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 (Supplemental)

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

In this supplemental document, we first provide more details of our decomposition scheme, model implementation, and training hyperparameters for different methods. Then we show some additional quantitative results related to the comparison with the sample-based S-LD method. Finally, we provide results about the robustness of pixel-based methods as mentioned in the discussion of the main paper.

We refer the readers to the supplemental interactive image viewer for additional qualitative results.

1. Decomposition Function Definition

In the most formal way, our decomposition scheme corresponds to a *complete full binary tree* with K leaf nodes, where the nodes correspond to decomposition modules and the leaves correspond to kernel-predicting denoisers. The input of the root node is the input color and the feature encoder's output. The input to all other nodes are one of the two the parents outputting component–feature pairs. The depth D of this tree can be computed as $D = \lceil \log_2 K \rceil + 1$. If K is a power of 2, the resulting decomposition framework will be a perfect binary tree with depth D.

Formally, our decomposition function is defined as follows. Let $\mathbf{c}^{(d,l)}$ denote the l-th noisy colors component at depth level d, and similarly for $\mathbf{f}^{(d,l)}$, with the network input as the root of the decomposition tree, *i.e.*, $\mathbf{c}^{(0,1)} = \mathbf{c}$ and $\mathbf{f}^{(0,1)} = \mathbf{f}$.

For each depth level $d = 1, \ldots, D-1$, there are $N_d \leq 2^d$ nodes, which we index with $l = 1, \ldots, N_d$ from left to right. Note that N_d is guaranteed to be an even number because of the complete full binary tree. For the l-th node at depth d, its parent node can be indexed with p_l at depth $d-1$, where $p_l = \lfloor \frac{l+1}{2} \rfloor$. Therefore, for $l = 1, 2, \ldots, N_d - 1$, components l and $l+1$ will be predicted from component p_l at depth $d-1$:

$$\{\mathbf{m}^{(d,l)}, \mathbf{f}^{(d,l)}, \mathbf{f}^{(d,l+1)}\} = h(\mathbf{c}^{(d-1,p_l)}, \mathbf{f}^{(d-1,p_l)}). \tag{1}$$

and

$$\mathbf{c}^{(d,l)} = \mathbf{m}^{(d,l)} \odot \mathbf{c}^{(d-1,p_l)}$$
$$\mathbf{c}^{(d,l+1)} = (\mathbf{1} - \mathbf{m}^{(d,l)}) \odot \mathbf{c}^{(d-1,p_l)}$$

with $\mathbf{c}^{(d,l)} + \mathbf{c}^{(d,l+1)} = \mathbf{c}^{(d-1,p_l)}$. The K leaf nodes of this decomposition tree will be the components that will be denoised.

2. Implementation details

Here we provide some additional implementation details about our network architecture, regularization loss and training scheme.

2.1. Network architecture (for comparing with pixel-based methods)

Decomposition module. We implement the decomposition function using a U-Net with 3 scales. The input to the decomposition module consists of 64-channel feature maps and an RGB color image. For the first decomposition module operating on the full color, the input feature vector is created by passing the input color and auxiliary features to a feature encoder consisting of 2 residual blocks [HZRS16] that outputs a 64-channel feature map (a residual block has 2 convolution layers and a residual connection by addition). The U-Net has feature maps with 64, 128 and 256 channels for the three scales, 2×2 strided convolution for down-sampling and nearest-neighbor interpolation for up-sampling of the feature maps between scales, and skip connections between the encoder and decoder on the same scale (implemented with concatenation). We use ReLU as the activation function and convolution kernels of size 3×3 for all convolution layers except for downsampling (2×2) and predicting masks and kernels (1×1).

The output of the decomposition module (component mask) is predicted by a 1×1 convolution layer from the 64-channel output of the U-Net with a sigmoid activation function. This raw output is then multiplied with the input image to obtain the first component (the left child of this node). The second component mask will be the complement of the first, making sure that the two components sum up to the input color image. Alongside the components, the decomposition module also predicts two sets of component-specific feature maps, each of which is predicted by a single (component-specific) residual block with 64 channels.

Denoising module. In this work, we adopt the 1-frame variant of KPAL [VRM*18] as the component denoiser. According to the suggestion in the KPAL work, we replace the ResNet with a U-Net for improved efficiency, and we confirm that this yields better performance at no loss in quality. We therefore use U-Nets for both our method and our implementation of the KPAL baselines.

The denoising module operates on the component color image $\mathbf{c}^{(k)}$ and the component auxiliary features $\mathbf{f}^{(k)}$. The final reconstructed image $\mathbf{d}$ is then produced by summing the denoised output $\mathbf{d}^{(k)}$ of each component. The U-Net backbone is the same as for the decomposition module, with different trainable parameters. As for the kernel reconstruction module at the end of the architecture, we adopt the multi-scale approach in KPAL [VRM*18] with 3 scales and 5×5 kernels instead of using one 21×21 kernel per pixel at the finest scale. The denoiser module is shared between all components.

2.2. Regularization: variance penalty

The regularization term we use at the beginning of training is defined as the per-pixel per-channel average of the variance of the masks $\{\mathbf{m}^{(k)}\}_{k=1}^{K}$ across different components:

$$R(\{\mathbf{m}^{(k)}\}) = \frac{1}{|\mathcal{I}|}\frac{1}{C}\frac{1}{K}\sum_{p\in\mathcal{I}}\sum_{c=1}^{3}\sum_{k=1}^{K}\left(\mathbf{m}_{p,c}^{(k)} - \overline{\mathbf{m}}_{p,c}\right)^2, \qquad (2)$$

where C is the number of channels in the mask ($C = 3$ for RGB masks), and $\overline{\mathbf{m}} = \frac{1}{K}\sum_k \mathbf{m}^{(k)}$ is the average mask, which is equal to $\frac{1}{K}\mathbf{1}$ because all masks sum to 1. This regularization term forces the masks to be uniform and can stabilize training in early steps. We use a weight of 10 for this term at the first 16384 steps and turn it off (*i.e.*, set its weight to 0) afterwards. Note that, when the number of components is not a power of 2 (*e.g.*, 3 or 6), leaf nodes are on two different depth levels of the tree. To enforce a balanced decomposition there, we multiply the masks from the deepest leaf nodes (*i.e.*, the ones that have been decomposed the most) by 2 before applying the variance penalty, so that masks from the same depth level are regularized to be the same.

2.3. Training scheme and hyper-parameters

The batch size is different for baselines and our decomposition methods: 12 for KPAL, 6 for Ours-C2 and 3 for Ours-C4. This makes the training time of different methods roughly the same at the same number of steps, and also means that the decomposition and denoising modules for different methods see roughly the same number of image patches. On the MITSUBA dataset, models that operate on noisy color directly (without user-defined decomposition) are trained for 1.8M steps, which takes approximately 5 days on a single RTX 2080Ti GPU. For denoising user-defined components (diffuse–specular or direct–indirect), the models are trained until 3M steps to ensure convergence. On the HYPERION dataset, all models are trained until 3M steps. We apply learning rate decay twice at 80% and 90% of the training, each time dividing the learning rate by 10. Unless otherwise mentioned, we evaluate the pixel-based methods using the models after the last training step. This is very close to the model with the best validation loss because of the learning rate decay, and because we do not observe overfitting behavior when training on our full datasets.

For sample-based methods, we follow Munkberg and Hasselgren [MH20] and train the models with batch size 4, initial learning rate 5×10^{-4}, and we halve the learning rate every 180k iterations. We also use the same learning rate decay scheme for the competing pixel-based methods, but with a starting learning rate of 10^{-4} because we observe training difficulty for these models at the higher learning rate. We also tried using 10^{-4} as the initial learning rate for S-LD methods but found that this leads to worse results compared to 5×10^{-4}.

Also, similar to the original work [MH20], we train the methods on the 8spp subset of our datasets. Because of the smaller training set size (which is similar to the dataset used by the original work in terms of the number and variety of scenes), over-fitting can be observed for both pixel-based and sample-based methods. We therefore report the metrics on the testing set using models with the best validation SMAPE loss.

3. Architecture improvement of the S-LD method

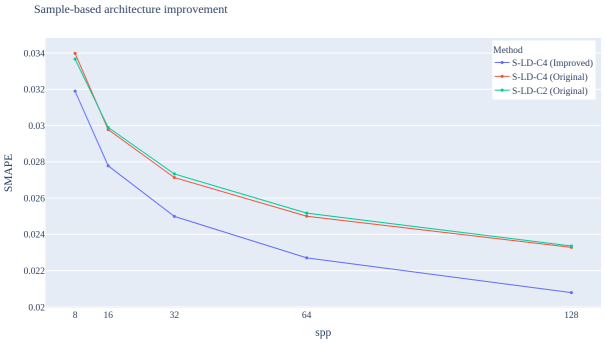

Figure 1: Quality benefit brought by our improved S-LD method (Improved) compared to the original method [MH20]. We show progression of SMAPE error with respect to sample count.

In Figure 1 we show the impact of our improved architecture of the S-LD method (using a larger denoising module and less trainable weights for sample partitioning). Our enhanced architecture is able to improve significantly the denoising quality of the original method. We also confirm that the 2- and 4- component variants of S-LD perform similarly, as reported by the original work [MH20].

4. Overfitting behavior of S-LD-C4

In our experiments, we find that the sample-based method S-LD [MH20] is more prone to overfitting than our method, which might explain why S-LD-C4 is not able to clearly outperform Ours-C4. We provide two pieces of evidence from our experiments.

First, Figure 2 shows the progression of the training and validation losses of our method and S-LD. We observe that the training loss of the sample-based method continues to decrease after the validation loss flattens, whereas the training loss of our method does

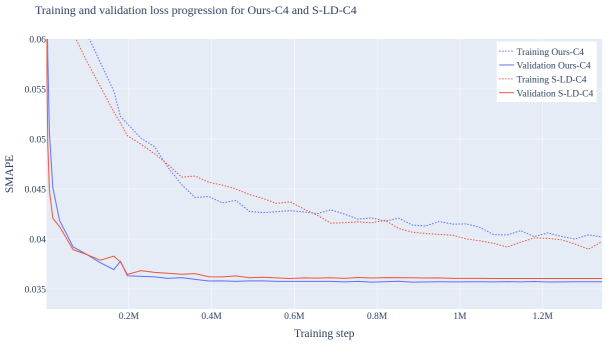

Figure 2: The training and validation loss progression of Ours-C4 and S-LD-C4. The training loss curves are smoothed to better reflect its trend.

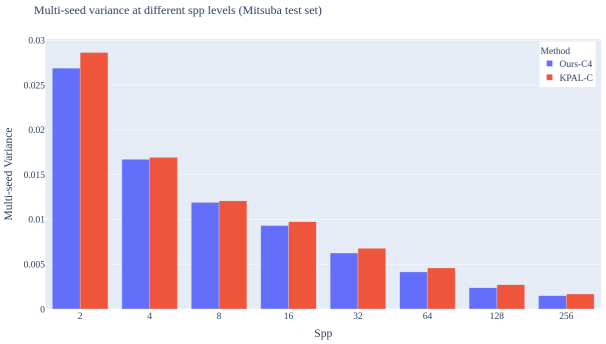

Figure 3: Multi-seed variance on MITSUBA test set. The denoised images of Ours-C4 from noisy input images with different random seeds are more stable than those of KPAL-C.

not decrease much after validation loss stops decreasing. Eventually, the training loss of S-LD-C4 becomes lower than that of Ours-C4, but the validation loss remains worse than our method. This indicates that the S-LD method is more prone to overfitting.

Second, S-LD sometimes produces degenerate reconstructions for some testing examples that are too different from the training set, which is not a problem for the compared pixel-based methods. One example is the "Motion Blur Ball 2" scene in the supplemental viewer, where the method fails to reconstruct the background texture with regular grid, which is not present in the training set. This artifact is present for both the original and improved architecture (though for partially different sets of testing examples), and it can also be found in the supplemental material from the concurrent work by Işik et al. [IFME21] (see Image Viewer (16spp), examples 008 and 013). *We identify and exclude these degenerate examples from the average loss computation*, for the comparisons in the main document and also results in Figure 1.

In our experiments, this behavior of S-LD happens after 200k steps of training, and it happens only for some of the sibling runs. Also, it is worth noting that these degenerate results do not happen on the validation set, which includes scenes that are generated from the same generator used for the training set. Therefore, it is likely that this is a result of overfitting, where the model adapts to the training examples too much and reaches regions in the parameter space where it is unable to produce stable results for the testing examples. Moreover, this stability issue also reflects the difficulties in handling noisy sample-based data, which can lead to high-magnitude gradients. From the analysis of the intermediate output from the model, we find that the feature maps produced by the U-Net in the partitioning module contain very large absolute values in degenerate regions.

In summary, the overfitting and instability of S-LD potentially reflects the difficulties of handling sample-based data, due to the noise, sparsity and high dimensionality [CHY21].

5. Evaluation of robustness against random noise

We evaluate the robustness of our (single-frame) approach and KPAL baseline against random noise by using these single-frame methods to process noisy images with different random seeds (on the same static scenes). The results on the MITSUBA test set are summarized in Figure 3, which show that our method produces less variation between denoised images with multiple seeds, and that this behavior is consistent across different spp levels. This experiment indicates that the additional decomposition step of our method should not increase temporal instability (which primarily results from different random noise between neighboring frames), when extended to multi-frame denoising.

References

[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

[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. 1

[IFME21] IŞIK M., FISHER M., MULLIA K., EISENMANN J.: Interactive Monte Carlo Denoising using Affinity of Neural Features. *ACM Trans. Graph 40* (2021). doi:10.1145/3450626.3459793. 3

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