Monte Carlo Denoising via Auxiliary Feature Guided Self-Attention

JIAQI YU, South China University of Technology, China
YONGWEI NIE[*], South China University of Technology, China
CHENGJIANG LONG, JD Finance America Corporation, USA
WENJU XU, OPPO US Research Center, InnoPeak Technology Inc, USA
QING ZHANG, Sun Yat-sen University, China
GUIQING LI, South China University of Technology, China

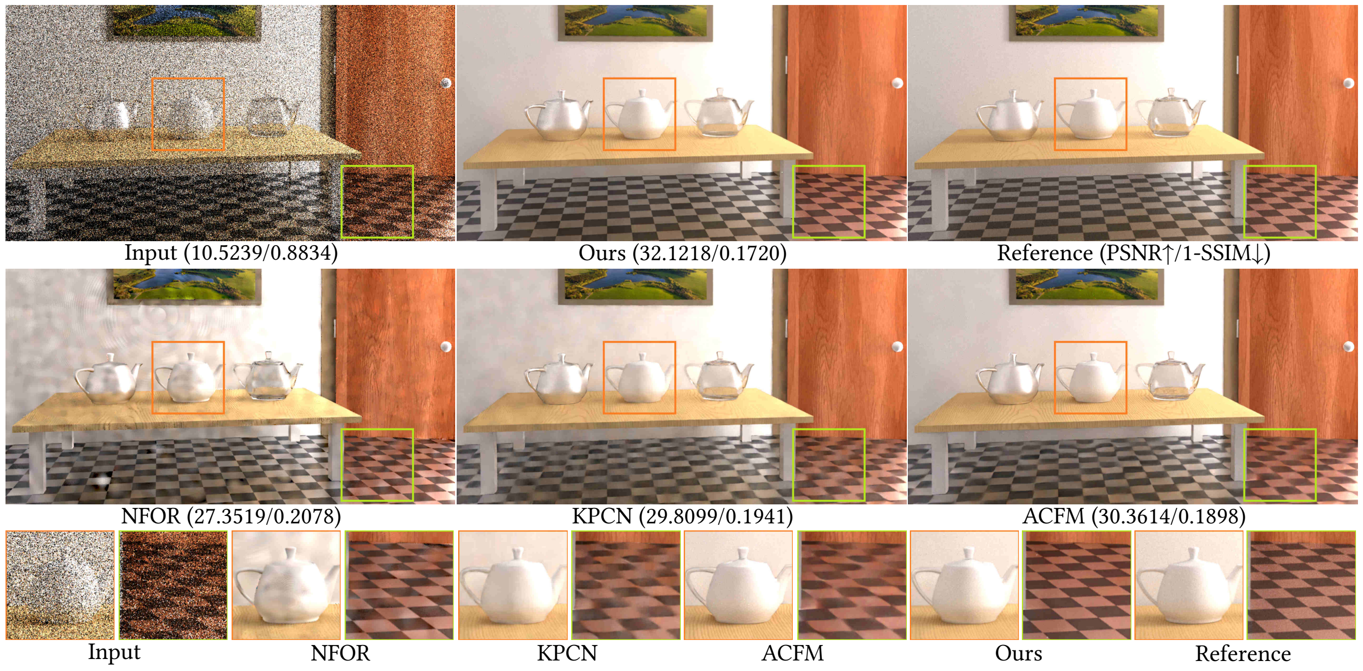

Fig. 1. Current state-of-the-art methods, including NFOR [Bitterli et al. 2016], KPCN [Bako et al. 2017] and ACFM [Xu et al. 2019], fail to produce a plausible denoised image for the scene "VeachAjar" because of the absence of specular albedo and the extremely noisy input. In contrast, our proposed model with auxiliary feature guided self-attention can gather the most relevant information for each pixel from its surrounding region in an edge-preserving manner, thus better restoring image details while preserving image structures and producing visually pleasing denoising results.

While self-attention has been successfully applied in a variety of natural language processing and computer vision tasks, its application in Monte Carlo (MC) image denoising has not yet been well explored. This paper presents a self-attention based MC denoising deep learning network based on the fact that self-attention is essentially non-local means filtering in the embedding space which makes it inherently very suitable for the denoising task. Particularly, we modify the standard self-attention mechanism to an auxiliary feature guided self-attention that considers the by-products (e.g., auxiliary feature buffers) of the MC rendering process. As a critical prerequisite to fully exploit the performance of self-attention, we design a multi-scale feature extraction stage, which provides a rich set of raw features for the later self-attention module. As self-attention poses a high computational complexity, we describe several ways that accelerate it. Ablation experiments validate the necessity and effectiveness of the above design choices. Comparison experiments show that the proposed self-attention based MC denoising method outperforms the current state-of-the-art methods.

[*]Yongwei Nie is the corresponding author.

CCS Concepts: • **Computing methodologies** → **Ray tracing**.

Additional Key Words and Phrases: Monte Carlo denoising, self-attention, non-local means.

1 INTRODUCTION

Monte Carlo (MC) path tracing is a popular realistic rendering technique widely used in computer animation, film production, video games, etc. Compared with other rendering techniques, MC methods are unbiased and exhibit high generality for a variety of visual rendering effects. Nevertheless, in order to produce high-quality images without visible noise, MC path tracing requires sampling a large number of rays which is extremely time-consuming. There is a long history that graphics researchers attempt to improve image quality using a reduced number of ray samples, such as ray histogram fusion [Delbracio et al. 2014] and sample-based denoising [Gharbi et al. 2019; Lin et al. 2021; Munkberg and Hasselgren 2020]. Besides these, image-space MC denoising is another popular technique that first renders a noisy image at a low sampling rate, and then applies a filtering operator to remove the noise and reconstruct the target high-quality image.

Traditional image-space MC denoising was achieved by weighted local regression [Moon et al. 2014], assuming the value of a pixel can be approximated locally by the Taylor polynomial expansion of a nearby pixel. However, these methods usually require a bias-variance trade-off. For example, zero-order regression methods [Overbeck et al. 2009; Rousselle et al. 2012] model the pixel neighborhood as a constant function [Bitterli et al. 2016], the solution of which is equivalent to applying a joint bilateral [Tomasi and Manduchi 1998] or non-local means filter [Buades et al. 2005; Rousselle et al. 2012]. The variance of these methods is the smallest, but the bias is significant. Increasing the order of the regression model can make it more flexible to fit the training data [Bauszat et al. 2011; Bitterli et al. 2016; Li et al. 2013; Moon et al. 2014], resulting in a smaller bias while the variance increases accordingly.

Recently, deep learning approaches have been proposed for MC denoising [Bako et al. 2017; Gharbi et al. 2019; Kalantari et al. 2015; Vogels et al. 2018; Xu et al. 2019]. As an early attempt of MC denoising using neural network, Kalantari et al. [Kalantari et al. 2015] employ a multi-layer perception (MLP) network to learn the bandwidth for joint bilateral and joint non-local means filters from paired noisy and non-noisy images. Bako et al. [Bako et al. 2017] use a convolutional neural network (CNN) to predict the filtering kernel for each pixel adaptively. Eliminating the intermediate filters, the network proposed in [Xu et al. 2019] directly outputs the denoised image given a noisy input.

While various CNN models can already produce impressive MC denoising results, this paper presents a self-attention based MC denoising model that achieves better results. Compared with CNNs, self-attention has several advantages. Firstly, self-attention can directly compute the interaction between any pair of features, naturally capturing long-range dependencies and having a global receptive field. In contrast, CNNs typically employ small convolutional kernels. Although the receptive field of a CNN can be enlarged by stacking more convolutional and pooling layers, the resultant interaction between two distant features is indirectly computed. Secondly, convolution is content-independent, i.e., the response at every spatial position is obtained with the same set of weights tailored for the whole image rather than for each local position. In contrast, self-attention is content-dependent, which computes the weights between any pair of pixels according to their features. Intuitively, gathering information in a large receptive field with directly computed adaptive weights can yield better denoising results. Moreover, Wang et al. [Wang et al. 2018] have shown that self-attention is essentially a non-local means filtering in the embedding space, thus sharing the same edge-preserving property as non-local means for image denoising. This is verified in Figure 1 in which previous state-of-the-art approaches tend to blur the edges of grids on the floor, while our self-attention based method perfectly preserves these edges.

Our self-attention based MC denoising starts with a multi-scale feature extraction stage which uses different sized convolutional filters to extract features under different receptive fields, as what has been done in Inception [Szegedy et al. 2015]. This provides more raw features that are later processed by the self-attention module. Specifically, we stack a total of five self-attention based neural blocks. To take advantage of the auxiliary feature buffers, including normal, depth, and albedo, we modify the standard self-attention mechanism by introducing auxiliary features into the computation of queries and keys that are the core ingredients of self-attention, proposing the auxiliary feature guided self-attention mechanism. Finally, the output of the last self-attention block is transformed to a high-quality denoised image by a simple decoder module. Since self-attention is computationally expensive, we show several ways to accelerate it, enabling it to run on the consumer graphics card GeForce RTX 3090.

In summary, the main contributions of this paper are three-fold:

- We propose a novel self-attention based MC denoising deep learning network comprising a multi-scale feature extractor and self-attention based neural blocks. To the best of our knowledge, we are the first that utilize self-attention mechanism for Monte Carlo image denoising.
- Specifically for MC denoising, we revise the standard self-attention mechanism to auxiliary feature guided self-attention, which effectively involves the auxiliary features into the complex denoising process.
- Ablation and comparison experiments show that the proposed self-attention based MC denoising model is effective and achieves superior performance over the state-of-the-art approaches.

2 RELATED WORK

Traditional state-of-the-art MC denoising approaches are mainly based on local neighborhood regression models. More recent methods utilize deep neural networks to achieve impressive denoising effects. In the following, we briefly review the above two kinds of methods. For a more comprehensive introduction of MC denoising methods, please refer to the survey papers [Huo and Yoon 2021; Zwicker et al. 2015]. Besides, since our method employs Transformer-based model and self-attention mechanism, we also introduce their recent advances and applications.

Traditional Monte Carlo Denoising. Denoising Monte Carlo rendering can be traced back to the nonlinear image space filter [Rushmeier and Ward 1994] proposed by Rushmeier et al. This pioneering idea becomes the basis of later Monte Carlo denoising methods, such as the cross bilateral filter [Eisemann and Durand 2004; Petschnigg et al. 2004; Xu and Pattanaik 2005], the edge-avoiding filter [Dammertz et al. 2010] and the non-local means filter [Buades et al. 2005]. As shown in [Bitterli et al. 2016], these methods actually perform zero-order regressions. McCool et al. [McCool 1999] are the first adding normal, depth, etc., as auxiliary information into the filtering process, yielding better denoising results, which becomes an integral part of later state-of-the-art methods. For example, methods of [Overbeck et al. 2009; Rousselle et al. 2012] utilize auxiliary feature buffers to guide the non-local means filtering, making the denoising process more robust [Moon et al. 2013; Rousselle et al. 2013]. Many other works consider using first-order [Bauszat et al. 2011; Bitterli et al. 2016; Moon et al. 2014] and even higher-order regression models [Moon et al. 2016]. Although higher-order regression models can better approximate ground truths than lower ones, they sometimes suffer from overfitting and cannot generalize to complicated scenes.

Learning-based Monte Carlo denoising. Many deep learning based methods now outperform their traditional counterparts for a variety of tasks like visual recognition [Hu et al. 2021], object detection and localization [Islam et al. 2020], and image generation [Xu et al. 2021], thanks to the strong fitting capabilities of deep neural networks. Similarly, in the MC denoising domain, Kalantari et al. [Kalantari et al. 2015] utilize a multi-layer perceptron to learn the filter weights for cross-bilateral and cross non-local means filters from training data. Bako et al. [Bako et al. 2017] utilize a CNN to predict filtering kernels themselves for each pixel instead of the weightings of a filter shared by all pixels. This scheme is later adopted by [Back et al. 2020; Gharbi et al. 2019; Lin et al. 2020; Vogels et al. 2018], further unleashing the capabilities of kernel-based denoising. Recently, orthogonal to kernel prediction, a new type of scheme directly predicts per-pixel radiance [Lu et al. 2020; Wong and Wong 2019; Yang et al. 2018, 2019]. For example, Xu et al. [Xu et al. 2019] propose an adversarial approach for MC denoising in which a conditioned feature modulation module deeply incorporates auxiliary features into the denoising process. Other than image-space denoising methods mentioned above, recent MC denoising researchers also pay attention to sample-based denoising [Gharbi et al. 2019; Lin et al. 2021], real-time denoising [Meng et al. 2020], and interactive denoising based on video sequences rather than a single image [Chaitanya et al. 2017; Işik et al. 2021; Vogels et al. 2018]. Inspired by the remarkable denoising results in [Xu et al. 2019], we also adopt the radiance predicting scheme and adversarial learning in our model.

Transformer And Self-attention. Transformer was originally proposed in [Vaswani et al. 2017] for neural machine translation. Subsequently, the architecture has been extensively researched, modified, and applied to a variety of downstream tasks [Devlin et al. 2018; Raffel et al. 2019]. One of the most outstanding works is GPT-3 [Brown et al. 2020], a massive Transformer-based network with up to 175 billion parameters that can be applied to other tasks even without fine-tuning. Inspired by the great accomplishment of Transformer achieved in the Natural Language Processing field, an increasing number of computer vision researchers are applying the Transformer to their works, including image classification [Chen et al. 2020], super-resolution [Yang et al. 2020], and image synthesis [Zhang et al. 2019]. Although Transformer is now becoming a prominent force, we have not found Transformer-based models being applied to MC denoising.

A key component of Transformer is the self-attention mechanism, which enables effective modeling of the long-range dependencies in features within one Transformer block. This is different from CNNs that require stacking multiple convolutional layers in order to enlarge receptive field so that long-range information can be indirectly captured. Specifically, self-attention enables a feature in the embedding space to absorb the information at all other positions on the feature map in a weighted average manner, controlled by the attention scores. However, the quadratic computation and memory complexity limit its scalability. Many works [Child et al. 2019; Ho et al. 2019; Katharopoulos et al. 2020] are therefore devoted to address the quadratic issue. For example, in [Ramachandran et al. 2019], the attention computation is restricted in a local region. [Wang et al. 2020] utilize two consecutive axial self-attention layers to calculate attention in the vertical and horizontal directions respectively, significantly reducing the computation complexity. [Liu et al. 2021; Vaswani et al. 2021] propose two different memory-friendly self-attention mechanisms. We employ these pioneering works to accelerate our self-attention based MC denoising network while reducing the memory usage to make it run on a consumer GPU.

3 METHODOLOGY

Monte Carlo path tracing is one of the most important techniques in realistic rendering which however suffers from extremely high computational consumption. That is, in order to generate a high-quality image, it needs to sample a large number of paths for each pixel, resulting in hours of time cost. To mitigate this problem, a large amount of work is done to first generate a noisy image with a low path sampling rate, and the noisy image is then denoised to obtain the high-quality noise-free image, which significantly reduce the overall usage time. Auxiliary feature buffers, including normal, albedo and depth, are usually involved in the denoising process which provide helpful guidance.

In this paper, we view Monte Carlo denoising as a specific instantiation of the general image denoising problem for which non-local means filtering [Buades et al. 2005] has been proven to be very effective [Davy et al. 2019; Lefkimmiatis 2017; Xie et al. 2019]. Non-local means has also been popularly adopted for Monte Carlo denoising [Kalantari et al. 2015; Rousselle et al. 2012; Zimmer et al. 2015]. Inspired by the fact that the recently developed self-attention mechanism for machine translation [Vaswani et al. 2017] is essentially a special type of non-local means filter that operates in a high-dimensional embedding space (see [Wang et al. 2018] or the supplemental of this paper), we propose a self-attention based Monte Carlo denoising method. To better utilize the information in the auxiliary buffers, we propose auxiliary feature guided self-attention, which computes the interactions between pixels in an image not only by the content of the image but also by that of the auxiliary buffers. In the following, before describing our self-attention based MC denoising network, we give some mathematical background of the self-attention mechanism.

3.1 Transformer and Self-Attention

Self-attention mechanism was first proposed by Vaswani et al. [Vaswani et al. 2017] for the task of machine translation. Based on self-attention, they proposed a new network architecture called "Transformer" which comprises a series of Transformer blocks (we call them self-attention based neural blocks in the above) that perform self-attention computations.

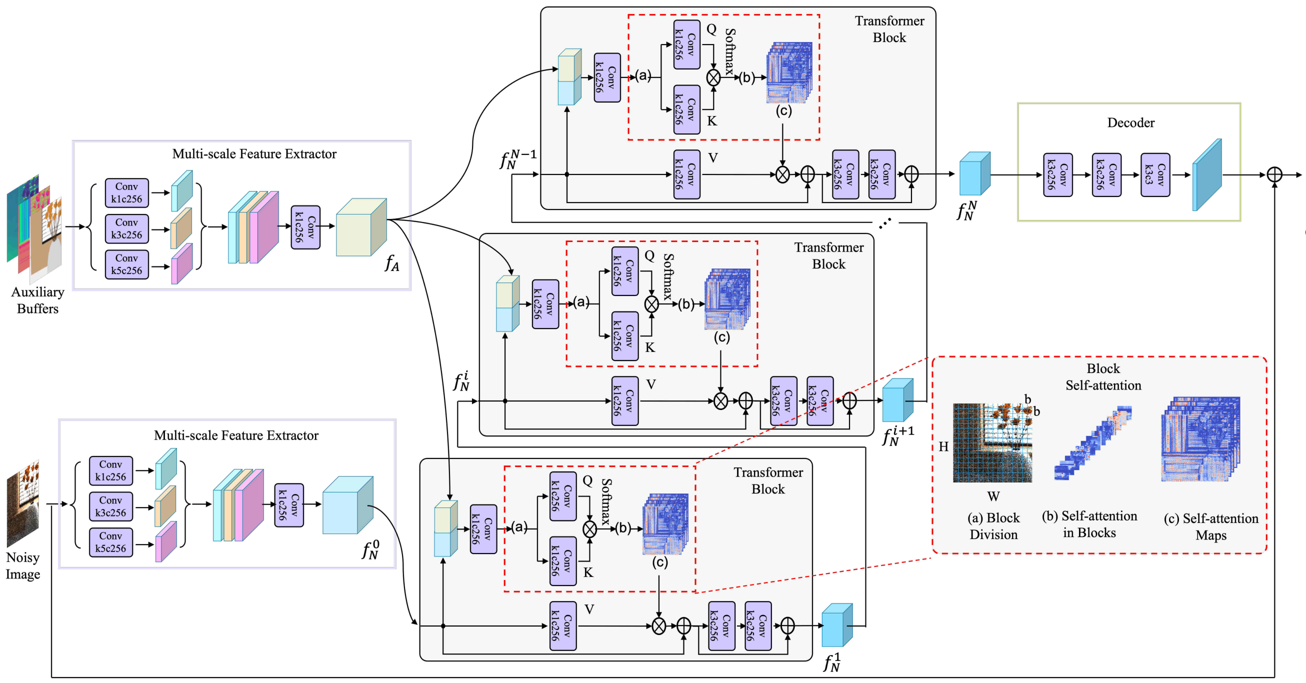

Fig. 2. Overview of our self-attention based MC denoising model. Given a noisy image and three auxiliary buffers, our model outputs the residual between the input noisy image and the target high-quality noise-free image. The model consists of three main components: multi-scale feature extractors, Transformer blocks, and a decoder. The multi-scale feature extractors extract features from the input noisy image and auxiliary buffers. Then five sequentially connected transformer blocks are used to process noisy image features by auxiliary feature guided self-attention. The output of the last transformer block is finally processed by a simple decoder to obtain the residual. We implement block self-attention for computation acceleration and memory saving. The whole model is trained in an adversarial manner. For simplicity, we omit the the discriminator in this figure.

Let X^{i-1} be the input to the ith Transformer block, the self-attention operator is defined as:

$$\mathrm{SA}(X^{i-1}) = \mathrm{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{1}$$

where,

$$Q = W_Q X^{i-1}, K = W_K X^{i-1}, V = W_V X^{i-1},$$

with W_Q, W_K, and W_V being the learnable linear transformations, and d_k being the feature channel dimension of Q and K. As can be seen, the term QK^T gives attention scores between the vector slots of Q and K, which are finally used to average vector slots of V. Since Q, K, V are all computed from X^{i-1}, the attention operator defined in Equation 1 is called "self-attention", i.e., the attention between the vector slots of X^{i-1} themselves.

With the self-attention operator, a Transformer block is easily implemented as:

$$\begin{aligned}\hat{X}^{i-1} &= \mathrm{LN}\left(\mathrm{SA}\left(X^{i-1}\right)\right) + X^{i-1},\\ X^{i} &= \mathrm{LN}\left(\mathrm{FFN}\left(\hat{X}^{i-1}\right)\right) + \hat{X}^{i-1},\end{aligned} \tag{2}$$

where LN denotes layer normalization [Ba et al. 2016], FFN denotes a feed-forward network, and X^i is the output of the ith Transformer block.

3.2 Self-Attention based Monte Carlo Denoising

Figure 2 shows the overview of the proposed self-attention based Monte Carlo denoising network, comprising three main components: multi-scale feature extractors, Transformer blocks with auxiliary feature guided self-attention mechanism, and a decoder. Our network takes a noisy image and three auxiliary buffers (including normal, depth, and albedo) as input, and outputs the residual between the input noisy image and the target high-quality image. Note that most of previous approaches [Bako et al. 2017; Lu et al. 2020; Xu et al. 2019] separate the noisy image into diffuse and specular components and process them separately. Since self-attention brings heavy computation and memory loads, we directly process the noisy image in a single pipeline. Our model is trained adversarially, which means the network shown in Figure 2 is a generator of a Generative Adversarial Network [Goodfellow et al. 2014], but for simplicity the discriminator is omitted in the figure.

3.2.1 Multi-scale feature extractor. Firstly, we need to transform the input noisy image and auxiliary buffers from image space to feature space. This is usually achieved by convolutional neural networks. For example, one can simply use a convolutional layer with 256 kernels of size 3×3 to transform a noisy image in space $\mathbb{R}^{H \times W \times 3}$ to space $\mathbb{R}^{H \times W \times 256}$, where W and H are width and height of the input image. However, we find that features calculated by a single convolutional layer limit the performance of subsequent self-attention

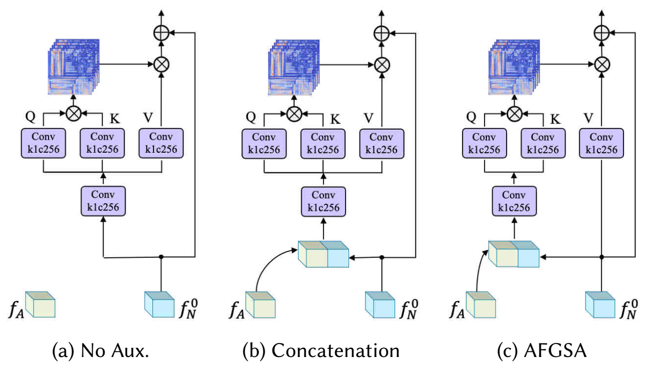

Fig. 3. (a) Auxiliary features are not utilized. (b) Auxiliary features are concatenated with noisy image features before feeding into self-attention module. (c) The proposed auxiliary feature guided self attention in which the auxiliary features are just involved in the calculation of Q and K.

modules since features extracted from a single-scale receptive field cannot fully reflect the original information of the image. To mitigate this problem, we propose multi-scale feature extractors as shown in Figure 2. Specifically, we design 3 convolutional layers all with 256 kernels but the kernel size of the first convolutional layer is 1×1, the second is 3×3, and the third is 5×5. The three convolutional layers are used to extract features at different scales of receptive fields, a strategy that has also been adopted in Inception [Szegedy et al. 2015]. By multi-scale feature extraction, we provide self-attention modules with more raw features, facilitating the self-attention mechanism to discover more useful information. Before inputting to self-attention modules, the multi-scale features are concatenated together in the channel dimension, and further processed by a convolutional layer with 256 kernels of size 1×1. The input noisy image and auxiliary buffers are processed by two different multi-scale feature extractors. For noisy image, we obtain a feature map, denoted by f_N^0, in space $\mathbb{R}^{H \times W \times 256}$. For auxiliary features, the obtained feature map, defined as f_A, is also in space $\mathbb{R}^{H \times W \times 256}$.

3.2.2 *Auxiliary feature guided self-attention.*

After the multi-scale feature extraction stage, we sequentially stack five Transformer blocks to extract more representative features. We use the standard Transformer pipeline as defined in Equation 2 except for the layer normalization (in practice adding the layer normalization yields worse results). But for the self-attention operator, there are several design choices to consider regarding how to make effective use of the auxiliary features.

Firstly, as shown in Fig. 3 (a), one can simply ignore the auxiliary features and use the standard self-attention operator defined in Equation 1, for which the computation of a Transformer block is:

$$\begin{aligned} \hat{f}_N^{i-1} &= \mathrm{SA}\left(f_N^{i-1}\right) + f_N^{i-1}, \\ f_N^{i} &= \mathrm{FFN}\left(\hat{f}_N^{i-1}\right) + \hat{f}_N^{i-1}, \end{aligned} \tag{3}$$

where f_N^{i-1} is the input to the ith Transformer block and f_N^i is the output, with f_N^0 being the features computed from the noisy image in the multi-scale feature extraction stage. In practice, this scheme yields relatively poor denoising results because of not using the helpful auxiliary information. A straightforward way to incorporate the auxiliary features f_A into the denoising process is replacing $\mathrm{SA}(f_N^{i-1})$ of Equation 3 by $\mathrm{SA}((f_N^{i-1}; f_A))$, where $(f_N^{i-1}; f_A)$ indicates the concatenation of f_N^{i-1} and f_A, as shown in Figure 3 (b). In this scheme, both the input noisy image and the auxiliary buffers are involved into the computation of Q, K and V, which is not intuitively reasonable. We argue that the auxiliary buffers should only be used to compute Q and K but not V. The reasons are two-fold. Firstly, the auxiliary buffers, including depth, normal and albedo, usually have clearer edges than the noisy image, making them very suitable for the computation of edge-preserving weighting (or attention) scores, i.e., QK^T. Secondly, since V actually represents image pixel values, it should not be contaminated by the auxiliary features. Based on this observation, we propose the Auxiliary Feature Guided Self-Attention (AFGSA) operator:

$$\mathrm{AFGSA}(f_N^{i-1}, f_A) = \mathrm{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V, \tag{4}$$

where,

$$\begin{aligned} Q &= W_Q\left(f_N^{i-1}; f_A\right), \\ K &= W_K\left(f_N^{i-1}; f_A\right), \\ V &= W_V\left(f_N^{i-1}\right), \end{aligned}$$

in which Q and K are computed from the concatenation of both the image and auxiliary features, while V is just from the image features. Our AFGSA based Transformer block, as shown in Figure 3 (c), is then defined as:

$$\begin{aligned} \hat{f}_N^{i-1} &= \mathrm{AFGSA}\left(f_N^{i-1}, f_A\right) + f_N^{i-1}, \\ f_N^{i} &= \mathrm{FFN}\left(\hat{f}_N^{i-1}\right) + \hat{f}_N^{i-1}. \end{aligned} \tag{5}$$

In our implementation, Q, K, and V are computed by different 1×1 convolutional layers, FFN is a feed-forward network comprising two 3×3 convolutional layers.

3.2.3 *Decoder.*

After five Transformer blocks, the obtained feature map is f_N^5. We design a decoder to transform it from the feature space to the image space. The structure of the decoder is very simple, sequentially stacking three 3×3 convolutional layers among which the first two have 256 kernels and the last one has 3 kernels.

3.3 Network Implementation and Acceleration

In Figure 2, we have annotated the implementation details of all the convolutional layers. For example, "Conv k1c256" means the convolutional layer has 256 kernels of size 1×1. For all the convolutional layers, the padding type is set to "reflect" and the stride is set to 1, while normalization layer is not adopted. Let H and W be the spatial shape of the input noisy and auxiliary images, the two feature maps f_N^0 and f_A outputted by the multi-scale feature extractors are both of size $H \times W \times C_{fea}$ (where H and W are both 128 at training, and $C_{fea} = 256$). Q, K, and V computed in self-attention blocks are also of size $H \times W \times C_{fea}$. Note that in order to avoid losing any spatial information that is essential for image denoising, we do not perform any type of downsampling operation in the model.

A problem is that the self-attention defined in Equation 4 is in fact a global self-attention, which is notorious for its high computation complexity and large memory usage, as the product QK^T requires each element in the feature map to compute attention to all the other elements (as shown in Figure 4 (a)):

$$f_{i,j} = \sum_{a \in [1,H], b \in [1,W]} \text{Softmax}_{a,b} \left(\frac{q_{i,j} k_{a,b}^{\top}}{\sqrt{d_k}} \right) v_{a,b}, \tag{6}$$

where $q_{i,j}$ indicates the query feature at position (i, j), $k_{a,b}$ represents the key feature at position (a, b), $v_{a,b}$ stands for the value feature at position (a, b), and $f_{i,j}$ is the resulting feature at position (i, j). The computation budget and memory needed for storing the attention matrix of QK^T are quadratic to the spatial dimensions of the input feature map, i.e., $O(H^2W^2)$. As a result, global self-attention can only be used on downsampled or small feature maps, limiting its scalability. In the following, we introduce several ways to optimize it, from local self-attention, to block self-attention.

3.3.1 Local self-attention. In order to reduce computation, one straightforward way is to restrict the attention computation in a local region (as shown in Figure 4 (b)) rather than the global feature map [Ramachandran et al. 2019]:

$$f_{i,j} = \sum_{a \in [i - \frac{m}{2}, i + \frac{m}{2}], b \in [j - \frac{m}{2}, j + \frac{m}{2}]} \text{Softmax}_{a,b} \left(\frac{q_{i,j} k_{a,b}^{\top}}{\sqrt{d_k}} \right) v_{a,b}, \tag{7}$$

where m indicates the size of the local neighborhood. Adding such local region constraint significantly reduces the theoretical computational complexity from $O(H^2W^2)$ to $O(HWm^2)$. However, although this effectively reduces computation budget, it consumes even more memory. That is because in practice we have to extract all local regions and store them in memory before we can perform self-attention computations in the local neighborhoods. These local regions are however seriously overlapped with each other, e.g., local patches of two adjacent pixels share an overlapping area of $m \times (m - 1)$, which are memory-wasteful.

3.3.2 Block self-attention. In order to balance the computation and memory usage, we can divide the input feature map into non-overlapping blocks with spatial size $b \times b$, resulting in $\frac{H}{b} \times \frac{W}{b}$ blocks. Then we can run global self-attention in each block (as shown in Figure 4 (c)), the computation complexity of which is only $O(b^2b^2)$. The whole complexity is $O(\frac{H}{b}\frac{W}{b}b^4) = O(HWb^2)$. Since there is no overlapping between blocks, no memory wastes when storing the blocks. However, this prohibits the transfer of information between adjacent blocks. For example, for pixels near the boundary of a block, despite being very close to some pixels in adjacent blocks, the attention calculation of them cannot gather information from adjacent blocks. This results in color discrepancy on the boundaries of each block. Vaswani et al. [Vaswani et al. 2021] solve this problem by expanding a band of pixels, called "halo", around each block (as shown in Figure 4 (d)), making the size of the block become $(b + 2h) \times (b + 2h)$ where h is the width of the band. The band pixels are copied from adjacent blocks if they exist, otherwise are padded with zero. By the expanded "band", they actually add some overlapping between adjacent blocks. Although this introduces a little waste of memory, it allows a certain extent of information exchanging between adjacent blocks, enabling smooth information transition between them. We follow this scheme in implementing our self-attention mechanism. But note that we just need to expand K and V while for Q the expansion is prohibited.

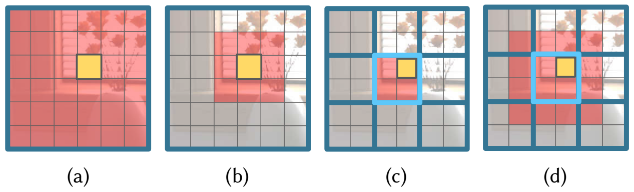

Fig. 4. (a) Global self-attention. The receptive field (red) covers the whole feature map. (b) Local self-attention. The receptive field is confined in the local window centered at the query point (yellow). (c) Block self-attention. The receptive field is confined in the block containing the query point. (d) Block self-attention with expanded band. Besides the block containing the query point, the receptive field is expanded into the surrounding band occupying pixels of other blocks.

3.4 Training Losses

To ensure pixel-level accuracy and perceptual quality, we use two types of loss functions as our optimization objectives: a pixel reconstruction loss and an adversarial loss. The overall loss function is:

$$L = \lambda_{rec} L_{rec} + \lambda_{adv} L_{adv}, \tag{8}$$

where λ_{rec} and λ_{adv} are two hyper-parameters that control the balance of the two terms.

Pixel reconstruction loss. We adopt the L_1 distance between the image generated by the proposed model and the ground truth as our pixel reconstruction loss:

$$L_{rec} = \left\| \tilde{I} - I_{gt} \right\|_1, \tag{9}$$

where $\tilde{I}$ is the denoised output of our network and I_{gt} is the ground-truth.

Adversarial loss. Following [Xu et al. 2019], we view the proposed self-attention based MC denoising network as a generator and train it in an adversarial manner. The discriminator used in this paper is borrowed from [Xu et al. 2019]. To stabilize the training, WGAN-GP [Gulrajani et al. 2017] loss is minimized:

$$\begin{aligned} L_{adv} = & \min_{G} \max_{D} \mathop{\mathbb{E}}_{\tilde{I} \sim \mathbb{P}_g} [D(\tilde{I})] - \mathop{\mathbb{E}}_{I_{gt} \sim \mathbb{P}_r} [D(I_{gt})] \\ & + \lambda \mathop{\mathbb{E}}_{\hat{I} \sim \mathbb{P}_{\hat{I}}} [(\left\| \nabla_{\hat{I}} D(\hat{I}) \right\|_2 - 1)^2], \end{aligned} \tag{10}$$

where G indicates the generator, D represents the discriminator, λ is a balance weight, $\hat{I}$ stands for the interpolation between the real sample I_{gt} and the generated sample $\tilde{I}$.

4 EVALUATION

In the following we evaluate our method and compare it with the previous state-of-the-art approaches. The code and dataset can be

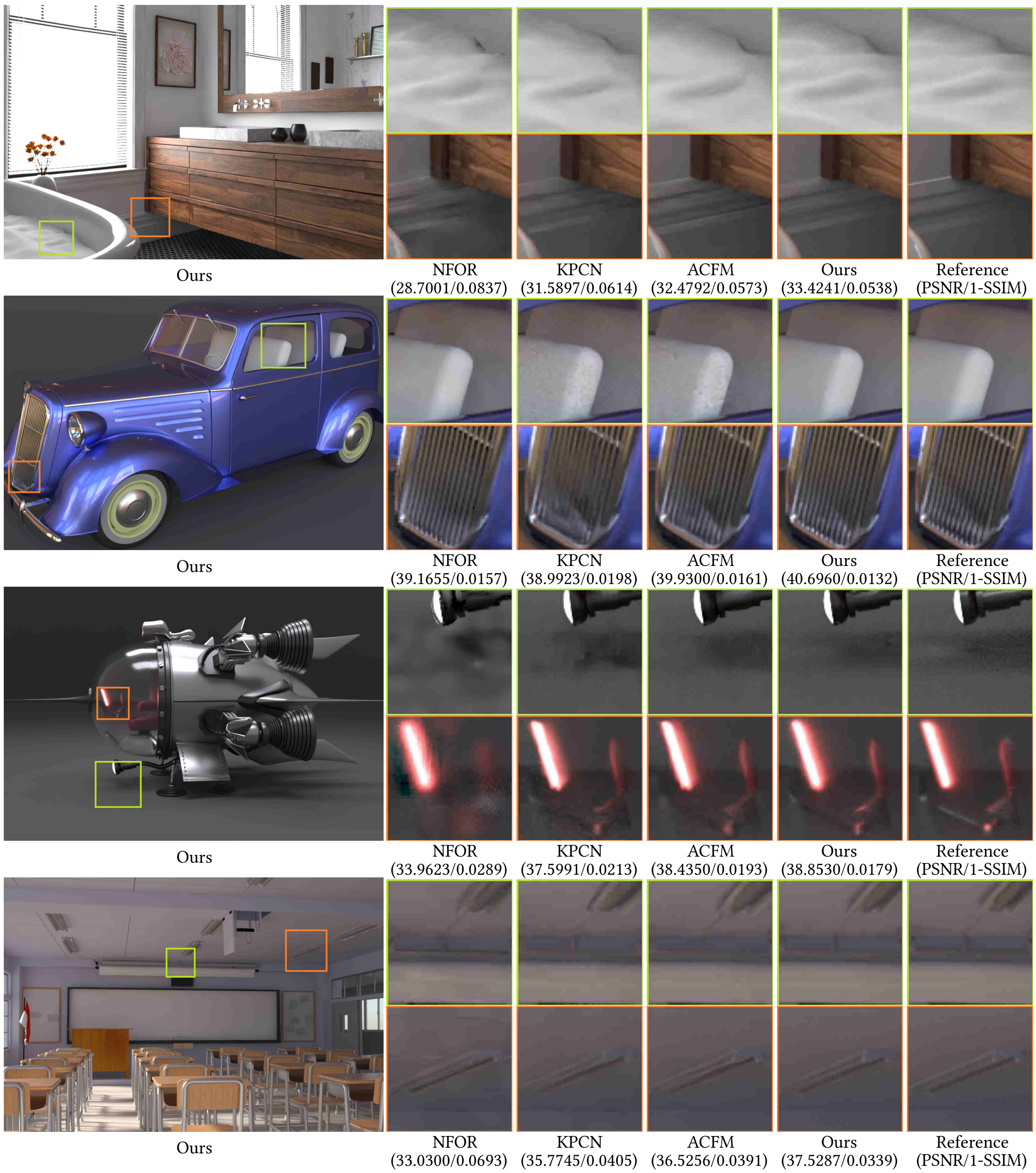

Fig. 5. Qualitative comparisons with NFOR [Bitterli et al. 2016], KPCN [Bako et al. 2017] and ACFM [Xu et al. 2019]. The noisy inputs are rendered at 32 spp. Our method can better restore details and structures of noisy images.

Table 1. Quantitative comparisons with NFOR [Bitterli et al. 2016], KPCN [Bako et al. 2017] and ACFM [Xu et al. 2019] on the test set. Learning-based model are trained on 32 spp images. The "re" after a model name means the model is retrained by ourselves. For ACFM, the results of the authors' published model are also reported. The RMSE, PSNR, and 1-SSIM metrics are measured for each compared model. Bold numbers are the best, while numbers with a hat are the second best. The differences (by subtraction) between the best and the second best numbers are calculated.

Model	SPP	RMSE(10^{-3})↓	PSNR↑	1-SSIM↓
NFOR	4	27.469	29.2585	0.1447
	8	16.010	31.6135	0.1016
	16	8.460	33.6525	0.0751
	32	4.682	35.7583	0.0544
	128	1.748	39.2909	0.0323
KPCN (re)	4	11.512	30.8417	0.1219
	8	6.631	32.9486	0.0888
	16	4.132	34.9944	0.0669
	32	2.840	36.6687	0.0509
	128	1.443	39.6035	0.0300
ACFM	4	10.668	31.5617	0.1166
	8	6.019	33.6399	0.0823
	16	3.697	35.6662	0.0593
	32	2.383	37.4461	0.0446
	128	1.192	40.2327	**0.0281**(-0.0002)
ACFM (re)	4	$\hat{10.217}$	$\hat{31.9081}$	$\hat{0.1076}$
	8	$\hat{5.365}$	$\hat{33.9380}$	$\hat{0.0781}$
	16	$\hat{3.545}$	$\hat{35.7812}$	$\hat{0.0586}$
	32	$\hat{2.369}$	$\hat{37.5137}$	$\hat{0.0443}$
	128	$\hat{1.189}$	$\hat{40.2571}$	$\hat{0.0283}$
Ours	4	**9.979**(-0.24)	**32.3770**(+0.47)	**0.1023**(-0.053)
	8	**4.855**(-0.51)	**34.6834**(+0.75)	**0.0713**(-0.068)
	16	**2.766**(-0.78)	**36.7625**(+0.98)	**0.0521**(-0.065)
	32	**1.839**(-0.53)	**38.3812**(+0.87)	**0.0410**(-0.033)
	128	**1.051**(-0.14)	**40.8028**(+0.55)	0.0286

found at https://github.com/Aatr0x13/MC-Denoising-via-Auxiliary-Feature-Guided-Self-Attention.

4.1 Experimental Settings

Dataset. We use the dataset released by ACFM [Xu et al. 2019] for training and validation. The dataset contains 1109 shots of 8 scenes provided by [Bitterli 2016] rendered with Tungsten. Each shot provides a noisy color image, a noisy diffuse image, a noisy specular image and three noisy auxiliary feature buffers including depth, normal, and albedo. The corresponding ground truths of these noisy images are also provided. All the noisy images are rendered at 32 samples per pixel (spp) while the ground truth images are rendered at 32k spp. Following [Xu et al. 2019], we randomly choose 95% shots as the training dataset, while the remaining 5% shots as the validation dataset. All shots are divided into patches of size 128×128 by the sampling strategy in [Bako et al. 2017], obtaining 286,649 patches in total, among which 272,230 come from the training shots and 14,419 from the validation ones.

As stated above, the dataset provided by ACFM [Xu et al. 2019] contains only noisy images rendered at 32 spp. In order to conduct comparisons more adequately, we render a test set by ourselves. Specifically, the newly rendered test dataset contains noisy images rendered at 4, 8, 16, 32, 128, 256, 512, 1024 spps for each of the 14 default Tungsten scenes [Bitterli 2016], whose ground-truth images

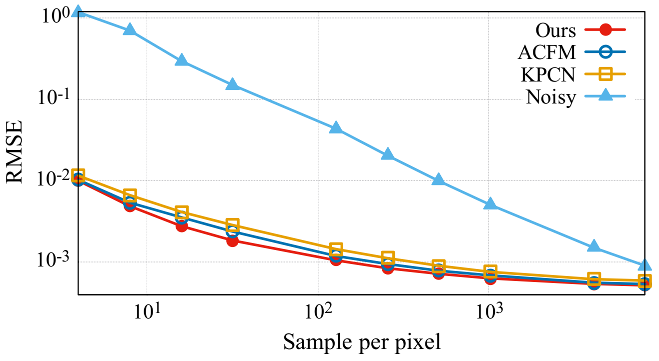

Fig. 6. RMSE convergence plots of MC path tracing (Noisy), KPCN [Bako et al. 2017], ACFM [Xu et al. 2019], and our method (Ours), which show the average RMSE values of the four methods on the test dataset as the sampling rate increases from 4 spp to 8192 spp.

are provided with the scene files. The format of these images is kept the same as that of the ACFM dataset. During testing, we also split them into patches of 128×128 size, resulting in 3192 patches. Note that although we use 128×128 patches during training, validation, and testing for comparison, our model can accept larger images as input for practical use.

Pre-and post-processing. Our method takes a noisy image as input rather than diffuse and specular components separately. Like what has done to the specular component in [Bako et al. 2017], we apply a log transform to the noisy image before feeding it into our model. Correspondingly, the output of our model is applied with an inverse-log transformation.

Comparison Metrics. We use three metrics to evaluate the denoising results, which are Relative Mean Square Error (RMSE), Peak Signal-to-Noise Ratio (PSNR) and One Minus Structural Similarity Index Measure (1-SSIM).

Training Settings. Our model has 5 Transformer blocks. For the block self-attention, we divide feature maps into blocks of size $b = 8$. The expanded band is of size $h = 3$. We implement our method using PyTorch [Paszke et al. 2019] and also Jittor [Hu et al. 2020], while only the experimental data of PyTorch are provided below. We run our method on a single NVIDIA GeForce RTX 3090 GPU with 24GB memory. Some variants of our model in the ablation study demand more than 24GB memory at training. In order to circumvent this problem, we utilize the technique of gradient checkpoint of PyTorch to reduce memory usage. During training, Adam optimizer is used, learning rate is set to 1e-4 initially and halved every 3 epochs for a total of 12 epochs, and the batch size is set to 8. The weights in the loss function are set as $\lambda_{rec} = 1$, $\lambda_{adv} = 5e-3$ and $\lambda = 10$.

4.2 Comparison with State-of-The-Art Methods

We compare our model with a traditional method NFOR [Bitterli et al. 2016] and two learning-based image-space MC denoising methods including KPCN [Bako et al. 2017] and ACFM [Xu et al. 2019]. NFOR is a traditional non-local denoising method and is shipped with the tungsten renderer. KPCN is based on kernel prediction and ACFM is the current state-of-the-art based on radiance prediction. Both of them have published their codes and model weights. The published KPCN model was trained on 128 spp. In order to compare with it on more difficult 32 spp images, we re-train KPCN on the same training set of ours, using the settings suggested by the authors of KPCN. We also re-train ACFM, and our re-trained model performs better than the published ACFM model.

Quantitative comparison results are shown in Table 1. All the deep learning methods are trained on 32 spp images, but evaluated on 4, 8, 16, 32, 128, 256, 512, and 1024 spp images. Due to space limit, we only show the results of 4 to 128 spps, and please refer to the supplemental material for the results of higher spps. For KPCN, we report the results by our re-trained model. For ACFM, we report both the results computed by the published model and our re-trained model. In the table, bold numbers indicate the best results. As can be seen, for 4 to 32 spp inputs and all the three comparison metrics RMSE, PSNR, and 1-SSIM, our method achieves the best results. One exception occurs for the 128 spp input and the 1-SSIM metric for which ACFM achieves the best results, but our result is comparable to that of ACFM (0.0286 vs. 0.0281). We also mark the second best results in the table, i.e., the numbers with a hat on them, and compute the difference (by subtraction) between the best and the second best numbers. The differences tell that our method surpasses the other methods the most for the 16 spp inputs.

In Figure 6, we plot the RMSE convergence curves of KPCN, ACFM and our method to illustrate the RMSE values of the three methods over the entire test set as the input noisy image spp increases. The "Noisy" curve shows the convergence plot of the MC path tracing algorithm, while the other three curves show the convergence plots of KPCN, ACFM, and our method, respectively. The convergence plots further confirm the effectiveness of our method.

In Figure 5, we choose four different scenes on which qualitative comparisons are shown. They are "Bathroom", "VintageCar", "Spaceship" and "Classroom". These scenes contain mirrors, reflections, refractions, dim regions and tiny structures, covering as many aspects as possible to fully test a model's denoising performance. As can be seen from Figure 5, our model not only accurately recovers low-frequency features, but also does a good job on restoring high-frequency details. For example, in the green box of "VintageCar", KPCN and ACFM still have visible noise in the seat, while our denoised image is clear and smooth; for the heat sink in the orange box, both KPCN and ACFM have varying degrees of blurring, while our denoised result has the structure and edge details well restored.

Although our model has high computational complexity and consumes more memory during training, the time and memory usage drop much at test time since there is no need to compute gradients and also the discriminator is discarded. Specifically, at test time, the time and memory costs for a batch (the batch size is 8) of 128×128 images by KPCN [Bako et al. 2017] are 327.5ms and 5.5GB, while those by [Xu et al. 2019] are 316.1ms and 2.7GB, and by our method are 382.5ms and 5.2GB.

5 ABLATION STUDY

In this section, we modify the proposed model in different ways and investigate the effectiveness of each design choice of our method.

Table 2. Ablation study on different types of feature extractor architectures, where, for example, 3×3 means a single convolutional layer of kernel size 3×3 is used for feature extraction, while 1357 means four parallel convolutional layers of kernel sizes 1×1, 3×3, 5×5 and 7×7 are used to extract features.

Encoder	RMSE(10^{-3})↓	PSNR↑	1-SSIM↓
Multi-scale (1357)	2.807	37.0931	0.0586
Multi-scale (135)	**2.616**	**37.4551**	**0.0531**
Multi-scale (357)	2.871	36.9197	0.0618
3×3	2.817	37.1675	0.0594
5×5	2.821	37.0746	0.0609
7×7	2.929	36.7243	0.0661

Table 3. Ablation study on different number of Transformer blocks.

Num.T.	RMSE↓	PSNR↑	1-SSIM↓
4	3.089	37.2391	0.0560
5	**2.616**	**37.4551**	**0.0531**
6	2.701	37.1491	0.0584

We use the dataset provided by ACFM [Xu et al. 2019] for the ablation study, among which 60% shots are used for training, 20% for validation, and 20% for testing. Directly performing ablation studies on the whole dataset is too time-consuming. Instead, we sample subsets from the corresponding full datasets, obtaining 8414, 2770, and 2793 patches for training, validation, and testing, respectively. Instead of 12, we train the ablation models for 24 epochs. In the following, when performing ablation experiments for some part of our model, the other parts are kept the same as the full model.

5.1 Multi-Scale Feature Extractor

Firstly, we analyze the role of the multi-scale feature extractor which is one of the key parts of our network. Six settings are compared, including single-scale feature extraction with convolutional kernel sizes of 3, 5, and 7 respectively, and multi-scale feature extraction with kernel size combinations of (1,3,5), (1,3,5,7) and (3,5,7). Their denoising results are shown in Table 2.

For single-scale schemes, 3×3 and 5×5 have similar performance and outperform 7×7. We conjecture that a 7×7 kernel may over-smooth the original pixels. Similar performance drop can be observed from the multi-scale settings of (1,3,5,7) and (3,5,7) that include 7×7 convolutional kernels, with (3,5,7) showing a more severe performance degradation. Therefore, we finally adopt the setting of (1,3,5) which provides both the original information of pixels (by 1×1 kernels) and their neighborhood information (by 3×3 and 5×5 kernels) to the later self-attention blocks.

5.2 Number Of Transformer Blocks

The number of Transformer blocks has a non-negligible impact on the denoising accuracy, computation time and memory usage of our model. We conduct three experiments having 4, 5, and 6 Transformer blocks each, the results of which are presented in Table 3. As can be seen, the best performance is achieved when 5 Transformer blocks are used. When there are 4 Transformer blocks, the denoising process may be incomplete which yields worse performance. The results of 6 blocks show that more blocks do not certainly guarantee better performance. But with the increase of the number of blocks,

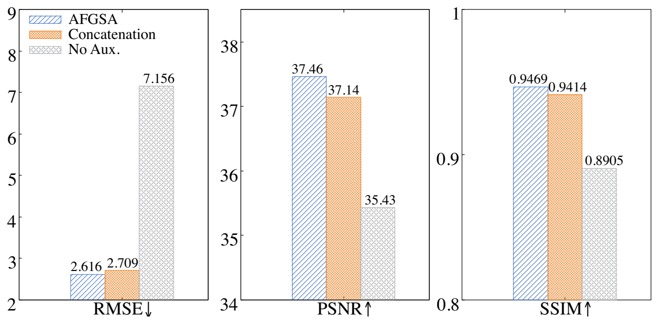

Fig. 7. Ablation study on the proposed auxiliary feature guided self-attention (AFGSA). AFGSA is compared with two variant models. One is without auxiliary input (No Aux.), and the other one concatenates noisy image and auxiliary features together (Concatenation).

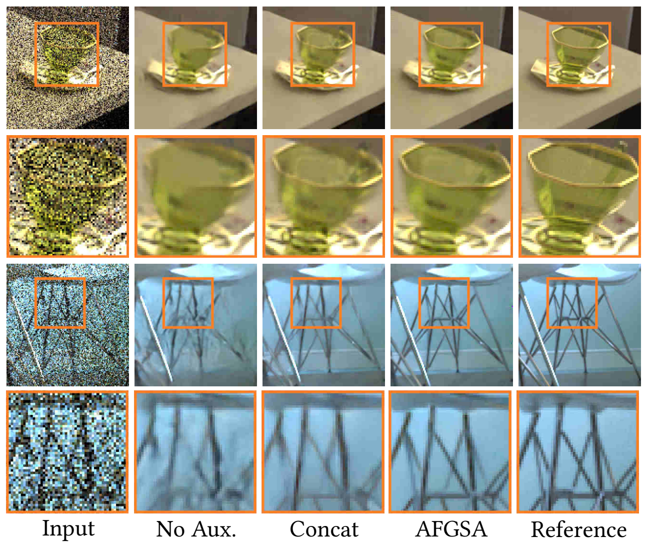

Fig. 8. Denoising results of different ways of utilizing auxiliary features in the self-attention mechanism. "No Aux." indicates the model does not use auxiliary features at all; "Concat" stands for directly using the concatenation of noisy and auxiliary features as input; "AFGSA" represents our proposed method.

the time and memory consumption at training and testing time steadily increases.

5.3 Auxiliary Feature Guided Self-Attention

We proposed the auxiliary feature guided self-attention module (AFGSA) in Section 3.2.2 which effectively involves the auxiliary features into the attention computation process. To verify its contribution, we compare it with two variants. The first variant (No Aux.) does not use auxiliary features at all, and the second (Concatenation) concatenates noisy image and auxiliary features together and then feeds them into the Transformer blocks. The ablation numerical results are illustrated in Figure 7, showing that AFGSA yields the best denoising performance. Figure 8 shows some denoising examples

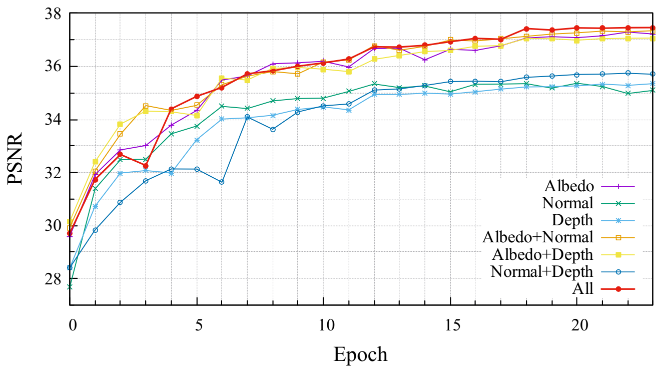

Fig. 9. Comparison of the denoising performance of different combinations of auxiliary features in terms of PSNR.

of the two variants and the AFGSA model. As can be seen, AFGSA produces the most visually pleasing denoising results with sharper edges and clearer structures. The reasons have been described in Section 3.2.2: AFGSA separates the auxiliary and image features in computing Q, K and V, only allowing the auxiliary features to participate the computation of Q and K. This on one hand takes advantage of the auxiliary features for edge-preserving filtering weights computation, and on the other hand prevents them from contaminating the computation pipeline of the pixel features.

5.4 Importance of Each Auxiliary Feature Buffer

To investigate which auxiliary feature plays a more important role in guiding the denoising process, we set up seven experiments, each using a different group of auxiliary feature buffers. The denoising performance in terms of PSNR as the number of training iterations increases is illustrated in Figure 9. The self-attention guided by all the auxiliary features achieves the best results. “Albedo+Normal”, “Albedo+Depth” and “Albedo” are belonging to the second best camp. “Normal+Depth”, “Normal”, and “Depth” belong to the third camp. This ablation study shows that albedo is essential for good denoising, while depth has the least effect.

5.5 Local and Block Self-Attention

Section 3.3 has introduced the local and block self-attentions for accelerating our self-attention computation. Here we analyze their performance under different settings. For local self-attention, the memory is easily blown up because of the large amount of duplicate content that are extracted and stored for local self-attention computation. During training, the GeForce RTX 3090 GPU with 24GB memory can only support a maximum local window of size 9. We set up two experiments for the local method, using windows of size 7 and 9 respectively. For the block self-attention with expanded band, the memory utilization is lower. With the same memory budget, the block method can perform attention calculation over a larger local region. We experiment with two settings for the block method, one with a small block size of $b = 8$ and band size of $h = 3$, the other one with a large block size of $b = 16$ and band size of $h = 5$.

Table 4. Ablation study on different settings of local and block self-attention, where m is the size of the local window centered around each query pixel. b and h represent block size and band width defined in the block self-attention.

Method	Configuration	RMSE(10^{-3})↓	PSNR↑	1-SSIM↓	Time(ms)↓	Memory(GB)↓
Local Window	m=7	5.187	34.7629	0.0835	**348.3**	9.2
	m=9	5.163	34.8305	0.0829	404.8	14.3
Block with Band	b=8 h=3	2.616	37.4551	**0.0531**	382.5	**5.2**
	b=16 h=5	**2.532**	**37.5019**	0.0558	477.4	7.8

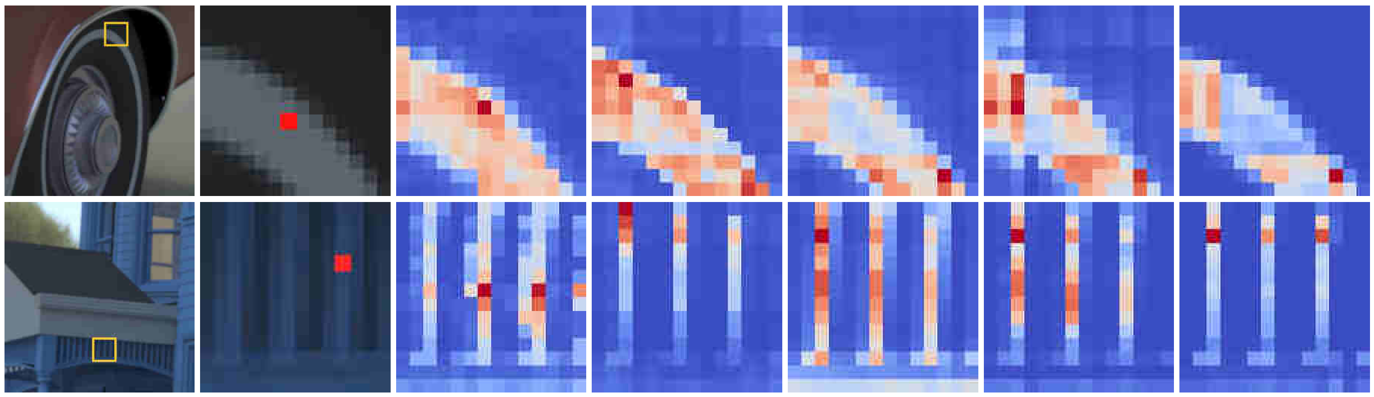

Fig. 10. From left to right: the denoised images by our model with yellow boxes specifying selected patches; zoomed-in patches with red points representing query pixels for which attention maps to be visualized; self-attention maps of all five Transformer blocks, in which a deeper red color means a higher attention score while a deeper blue color represents a lower attention score.

Table 4 gives the ablation results which show that block method outperforms the local method by a large margin while consuming less time and memory. For example, the setting of $b = 8$ and $h = 3$ needs to compute attention in a larger local block of size 14×14 $(b + 2 \times h)$, but it consumes less time and memory than the setting of $m = 9$ for the local method which just needs to compute attention in a 9×9 window. We observe that when using a larger block size ($b = 16$, $h = 5$), metrics of RMSE and PSNR become better, but 1-SSIM becomes a little worse. We conjecture this is because larger receptive field may lead to over-smoothing which destroys structures of image.

6 SELF-ATTENTION MAPS VISUALIZATION

In Figure 10, we visualize the attention maps to show the effects of our Transformer blocks. The first column shows the chosen two examples from which we select two patches indicated by the yellow boxes. The second column shows the zoomed-in patches. There is a red point in each patch. The third to seventh columns present the attention maps of the red point with respect to all the other points in the corresponding patch computed by the five Transformer blocks of our model, where a deeper red color means a stronger relationship between points while a deeper blue means a weaker relationship.

In the first example, the chosen patch contains three regions, one white region from wheel hub, and two black regions from the tire separated by the white region. The red point is selected from the white region. As can be seen in the attention maps, the red point always has higher attention scores in the white region. In early attention maps, some pixels in the dark regions have colors other than deep blue, but in the last attention map they become deep blue. This tells that the later self-attention blocks have learned that the red point belongs to the white region and therefore they can stop the information in other regions from denoising the red point. In the second example, the problem becomes more challenging since the red point is similar to points in distant regions, but the Transformer blocks still recognize these distant regions and assign high attention scores to them.

7 LIMITATIONS

Despite being optimized, our model still consumes a significant amount of memory. When processing a 1280×720 image, we need to split it into two halves and denoise them separately to avoid running out of memory, and finally combine them together to form the output image. Such a problem also prevents us from processing specular and diffuse components in two parallel pipelines, which however may bring additional accuracy gain.

As shown in the first and second rows of Figure 11, our model restores more details than KPCN and ACFM. But compared with the ground truth, our results still lose some details. In the third row, it seems that our method is too aggressive to recover textures but ignores the specular. This can be solved by separating diffuse component from specular, as verified in [Xu et al. 2019]. The last row shows another example regarding our method being too aggressive on recovering image structures (e.g., shadow boundaries) but leading to wrinkle-like artifact.

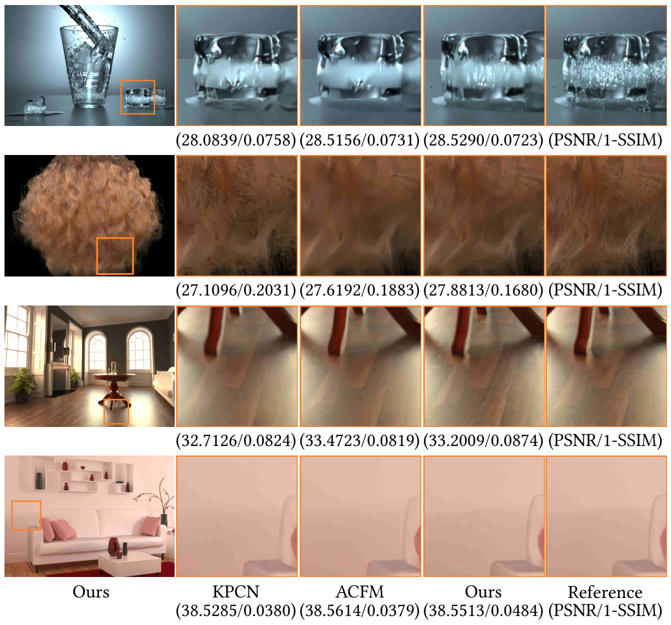
(28.0839/0.0758) (28.5156/0.0731) (28.5290/0.0723) (PSNR/1-SSIM)
(27.1096/0.2031) (27.6192/0.1883) (27.8813/0.1680) (PSNR/1-SSIM)
(32.7126/0.0824) (33.4723/0.0819) (33.2009/0.0874) (PSNR/1-SSIM)
Ours
KPCN
ACFM
Ours
Reference
(38.5285/0.0380) (38.5614/0.0379) (38.5513/0.0484) (PSNR/1-SSIM)

Fig. 11. Limitations of our method. For the "GlassOfWater" and "CurlyHair" scenes, our results are better than KPCN and ACFM, but there is still some degree of detail loss compared to the ground truth. For the "LivingRoom" scene, our result's specular reflection is not as apparent as that of KPCN and ACFM. For the last scene, KPCN and ACFM oversmooth the shadow boundary in the middle of the zoomed-in patch. Although our method recovers the shadow boundary, it introduces wrinkle-like artifact.

Moreover, since Tungsten dataset does not provide scenes of challenging effects like motion blur, depth of field, and volumetric media, we have not evaluated our method on them as what has been done in KPCN [Bako et al. 2017].

8 CONCLUSION AND FUTURE WORK

We propose the first self-attention based model for Monte-Carlo denoising problem and experiments show that our method is especially good at recovering structures and textures of images. This is because self-attention is essentially a non-local means filter in the high-dimensional space, sharing edge-preserving properties of non-local denoising methods. Specifically for our self-attention based MC denoising model, we design multi-scale feature extractors and auxiliary feature guided self-attention mechanism to enhance its denoising performance. Our model reaches a new state-of-the-art with improvements in terms of both quantitative metrics and qualitative effects. For future improvements, we can further optimize the computational complexity and memory usage of self-attention, borrowing ideas such as kernel attention [Katharopoulos et al. 2020] or investigating feature map downsampling approaches while retaining spatial information. Methods for better utilizing auxiliary features are also worth exploring, using more advanced attention mechanisms such as memory augmented self-attention [Cornia et al. 2020] or cross-attention [Hou et al. 2019].

ACKNOWLEDGMENTS

This research is supported in part by the National Natural Science Foundation of China (62072191), in part by the Fundamental Research Funds for the Central Universities (D2190670), in part by the National Natural Science Foundation of China (61802453, 61972160), and in part by the Natural Science Foundation of Guangdong Province (2019A1515010860, 2021A1515012301).

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. 2020. Deep combiner for independent and correlated pixel estimates. *ACM Trans. Graph.* 39, 6 (2020), 1–12.

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 (2017), 97–1.

Pablo Bauszat, Martin Eisemann, and Marcus Magnor. 2011. Guided image filtering for interactive high-quality global illumination. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 1361–1368.

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

Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, José A Iglesias-Guitián, David Adler, Kenny Mitchell, Wojciech Jarosz, and Jan Novák. 2016. Nonlinearly weighted first-order regression for denoising Monte Carlo renderings. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 107–117.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

Antoni Buades, Bartomeu Coll, and J-M Morel. 2005. A non-local algorithm for image denoising. In *CVPR*, Vol. 2. IEEE, 60–65.

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 (2017), 1–12.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. 2020. Generative pretraining from pixels. In *International Conference on Machine Learning*. PMLR, 1691–1703.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating Long Sequences with Sparse Transformers. arXiv:1904.10509 [cs.LG]

Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. 2020. Meshed-memory transformer for image captioning. In *CVPR*. 10578–10587.

Holger Dammertz, Daniel Sewtz, Johannes Hanika, and Hendrik PA Lensch. 2010. Edge-avoiding a-trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics*. Citeseer, 67–75.

Axel Davy, Thibaud Ehret, Jean-Michel Morel, Pablo Arias, and Gabriele Facciolo. 2019. A non-local CNN for video denoising. In *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2409–2413.

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 (2014), 1–15.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

Elmar Eisemann and Frédo Durand. 2004. Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph.* 23, 3 (2004), 673–678.

Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. 2019. Sample-based Monte Carlo denoising using a kernel-splatting network. *ACM Trans. Graph.* 38, 4 (2019), 1–12.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661* (2014).

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028* (2017).

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. 2019. Axial Attention in Multidimensional Transformers. arXiv:1912.12180 [cs.CV]

Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. 2019. Cross Attention Network for Few-shot Classification. arXiv:1910.07677 [cs.CV]

Shi-Min Hu, Dun Liang, Guo-Ye Yang, Guo-Wei Yang, and Wen-Yang Zhou. 2020. Jittor: a novel deep learning framework with meta-operators and unified graph execution. *Information Sciences* 63, 222103 (2020), 1–21.

Tao Hu, Chengjiang Long, and Chunxia Xiao. 2021. A Novel Visual Representation on Text Using Diverse Conditional GAN for Visual Recognition. *IEEE Transactions on Image Processing* 30 (2021), 3499–3512.

Yuchi Huo and Sung-eui Yoon. 2021. A survey on deep learning-based Monte Carlo denoising. *Computational Visual Media* (2021), 1–17.

Mustafa Işik, Krishna Mullia, Matthew Fisher, Jonathan Eisenmann, and Michaël Gharbi. 2021. Interactive Monte Carlo denoising using affinity of neural features. *ACM Trans. Graph.* 40, 4 (2021), 1–13.

Ashraful Islam, Chengjiang Long, Arslan Basharat, and Anthony Hoogs. 2020. DOA-GAN: Dual-order attentive generative adversarial network for image copy-move forgery detection and localization. In *CVPR*. 4676–4685.

Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. 2015. A machine learning approach for filtering Monte Carlo noise. *ACM Trans. Graph.* 34, 4 (2015), 122–1.

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. 2020. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Stamatios Lefkimmiatis. 2017. Non-local color image denoising with convolutional neural networks. In *CVPR*. 3587–3596.

Xian-Ying Li, Yan Gu, Shi-Min Hu, and Ralph R Martin. 2013. Mixed-domain edge-aware image manipulation. *IEEE Transactions on Image Processing* 22, 5 (2013), 1915–1925.

Weiheng Lin, Beibei Wang, Lu Wang, and Nicolas Holzschuch. 2020. A detail preserving neural network model for Monte Carlo denoising. *Computational Visual Media* 6, 2 (2020), 157–168.

Weiheng Lin, Beibei Wang, Jian Yang, Lu Wang, and Ling-Qi Yan. 2021. Path-based Monte Carlo Denoising Using a Three-Scale Neural Network. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 369–381.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030* (2021).

YiFan Lu, Ning Xie, and Heng Tao Shen. 2020. DMCR-GAN: Adversarial Denoising for Monte Carlo Renderings with Residual Attention Networks and Hierarchical Features Modulation of Auxiliary Buffers. In *SIGGRAPH Asia 2020 Technical Communications*. 1–4.

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

Xiaoxu Meng, Quan Zheng, Amitabh Varshney, Gurprit Singh, and Matthias Zwicker. 2020. Real-time Monte Carlo Denoising with the Neural Bilateral Grid. (2020).

Bochang Moon, Nathan Carr, and Sung-Eui Yoon. 2014. Adaptive rendering based on weighted local regression. *ACM Trans. Graph.* 33, 5 (2014), 1–14.

Bochang Moon, Jong Yun Jun, JongHyeob Lee, Kunho Kim, Toshiya Hachisuka, and Sung-Eui Yoon. 2013. Robust image denoising using a virtual flash image for Monte Carlo ray tracing. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 139–151.

Bochang Moon, Steven McDonagh, Kenny Mitchell, and Markus Gross. 2016. Adaptive polynomial rendering. *ACM Trans. Graph.* 35, 4 (2016), 1–10.

Jacob Munkberg and Jon Hasselgren. 2020. Neural denoising with layer embeddings. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 1–12.

Ryan S Overbeck, Craig Donner, and Ravi Ramamoorthi. 2009. Adaptive wavelet rendering. *ACM Trans. Graph.* 28, 5 (2009), 140.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. 2004. Digital photography with flash and no-flash image pairs. *ACM Trans. Graph.* 23, 3 (2004), 664–672.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683* (2019).

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. 2019. Stand-Alone Self-Attention in Vision Models. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/3416a75f4cea9109507cacd8e2f2aefc-Paper.pdf

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2012. Adaptive rendering with non-local means filtering. *ACM Trans. Graph.* 31, 6 (2012), 1–11.

Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. 2013. Robust denoising using feature and color information. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 121–130.

Holly E. Rushmeier and Gregory J. Ward. 1994. Energy Preserving Non-Linear Filters. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94)*. Association for Computing Machinery, New York, NY, USA, 131–138. https://doi.org/10.1145/192161.192189

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR*. 1–9.

Carlo Tomasi and Roberto Manduchi. 1998. Bilateral filtering for gray and color images. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*. IEEE, 839–846.

Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. 2021. Scaling Local Self-Attention For Parameter Efficient Visual Backbones. arXiv:2103.12731 [cs.CV]

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

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 (2018), 1–15.

Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. 2020. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *ECCV*. Springer, 108–126.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2018. Non-local neural networks. In *CVPR*. 7794–7803.

Kin-Ming Wong and Tien-Tsin Wong. 2019. Deep residual learning for denoising Monte Carlo renderings. *Computational Visual Media* 5, 3 (2019), 239–255.

Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. 2019. Feature denoising for improving adversarial robustness. In *CVPR*. 501–509.

Bing Xu, Junfei Zhang, Rui Wang, Kun Xu, Yong-Liang Yang, Chuan Li, and Rui Tang. 2019. Adversarial Monte Carlo denoising with conditioned auxiliary feature modulation. *ACM Trans. Graph.* 38, 6 (2019), 224–1.

Ruifeng Xu and Sumanta N Pattanaik. 2005. A novel Monte Carlo noise reduction operator. *IEEE Computer Graphics and Applications* 25, 2 (2005), 31–35.

Wenju Xu, Chengjiang Long, Ruisheng Wang, and Guanghui Wang. 2021. DRB-GAN: A Dynamic ResBlock Generative Adversarial Network for Artistic Style Transfer. *arXiv preprint arXiv:2108.07379* (2021).

Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. 2020. Learning texture transformer network for image super-resolution. In *CVPR*. 5791–5800.

Xin Yang, Dawei Wang, Wenbo Hu, Lijing Zhao, Xinglin Piao, Dongsheng Zhou, Qiang Zhang, Baocai Yin, Qiang Cai, and Xiaopeng Wei. 2018. Fast reconstruction for Monte Carlo rendering using deep convolutional networks. *IEEE Access* 7 (2018), 21177–21187.

Xin Yang, Dawei Wang, Wenbo Hu, Li-Jing Zhao, Bao-Cai Yin, Qiang Zhang, Xiao-Peng Wei, and Hongbo Fu. 2019. DEMC: A deep dual-encoder network for denoising Monte Carlo rendering. *Journal of Computer Science and Technology* 34, 5 (2019), 1123–1135.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2019. Self-attention generative adversarial networks. In *International conference on machine learning*. PMLR, 7354–7363.

Henning Zimmer, Fabrice Rousselle, Wenzel Jakob, Oliver Wang, David Adler, Wojciech Jarosz, Olga Sorkine-Hornung, and Alexander Sorkine-Hornung. 2015. Path-space motion estimation and decomposition for robust animation filtering. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 131–142.

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. https://doi.org/10/f7k6kj