Neural Temporal Denoising for Indirect Illumination

Yan Zeng
Shandong University
Jinan, China
yanzeng@mail.sdu.edu.cn

Lu Wang
Shandong University
Jinan, China
luwang_hcivr@sdu.edu.cn

Yanning Xu
Shandong University
Jinan, China
xyn@sdu.edu.cn

Xiangxu Meng
Shandong University
Jinan, China
mxx@sdu.edu.cn

Figure 1. We present a novel learning-based temporal denoiser for indirect illumination of sparse Monte Carlo path traced sequences at real-time rates. Our method significantly alleviate the overblur and ghosting artifacts in dynamic occluded regions. We compare with the state of the art methods: SVGF [24], TRMV [29] and NTASD [9].

Abstract

The state of the art temporal reuse methods with traditional motion vectors cause artifacts in motion occlusions. We propose a novel neural temporal denoising method for indirect illumination of Monte Carlo (MC) ray tracing animations at 1 sample per pixel. Based on the end-to-end multi-scale kernel-based reconstruction, we apply temporally reliable dual motion vectors to better reconstruct the occlusions, and introduce additional motion occlusion loss to reduce ghosting artifacts. Experiments show that our method significantly reduces the overblur and ghosting artifacts while generating high quality images at real-time rates.

1. Introduction

MC integration [23] is pervasively applied in rendering physics-based realistic images from virtual modeling scenes in recent years [19]. However, it requires heavy computer calculations and enormous samples per pixel, usually taking hours to generate a plausible image. Due to the requirements of real-time (at 30 FPS) and the limitations of hardware, MC ray tracing can only use 1 path per pixel, which brings inevitable and unbearable noise to rendered images. Since the improvement of NVIDIA RTX hardware could hardly achieve the requirements of real-time ray tracing for the foreseeable future, researchers focus on reconstruction algorithm with low sample counts to obtain noise-free images at interactive or even real-time rates.

Recent learning-based offline denoising methods bring the possibility of sparse MC reconstruction. Bako *et al*. [2] propose Kernel Predict Convolutional Networks (KPCN) to predict the locally optimal neighborhood weights per pixel, Vogels *et al*. [27] introduce hierarchy multi-scale module to KPCN, and extend the network to temporal domain. These methods generate high-quality results, however, they need 16-32 spp, which are far more than the sample counts tolerable at real-time rates.

The Recurrent Auto-Encoder (RAE) network [6] denoises the MC renderings at interactive rates by adding a recurrent convolutional block to each encoder layer of a U-Net [21] architecture. The Spatiotemporal Variance-Guided Filtering (SVGF) method [24] reconstructs direct and indirect illumination separately, using temporal accumulation and Temporal Anti-Alising (TAA) [25] to generate temporally stable sequences. These methods denoise the sparse rendered images at interactive or real-time rates and reduce flickering between successive frames, however, still remain significant overblur and ghosting artifacts, especially when occlusion happens.

Inspired by hierarchy KPCN [27], The Neural Temporal Adaptive Sampling and Denoising (NTASD) method [9] applies multi-scale kernel predict module to the Deep Adaptive Sampling and Reconstruction (DASR) network [1], adds temporal optimization to sample predictor and denoiser, thus produces better images with high-frequency details and increases temporal stability. The quality of all those TAA based methods is related to the correctness of motion vectors. Zeng *et al*. [29] propose reliable motion vectors for shadows, glossy reflections and occlusions separately, but the quality of denoising results is unstable caused by empirically based temporal filtering. Since indirect illumination is very important for complex scenes with a large number of occlusions, we pursue stable diffuse indirect lighting effects in the occlusion case. Note that the direct illumination can be reasonably denoised using the Linear Transformed Cosines (LTC)[10], however, the indirect illumination is difficult to be solved by LTC, and there are serious ghosting artifacts when moving objects or viewpoints changing rapidly in the scene, so we focus on indirect illumination denoising.

Based on the idea of NTASD [9] and hierarchy KPCN [27], we propose a learning-based end-to-end denoising network. In addition to the generation of high-quality noise-free images and the preservation of temporal stability, our method aims at generating reasonable warped previous images with temporally reliable dual motion vectors (TRMV) [29], and using motion occlusion loss function to reduce the ghosting artifacts. The main contributions are as follows:

- propose a learning-based temporal denoiser network for indirect illumination of MC path tracing at sparse sample counts (1 ray per pixel) at real-time rates,
- generate a more plausible warped previous denoised image aligned with current frame in temporal reproject process by using dual motion vectors, and
- introduce a motion occlusion loss function to substantially reduce the ghosting artifacts in dynamic occlusions.

2. Related work

2.1. Traditional adaptive sampling and reconstruction methods

To reduce the visually annoying noise caused by relative low sample counts in classical MC rendering estimations [23], traditional algorithms [30] control the sampling density and aggregate samples. Some of traditional methods [26, 15, 11] obtain sampling rates and filters by analyzing the light equations [12], other image-space methods [20, 22, 4, 5] based on linear regression usually exploit auxiliary features (such as per-pixel albedo, depth, normal or diffuse reflectance), estimate the errors of the filtered output and compute the kernel weights. However, these traditional methods are still far from real-time and rarely used in animation sequences.

2.2. Learning-based post-processing methods

Thanks to recent advanced deep learning frameworks [14], MC denoising methods have made great progress in post-processing reconstruction [28]. Among which the pixel denoising methods stand out, this kind of methods predict the per-pixel results, become the most general and popular solutions. The pixel denoising researches can be divided into categories by the prediction targets (parameter, kernel and radiance) [28], and our research mainly focuses on the kernel prediction methods.

Bako *et al*. [2] propose a supervised KPCN, instead of their previous direct prediction framework (DPCN) that directly predicts the final pixel value by training a deep convolutional network (CNN) [14], this network manages to predict the local reconstruction kernel weights of each pixel, perform a filter operator by applying the kernel to each pixel, and then obtain the final pixel color indirectly.

To deal with the temporal stability, hierarchy KPCN [27] extend the network to temporal domain. They also introduce a hierarchy multi-scale module to increase robustness and an asymmetric loss function to preserve details. Sample-based Kernel-Splatting network [16] reconstructs MC renderings directly from the samples by splatting them onto neighbor pixels. Xu *et al*. [3] propose an adversarial approach that uses generative adversarial networks (GAN) [8], they also adapt a feature modulation method to utilize auxiliary features.

These methods maximize the use of relative a few samples (8-32 spp) and generate fine-grained results, but the requirements of the sample counts are still too high for real-time rates.

2.3. Interactive and real-time denoising methods

In order to achieve plausible images at interactive and real-time speed, researchers pay more attention to reconstruct noise-free results with only 1 ray per pixel. RAE [6] adds a recurrent convolutional block to each encoder layer of a U-Net [21] architecture, which uses the hidden information of previous frame as part of the input of the current encoder layer.

Inspired by TAA [25], real-time SVGF method [24] uses temporal accumulation to preserve temporal stability and applies a clamping operator to reduce the serious ghosting. However, they still remain apparent overblur and ghosting artifacts in occlusions. Since it is not a learning-based kernel prediction network, the results are not sufficiently fine-grained.

NTASD method [9] denoises at near real-time rates, it applies multi-scale kernel predict module [27] to the DASR network [1], adds temporal optimization to sample predictor and denoiser, thus produces better images with high-frequency details and increases temporal stability. Similarly, our network applies a circulate feedback block to classical U-Net to reuse the temporal features, and uses a hierarchy multi-scale kernel prediction module to increase robustness.

2.4. Motion vectors

Previous methods [24, 27, 9] use the traditional screen-space motion vector, a two-dimensional vector points from each pixel coordinate of the current frame to the corresponding pixel coordinate of the previous frame, to obtain the reprojected warped previous denoised image.

The utilization of motion vectors makes better use of historical information to smooth temporal flickering. But in some cases, the motion vectors are incorrect or even no-exist [29], for example, an occluded region in the background is blocked by an object in previous frame, and then appears in current frame as the moving object leaves (a discussion of another two failure cases: shadows and glossy reflections, is available in [29]).

In this paper, we generate reasonable warped previous images with temporally reliable dual motion vectors [29], and exploit them (both traditional and dual motion vectors) to provide the novel motion occlusion loss function. In order to avoid being affected by another two failure cases (ignore the shadows and glossy reflections), we apply our network to indirect illumination (focus on occlusions).

3. Background and motivation

3.1. Temporal accumulation and learnable per-pixel blending weight

Many methods [24, 27, 9] use traditional screen space motion vectors to generate warped previous images or take them as part of network input. In fact, traditional screen space motion vectors do introduce historical reuse accumulation information and remove temporal flickering artifacts of animation sequences. However, they fail in the occluded regions.

With the help of traditional motion vectors, pixels in the occluded regions will always point to the previous occluders, which often have completely different colors with background. The final pixel value in temporal accumulation method is obtained with follow blending equation:

$$\bar{c}_{current}(x,y) = \alpha \cdot \widetilde{c}_{current}(x,y) + (1-\alpha) \cdot \bar{c}_{warped}(x,y), \quad (1)$$

where α is a fixed scalar weight (usually set to 0.1 or 0.2 in practice) for blending the color of current filtered image $\widetilde{c}_{current}$ and warped previous blended one $\bar{c}_{warped}$, that means their heavy temporal dependences on previous frames causing significant ghosting artifacts.

To reduce the main ghosting problems caused by incorrect historical information, SVGF [24] applies a clamping operator [25], that clamps the previous pixel color to the current value interval by using a local Gaussian distribution. Despite the presence of clamping operator, fixed scalar weight is still not flexible for different parts of image. SVGF forcibly reuses much historical information which are still incorrect after clamping, causing noticeable bright color blocks at the tails of moving objects.

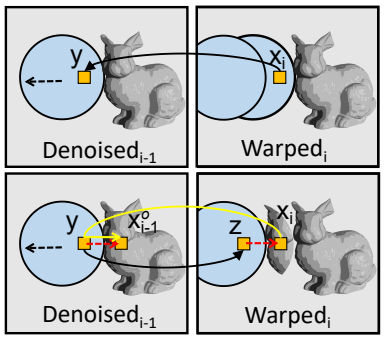

Figure 2. Illustration of the computation of traditional (first line) and dual motion vectors (second line) for occlusions. For a pixel x_i that is visible now but was occluded in the previous frame at y, using the traditional motion vector will directly assign the color of y to x_i, while the dual motion vector finds where the occluder y is in the current frame at z, then assigns the color of x^O_{i-1} to x_i (yellow arrow).

Figure 3. Visualizations of traditional motion vectors (Trad mvec) and dual motion vectors (Dual mvec) as well as the warped reprojections they produced. We set the opacity of motion vectors to 0.8 for intuition.

Different from non-neural temporal reuse methods [24, 29], we use a net similar to other learning-based networks [27, 9], to implicitly train the spatial filter and the per-pixel scalar weight α (temporal accumulate) to get more stable denoising results. We use a trainable weight tensor of the same size as the images, that is, each pixel has a trainable weight instead of a fixed 0.1 or 0.2 value.

Although clamping operator and learnable blending weight solve the most obvious ghosting problems, there are still overblur and minor ghosting artifacts in occlusions. All these are caused by the large differences between warped history and current in the occluded areas.

3.2. Dual motion vectors and warped previous reprojection

In our method, we use the temporally reliable dual motion vector [29], a motion vector for the just-appeared region occluded previously, to find a similar correspondence and then close the gap between $\widetilde{c}_{current}$ and $\bar{c}_{warped}$.

We reproduced the explanation of dual motion vectors [29] here for completeness: See Fig. 2, the traditional motion vector gives the $x_i \rightarrow y$ correspondence but unfortunately cannot be easily used. The dual motion vector tracks the movement of $y \rightarrow z$ from the previous frame to the current, using the motion of the occluder. Then, based on the relative positions of x_i and z (red dotted arrow), it finds the location x_{i-1}^{O} in the previous frame. This process can be simply represented as:

$$x_{i-1}^{O} = y + (x_i - z)\,, \tag{2}$$

where $y = P_{i-1}T^{-1}(x_i)\,P_i^{-1}x_i$ and $z = P_iT(y)P_{i-1}^{-1}y$ (P: the viewport*modelview projection transformation per frame, T: the geometry transformation between frames).

As can be seen from Fig.3, The box and the ball move from right to left, the traditional motion vector tightly fits the edges of the moving object, whereas the dual motion vector appears to have a tail in the opposite direction of the object's movement, as a result, in the occlusion area, the former (Trad mvec) creates the repetition of moving objects whose colors are often different from the background, while the latter (Dual mvec) reuses background information to solve the problem.

4. Method

4.1. Network

Inspired by NTASD [9], we propose a learning-based end-to-end denoising network, an overview is shown in Fig.4. The main differences between our network and NTASD are:

- the motion vectors used in reproject process (NTASD uses traditional motion vectors, while we use dual motion vectors),
- the warped previous denoised images (part of the input), and
- our novel motion occlusion loss function.

4.2. Temporal reprojection process

Based on a classical U-net [21], our denoiser has a circulate feedback block to reuse the temporal features, this module aligns the previous denoised result $\bar{c}_{history}$ with the current noisy frame $c_{current}$ by using dual motion vectors [29] $m_{x,y}$, obtains the warped previous denoised image $\bar{c}_{warped}$, and sends both $c_{current}$ and $\bar{c}_{warped}$ into the network along with auxiliary feature buffers (albedo, depth, normal) to generate the current filtered result $\widetilde{c}_{current}$. The temporal reprojection process is as follows:

$$\bar{c}_{warped}(x, y) = \bar{c}_{history}\,(x + m_x, y + m_y)\,. \tag{3}$$

In this paper, we use temporally reliable dual motion vectors [29] to better generate the warped previous in occlusions. We apply PyTorch's $grid\_sample$ function with bilinear interpolation mode to implement this reproject step. The difference of traditional and temporally reliable dual motion vectors as well as the reprojected warped results are demonstrated in Fig. 5.

Our warped results are superior to the others that use traditional motion vectors, since the repeated background area often has a closer color to the occluded part than the foreground moving object. In other words, our approach provides more efficient and valid temporal information at the reprojection and recurrent feedback process.

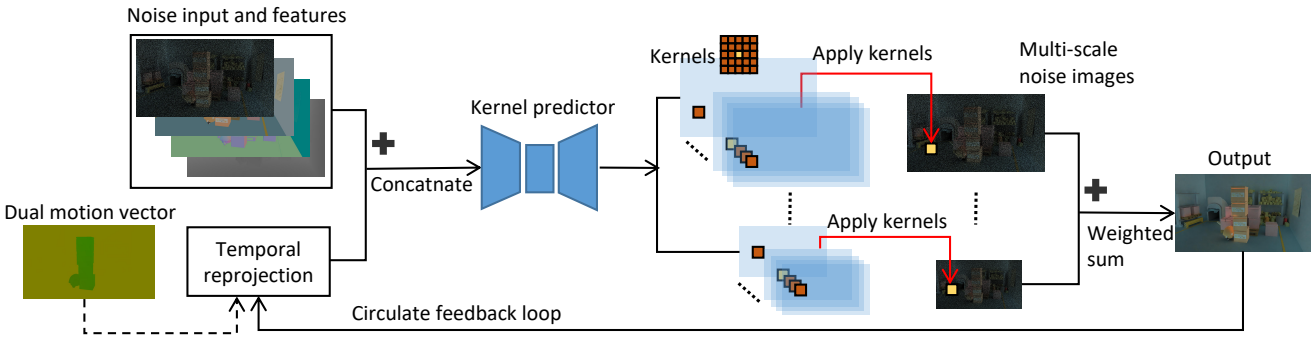

Figure 4. An overview of our network. It takes the concatenation of noise, auxiliary features (albedo, normal, depth) and temporal reprojection (the warped denoised image of previous frame) as input, predicts per-pixel kernels in order to filter the multi-scale noisy images, then combines them using learnable per-pixel weights. Particularly, we apply the dual motion vector [29] instead of the traditional motion vector to temporal reproject process, and introduce novel motion occlusion loss function.

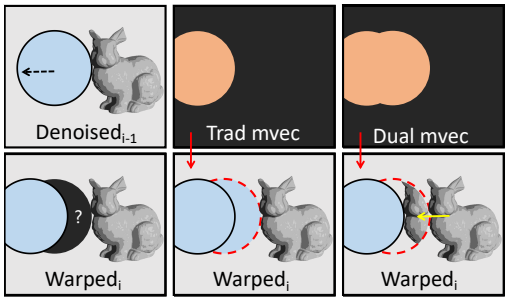

Figure 5. The blue circle moves from right to left, leaving an occluded area marked by black shadow (left column). The current traditional motion vector (Trad mvec) and dual motion vector (Dual mvec) reproject their warped previous images differently (middle and right column).

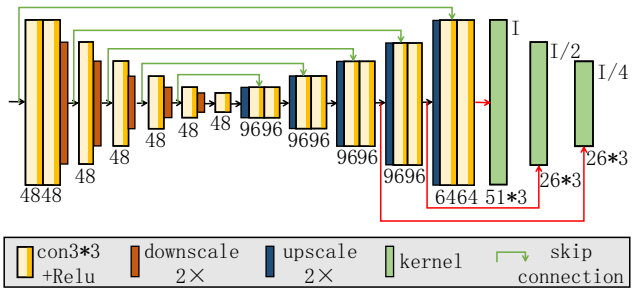

Figure 6. Structure of kernel predictor. In last three layers of U-net, we predict per-pixel kernels and scalar layer blending weights ($26 = 5{\times}5 + \alpha$). At the finest level, we additionally predict a 5×5 temporal kernel which is applied to the warped previous denoised output ($51 =$ two $5{\times}5 + \alpha^t$).

4.3. Multi-scale kernel predict

We apply multi-scale pixel-wise kernel predict module [27] to our framework, as can be seen from Fig.6, our network predicts three kernels at last three layers of U-net, each with 5×5 per-pixel filter weights, to filter multi-scale current noisy images (1, 1/2 and 1/4 of full resolution), and then sums the filtered results of these three scales noisy images and warped previous denoised one using learned blending weights. The coarse and fine images are combined using:

$$\mathbf{o}_p = \mathbf{i}_p^f - \mathbf{U}\left[\alpha^c\left[\mathbf{D}\mathbf{i}^f\right]\right]_p + \mathbf{U}\left[\alpha^c\left[i^c\right]\right]_p, \qquad (4)$$

where i^f/i^c represents the relative fine-scale/coarse-scale filtered output. D and U are 2×2 maximum pooling downsampling and bi-linear up-sampling operators respectively. We obtain the final output color of pixel p by combining i^f and i^c using the trainable weight α^c.

In addition to three *spatial* scales, our network predicts a *temporal* 5×5 kernel at the last layer. As shown in Fig.6, at the finest scale, 51 means a 5×5 temporal kernel + a 5×5 spatial kernel + α^t, at the coarser scales, 26 means a 5×5 spatial kernel + α, while 3 means three RGB channels.

The fine-scale images maintain details and avoid producing overblur, while the coarse-scale images prevent main high-frequency and minor low-frequency noise caused by sparse sample counts. As stated in [27], the multi-scale kernel prediction structure generates better results than direct prediction and single-scale prediction.

4.4. Motion occlusion loss

Even though we solve the problem of the huge gap between $\widetilde{c}_{current}$ and $\bar{c}_{warped}$ in section 4.2, the occluded areas of final denoised output are still unsatisfactory visually. Since the occlusions are too small and narrow relative to the whole input tensors, and the network could not spontaneously pay additional attention to these regions, as a result, the convergence is not sufficient enough in occlusions. In order to further decrease the minor artifacts in these areas, we introduce a new motion occlusion loss function to increase the network's attention on them.

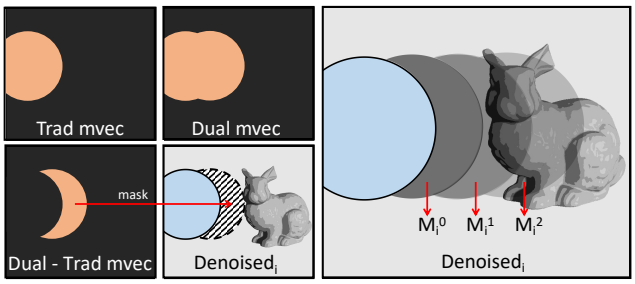

Figure 7. We obtain a mask from dual motion vector minus traditional motion vector (left). For frame i, we use three masks M_i^0, M_i^1 and M_i^2 to delimit the calculation range of our motion occlusion loss (right).

We combine traditional motion vectors and dual motion vectors to calculate the occlusion mask quickly. As shown in Fig.7, in order to calculate progressive loss temporally, for each current frame i, we have three masks: M_i^0, M_i^1 and M_i^2, they are computed by $MV_{dual}^i - MV_{tra}^i$ (dual motion vector minus traditional motion vector, where MV is the abbreviation of motion vector), $MV_{dual}^{i-1} - MV_{tra}^{i-1}$ (frame $i-1$) and $MV_{dual}^{i-2} - MV_{tra}^{i-2}$ (frame $i-2$) respectively. We use the following equation:

$$\begin{aligned} L_{occ} &= L_1\left(\mathring{I}_i, \mathring{R}_i\right) \\ \mathring{I}_i &= I_i \odot \textstyle\sum_{k=0}^{2} w_i^k \cdot M_i^k \\ \mathring{R}_i &= R_i \odot \textstyle\sum_{k=0}^{2} w_i^k \cdot M_i^k \end{aligned} \tag{5}$$

where I_i and R_i are denoised image and reference of frame i, $\odot$ is a Hadamard Product operator that multiplies the corresponding location values of two matrices, which is used for achieving the masked denoised image $\mathring{I}_i$ and reference $\mathring{R}_i$. w_i^0, w_i^1 and w_i^2 are the weights of these three masks M_i^0, M_i^1 and M_i^2, we set them to 1, 0.8 and 0.6 respectively in our practice. For some special cases like the first frame or second frame whose $i-1$ and $i-2$ frame are not exist, we simply initialize their M_i^1 and M_i^2 to M_i^0.

5. Implementation

Our network is conducted in PyTorch [18], using Xavier initialization [7] and Adam optimizer [13] to initialize the parameters and control the learning rate (the initial learning rate is 0.0001). All experiments are trained for 200 epochs.

We obtain the 1 spp noisy images, 2048 spp ground-truth images and auxiliary feature buffers (albedo, depth, normal, motion vector) from NVIDIA OptiX [17]. We take 281 animated sequence clips (142 for training and 139 for testing) from different perspectives of 4 scenes (APPLE, ROBOT, PINKROOM and PICA) as datasets, for a total of 2810 frames. Each clip of training data has 10 successive frames, where the first 9 frames are trained for backward propagation, and the last 1 frame is used for verification. For the first frame of each clip, we initialize the warped previous image and previous denoised one to the noise.

The total loss function is as follows:

$$L = L_1\left(I_i, R_i\right) + L_1\left(\Delta I_i, \Delta R_i\right) + L_{occ}, \tag{6}$$

it consists of three parts: a spatial loss $L_{spat} = L_1\left(I_i, R_i\right)$, a temporal loss $L_{temp} = L_1\left(\Delta I_i, \Delta R_i\right)$ and our novel motion occlusion loss L_{occ}, they are all $L1$ loss and have equal weight. I_i and R_i are filtered result and ground truth image of frame i. The temporal gradients are $\Delta I_i = I_i - I_{i-1}$ and $\Delta R_i = R_i - R_{i-1}$. For the first 100 epochs, we apply spatial and temporal loss, and for the last 100 epochs, we add motion ooclusion loss function. More details about motion occlusion loss can be found in section 4.4.

In order to prevent the wrong information from being accumulated incorrectly at an early stage, we calculate the total loss function on the first 9 frames of each clip. Note that all loss functions are applied to the final filtered results and references, not the intermediate warped previous images.

6. Results

6.1. Runtime cost

All images are rendered with 1280×720 resolution and trained on NVIDIA GeForce RTX 2080 Ti with 12GB video memory. The average denoising time of our network is 7.9ms per frame, which can be divided into four parts: extracting features and predicting per-pixel kernels (3.1ms), applying the kernels to multi-scale images (3.4ms), blending the spatial and temporal filtered results (0.8ms), and calculating temporal reprojection(0.6ms). These numbers are averages over 1390 testing frames.

6.2. Compare with state of the arts

We compare with several real-time temporal reuse and learning-based post-processing denoising methods: SVGF [24], TRMV [29] and NTASD [9]. To compare with learning-based NTASD method in a similar setting, we alter NTASD to use the same network structure, initialization

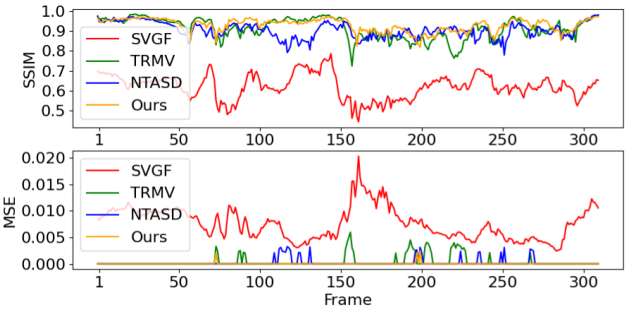

Figure 8. Results of SSIM and MSE errors of 310 consecutive frames on the APPLE scene.

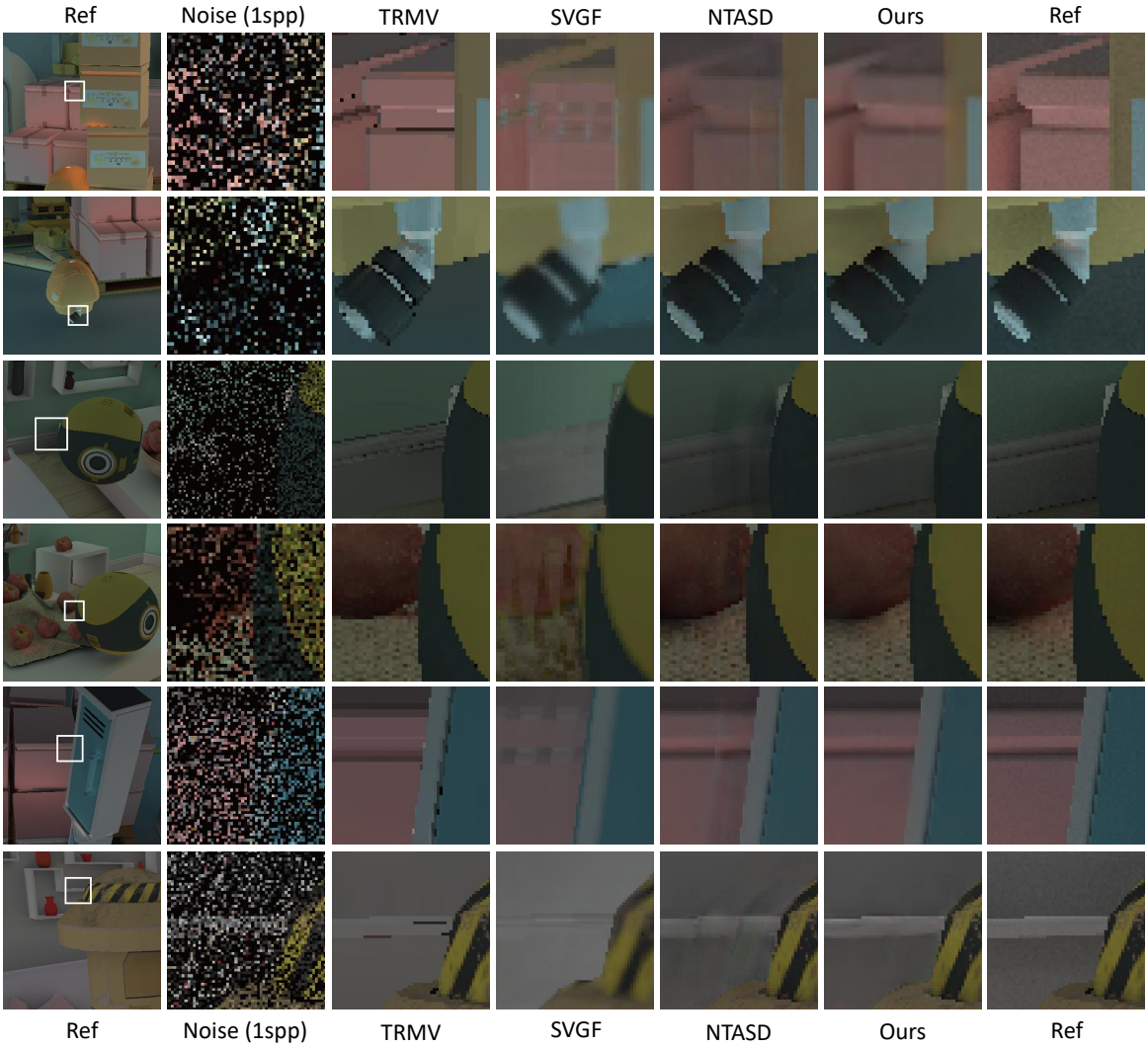

Figure 9. We compare with the state of the arts: SVGF [24], TRMV [29] and NTASD [9]. All objects move from left to right except for the second row.

Name	Motion vectors	Loss function
A	Traditional	spatial+temporal
B	Dual	spatial+temporal
C(Ours)	Dual	spatial+temporal+mask

Table 1. Setup of ablation study.

Name	PSNR↑	SSIM↑	L1↓	MSE↓
SVGF	69.7086	0.6517	0.0652	0.0087
TRMV	78.4319	0.8960	0.0215	0.0007
NTASD	77.7679	0.8983	0.0283	0.0009
Ours	**80.1891**	**0.9115**	**0.0206**	**0.0002**

Table 2. Errors of four metrics. For PSNR and SSIM, higher is better, for L1 and MSE, lower is better. The scores are averaged over 1390 test frames.

and learning rate as our method, and remove the adaptive sampling part of NTASD.

Fig.9 illustrates the results of comparison. Even though with clamping operator, SVGF [24] still exploits the temporal information improperly, results in significant bright color blocks, overblur and ghosting artifacts.

By using dual motion vectors and incident radiance, TRMV [29] is able to achieve much cleaner results that completely eliminate the overblur and ghosting artifacts. But it introduces obvious shading aliasing, thus gives us the inspiration of using dual motion vectors in neural-based approach.

For learning-based methods, NTASD [9] performs well enough except for occlusions, because it only uses traditional motion vectors, results in unnatural boundaries in the occluded area.

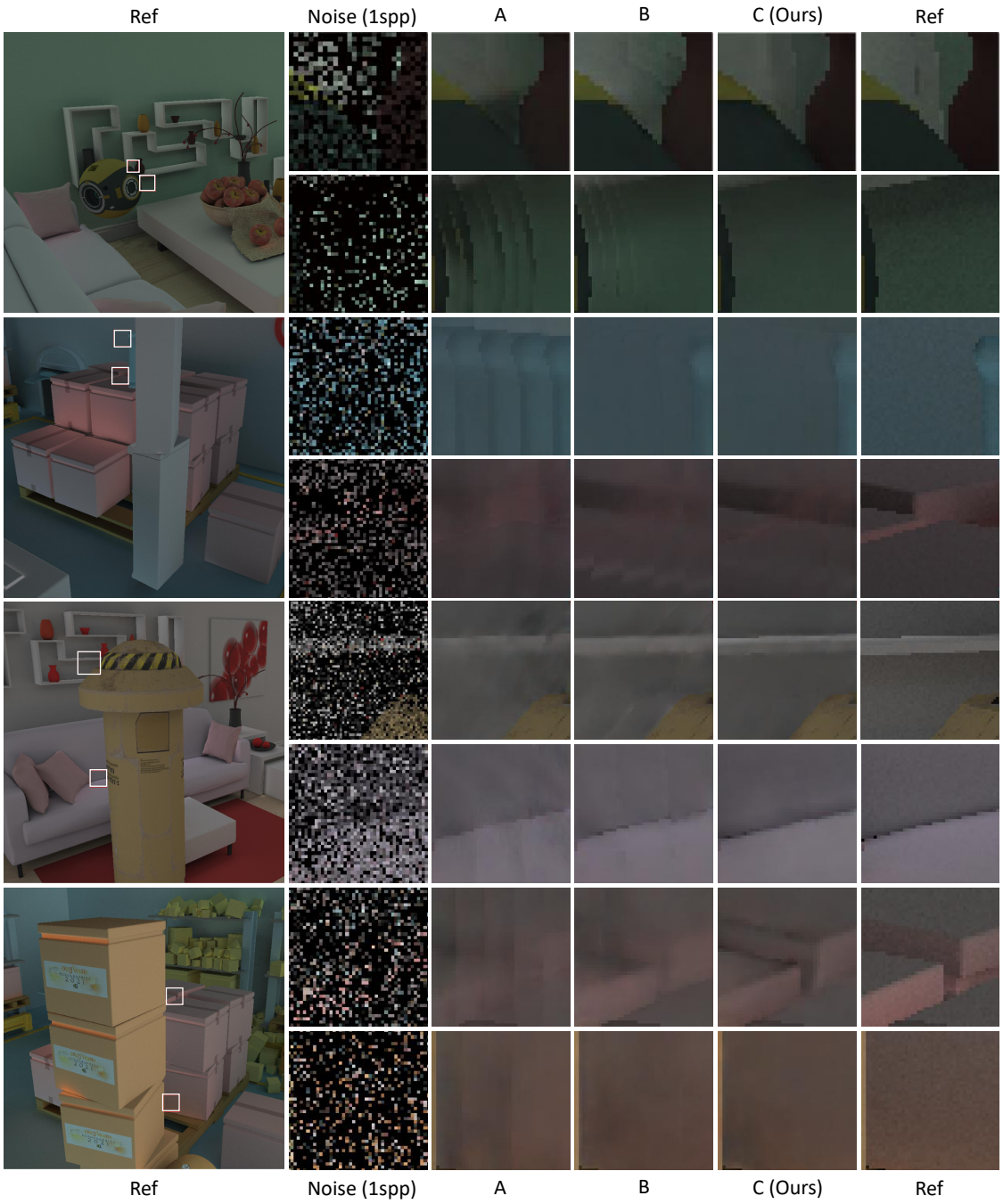

Figure 10. Results of ablation study. Experiment A (baseline) is multi-scale kernel prediction network, experiment B replaces traditional motion vectors with dual motion vectors, experiment C (our overall network) adds motion occlusion loss on the basis of B. The results show that our method can significantly reduce the overblur and ghosting artifacts in occlusions.

Thanks to the application of temporally reliable dual motion vectors and our novel motion occlusion loss function, we solve those artifacts while preserving details as much as possible. As can be seen from Fig.9, our results perform better than methods only using traditional motion vectors.

To evaluate denoising quality of occlusions, we apply four metrics: peak signal-to-noise (PSNR), structural similarity (SSIM), L1 loss and mean squared error (MSE), on the tonemapped results of our method and the state of the arts. Tab.2 shows that our method performs best on four metrics. Fig.8 displays the errors of SSIM and MSE of consecutive frames on the APPLE scene, and in order to show

the effect of TAA, we ignore the first frame and demonstrate errors from the second frame.

6.3. Ablation study

We conduct an ablation study to demonstrate the contribution of each part in our network. In practical, we regard the multi-layer kernel prediction (using traditional motion vectors) as the baseline, and denote it as experiment A. The following experiment B replaces the traditional motion vectors with temporally reliable dual motion vectors, and the final experiment C adds the motion occlusion loss function on the basis of B. The setup of each experiment is shown in the Tab.1.

The test results of different experiments are shown in Fig.10. In the just-appeared regions that are occluded by moving objects previously (the areas highlighted with boxes), experiment A presents serious ghosting and overblur artifacts. The results of B are much better than that of A, as can be seen, experiment B reduces the major artifacts but still remains minor ghosting problems. Experiment C (our whole network) resolves all the artifacts encountered in previous experiments. So the conclusion from the ablation study is that, all parts of our network are valid and effective, and our method can significantly reduce the overblur and ghosting artifacts.

6.4. Failure cases

As can be seen from Fig.11, there are two conditions that can cause failure of dual motion vectors: the occluded areas are near the boundary of image, or the foreground object moves too fast in the dark to light background area.

Consider that a moving object has just completely emerged from the image boundary, leaving an occluded region near the boundary, according to the dual motion vector, the pixel in occlusions will point out of the screen. Since the expected value cannot be obtain in this case, the performance of dual motion vector degenerated into the traditional: pointing to the occluder.

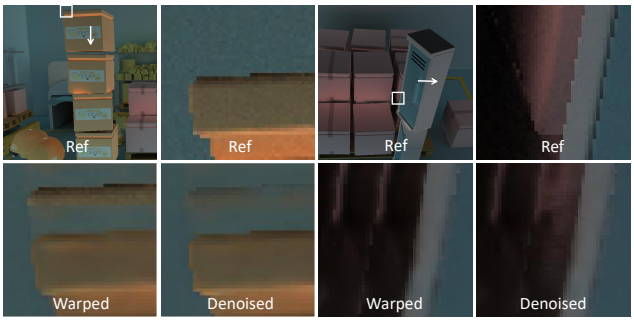

Figure 11. Two failure cases: the occlusions near the image boundary (left) or the foreground object moves too fast in the dark to light background area (right).

Another failure case usually happens when the object moves too fast, leaving significant background changes in the occlusions where it passed by, especially from a very dark block to a lighter area. In this case, the dual motion vector pastes the dark block into the occluded area whose reference is lighter, resulting in the wrong blending result.

In the future work, we may try to solve these cases by improving the dual motion vector or the network structure.

7. Conclusion

We have presented a learning-based temporal denoising method for indirect illumination of sparse MC renderings at real-time rates. We applied temporally reliable dual motion vectors to reprojection step of multi-scale kernel-predicting networks, and inspired by the visualizations of traditional and dual motion vectors, we defined novel motion occlusion loss function which is able to solve the residual minor ghosting artifacts of our occlusions' reconstruction. We also demonstrated the superiority in terms of occlusions' reconstruction over other temporal reuse and learning-based methods that only using traditional motion vectors. In the future, we would like to explore temporally stable denoising method for glossy indirect lighting effects. and further explore the integration of other types of motion vectors, for example, the shadow motion vector for dynamic shadows, and the stochastic glossy reflection motion vector for glossy reflections.

References

[1] K. Alexandr, K. Nima, and R. Ravi. Deep adaptive sampling for low sample count rendering. *Comput. Graph. Forum*, 37:35–44, 2018. 2, 3

[2] S. Bako, T. Vogels, B. McWilliams, M. Meyer, J. Novák, A. Harvill, P. Sen, T. Derose, and F. Rousselle. Kernel-predicting convolutional networks for denoising monte carlo renderings. *ACM Trans. Graph.*, 36(4):97–1, 2017. 2

[3] X. Bing, Z. Junfei, W. Rui, X. Kun, Y. Yongliang, L. Chuan, and T. Rui. Adversarial monte carlo denoising with conditioned auxiliary feature modulation. *ACM Trans. Graph.*, 38(6), 2019. 2

[4] M. Bochang, C. Nathan, and Y. Sung-eui. Adaptive rendering based on weighted local regression. *ACM Trans. Graph.*, 33(5), 2014. 2

[5] M. Bochang, M. Steven, M. Kenny, and G. Markus. Adaptive polynomial rendering. *ACM Trans. Graph.*, 35(4), 2016. 2

[6] C. R. A. Chaitanya, A. S. Kaplanyan, C. Schied, M. Salvi, A. Lefohn, D. Nowrouzezahrai, and T. Aila. Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Trans. Graph.*, 36(4):1–12, 2017. 2, 3

[7] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*,

volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. 6

[8] I. J. Goodfellow, J. P. Abadie, M. Mirza, B. Xu, D. W. Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014. 2

[9] J. Hasselgren, J. Munkberg, M. Salvi, A. Patney, and A. Lefohn. Neural temporal adaptive sampling and denoising. *Comput. Graph. Forum*, 39(2):147–155, 2020. 1, 2, 3, 4, 6, 7

[10] E. Heitz, J. Dupuy, S.Hill, and D. Neubelt. Real-time polygonal-light shading with linearly transformed cosines. *ACM Transactions on Graphics (TOG)*, 35:1–8, 2016. 2

[11] M. Jacob, H. Jon, C. Petrik, A. Magnus, and A. Tomas. Texture space caching and reconstruction for ray tracing. *ACM Trans. Graph.*, 35(6), 2016. 2

[12] J. T. Kajiya. The rendering equation. In *Computer Graphics*, pages 143–150, 1986. 2

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the Third International Conference for Learning Representations*, 2015. 6

[14] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015. 2

[15] Y. Ling-Qi, M. S. Uday, R. Ravi, and D. Fredo. Fast 4d sheared filtering for interactive rendering of distribution effects. *ACM Trans. Graph.*, 35(1), 2016. 2

[16] G. Michaël, L. Tzu-mao, A. Miika, L. Jaakko, and D. Frédo. Sample-based monte carlo denoising using a kernel-splatting network. *ACM Trans. Graph.*, 38(4), 2019. 2

[17] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich. Optix: A general purpose ray tracing engine. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, New York, NY, USA, 2010. Association for Computing Machinery. 6

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, L. Zeming, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*, 2017. 6

[19] M. Pharr, W. Jakob, and G. Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016. 1

[20] S. Pradeep and D. Soheil. On filtering the noise from the random parameters in monte carlo rendering. 31(3), 2012. 2

[21] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. pages 234–241. Springer International Publishing, 2015. 2, 3, 4

[22] F. Rousselle, M. Manzi, and M. Zwicker. Robust denoising using feature and color information. *Comput. Graph. Forum*, 32(7):121–130, 2013. 2

[23] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo method*, volume 10. John Wiley & Sons, 2016. 1, 2

[24] C. Schied, A. Kaplanyan, C. Wyman, A. Patney, C. R. A. Chaitanya, J. Burgess, L. Shiqiu, C. Dachsbacher, A. Lefohn, and M. Salvi. Spatiotemporal variance-guided filtering: Real-time reconstruction for path-traced global illumination. Association for Computing Machinery, 2017. 1, 2, 3, 4, 6, 7

[25] N. Tatarchuk, B. Karis, M. Drobot, N. Schulz, J. Charles, and T. Mader. Advances in real-time rendering in games, part i. SIGGRAPH '14. Association for Computing Machinery, 2014. 2, 3

[26] M. S. Uday, Y. JiaXian, R. Ravi, and D. Fredo. Factored axis-aligned filtering for rendering multiple distribution effects. *ACM Trans. Graph.*, 33(4), 2014. 2

[27] T. Vogels, F. Rousselle, B. McWilliams, G. Röthlin, A.Harvill, D. Adler, M. Meyer, and J. Novák. Denoising with kernel prediction and asymmetric loss functions. *ACM Trans. Graph.*, 37(4):1–15, 2018. 2, 3, 4, 5

[28] H. Yuchi and Y. Sung-eui. A survey on deep learning-based monte carlo denoising. *Computational Visual Media*, 7:169–185, 2021. 2

[29] Z. Zheng, L. Shiqiu, Y. Jinglei, W. Lu, and Y. Lingqi. Temporally reliable motion vectors for real-time ray tracing. *Comput. Graph. Forum*, 40(2):79–90, 2021. 1, 2, 3, 4, 5, 6, 7

[30] M. Zwicker, W. Jarosz, J. Lehtinen, B. Moon, R. Ramamoorthi, F. Rousselle, P. Sen, C. Soler, and S. E. Yoon. Recent advances in adaptive sampling and reconstruction for monte carlo rendering. *Comput. Graph. Forum*, 34(2):667–681, 2015. 2