DOI: 10.1111/cgf.13919
EUROGRAPHICS 2020 / U. Assarsson and D. Panozzo
(Guest Editors)

Volume 39 (2020), Number 2

Neural Temporal Adaptive Sampling and Denoising

J. Hasselgren, J. Munkberg, M. Salvi, A. Patney and A. Lefohn

NVIDIA

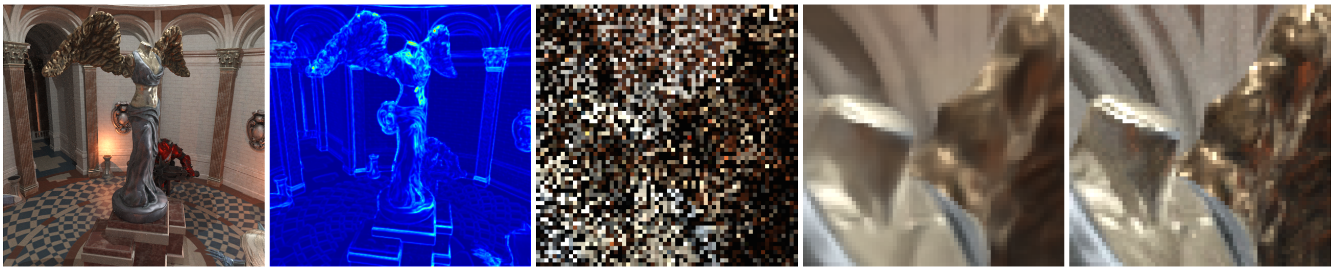


Figure 1: *We propose a novel method for temporal adaptive sampling and denoising of sparse Monte Carlo path traced animations at interactive rates.*

Abstract

Despite recent advances in Monte Carlo path tracing at interactive rates, denoised image sequences generated with few samples per-pixel often yield temporally unstable results and loss of high-frequency details. We present a novel adaptive rendering method that increases temporal stability and image fidelity of low sample count path tracing by distributing samples via spatio-temporal joint optimization of sampling and denoising. Adding temporal optimization to the sample predictor enables it to learn spatio-temporal sampling strategies such as placing more samples in disoccluded regions, tracking specular highlights, etc; adding temporal feedback to the denoiser boosts the effective input sample count and increases temporal stability. The temporal approach also allows us to remove the initial uniform sampling step typically present in adaptive sampling algorithms. The sample predictor and denoiser are deep neural networks that we co-train end-to-end over multiple consecutive frames. Our approach is scalable, allowing trade-off between quality and performance, and runs at near real-time rates while achieving significantly better image quality and temporal stability than previous methods.

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

1 Introduction

There has been substantial recent progress in denosing for offline rendering. A great overview is available in the survey from Zwicker et al. [ZJL*15]. Recent deep learning denoisers use large convolutional neural networks (CNN) to predict unique filter kernels per pixel [BVM*17]. A similar network has been applied to time sequences, by running the network in parallel over multiple frames [VRM*18]. Adaptive sampling is supported through a learned error-predicting module. Xu et al. [XZW*19] use an adversarial loss function when training a CNN denoiser. These approaches generate state-of-the-art denoising quality from 16-64 samples per pixel (spp), but come at significant cost: seconds per frame on a high end GPU.

More recently Gharbi et al. [GLA*19] apply a CNN on individual samples, typically 8-32 samples per pixel. They learn a sample embedding per-pixel, process the embeddings by U-nets [RFB15], and finally splat large (21×21 pixel) kernels per sample onto the framebuffer. They show high quality results on still images, albeit at a very high computational cost and memory requirements. The network input consists of 74 scalar features per sample in each pixel. Runtime cost is measured in seconds or minutes and scales linearly with the input sample count. In this paper, we strive for interactive denoising, with total cost of adaptive sampling and denoising measured in milliseconds.

Kuznetsov et. al [KKR18] introduce deep adaptive sampling and reconstruction (DASR) for low sample count rendering. Their

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

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

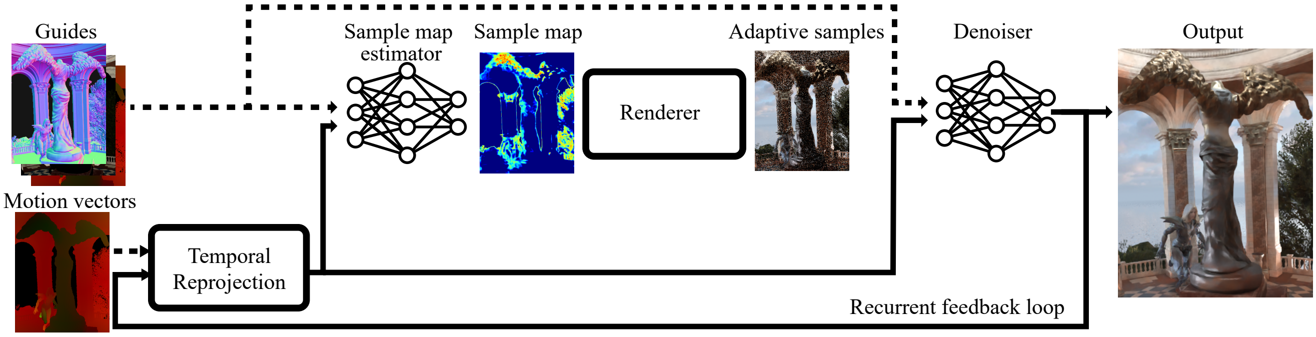


Figure 2: *An overview of our approach. The sample map estimator network outputs a sample map, which dictates how many paths the renderer should trace for each pixel. The adaptively sampled image is then denoised by a denoiser network, producing the output image. Our approach relies on a recurrent feedback loop where the previous denoised frame is reprojected and added as an input to both the sample map estimator and the denoiser.*

architecture uses two CNNs: one network learns to predict a sample density map and a second network learns to denoise the adaptively sampled rendered image. The scene is first rendered at 1 spp, which is provided as input to the sample density map estimation network. This network outputs a sample density map, that guides where to trace additional paths. These new, adaptively distributed samples, are added to the initial samples, and the final image is produced by the denoiser network. The system is trained end-to-end, optimizing both networks jointly, using just the loss of the final image. Thus, the sampler network can insert more samples where the denoiser network does a poor job, and vice versa.

To denoise low-sample count renderings at interactive rates, Chaitanya et al. [CKS*17] propose to use a U-net with recurrent convolutional blocks at each encoder level, allowing the network to exploit temporal information. At inference time, the network is applied to a single frame at a time, but keeps internal state at each encoder level, similar to an IIR filter, so it is significantly faster than the networks for offline rendering running in parallel over multiple frames. However, the hierarchy of recurrent blocks is expensive, and capturing frame-to-frame reprojection with small convolution primitives is challenging. Consequently, their algorithm is mostly suitable for small motion between frames, and they employ temporal anti-aliasing [Kar14] as a post-processing step to reduce remaining temporal artifacts. A similar network structure with hierarchical temporal recurrence is used by Kaplanyan et al. [KSL*19] but in the context of foveated rendering.

Similar recurrence loops have been applied in video-to-video synthesis [WLZ*18] and video super-resolution [SVB18]. They use a sequential generator to generate the t-th frame $\hat{\mathbf{x}}_t$ in the form of a feed-forward network, which takes the previously generated result, $\hat{\mathbf{x}}_{t-1}$, warped using optical flow, as one of the inputs.

Spatiotemporal variance-guided filtering [SKW*17, SPD18] aggressively accumulates samples over time based on heuristics. It combines selective temporal accumulation with large bilateral spatial kernels to reach impressive denoising results from low sample count input. The quality is plausible, if somewhat lower than the learned approaches, but performance is attractive for interactive applications: a 1920 × 1080 frame is denoised in a few milliseconds. A similar approach using blockwise regression was recently proposed by Koskela et al. [KIM*19].

We propose a novel approach for interactive, temporally stable, adaptive sampling and denoising by extending the end-to-end training procedure proposed by Kuznetsov et. al [KKR18] to the temporal domain. This drastically increases the effective sample count, substantially improves temporally robustness and allows us to avoid the initial sampling pass. We use a small U-net architecture with hierarchical kernel prediction (KP) [VRM*18] to create a robust and efficient denoiser. Similar to previous temporal machine learning denoisers, we use recurrence, but restrict it to the denoised output at full resolution instead of recurrence at each level, following the approach taken in recent work in video-to-video synthesis [WLZ*18] and video super-resolution [SVB18]. In contrast to their work which uses optical flow, we warp the previous output using high quality motion vectors readily available by the renderer. This is key to the effectiveness of our approach. Since warping is applied on a per-pixel level with accurate motion vectors we get high temporal reuse except in areas with disocclusion, or highly view dependent shading, where no temporal data is available. This changes the problem from learning to track the motion of noisy image features [CKS*17] to learn to detect where temporal reuse is appropriate. This is a much simpler problem, well suited to CNNs, which have been demonstrated to excel at image recognition tasks. A version of our architecture with 10× fewer weights than Chaitanya et al.'s network still generates higher quality results on animated sequences.

Our approach generalizes to new data and runs at interactive rates at 1920 × 1080 pixel resolution. Our main contributions are:

- Temporally stable adaptive sampling at low sample counts.
- Adaptive sampling driven by *warped* temporal feedback instead of an initial sampling pass.
- An interactive, temporally-stable denoiser network based on hierarchical kernel prediction and *warped* temporal feedback, which

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

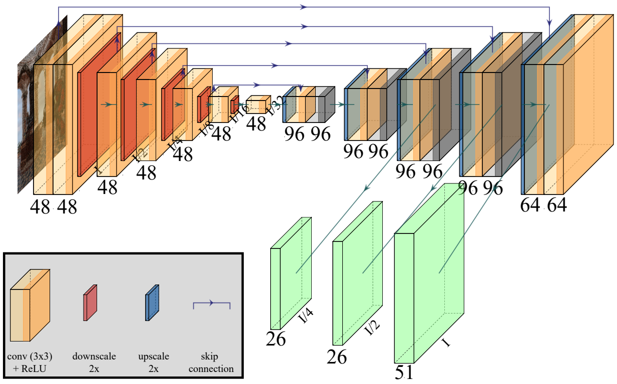


Figure 3: *U-net architecture for the adaptive sampler and denoiser network. The adaptive sampler network outputs a scalar per pixel, representing the sample map. For the denoiser, we additionally use three levels of kernel prediction (green boxes in lower part). At each level, we output per-pixel weights for a 5×5 kernel and a scalar layer blend weight. At the finest level, we additionally output a 5×5 kernel which is applied to the warped previous denoised output.*

is substantially faster and generates higher image quality than previous hierarchical recurrent denoisers.

- A scalable architecture with high image quality for larger networks, that still outperforms previous work while scaled down to real-time performance.

2 Our Approach

Figure 2 presents an overview of our approach. The design extends deep adaptive sampling and reconstruction (DASR) [KKR18], with a recurrent feedback loop and temporal reprojection. This increases the effective sample count and enforces a temporal error metric, which improves the temporal stability of the denoiser. Our method accurately predicts the sample density map using only temporal reprojected data and the geometry buffer of the current frame, allowing us to remove the initial sampling pass. In addition, we use hierarchical kernel prediction in the denoiser network for increased quality and robustness.

2.1 Networks

For both the sample map estimator and denoiser networks, we use U-nets [RFB15] with 3×3 convolution kernels throughout, as shown in Figure 3. The main differences between the two networks are the input feature guides, and how the output is processed.

The temporal reprojection steps uses per-pixel motion vectors $\mathbf{m}$ generated by the renderer at the primary intersection point, and warps, using bilinear filtering, the previous frame (c_{history}) to align with the current frame:

$$c_{\text{warped}}(x,y) = c_{\text{history}}(x+m_x, y+m_y)\,. \tag{1}$$

We use PyTorch's `grid_sample` function to implement this image warp during training.

Adaptive Sampler The adaptive sampling network is fed with the warped previous denoised output (warped using application-provided motion vectors) along with feature buffers for the current frame (normals, depth, motion vectors and albedo at first hit). The sample map is then produced using *direct prediction*, by combining all output features into a single component gray-scale image generated by a final convolutional layer. The sample map is normalized as suggested by Kuznetsov et. al [KKR18] to reach a desired average sample count:

$$\widehat{s}(p) = \text{round}\left(\frac{M \cdot e^{s(p)}}{\sum_{i=1}^{M} e^{s(i)}} \cdot n\right). \tag{2}$$

Here, M is the number of pixels in the image, n is the average number of samples per pixel, and $s(p)$ is the unnormalized output of the network. The normalization is similar to a *softmax* operation, and we have found that implementing this using a softmax over the unrolled image tensor improves the stability of the gradient computation as opposed to chaining the individual operations. We use Kuznetsov et. al's [KKR18] approximation of the renderer gradient during back-propagation.

Denoiser As shown in Figure 2 the input to the denoiser network consists of the adaptively sampled noisy image, rendered as dictated by the sample map, and all inputs of the sampler network, including the warped previous denoised output. The outputs of the last three hierarchical levels of the decoder network are fed through a convolutional layer predicting filter weights. Thus, we implement a multi-scale kernel predicting network as suggested by Vogels et al. [VRM*18]. Our denoiser architecture is illustrated in Figure 3. We use three levels of kernel prediction, each with a 5×5 pixel kernel, applied to a full resolution noisy image and, 2×, and 4× downscaled versions. The scaled and filtered images, $\mathbf{i}$ are combined using the scale composition suggested by Vogels et al. (Eq. 5), reproduced here for completeness. Let $\mathbf{i}^c$ represent the coarse-scale image, $\mathbf{i}^f$ the fine-scale image and α a trainable per-pixel scalar weight. $\mathbf{D}$ and $\mathbf{U}$ are 2× 2-downsampling and nearest-neighbor upsampling operators, respectively. The coarse and fine images are combined using:

$$\mathbf{o}_p = \mathbf{i}_p^f - \alpha_p \left[\mathbf{UDi}^f\right]_p + \alpha_p \left[\mathbf{Ui}^c\right]_p. \tag{3}$$

This composition is applied recursively from coarsest to finest scale.

Additionally, we predict a *temporal* 5×5 pixel kernel at the finest scale, which is applied to the reprojected denoised image of the previous frame (history). The filtered history is linearly blended with the result of the multi-scale filtered current frame using a trainable per-pixel scalar, α^t. As shown in Figure 3, this results in 51 per-pixel features at the finest scale: (two 5×5 kernels + α^t), and 26 features at the coarser scales (a 5×5 kernel + α). Similarly to Vogels et al., we observe higher quality networks when using kernel prediction compared to direct prediction. We compare direct prediction against kernel prediction in Section 3.

Training To account for the recurrent term (the reprojected denoised previous frame), we employ back-propagation through time [GBC16], training on sequences of N frames. Typically, we use sequences of five frames, where one frame is used for initialization

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

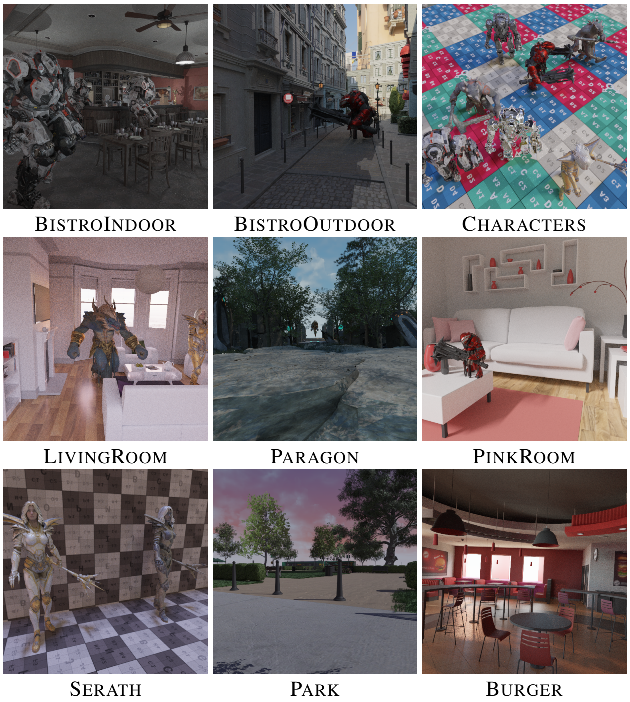


Figure 4: *Animations used to train the networks. For each scene, we dump between 16-25 clips, each with eight frames. We store 64 individual samples for each pixel of the input images (to allow for adaptive sampling). The target images are rendered with 1024 spp.*

and we train with the four subsequent unrolled iterations of the network. For the first frame, we initialize the recurrent term to the noisy uniformly sampled image at our target sample count.

We train sample map estimation and denoising end-to-end, which means that the loss term is only computed on the final denoised image. There is no specific loss computed for the sample map. We use both a spatial L_1 and a *temporal* L_1 loss term [CKS*17], weighted equally. The temporal L_1 term is intended to suppress temporal flickering and is computed as the L_1 norm of the temporal finite differences between frame i and frame $i-1$. Let x_i be the denoised frame and y_i be the corresponding reference frame, where i is the current time step. The temporal gradients are $\Delta x_i = x_i - x_{i-1}$ and $\Delta y_i = y_i - y_{i-1}$, and our loss, L, is

$$L = L_1(x_i, y_1) + L_1(\Delta x_i, \Delta y_i). \tag{4}$$

When training, the spatial loss is only applied to the last frame of each sequence, and the temporal loss by necessity involves the last two frames. This procedure ensures that the network has access to a reasonable amount of temporal information, and avoids transients from initializing the recurrent term.

2.2 Implementation

We implemented our technique in PyTorch [PGC*17] and integrated it with the Falcor rendering framework [BYF*19]. We initialize the trainable parameters of our networks using Xavier initialization [GB10] and train with the Adam optimizer [KB15] with an initial learning rate of 0.001. All networks are trained for 1000 epochs. We clamp input values to $([0, 65535])$ and transform input radiance values, x, according to $x' = \log(x+1)^{\frac{1}{2.2}}$. The log-squeeze is motivated by the 2^e exposure scaling of most tonemappers. Since we gamma correct the image before display we also included this transform. We recommend to match the training loss to the tonemapper used in the rendering system (if known).

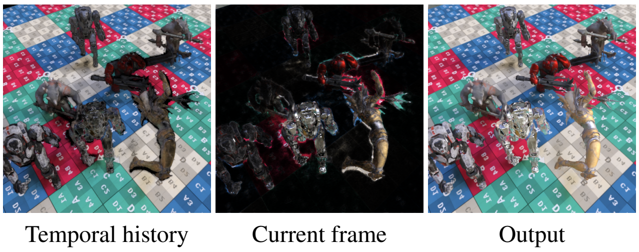


Figure 5: *The network aggressively uses temporal information. An example frame from the* CHARACTERS *scene where we visualize the filtered history (left) and the filtered image of the current frame (center) that are used to construct the denoised output (right). Note that most information is taken from the history. The animation has moving characters but a static camera.*

For the adaptive sampler network, we convert all color inputs (albedo and warped previous denoised output) to gray scale using the CCIR 601 weights $v = 0.2989\mathrm{r} + 0.587\mathrm{g} + 0.114\mathrm{b}$. The rationale behind this is that the adaptive sampling should be independent of chroma and motivated by noise levels, geometric complexity, or animation/disocclusion. To implement the renderer during training, we follow the pre-computation approach of Kuznetsov et al. [KKR18] and store pre-computed images of (independent) renderings with 2^n samples for $n \in [0, 5]$. During training, we generate desired per pixel sample counts by combining the appropriate layers. For example to get 13 spp, we combine the sample layers with 1+4+8 samples.

We train with input augmentations (random crops, flip x/y, rotate 90 degrees) and shuffle the training data each epoch. For denoisers with direct prediction networks, we also include hue permutations and a grayscale augmentation as this reduces color shifts.

We use nine animated scenes, shown in Figure 4 for training. For each scene we use 16-25 animation clips, each with eight frames, for a total of 1368 frames, each stored with per-sample info up to 64 spp and per-pixel feature guides (normals, depth, albedo and motion vectors). We render references with 1k samples. The rationale behind this setup is to maximize training diversity while keeping data generation and storage costs reasonable. Using separate clips rather than all frames of a longer animation helps the network rapidly see significantly different views of the scene, and training a CNN with lower quality references can be expected to converge with similar quality and speed as training with ground truth. For evaluation we use longer video clips and references with 4k spp.

3 Results

A main contribution of this work is temporally stable denoising. Figure 5 shows one example of how the denoiser network mixes

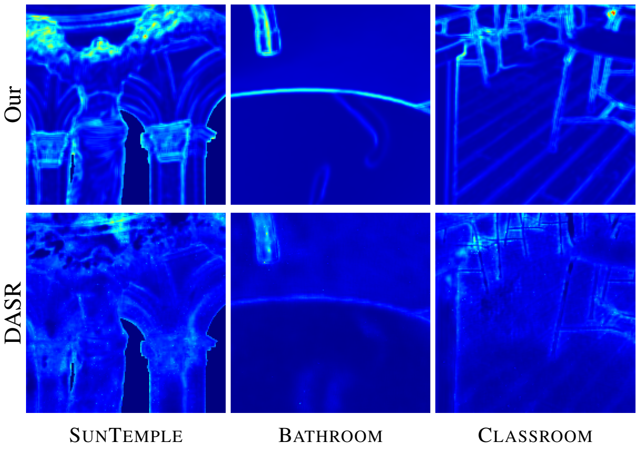


Figure 6: *Sample map comparison between our method and DASR, visualized as heatmaps. Our sample map is typically of higher quality than DASR since it is computed from the denoised & reprojected data rather than a low (1spp) initial sampling.*

the temporal history with the current denoised frame. As can be seen, it aggressively re-uses temporal data except for in areas of disocclusion or with view-dependent shading. We refer to the video for results on temporal stability as it is hard to convey in still images.

The temporal history greatly improves the adaptive sampling stage. In Figure 6 we compare the sample maps between our method and DASR. Our sample map is typically of higher quality than DASR since it is computed from the denoised & reprojected data rather than a low (1spp) initial sampling.

To evaluate quality, we present both peak signal-to-noise ratio (PSNR) and *temporal PSNR* (tPSNR), which is PSNR computed on the temporal finite differences $\Delta x_i = x_i - x_{i-1}$. Both metrics are applied on the tonemapped output. Figure 7 shows a breakdown of PSNR and tPSNR scores for two animation clips of the SUNTEMPLE scene rendered at 4 spp. We start with a simple direct prediction U-net (Direct prediction) with feature counts as reported in Figure 3, and observe how image quality is improved by adding three-level hierarchical kernel prediction (KPN), temporal recurrence (Our uniform), and adaptive sampling (Our adaptive). Note that the adaptive network consistently outperforms the corresponding uniform networks (wherein each pixel is sampled with exactly four samples), and is close to on par with 8 spp uniform for the first clip. A visual comparison for a selected frame is shown in Figure 8. The recurrent methods require a short transition period of 2-3 frames between clips, before reaching full quality. Such a period could easily be hidden in a cross-fade or transition, so we do not consider this a limitation. Other than for abrupt camera transitions (cuts), the temporal methods are stable even for large and complex camera and object motion.

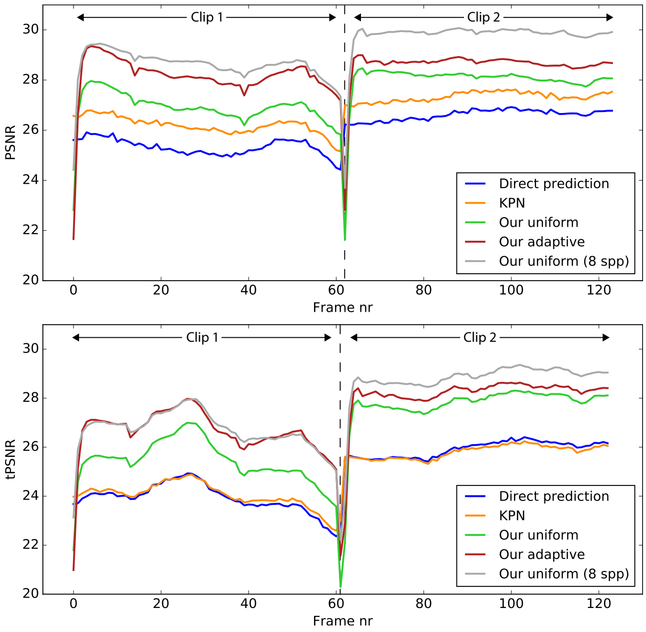
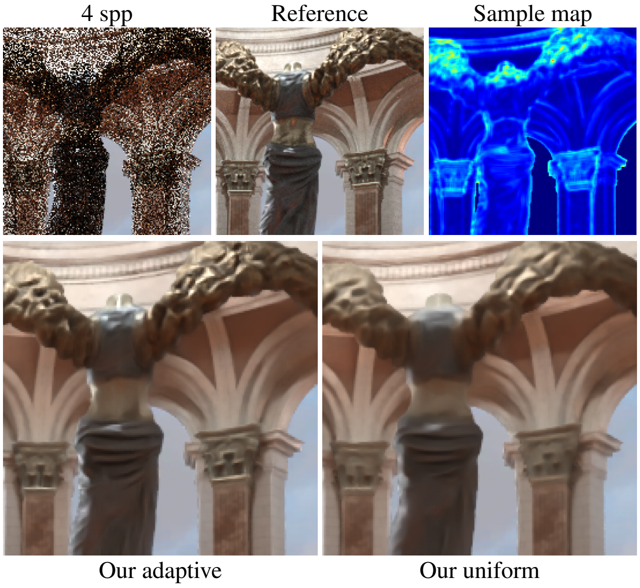


Figure 7: *PSNR and tPSNR scores for the* SUNTEMPLE *animation. Our baseline is a U-net directly predicting the colors (Direct Prediction), and observe how quality is improved by adding kernel prediction (KPN), temporal recurrence (Our uniform), and adaptive sampling (Our adaptive). The results are for denoising 4 spp images unless otherwise noted.*

We evaluate our method quantitatively in Figure 9, where we compare it with competing algorithms for the SUNTEMPLE animation. Visual comparisons are shown in Figure 10 and a larger collection of images is included in the supplemental material. In both figures, we compare our algorithm to recursive denoising autoencoders [CKS*17] (RAE), deep adaptive sampling and reconstruction [KKR18] (DASR) and spatiotemporal variance-guided filtering [SKW*17] (SVGF). Note that SVGF is designed for 1 spp input with features from a rasterized geometry buffer, and does not directly fit with our ray tracing framework. We therefore generated these results by tracing four paths from each rasterized primary pixel, which unfortunately means that SVGF results do not benefit from anti-aliasing. In the supplemental material we include a version of SVGF applied on an image rendered at 2× higher resolution, with one path per pixel, followed by 2× downscaling. This comes at roughly 4× the filtering cost, and improves PSNR scores by about one decibel on average.

To compare the methods in a similar setting, we altered DASR to use the same, slightly larger, U-nets as our method (~850k weights, see Figure 3). We use the RAE architecture as described in their paper (~2.6M weights, a larger network than ours) but we passed the albedo as an additional guide instead of doing the albedo demodulation. We trained both variants, and the former approach generated higher quality results for our dataset. We use the same loss function (spatial and temporal L_1) and log-space training for all learning-based methods, and as such we do not expect results exactly comparable to what they report. It is straightforward to increase or decrease the U-net feature counts, or tweak training parameters for all methods. We also feed DASR, RAE, and our method the exact same feature guides. The visibility guide of the DASR paper is omitted in this comparison since it is hard to generalize for scenes

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.


4 spp

Reference

Sample map

Our adaptive

Our uniform

Figure 8: *Quality comparison between our adaptive and uniform algorithms for the* SunTemple *scene. The sample map shows that our sampling network prioritizes the specular highlights on the statue, and the details on the pillars in the background. The denoised image is significantly improved in those regions.*

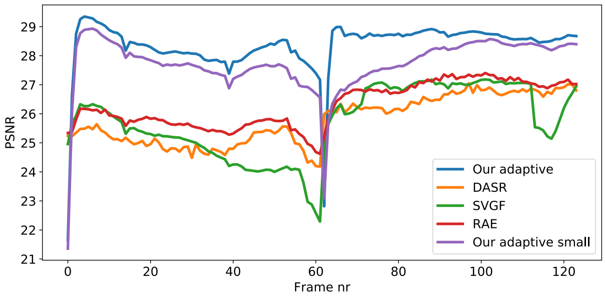


Figure 9: *Image quality comparison between our adaptive method (we also include a scaled down version of the architecture: our adaptive small), deep adaptive sampling and reconstruction (DASR), spatiotemporal variance-guided filtering (SVGF), and the recurrent autoencoder (RAE) on the* SunTemple *scene.*

with large light counts or environment lighting, and we found it to only have a small impact on image quality.

Scene	Our adapt.	Our unif.	DASR	Direct prediction
SunTemple	7.0	9.8	10.0	14.8
Classroom	7.0	10.6	12.4	15.0
Bathroom	7.3	9.7	12.6	12.1

Table 1: *Relative mean square error (rMSE) for the* SunTemple, Classroom, *and* Bathroom *scenes. For clarity, we factor out 0.001 from all rMSE values.*

Scene	Clips	Hold-out	Specialized
SunTemple	16	28.1 dB	28.3 dB
BistroIndoor	16	32.4 dB	32.5 dB
PinkRoom	25	31.8 dB	32.5 dB

Table 2: *PSNR scores for different training sessions with the current scene eliminated (hold-out), or specialized for that scene. PSNR scores are computed on one frame per clip per scene, and the individual scores are averaged.*

The RAE paper states that *the recurrent loops at finer resolutions have insufficient temporal receptive field to reproject the high-frequency features*. We observe a similar behavior: the hierarchical recurrent architecture of RAE is significantly less temporally stable than our architecture, likely because their recurrent state is not reprojected to align with the current frame. They mitigated this drawback somewhat by applying temporal anti-aliasing (TAA) [Kar14] as a supplemental post-processing pass to all methods in the evaluation. With our warped recurrent architecture, there is no need for a last-mile post-processing step to improve temporal stability, instead, our trained temporal reprojection is integrated in the network architecture. Please refer to the accompanying video to see temporal robustness comparisons without any TAA post-processing. We argue that our proposed network is significantly more temporally stable, and comes at a much lower computational and storage cost compared to the full hierarchy of (non-warped) convolutional RNN layers with accompanying state.

In Table 1 we report relative mean square error (rMSE) for a set of validation images not a part of the training set (full image is the supplemental material). The relative quality of the different algorithms are similar to our PSNR results even though this metric is measured on high dynamic range data. The Bathroom scene is a notable outlier, where DASR fails to outperform a direct prediction denoiser. We speculate that this is because of very high noise levels coupled with bad feature guides, due to the many reflective surfaces, which makes it hard to predict the sample map from the 1 spp initial sampling. Our adaptive method also performs worse for this scene, but we note that the higher quality temporal data still allows it to outpace the uniform version.

The Bathroom scene is designed to be a hard case for our algorithm. For mirrors, motion vectors will match the motion of the mirror surface, rather than the motion of the reflected object, which means that reprojection is incorrect for large parts of this scene. A similar problem appear in shadows from moving objects. RAE and SVGF have the same challenges when reprojection fails. Still, we note from the images and video that our algorithm tends to outperform the competition while remaining temporally stable. We show an example in Figure 11.

3.1 Generalization

In order to evaluate how well the method generalizes, we trained different versions with *hold-out*. That is, we select a scene and train two versions: one where the scene is eliminated from the training set, and one version specialized only for that particular scene. We

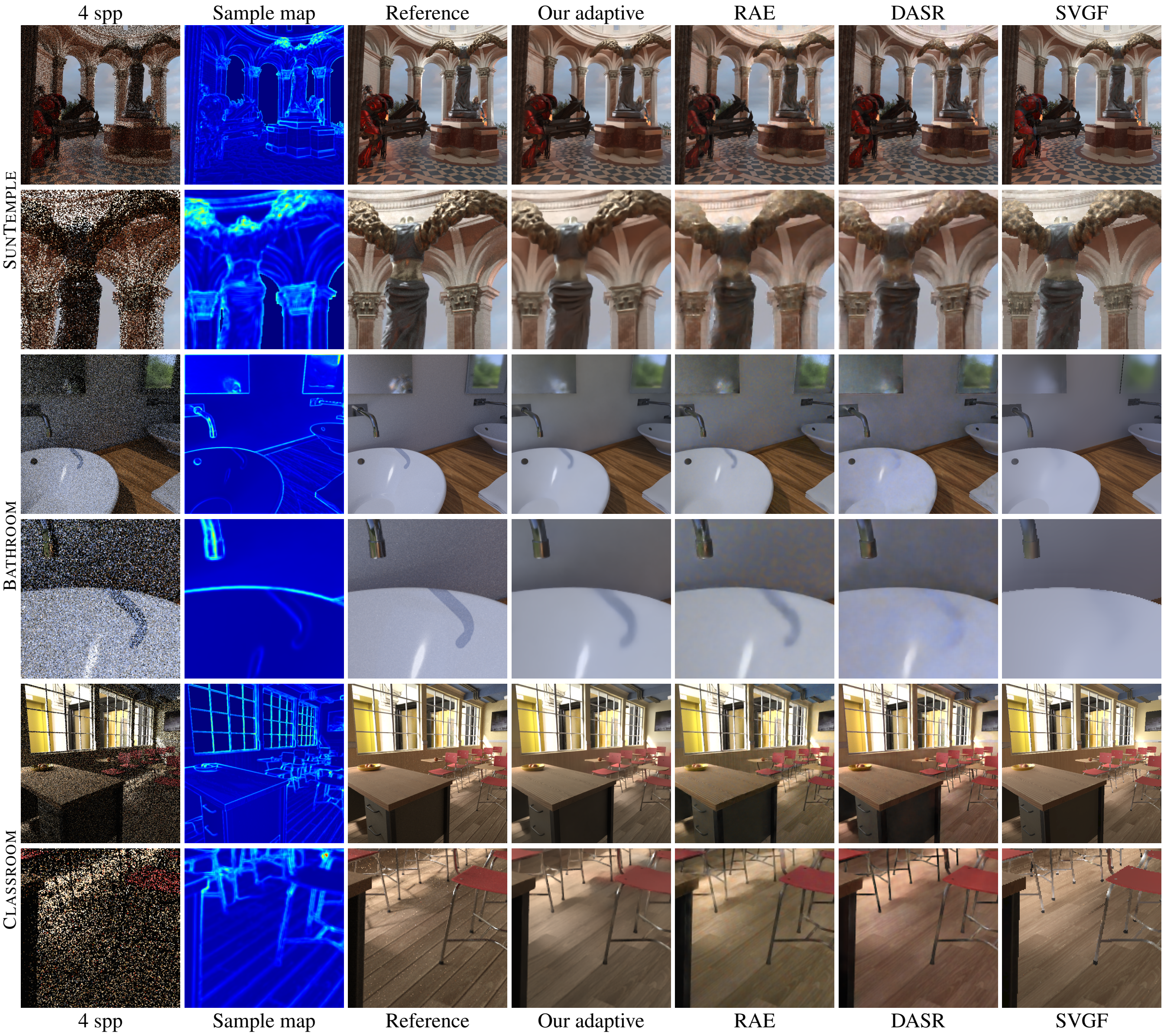


Figure 10: *Denoising quality from an average of 4 spp. These scenes were not part of the training set. We show comparison with RAE (recurrent autoencoder), SVGF (spatiotemporal variance guided filtering), and DASR (deep adaptive sampling and reconstruction).*

then evaluate image quality on that scene to determine the quality difference of a generalized network compared to the practical upper limit. The PSNR scores of this study are summarized in Table 2. As can be seen, the method appears to generalize well, with the hold-out versions of the network often having quality on par with the specialized network.

3.2 Performance and Scaling

Compared to DASR, the overhead at inference time of the temporal recurrence loop and hierarchical kernel evaluation of our method is less than 1 ms at 1920 × 1080 pixel resolution. Reprojection is implemented as a GPU texture lookup, so it's extremely fast. Given the improvements to image quality and temporal stability, we believe this is a motivated cost, even for real-time rendering applications.

Config.	Sample map est.	Denoiser	Kernel	Total
LARGE	20 ms	21 ms	1 ms	42 ms
SMALL	3 ms	10 ms	1 ms	14 ms

Table 3: *Performance breakdown for our large (high quality) and small (performance) network configurations, running on an NVIDIA Geforce RTX 2080 Ti graphics card at* 1920 × 1080 *pixel resolution.*

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

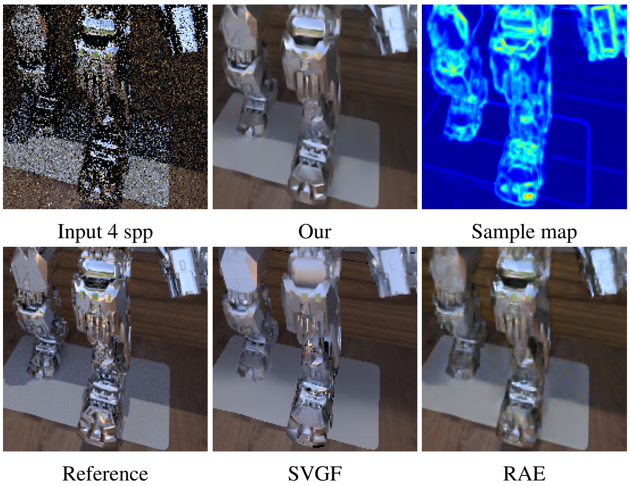


Figure 11: *A hard case with moving shadows and reflections where motion vectors are inaccurate. Note that our sample map fails to detect the moving shadow below the left foot. This is a challenging case for all algorithms relying on temporal reprojection.*

We have highlighted the network architectural components which have significant impact on quality, e.g., kernel prediction, adaptive sampling and temporal feedback (see Figure 7). Additionally, the number of network layers, feature counts, and kernel sizes can be tweaked to trade quality for performance. In our experience most such optimizations scale gracefully with runtime performance being roughly inversely proportional to image quality. To that end, Table 3 shows run-time performance of both the LARGE network configuration, presented in this paper, and an optimized SMALL configuration that trades image quality for additional performance. The small version improves performance by 3× while reducing the PSNR score of the SUNTEMPLE animation by a modest 0.6 dB, still outperforming competing algorithms, as can be seen in Figure 9. The implementation details of the SMALL network can be found in the supplemental material.

To evaluate the impact of a larger recurrent state, we configured our network with 64 additional recurrent features (in addition to the denoised color output). This allows the network to, e.g., pass through additional data such as input samples and sample map from previous frame, or freely compute additional data to be reprojected and reused for the current frame. This configuration (with larger recurrent state) had 0.2 dB lower PSNR on our validation set compared to the LARGE network, which is within expected variations due to training with stochastic gradient descent.

4 Discussion

We have described a rendering approach using deep learning, temporal reprojection and adaptive sampling to achieve high quality, temporally stable, denoising of path traced animations at near real-time rates. By combining temporal reuse and adaptive sampling, the network learns to reconstruct difficult temporal effects, such as disocclusion and view dependent shading.

Our system is scalable in that network complexity controls the ratio between quality and performance, and we have shown that this holds for larger, high quality, networks all the way down to the real-time performance realm. Even for the smallest network design, we note that our architecture outperforms previous work significantly both in quality as well as in performance.

Seemingly, the most objectionable artifact from our method is ghosting and the network tends to error on the side of aggressive temporal reuse. See, for example, the CLASSROOM scene in Figure 10. While this is particularly visible in still images, it tends to be much less pronounced in motion, though if one looks carefully it is still possible to notice the artifacts in the accompanying video.

View dependent shading effects, such as reflections, remain an open issue. Such effects makes it hard to compute accurate motion vectors, and different shading terms can have different motion. Consequently our warping will be less accurate and temporal reuse is limited. This is stressed in the BATHROOM scene, which contains many mirror surfaces. The results for this scene is still surprisingly good and we perceive it as temporally stable. Seemingly the motion vectors are accurate enough for the planar and simple reflective objects. The more complex robot object consists of rotating planar mirrors. This makes temporal reuse virtually impossible, but also makes it very hard to perceive if the result is temporally stable.

Acknowledgments

We thank Bill Dally, and David Luebke for discussions and supporting the research; NVIDIA Research staff for suggestions and discussions; Donald Brittain for deep learning framework code used in performance measurements; Petrik Clarberg, Nir Benty and Kai-Hwa Yao for Falcor support.

References

[BVM*17] BAKO S., VOGELS T., MCWILLIAMS B., MEYER M., NOVÁK J., HARVILL A., SEN P., DEROSE T., ROUSSELLE F.: Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings. *ACM Trans. Graph. 36*, 4 (2017). 1

[BYF*19] BENTY N., YAO K.-H., FOLEY T., OAKES M., LAVELLE C., WYMAN C., CLARBERG P.: The Falcor rendering framework, 10 2019. URL: https://github.com/NVIDIAGameWorks/Falcor. 4

[CKS*17] CHAITANYA C. R. A., KAPLANYAN A. S., SCHIED C., SALVI M., LEFOHN A., NOWROUZEZAHRAI D., AILA T.: Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. *ACM Trans. Graph. 36*, 4 (2017), 98:1–98:12. 2, 4, 5

[GB10] GLOROT X., BENGIO Y.: Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (2010), Teh Y. W., Titterington M., (Eds.), vol. 9 of *Proceedings of Machine Learning Research*, pp. 249–256. 4

[GBC16] GOODFELLOW I., BENGIO Y., COURVILLE A.: *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 3

[GLA*19] GHARBI M., LI T.-M., AITTALA M., LEHTINEN J., DURAND F.: Sample-based Monte Carlo Denoising Using a Kernel-splatting Network. *ACM Trans. Graph. 38*, 4 (july 2019), 125:1–125:12. 1

[Kar14] KARIS B.: High-Quality Temporal Supersampling. In *SIGGRAPH Courses: Advances in Real-Time Rendering in Games* (2014). 2, 6

[KB15] KINGMA D. P., BA J.: Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference for Learning Representations* (2015). 4

[KIM*19] KOSKELA M., IMMONEN K., MÄKITALO M., FOI A., VIITANEN T., JÄÄSKELÄINEN P., KULTALA H., TAKALA J.: Blockwise Multi-Order Feature Regression for Real-Time Path-Tracing Reconstruction. *ACM Trans. Graph. 38*, 5 (2019), 138:1–138:14. 2

[KKR18] KUZNETSOV A., KALANTARI N. K., RAMAMOORTHI R.: Deep Adaptive Sampling for Low Sample Count Rendering. *Computer Graphics Forum 37* (2018), 35–44. 1, 2, 3, 4, 5

[KSL*19] KAPLANYAN A. S., SOCHENOV A., LEIMKÜHLER T., OKUNEV M., GOODALL T., RUFO G.: Deepfovea: neural reconstruction for foveated rendering and video compression using learned statistics of natural videos. *ACM Trans. Graph. 38*, 6 (2019), 212:1–212:13. 2

[PGC*17] PASZKE A., GROSS S., CHINTALA S., CHANAN G., YANG E., DEVITO Z., LIN Z., DESMAISON A., ANTIGA L., LERER A.: Automatic differentiation in PyTorch. In *NIPS-W* (2017). 4

[RFB15] RONNEBERGER O., FISCHER P., BROX T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (2015), vol. 9351, pp. 234–241. 1, 3

[SKW*17] SCHIED C., KAPLANYAN A., WYMAN C., PATNEY A., CHAITANYA C. R. A., BURGESS J., LIU S., DACHSBACHER C., LEFOHN A., SALVI M.: Spatiotemporal Variance-guided Filtering: Real-time Reconstruction for Path-traced Global Illumination. In *Proceedings of High Performance Graphics* (2017), HPG '17, pp. 2:1–2:12. 2, 5

[SPD18] SCHIED C., PETERS C., DACHSBACHER C.: Gradient Estimation for Real-time Adaptive Temporal Filtering. *Proc. ACM Comput. Graph. Interact. Tech. 1*, 2 (2018), 24:1–24:16. 2

[SVB18] SAJJADI M. S. M., VEMULAPALLI R., BROWN M.: Frame-Recurrent Video Super-Resolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2018). 2

[VRM*18] VOGELS T., ROUSSELLE F., MCWILLIAMS B., RÖTHLIN G., HARVILL A., ADLER D., MEYER M., NOVÁK J.: Denoising with Kernel Prediction and Asymmetric Loss Functions. *ACM Trans. Graph. 37*, 4 (2018), 124:1–124:15. 1, 2, 3

[WLZ*18] WANG T.-C., LIU M.-Y., ZHU J.-Y., LIU G., TAO A., KAUTZ J., CATANZARO B.: Video-to-Video Synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)* (2018). 2

[XZW*19] XU B., ZHANG J., WANG R., XU K., YANG Y.-L., LI C., TANG R.: Adversarial Monte Carlo Denoising with Conditioned Auxiliary Feature Modulation. *ACM Trans. Graph. 38*, 6 (2019), 224:1–224:12. 1

[ZJL*15] ZWICKER M., JAROSZ W., LEHTINEN J., MOON B., RAMAMOORTHI R., ROUSSELLE F., SEN P., SOLER C., YOON S.-E.: Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports) 34*, 2 (2015), 667–681. 1