicting methods but are less robust and converge slower during training [BVM*17, VRM*18]. Notably, Chaitanya et al. [CKS*17] use a recurrent neural network to denoise low sample-count images while also leveraging temporal information from previous frames, and Xu et al. [XZW*19] use adversarial training [GPAM*14] for Monte Carlo denoising.

Deep sample-based denoising. With increasing computation power, deep-learning-based denoising methods that work with sample-based data have recently surfaced. They operate directly on the noisy samples instead of statistics (*e.g.*, mean and variance) aggregated at the pixel level, and are thus only practical in low-sample-count regimes. Gharbi et al. [GLA*19] propose a network that distributes each sample's radiance over neighboring pixels, improving denoising quality at low sample count (preview quality) compared to pixel-based methods. Lin et al. [LWY*21] propose a method that utilizes information at a sub-sample (*i.e.*, sub-paths of a light path) level, which is even more costly and is currently out of reach for final-quality scenarios.

Munkberg and Hasselgren [MH20] propose to learn to partition each sample into different layers and denoise the resulting layers instead of the individual samples. This approach mitigates the linear increase in the number of kernel predictions and applications introduced by purely sample-based methods [GLA*19] while preserving their quality. Its resulting learned layers are analogous to our learned components, but they are created using *sample-based* information while our components are predicted from only averaged *pixel-based* statistics.

Concurrent to our work, Cho et al. [CHY21] use an additional contrastive loss to weakly supervise path-space features of individual samples based on their corresponding pixel colors. This addresses challenges with information flow between loss on reconstructed pixel values and high-dimensional sparse features of individual samples. Further, Işik et al. [IMF*21] propose another method to improve the reconstruction quality from individual samples by indirectly predicting kernels based on the affinity of learned features.

3. Methodology

Our work builds on the kernel-predicting denoiser from [VRM*18] by additionally learning a decomposition of its input (a noisy color image) from per-pixel statistics. The resulting components will be denoised individually in multiple passes. In this section, we formally define the task of kernel prediction denoising and color decomposition, based on which we describe our proposed method in detail. We follow the notation of KPCN [BVM*17, VRM*18].

3.1. Pixel-based denoising

When operating on noisy rendered pixel color estimates $\mathbf{c}$, the denoiser g can take advantage of *feature maps* $\mathbf{f}$ to estimate the reference image $\mathbf{r}$, *i.e.*, $\mathbf{d} = g(\mathbf{c},\mathbf{f}) \approx \mathbf{r}$. The feature maps can be either byproducts of the rendering or learned from a network. Rendered features include surface normals and reflectance (albedo) texture values which often correlate well with the reference image, contain less noise than color estimates and are inexpensive to compute. Additionally, variance estimates of the pixel color and features can also be included in the rendered features. For notation simplicity, we define $\mathbf{x} = [\mathbf{c},\mathbf{f}]$ as the concatenation of noisy color and feature maps.

While traditional methods employ hand-crafted denoising functions [BCM05, DFKE06, RMZ13, BRM*16], we follow recent deep learning methods [CKS*17, BVM*17, VRM*18]. We parameterize the denoiser as a convolutional neural network whose parameters θ are optimized to minimize the average pixel-wise loss over a dataset of noisy–clean image pairs. As a training loss, we opt for symmetric mean absolute percentage error (SMAPE) [VRM*18] due to its stable behavior with high-dynamic-range (HDR) images. For a pixel p we define our training loss $\ell(\cdot,\cdot)$ as:

$$\ell(\mathbf{r}_p,\mathbf{d}_p) = \frac{|\mathbf{r}_p - \mathbf{d}_p|}{|\mathbf{r}_p| + |\mathbf{d}_p| + \varepsilon}, \qquad \varepsilon = 10^{-2}. \tag{1}$$

In this work, we adopt kernel prediction [BVM*17, VRM*18], in which the denoised color of pixel p is computed as a weighted sum of its neighbors in an $l \times l$ neighborhood $\mathcal{N}(p)$:

$$\mathbf{d}_p = g_p(\mathbf{x};\theta) = \sum_{q\in\mathcal{N}(p)} w_{pq}(\mathbf{x};\theta)\,\mathbf{c}_q. \tag{2}$$

The weights w_{pq} are predicted by a neural network based on the noisy image and feature maps. They are the output of a *softmax* layer applied to the raw network output $\mathbf{z}$:

$$w_{pq} = \frac{\exp(\mathbf{z}_{pq})}{\sum_{q'\in\mathcal{N}(p)}\exp(\mathbf{z}_{pq'})}, \tag{3}$$

which ensures that for each pixel p, the set of weights $\{w_{pq}\}$ is convex: they are non-negative and sum to unity.

3.2. Compositional denoising

Instead of denoising the noisy color directly, we can alternatively decompose the image into components and denoise each component separately. In this case, the sum of denoised components constitutes the final denoising result. In the context of kernel-predicting denoising, such a divide-and-conquer approach can often lead to significant improvements in quality if the input color image is composed of complex, superimposed signals that can be decomposed into simpler-to-denoise signals [ZRJ*15, VRM*18, BVM*17].

We define the additive decomposition of a noisy color image $\mathbf{c}$ as a set of non-negative image components $\{\mathbf{c}^{(k)}\}_{k=1}^{K}$ that sums up to $\mathbf{c}$. This decomposition into components can equivalently be defined by a set of masks $\{\mathbf{m}^{(k)}\}_{k=1}^{K}$ such that $\mathbf{c}^{(k)} = \mathbf{c}\odot\mathbf{m}^{(k)}$ where $\odot$ is the per-element multiplication operator. To ensure non-negativity of the components, the masks also follow the convex constraint for each pixel p:

$$\sum_{k=1}^{K}\mathbf{m}_p^{(k)} = 1 \;\text{ and }\; 0 \le \mathbf{m}_p^{(k)} \le 1, \forall k = 1\ldots K. \tag{4}$$

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

Given a decomposition, we denoise each component as in Equation (2), and sum the results:

$$\mathbf{d}_p = g_p(\mathbf{x};\theta) = \sum_{k=1}^{K} g_p^{(k)}(\mathbf{x}^{(k)};\theta^{(k)}), \quad \mathbf{x}^{(k)} = \left[\mathbf{c}^{(k)},\mathbf{f}^{(k)}\right]. \tag{5}$$

Components can either be denoised with separate denoisers $g^{(k)}$, or one denoiser can be used for all components. Each component $\mathbf{c}^{(k)}$ can be accompanied by component-specific feature maps $\mathbf{f}^{(k)}$, which can enhance the denoising quality. This is especially useful when the components are very different, *e.g.*, when one component contains the reflection and the other refraction from a dielectric surface. Note that the per-component feature maps do not always have physical meanings like normals or albedo. They can be predicted by a neural network (as in our case) with the goal of improving denoising quality.

A consequence of compositional denoising is that the denoiser is free to predict different kernels for each components. Even if the component denoisers are the same, they receive different inputs, and thus they can specialize the predicted kernels. This allows the denoiser module to adapt the kernels to each component's noise and frequency characteristics (recall Figure 1).

3.3. Learned decomposition for kernel prediction

We now describe our proposed method, which learns beneficial decompositions for compositional denoising from per-pixel information only. In the spirit of the modular pipeline described in [VRM*18], we extend the kernel-predicting denoiser with a *decomposition module*. As the kernel-predicting denoiser is only one part of our modular architecture, we refer to it as the *denoising module*. We will first introduce our pipeline in the simple case of $K = 2$ components and then extend the approach to $K \geq 2$ components via a hierarchical decomposition scheme.

Two-way decomposition. The decomposition module consists of a trainable decomposition function followed by a mask multiplication. The decomposition function h is a convolutional neural network with trainable parameters ϕ and takes as input an image $\mathbf{c}$ and a feature map $\mathbf{f}$. The feature map is the output of our *feature encoder* which is applied right before the first decomposition to encode the features from the renderer. Our decomposition module outputs a mask $\mathbf{m}$ containing values in the range $[0,1]$ and two feature maps (one for each output component):

$$h(\mathbf{c},\mathbf{f};\phi) = \{\mathbf{m},\mathbf{f}^{(0)},\mathbf{f}^{(1)}\} \tag{6}$$

Note that the mask $\mathbf{m}$ has the same number of channels as the input image $\mathbf{c}$. In other words, both the mask and the input contain RGB triplets. The number of channels in the feature maps is a hyperparameter of our pipeline.

Element-wise multiplication of the mask (and its complement) with the input image splits the input into two components,

$$\mathbf{c}^{(0)} = \mathbf{m} \odot \mathbf{c}$$
$$\mathbf{c}^{(1)} = (1-\mathbf{m}) \odot \mathbf{c}$$

where by construction $\mathbf{c}^{(0)} + \mathbf{c}^{(1)} = \mathbf{c}$. The two component–feature pairs $\{\mathbf{c}^{(0)},\mathbf{f}^{(0)}\}$ and $\{\mathbf{c}^{(1)},\mathbf{f}^{(1)}\}$ are the output of the decomposition module.

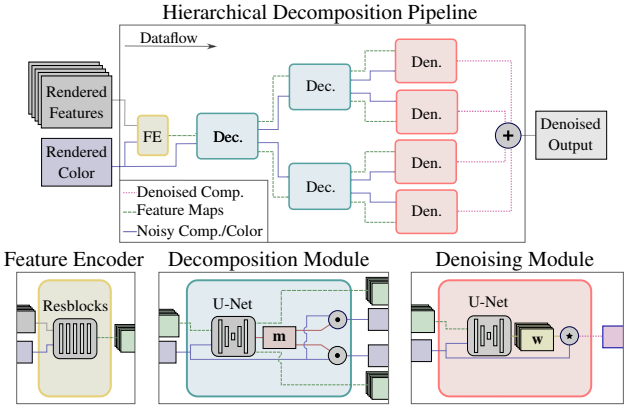


Figure 2: *Hierarchical 4-way decomposition. Starting from the left, the decomposition module receives the noisy color accompanied by rendered features (*e.g.*, albedo, normal) that are converted by a feature encoder into learned feature maps. The decomposition module produces two component–feature pairs. The component colors result from element-wise multiplication $\odot$ with the predicted mask $\mathbf{m}$ (and its complement), while the component feature maps are directly predicted by the decomposition module. After two levels of decomposition, we have four noisy components that are processed by a denoising module that is trained end-to-end with our decomposition module. The denoising module predicts per-pixel kernels $\mathbf{w}$ that are convolved $\circledast$ with the noisy components. The final image is the element-wise sum $\oplus$ of all denoised component images.*

These pairs are separately passed to the denoising modules for further processing. In our architecture, the denoising modules used for both components share the same weights, *i.e.*, $\theta^{(1)} = \theta^{(2)}$. The final image is produced by adding all denoised components (see Equation (5)).

The benefit of producing per-component feature maps alongside the components is twofold. First, it accompanies component colors with relevant features, which helps with their denoising. Second, when the decomposition module is trained jointly with the denoising module, the feature maps enable information sharing between the modules.

***K*-way decomposition.** To decompose the color into more than two components, we exploit our pipeline's modular architecture by concatenating decomposition modules *hierarchically* as illustrated in Figure 2. To achieve this, we make sure all the inputs and outputs of the decomposition modules are in a common representation space. The feature encoder projects the raw feature maps from the renderer into this representation space. This gives the network the flexibility to choose an appropriate feature representation for the communication between its trainable modules instead of being restricted to the format and dimensionality of features produced by the renderer.

We can produce an arbitrary number of components ($K \geq 2$) by

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

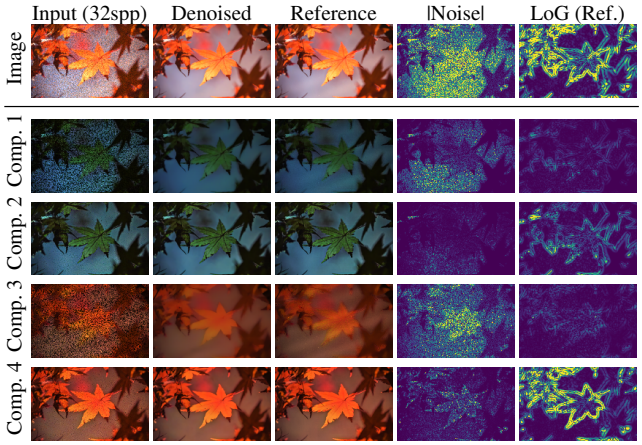


Figure 3: *Our predicted components on a rendered painting. The first row corresponds to the full image, while the rows below correspond to components generated by our decomposition module. In this case, the decomposition module creates components with either high noise (4th column, bright is large), or much high-frequency detail (5th column, reference image filtered with Laplacian of Gaussian). This allows the kernel-predicting denoising module to use large kernels in noisy components without losing much detail, and to use fine-grained kernels in other components to preserve fine detail.*

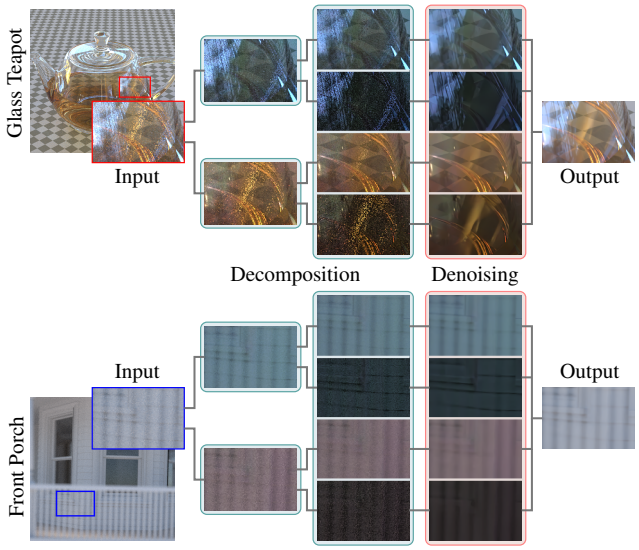


Figure 4: *Two decomposition examples using a 4-way hierarchical decomposition. From left to right: The color image is first decomposed into two components, which are then further decomposed, yielding a total of 4 components. Every component is denoised separately and finally summed to produce the output. Note that the same decomposition module is used for all three decomposition operations in the hierarchy.*

allowing decomposition modules in the hierarchy to be followed by modules with different types (*e.g.*, a denoising and a decomposition module). Even though we do not exercise this option, we provide a formal definition of the decomposition strategy that allows for an arbitrary number of components in Supplemental Material. In our experiments, the number of components and the resulting architecture are pre-defined before training. Note that the final result is independent of the denoising order of these K components since we apply the same denoising module on each one.

4. Decomposition analysis

Starting our analysis from our 4-way decomposition, we visualize the high-frequency content of the components reference signal and the noise magnitude in the components in Figure 3. To approximate the component reference signal, we render the same noisy image 256 times with different seeds, process these images with the same decomposition module, and average the resulting component inputs. We can then measure each component's noise magnitude by taking the absolute difference between the noisy input and the corresponding component reference. The high-frequency spatial content of the component reference signal is extracted with a Laplacian of Gaussian (LoG) filter.

We observe that components with higher noise magnitude (1 and 3) have less high-frequency content in their reference signal, and components with sharper details (2 and 4) are less noisy. Additionally, when comparing components 2 and 4, we observe that the sharp details in one component are not necessarily sharper in the other. We believe this suggests that our decomposition module can separate overlapping noisy signals to some degree, based on their inferred spatial frequency content and noise characteristics.

We demonstrate another example of the separation of different signals from our learned decomposition module in Figure 1, where the inset region contains overlapping noisy signals. In this example, reflected motion-blurred highlights are superimposed over a high-frequency background texture. Our decomposition module can produce components that resemble a separation of the signals. This separation allows the denoising pass over each component to utilize kernels that adapt to each component's content.

In Figure 4, we provide two more examples for a hierarchical 4-way decomposition that exemplify how our neural decomposition can disentangle complex overlapping signals. The Glass Teapot inset contains various high-frequency reflections that overlap with a textured floor. The first step of the decomposition mainly separates signals according to their hue. This leads to a clear separation of the different reflections. The second step of the decomposition appears to be separating some of the reflections from the floor.

The Front Porch inset contains overlapping signals due to depth of field. The lattice in the foreground leads to a vertically aligned out-of-focus structure that overlaps with the in-focus wooden facade that mainly contains horizontal structures. The first decomposition step separates again according to the color hue, but more interestingly also separates foreground and background, as most of the in-focus background ends up in the blue component. The second decomposition step further separates background and foreground and pushes noise into the darker component.

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

Method	Description
Ours	Proposed method, using learned decomposition from pixel-based statistics and a kernel-predicting denoiser
Ours-C4	Ours decomposing noisy **color** into 4 components
Ours-D4S4	Ours decomposing noisy **diffuse** and **specular** components into 4 learned sub-components each
Ours-d4i4	Ours decomposing noisy **direct** and **indirect** (illumination) components into 4 learned sub-components each
KPAL	Kernel-predicting denoiser (using the single-frame 3-scale variant) [VRM*18] on pixel-based data
KPAL-C	KPAL processing noisy **color**
KPAL-DS	KPAL processing noisy **diffuse** and **specular** components
KPAL-di	KPAL processing noisy **direct** and **indirect** (illumination) components
DnGAN	Direct-predicting denoising method with adversarial training [XZW*19] on pixel-based data
S-LD	Learned decomposition from sample-based data [MH20]
S-LD-C4	S-LD creating 4 learned components from input **color**

Table 1: *Method name abbreviations used in the results. Note that this list is not comprehensive, but other shorthands such as Ours-C2 and S-LD-C2 can be interpreted similarly.*

5. Experimental setup

5.1. Data

Datasets. Most of our experiments are conducted on academic datasets, which are generated from our own dataset generator from publicly available scene assets [Bit16,MKD*16,KMA*15,KHL19] with the Mitsuba renderer [Jak10]. We also present results from datasets rendered with the Hyperion renderer [BAC*18] to illustrate the usefulness of our method in a production environment. A summary of our training, validation and testing datasets can be found in Table 8.

Network input. As input features from the renderer, we use the *normal* vector, texture *albedo* and *depth* values collected from the first non-delta interaction point of the sample paths. We apply a log-like transform, $\eta(\mathbf{c}) = \log(1+\mathbf{c})$, on the noisy color images to prevent excessive input values due to the high dynamic range [BVM*17]. The predicted kernels are still applied on the linear RGB values instead of log-transformed ones. For pixel-based methods, we also compute single-channel per-pixel variance estimates for each color and feature buffer. To avoid excessive variance values, we use the relative variance of the noisy color values and rendered features [VRM*18].

In total, we use 14 floats *per pixel* (3 for color, 3 for normal, 3 for albedo, 1 for depth and 4 for variance estimates) as the input to pixel-based methods. Accordingly, we provide 10 floats *per sample* for sample-based methods which do not take variance estimates as input.

5.2. Implementation

We compare our methods with two main baselines, namely KPAL [VRM*18] among the pixel-based methods and the sample-based decomposition method (S-LD) of Munkberg and Hasselgren [MH20]. See Table 1 for description of the shorthands. To ensure fairness of comparison, we implement our proposed method differently when comparing with these two baselines, whose U-Net backbones and kernel reconstruction schemes are different, as described below.

5.2.1. Comparing with KPAL

Our single-frame KPAL implementation uses a 3-scale U-Net and multi-scale reconstruction with 3 scales and 5×5 kernels at each scale, following the suggestions of the KPAL work [VRM*18]. Two residual blocks each with two convolution layers, two ReLU activations and a residual connection [HZRS16] are used on each scale of the U-Net.

In the implementation of our method for comparing with KPAL, we use a U-Net with half the number of trainable parameters for both decomposition and denoising modules. Parameter sharing between the same type of modules ensures that our models have a similar number of trainable parameters as KPAL, regardless of the number of learned components.

Both KPAL and our models include the feature encoder, composed of two residual blocks, to process the rendered input features.

5.2.2. Comparing with S-LD

We closely follow the open-source implementation of S-LD [MH20] which utilizes a 5-scale U-Net and single-scale reconstruction with kernels of size 17×17.

The original method from Munkberg and Hasselgren [MH20] uses a large U-Net for sample partitioning (decomposition module) but only a few convolution layers for kernel prediction (denoising module). This architecture can be seen as potentially optimized for speed because the denoising module needs to be used for each component. Through experiments we discovered that the denoising quality of this method can be improved significantly by using a larger denoising module and shrinking the size of the decomposition module (see Supplemental Material for details). We thus use this improved architecture of the S-LD method in our comparisons, which results in two equally sized U-Nets that sum up to the number of parameters as in the original method.

When comparing our method and KPAL against S-LD, we use a similar 5-scale U-Net architecture with comparable total number of trainable parameters.

5.3. Training

We use the Adam [KB14] optimizer to train all models to minimize SMAPE loss (Equation (1)). Particularly, our models are trained end-to-end to minimize the final denoising error.

During training, we take random 128×128 crops from the images and reject less interesting patches following a similar scheme as Bako et al. [BVM*17]. We use a learning rate of 10^{-4} to train

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

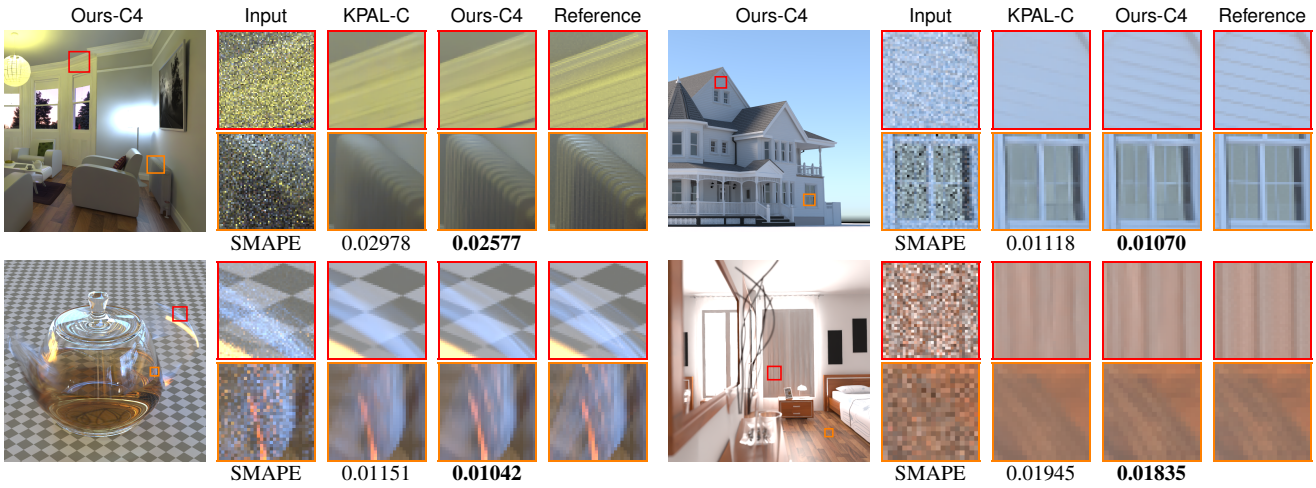


Figure 5: *The effect of learned decomposition for color-only denoising. Our method improves the reconstruction of details compared to the baseline KPAL-C,* e.g.*, in regions with changing surface normals, objects behind glass, motion blur, reflections, and fine texture. The exposure of some crops is adjusted to better illustrate the results.*

KPAL and our methods. For comparison with sample-based methods, we follow the training scheme used for S-LD [MH20] in order to make a fair comparison.

To prevent the component masks from collapsing to unity (all 1) or blank (all 0), we add a regularization term to our training loss at the beginning, which encourages a uniform decomposition in early training steps to stabilize training. More details of the training schemes can be found in Supplemental Material.

6. Results

We extensively evaluate the denoising capability of our proposed method in different scenarios. First, we evaluate our method when decomposing the noisy input color and compare it to pixel-based baselines. Next, we evaluate our method in combination with user-defined decompositions. We then compare our method with recent sample-based methods and show that for the purpose of high-quality rendering, current sample-based methods are not suitable. Finally, we examine the major design choices of our proposed model in ablation studies.

We measure the denoising quality of different methods with three error metrics: SMAPE, DSSIM (1−SSIM [WBSS04]), and ꟻLIP [ANAM*20]. Among these three error metrics, SMAPE is the loss function that we optimize for, DSSIM highlights the structural dissimilarity between the images, and ꟻLIP approximates the magnitude of perceived differences when flipping between two images.

For the majority of our comparisons, we choose to show the average *relative* error, which is computed by dividing the per-example *absolute* error of each competing method by that of a common baseline and averaging over the dataset. This avoids biasing the average metrics towards outliers and low sample-count testing examples. We also show a *win percentage* (W%), which measures how often a method ranks first among all competing methods. This shows the consistency of improvements brought by a method across large-scale datasets [XZW*19].

6.1. Color Decomposition

To show the effect of our proposed denoiser with learned decomposition, we compare our approach with KPAL baselines (see Section 5.2.1 for model details).

Here we focus on comparing models that process noisy color images directly (*i.e.*, no user-defined decompositions). Table 2 summarizes the quantitative comparison between our method and the KPAL baseline, showing the average relative metric values on testing examples from our test sets. Our model can improve the denoising quality consistently on both academic and production data.

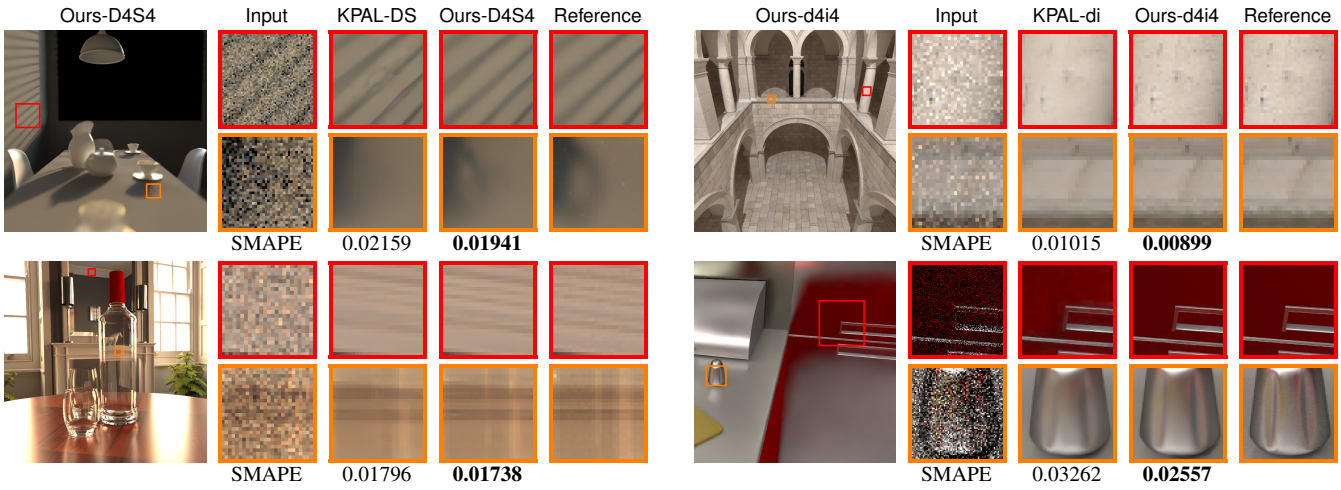
		Average relative error and Win percentage (W%)					
Dataset	K	SMAPE	W%	DSSIM	W%	ꟻLIP	W%
MITSUBA	1	1.000	1.9	1.000	2.5	1.000	5.2
	4	**0.912**	**98.1**	**0.914**	**97.5**	**0.939**	**94.8**
HYPERION	1	1.000	1.2	1.000	3.6	1.000	1.2
	4	**0.934**	**98.8**	**0.934**	**96.4**	**0.952**	**98.8**

Table 2: *Effect of decomposing noisy input color. Rows with $K = 1$ show results generated with KPAL-C and rows with $K = 4$ correspond to results generated with Ours-C4.*

Figure 5 shows the effect of our learned decomposition when operating on color by comparing it to a baseline method without decomposition. We observe overall quality improvements in detail regions, including reflections and high-frequency patterns. Our method also reconstructs motion blur effects with lower error than the baseline.

Decomposition for pre-trained denoiser. Our model is trained end-to-end, meaning that the decomposition and denoising modules are trained jointly, and they adapt to the behavior of each other. To demonstrate that our decomposition module helps in creating components that are easier to denoise, we train a model that uses

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.



(a) *Diffuse-specular decomposition.* **(b)** *Direct-indirect decomposition.*

Figure 6: *Combining learned decomposition with user-defined decomposition. Our method is also able to reconstruct finer details than the baselines, when further decomposing noisy user-defined components. The exposure of some crops is adjusted to better illustrate the results.*

a pre-trained KPAL-C model as its denoiser. That is, we are only optimizing the decomposition module.

Method	Denoising Module	Average relative error SMAPE	DSSIM	ꟻLIP
KPAL-C	Learned	1.000	1.000	1.000
Ours-C2	Pre-trained	0.984	0.982	0.982
Ours-C4	Pre-trained	**0.957**	**0.970**	**0.956**
Ours-C2	Learned	0.973	0.986	0.981
Ours-C4	Learned	**0.912**	**0.914**	**0.939**

Table 3: *Training only a decomposition module to work with a fixed pre-trained denoiser (KPAL-C) is beneficial.*

As shown in Table 3, training only the decomposition module with 2 or 4 components also leads to reduction in denoising error. Additionally, in both cases, we observe that the validation loss decreases quickly and surpasses the pre-trained KPAL-C denoiser in a short period of time. This suggests that our decomposition module is able to learn components that are easier for the denoiser to process, and that this partial training can be a fast way to improve the denoising quality of a pre-trained model.

6.2. Combining with user-defined decompositions

In addition to decomposing and denoising noisy color, our method can also be applied on components from user-defined decompositions such as diffuse–specular and direct–indirect. Here we also show results on both academic and production datasets, which demonstrate that our method can provide denoising quality that exceeds that of kernel-predicting and direct-predicting baselines that use the same type of user-defined decompositions. Our method is particularly beneficial in regions where the user-defined decompositions cannot effectively separate overlapping noisy signals with different noise characteristics.

6.2.1. Kernel-predicting baselines

Method	Average relative error and Win percentage (W%) SMAPE	W%	DSSIM	W%	ꟻLIP	W%
KPAL-DS	1.000	3.7	1.000	7.9	1.000	14.9
Ours-D4S4	**0.933**	**96.3**	**0.947**	**92.1**	**0.957**	**85.1**
KPAL-di	1.000	3.7	1.000	10.1	1.000	14.2
Ours-d4i4	**0.938**	**96.3**	**0.954**	**89.9**	**0.954**	**85.8**

a MITSUBA *dataset.*

Method	Average relative error and Win percentage (W%) SMAPE	W%	DSSIM	W%	ꟻLIP	W%
KPAL-DS	1.000	0.0	1.000	1.2	1.000	0.0
Ours-D4S4	**0.958**	**100.0**	**0.942**	**98.8**	**0.967**	**100.0**

b HYPERION *dataset.*

Table 4: *Our method can also be used in conjunction with user-defined decompositions and achieve additional quality benefits.*

In Table 4 we compare our methods against respective KPAL baselines using the same user-defined decompositions and Figure 6 shows selected testing examples from our MITSUBA test set. Similar to the color-only case, we observe consistent quantitative improvement over both academic and production datasets. However, the overall improvement is smaller than for color-only denoising. This is because user-defined decompositions already perform beneficial decompositions of superimposed signals in some situations, and the gains from further decomposing those components are not as large as using our method to decompose the color directly. Thus, the benefits of our method are most pronounced in regions where the user-defined decomposition is less effective, *e.g.*, shadows on diffuse surfaces for diffuse–specular and reflections for direct–indirect. This indicates that the benefits of our proposed method can complement those of user-defined decompositions.

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

Figure 7: *Comparison with KPAL on* HYPERION *test set. Our method can improve the reconstruction quality of details. © 2021 Disney*

Qualitative comparison on the production (HYPERION) dataset are shown in Figure 7. Our method improves the reconstruction of complex details and thin structures, including volumetric effects, fine textures, hair, and reflections.

6.2.2. Direct-predicting baselines

	Average relative error & Win percentage (W%)					
Method	SMAPE	W%	DSSIM	W%	ꟻLIP	W%
DnGAN	1.000	8.8	1.000	11.2	1.000	4.8
KPAL-DS	0.970	0.0	0.945	4.8	0.918	9.6
Ours-D4S4	**0.881**	**91.2**	**0.828**	**84.0**	**0.866**	**85.6**

Table 5: *Comparing with the direct-predicting baseline (DnGAN). All runs use diffuse–specular decomposition.*

We conduct additional comparisons with a recent state-of-the-art direct-predicting denoiser (DnGAN), which makes use of adversarial training to synthesize better details [XZW*19]. For this comparison, our model and KPAL-DS are trained on the 32spp subset of our MITSUBA training set because the GAN model is reported to be trained on 32spp data. We use the testing dataset provided by the authors, rendered with the Tungsten renderer. Quantitative results are summarized in Table 5. It can be seen that our method is able to outperform both baselines consistently. We refer the interested reader to the Supplemental Material for qualitative comparisons.

6.3. Equal Quality Comparisons

In order to produce renderings at a targeted quality, it is often necessary to render the noisy image with hundreds or thousands of samples per pixel before denoising, which can often take hours in production scenarios. Therefore, in this scenario, our improvements translate to significant savings in sampling budget required to reach the desired quality.

To examine the sampling budget saved by our methods, we expanded the MITSUBA test sets by rendering at 4-256spp with a step size of 4 spp. With this fine-grained test set, we evaluate the

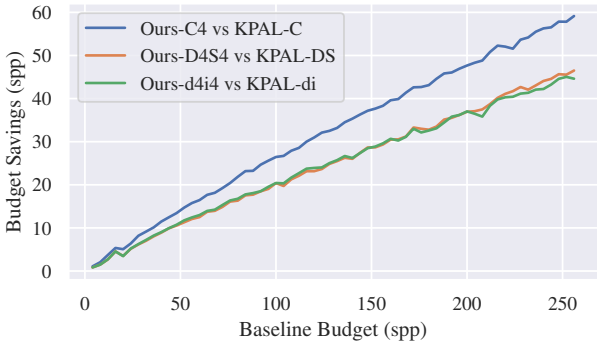


Figure 8: *Sampling budget savings due to decomposition. We visualize the average reduction in sampling budget achieved by our learned decompositions over KPAL baselines at different sampling budgets on the* MITSUBA *test sets.*

sampling budget our methods saved over KPAL baselines. Figure 8 shows the sampling budget savings provided by our method. These savings are computed by comparing the sampling budget required for our methods to achieve equal quality (with respect to SMAPE) as the corresponding KPAL baseline. Our decompositions can provide sampling budget reductions of approximately 25% when compared to a corresponding baseline without any decomposition (Ours-C4 vs. KPAL-C) and savings of approximately 18% when user-defined decompositions are utilized. On the HYPERION dataset, testing data generation is much more costly, and this experiment cannot be conducted as thoroughly. However, based on available data, we still observe approximately 15% savings in the sampling budget of Ours-C4 over KPAL-C.

6.4. Comparing with sample-based methods

Recently, deep sample-based denoising methods have been shown to benefit preview-quality denoising [GLA*19, MH20] by processing each sample independently. The S-LD [MH20] method in particular proposes to learn a partitioning of samples into components,

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

which shares the learned decomposition concept with our work. However, while their decomposition is based on per-sample information, ours is based on statistics gathered at the pixel level. This difference in the input structure results in fundamental differences in the decomposition module architectures.

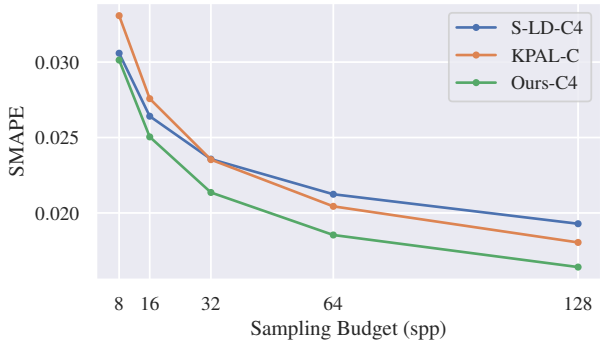


Figure 9: *Generalization to high-sample-count scenarios. For this comparison all methods were trained at 8 samples per pixel. Sample-based decomposition (S-LD-C4) performs well at low sample counts (where it was trained) but does not generalize at high-sample-count scenarios as well as pixel-based methods (KPAL-C, Ours-C4).*

Figure 9 summarizes the denoising quality of sample-based and pixel-based methods on 8-128spp MITSUBA testing examples, where all methods are trained on a 8spp training set. We see that S-LD performs well on 8spp testing data, outperforming the KPAL-C baseline, but our method is able to match its quality despite having access only to per-pixel statistics. On the other hand, sample-based decompositions do not generalize to high-spp scenarios as well as pixel-based methods, falling behind KPAL-C at 64 and 128spp, whereas our method is even better-performing in terms of generalization compared to the KPAL baseline.

In Figure 10 we show qualitative comparisons between these methods. We observe that at 8spp, sample-based decomposition can produce results of competitive quality but at 128spp it cannot match the quality of our method (Ours-C4), producing over-blurred reconstruction results.

In addition to the worse generalization behavior of S-LD at higher sample counts, we also observe that it is more prone to overfitting and less stable than Ours-C4 at 8spp. Even though the additional information stored in the per-sample data should theoretically improve the denoising quality of sample-based methods compared to pixel-based ones, the concurrent work from Cho et al. [CHY21] shows that the sparsity and high-dimensionality of sample-based information pose major challenges. We believe that this can explain why Ours-C4 can perform on par with the sample-based method at 8spp (see Supplemental Material for additional discussion and results).

6.5. Ablation studies

Our ablation studies focus on key design choices of the method, including the number of components and the input feature prediction.

#Components	Average relative error SMAPE	DSSIM	ꟻLIP
1 (baseline)	1.000	1.000	1.000
2	0.973	0.986	0.981
3	0.923	0.927	0.948
4	0.910	0.909	0.939
6	0.902	0.903	0.935
8	0.892	0.885	0.925
16	0.892	0.878	0.925

Table 6: *Diminishing return of additional quality benefits by using more learned components.*

Number of components. We evaluate our method with up to 16 learned components on the MITSUBA dataset. As shown in Table 6, using more components typically leads to lower denoising error, but the benefit diminishes as the number of components increases beyond 4. In our experiments we observed that training with 8 or 16 components can be less stable than training with 2 or 4 components. Taking both practicality and quality into consideration, we chose the 4-component models as our approach in our comparisons.

Predicting Input Features. By allowing the decomposition module to predict per-component feature maps in addition to masks, our denoising module can potentially better adapt to the different characteristics of each component. We experimentally verify this benefit by disabling per-component feature map prediction, which means the decomposition module only predicts component masks. We use the same set of feature maps for denoising all components, which are the output of the feature encoder at the beginning of the pipeline.

Method	Per-comp. feature maps	Average relative error SMAPE	DSSIM	ꟻLIP
KPAL-C	–	1.000	1.000	1.000
Ours-C4	–	0.962	0.985	0.972
Ours-C4	✓	**0.912**	**0.914**	**0.939**

Table 7: *Per-component feature prediction brings denoising quality benefits.*

As shown in Table 7, the prediction of per-component feature maps benefits the overall quality.

7. Discussion and future work

Computational efficiency. The computational cost of compositional image denoising scales linearly with the number of components. We measure an approximate runtime of 0.7s per 1280×720 image component on an RTX 2080Ti GPU for our method. Because typical offline production scenes take multiple hours to render, such timings are practically viable.

An interesting extension to our proposed method is *adaptive decomposition*. That is, if the usefulness of decomposing an image

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

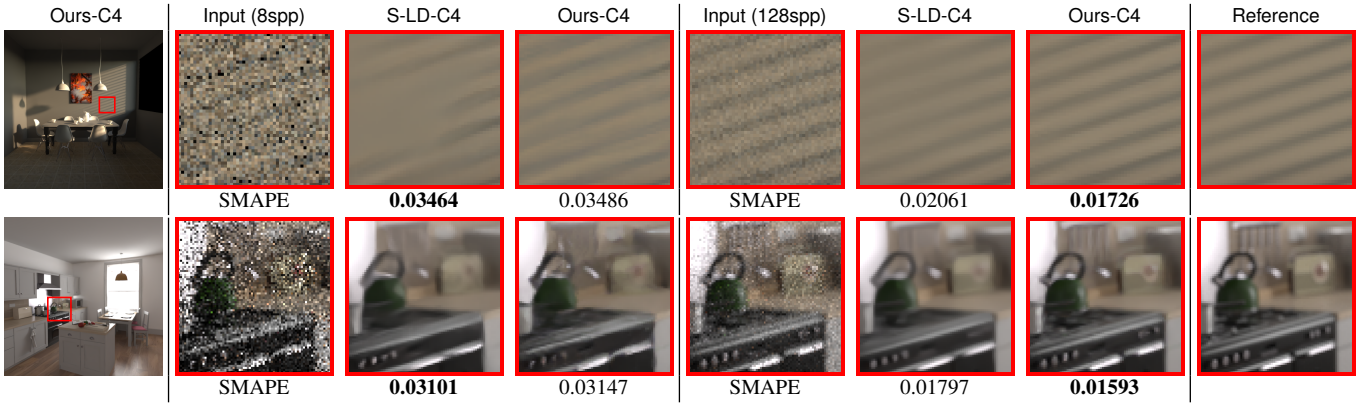


Figure 10: *Comparing our method with sample-based decomposition method [MH20]. Both methods were trained on our 8spp* MITSUBA *training set. On each row, we show denoising results on 8spp (left) and 128spp (right) input of the same scene. Despite being able to achieve relatively good quality at preview quality renderings, the sample-based method does not generalize as well to high-quality scenarios. Our method achieves comparable results in low-spp testing cases and generalizes nicely to higher sample count.*

can be cheaply estimated, we might split an image into many components if this is useful, or denoise it directly if not. This would improve the average runtime required to achieve a desired image quality.

Repeatability. The repeatability of a method can be defined by the variation between different *sibling runs*—identical models trained with different randomness. Because we optimize our model end-to-end, the decomposition module can reach different decomposition strategies across sibling runs. For example, our model with 4 learned components can create a strong separation of signals with different frequency, or predict components that are farther from each other with respect to chromaticity. While all sibling runs of our method yield similar average improvements over the KPAL-C baseline, the results on individual images can vary in quality from run to run. One possible way to improve the similarity of components across sibling runs is by posing more constraints on the components. For example, one can penalize differences in chromaticity to encourage frequency separation.

Non-additive decompositions. Our approach can possibly be combined with other types of decompositions, such as decomposition by multiplication or division. For instance, the user-defined *albedo division* [ZRJ*15, BVM*17, VRM*18] decomposes the reflected radiance into the product of irradiance and the typically higher-frequency textured reflectance. This decomposition typically leads to denoising quality improvements in image regions with diffuse textured surfaces. However, we do not use it in this work because we were able to identify examples where albedo division degrades denoising quality. More specifically, albedo division is only beneficial when the noisy diffuse and albedo buffers are highly correlated. Otherwise, *e.g.*, at blurred boundaries or when the diffuse channel contains both transmission and reflection, denoising with albedo division leads to noticeable artifacts. Predicting robust multiplicative decompositions might be an interesting area for future work.

Training stability. During the early training iterations, we use a regularizer that penalizes variance of the component masks (see Section 5.3). Without such a regularizer, the decomposition module sometimes degenerates to producing all-1 and all-0 masks early during training. Similar issues arise in KPAL [VRM*18], where the softmax-transformed kernel values sporadically degenerate to zeros and ones. This happens when large gradients in the beginning of training push the decomposition mask weights into the flat tails of the sigmoid, where the gradients go to zero, or push the kernel weights of KPAL into the equivalent regions of the softmax function. While our solution of regularizing through a variance penalty term addresses this problem, we see potential for more elegant solutions.

Temporal information and robustness. In the temporal domain, adjacent frames might contain information about how the signal should be decomposed. For instance, when the camera moves in a scene, the movement of glossy highlights relative to diffuse surfaces is different. While our proposed method focusses on single-frame denoising, in theory, our learned decomposition can be extended to decompose frame sequences into components that are beneficial for denoising. However, we leave the study of temporal denoising with learned decomposition to future work.

We measure the stability of our single-frame reconstruction method on sequences of 5 independently seeded frames of a static scene. We compare between Ours-C4 and KPAL-C the per-pixel variance of the reconstructed images across different frames. We observe that Ours-C4 leads to lower cross-frame variance than KPAL-C for the majority of testing examples. Compared to KPAL-C, our method reduces the cross-frame variance by 5.5% and 6.7% on average over the MITSUBA and HYPERION datasets, respectively. This shows that our method is more robust against noise than single-frame KPAL-C. Based on this experiment, we believe that our approach can be extended to benefit from temporal information. Additional results for the MITSUBA dataset can be found in Supplemental Material.

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

8. Conclusion

In this paper, we focus on improving the state-of-the-art for offline denoising of single-frame high-quality content. We have demonstrated that our neural decomposition module can learn to produce beneficial decompositions for state-of-the-art kernel-predicting denoisers.

We showed our neural decomposition module's flexibility as it can be applied on the noisy color input directly or on top of user-defined decompositions that proved effective in the past. Further, it can be trained jointly with a denoising module to maximize denoising performance, or it can be trained separately with a pre-trained denoiser when retraining of the denoiser is not feasible. In all these scenarios, we observed improvements in denoising quality over the corresponding baseline methods that do not use a neural decomposition.

The improvements from our decompositions come at the cost of a linear increase in denoising time with the number of components, but this cost is negligible compared to the render times saved by our method for final-quality content.

We also validated our method against sample-based methods. Even though our method is designed for high-quality setups and does not have access to the per-sample information, it performs roughly on par in terms of quality on low-sample-count renderings and outperforms them on high-quality renderings.

Finally, we emphasize that our method appeals to high-quality production rendering due to its low adoption cost. It uses the same inputs as standard per-pixel denoising methods, can be trained on the same data, and is robust enough to be used in various high-quality rendering scenarios.

Acknowledgements

We thank David Adler, Mark Meyer, and the anonymous reviewers for their constructive feedback, and Bing Xu for providing us with the evaluation datasets for the DnGAN work. We also thank the following Blendswap artists for creating scenes used in the MITSUBA dataset: Mareck, Wig42, SlykDrako, Jay-Artist, NewSee2l035, nacimus, aXel, thecali, piopis, cekuhnen, UP3D, MrChimp2313, NovaZeeke, and Delatronic. Our method was trained and tested on production imagery but the results were not part of the released productions.

References

[ANAM*20] ANDERSSON P., NILSSON J., AKENINE-MÖLLER T., OSKARSSON M., ÅSTRÖM K., FAIRCHILD M. D.: Flip: A difference evaluator for alternating images. *Proc. ACM Comput. Graph. Interact. Tech. 3*, 2 (Aug. 2020). URL: https://doi.org/10.1145/3406183, doi:10.1145/3406183. 7

[BAC*18] BURLEY B., ADLER D., CHIANG M. J.-Y., DRISKILL H., HABEL R., KELLY P., KUTZ P., LI Y. K., TEECE D.: The design and evolution of disney's hyperion renderer. *ACM Trans. Graph. 37*, 3 (July 2018). URL: https://doi.org/10.1145/3182159, doi:10.1145/3182159. 6

[BCM05] BUADES A., COLL B., MOREL J.-M.: A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation 4*, 2 (2005), 490–530. 2, 3

[Bit16] BITTERLI B.: Rendering resources, 2016. https://benedikt-bitterli.me/resources/. 6

[BRM*16] BITTERLI B., ROUSSELLE F., MOON B., IGLESIAS-GUITIÁN J. A., ADLER D., MITCHELL K., JAROSZ W., NOVÁK J.: Nonlinearly weighted first-order regression for denoising Monte Carlo renderings. *Computer Graphics Forum 35*, 4 (2016), 107–117. 2, 3

[BVM*17] BAKO S., VOGELS T., MCWILLIAMS B., MEYER M., NOVÁK J., HARVILL A., SEN P., DEROSE T., ROUSSELLE F.: Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Trans. Graphics (Proc. SIGGRAPH) 36*, 4 (July 2017), 97:1–97:14. 2, 3, 6, 11

[CHY21] CHO I.-Y., HUO Y., YOON S.-E.: Weakly-supervised contrastive learning in path manifold for monte carlo image reconstruction. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 38:1–38:14. URL: https://doi.org/10.1145/3450626.3459876. 3, 10

[CKS*17] CHAITANYA C. R. A., KAPLANYAN A. S., SCHIED C., SALVI M., LEFOHN A., NOWROUZEZAHRAI D., AILA T.: Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Trans. Graphics (Proc. SIGGRAPH) 36*, 4 (July 2017), 98:1–98:12. 3

[DFKE06] DABOV K., FOI A., KATKOVNIK V., EGIAZARIAN K.: Image denoising with block-matching and 3D filtering. In *Proc. SPIE* (2006), vol. 6064, pp. 606414–606414–12. 3

[FHF*17] FASCIONE L., HANIKA J., FAJARDO M., CHRISTENSEN P., BURLEY B., GREEN B.: Path tracing in production-part 1: production renderers. In *ACM SIGGRAPH 2017 Courses*. 2017, pp. 1–39. 2

[GLA*19] GHARBI M., LI T.-M., AITTALA M., LEHTINEN J., DURAND F.: Sample-based monte carlo denoising using a kernel-splatting network. *ACM Trans. Graph. 38*, 4 (2019), 125:1–125:12. 3, 9

[GPAM*14] GOODFELLOW I., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680. 3

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition* (2016), IEEE Computer Society, pp. 770–778. 6

[IMF*21] IŞIK M., MULLIA K., FISHER M., EISENMANN J., GHARBI M.: Interactive monte carlo denoising using affinity of neural features. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 1–13. URL: https://doi.org/10.1145/3450626.3459793, doi:10.1145/3450626.3459793. 3

[Jak10] JAKOB W.: Mitsuba renderer, 2010. 6

[JWB*19] JAKOB W., WEIDLICH A., BEDDINI A., PIEKÉ R., TANG H., FASCIONE L., HANIKA J.: Path tracing in production: part 2: making movies. In *ACM SIGGRAPH 2019 Courses*. 2019, pp. 1–41. 2

[Kaj86] KAJIYA J. T.: The rendering equation. *SIGGRAPH Computer Graphics 20*, 4 (Aug. 1986), 143–150. 2

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014). arXiv:1412.6980. 6

[KBS15] KALANTARI N. K., BAKO S., SEN P.: A machine learning approach for filtering Monte Carlo noise. *ACM Trans. Graphics (Proc. SIGGRAPH) 34*, 4 (July 2015), 122:1–122:12. 2

[KHL19] KETTUNEN M., HÄRKÖNEN E., LEHTINEN J.: Deep convolutional reconstruction for gradient-domain rendering. *ACM Trans. Graph. 38*, 4 (July 2019). URL: https://doi.org/10.1145/3306346.3323038, doi:10.1145/3306346.3323038. 6

[KMA*15] KETTUNEN M., MANZI M., AITTALA M., LEHTINEN J., DURAND F., ZWICKER M.: Gradient-domain path tracing. *ACM Trans. Graph. 34*, 4 (2015). 6

[LWY*21] LIN W., WANG B., YANG J., WANG L., YAN L.-Q.: Path-based monte carlo denoising using a three-scale neural network. In *Computer Graphics Forum* (2021), Wiley Online Library. 3

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

Renderer	Type	Scenes	Ref. spp	Resolution	Input spp	Examples	Samples?	Used in
Mitsuba (academic)	Training	575	8192	1280×720	2-256	4600	No	Sections 6.1, 6.2, 6.5
					8	575	Yes	Section 6.4
	Validation	55	8192	1280×720	2-256	440	No	Sections 6.1, 6.2, 6.5
					8	55	Yes	Section 6.4
Hyperion (production)	Training	382	Various	1920×804	16-3496	2962	No	Section 6.2.1

(a) *Training and validation datasets.*

Name / Renderer	Source	Scenes	Ref. spp	Resolution	Input spp	Examples	Samples?	Used in
Mitsuba 1 (academic)	Public scenes	47	65536	1280×720	2-256	376	No	Sections 6.1, 6.2.1, 6.5
					2-256	3055	No	Section 6.3 (spp saving)
					2-128	329	Yes	Section 6.4
Mitsuba 2 (academic)	Similar to [MH20]	27	65536	512×512	2-256	216	No	Sections 6.1, 6.2.1, 6.5
					2-256	1755	No	Section 6.3 (spp saving)
					2-128	189	Yes	Section 6.4
Tungsten (academic)	[XZW*19]	25	32768	Various	4-128	125	No	Section 6.2.2
Hyperion (production)	–	35	Various	1920×804	8-128	84	No	Section 6.2.1

(b) *Testing (evaluation) datasets.*

Table 8: *Dataset overview. Here we list the datasets used in our experiments, including training, validation and testing datasets, and point to the sections where the discussion is based on the results from the corresponding datasets. Notably, we have mainly pixel-based datasets which can scale to higher sample count, but we also create sample-based data for the comparison with sample-based methods. In the table, "Various" means the related value varies between examples.*

[LZZ*18] LIU P., ZHANG H., ZHANG K., LIN L., ZUO W.: Multi-level wavelet-cnn for image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2018), pp. 773–782. 2

[MBC*18] MILDENHALL B., BARRON J. T., CHEN J., SHARLET D., NG R., CARROLL R.: Burst denoising with kernel prediction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2502–2510. 2

[MCY14] MOON B., CARR N., YOON S.-E.: Adaptive rendering based on weighted local regression. *ACM Trans. Graphics 33*, 5 (Sept. 2014), 170:1–170:14. 2

[MH20] MUNKBERG J., HASSELGREN J.: Neural denoising with layer embeddings. In *Computer Graphics Forum* (2020), vol. 39, Wiley Online Library, pp. 1–12. 2, 3, 6, 7, 9, 11, 13

[MJL*13] MOON B., JUN J. Y., LEE J., KIM K., HACHISUKA T., YOON S.-E.: Robust image denoising using a virtual flash image for Monte Carlo ray tracing. *Computer Graphics Forum 32*, 1 (2013), 139–151. 2

[MKD*16] MANZI M., KETTUNEN M., DURAND F., ZWICKER M., LEHTINEN J.: Temporal gradient-domain path tracing. *ACM Transactions on Graphics (TOG) 35*, 6 (2016), 1–9. 6

[MSY16] MAO X., SHEN C., YANG Y.-B.: Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Advances in neural information processing systems* (2016), pp. 2802–2810. 2

[PJH16] PHARR M., JAKOB W., HUMPHREYS G.: *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016. 2

[RKZ12] ROUSSELLE F., KNAUS C., ZWICKER M.: Adaptive rendering with non-local means filtering. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 31*, 6 (Nov. 2012), 195:1–195:11. 2

[RMZ13] ROUSSELLE F., MANZI M., ZWICKER M.: Robust denoising using feature and color information. *Computer Graphics Forum 32*, 7 (2013), 121–130. 2, 3

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)* (1998), IEEE, pp. 839–846. 2

[VRM*18] VOGELS T., ROUSSELLE F., MCWILLIAMS B., RÖTHLIN G., HARVILL A., ADLER D., MEYER M., NOVÁK J.: Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018) 37*, 4 (2018), 124:1–124:15. doi:10.1145/3197517.3201388. 1, 2, 3, 4, 6, 11

[WBSS04] WANG Z., BOVIK A., SHEIKH H., SIMONCELLI E.: Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (April 2004), 600–612. 7

[XPG*19] XIA Z., PERAZZI F., GHARBI M., SUNKAVALLI K., CHAKRABARTI A.: Basis prediction networks for effective burst denoising with large kernels. *arXiv preprint arXiv:1912.04421* (2019). 2

[XZW*19] XU B., ZHANG J., WANG R., XU K., YANG Y.-L., LI C., TANG R.: Adversarial monte carlo denoising with conditioned auxiliary feature. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2019) 38*, 6 (2019), 224:1–224:12. 2, 3, 6, 7, 9, 13

[ZJL*15] ZWICKER M., JAROSZ W., LEHTINEN J., MOON B., RAMAMOORTHI R., ROUSSELLE F., SEN P., SOLER C., YOON S.-E.: Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Computer Graphics Forum (Proc. Eurographics) 34*, 2 (May 2015), 667–681. 2

[ZRJ*15] ZIMMER H., ROUSSELLE F., JAKOB W., WANG O., ADLER D., JAROSZ W., SORKINE-HORNUNG O., SORKINE-HORNUNG A.: Path-space motion estimation and decomposition for robust animation filtering. *Computer Graphics Forum 34*, 4 (2015), 131–142. 2, 3, 11

[ZZC*17] ZHANG K., ZUO W., CHEN Y., MENG D., ZHANG L.: Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Trans. Image Processing 26*, 7 (2017), 3142–3155. 2

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.