Eurographics Symposium on Rendering (2023)
T. Ritschel and A. Weidlich (Editors)

Deep Compositional Denoising on Frame Sequences

Xianyao Zhang[†1,2], Gerhard Röthlin[2], Marco Manzi[1], Markus Gross[1,2] and Marios Papas[†2]

[1]ETH Zürich, Switzerland
[2]DisneyResearchlStudios, Switzerland

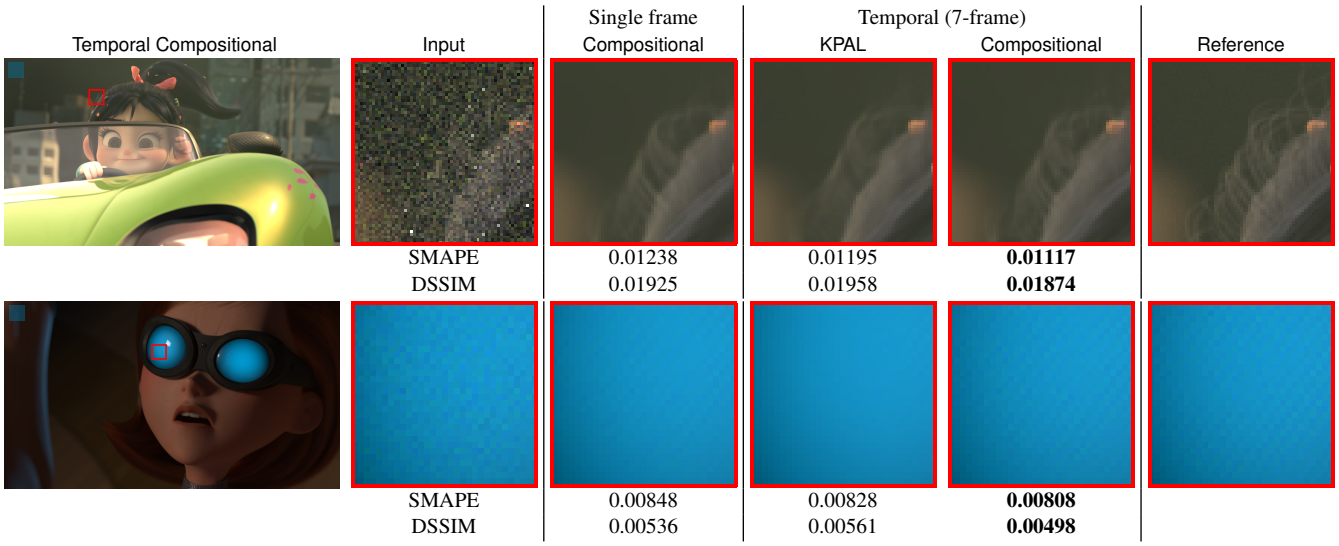

Figure 1: *Visual results from the single-frame compositional denoiser [ZMV*21], the multi-frame temporal kernel-predicting denoiser (KPAL [VRM*18]) and our multi-frame compositional denoiser, showcasing the improved detail reconstruction from the temporal compositional denoiser over both counterparts. The first image was created by the Walt Disney Animation Studios (© Disney), and the second by Pixar Animation Studios (© Disney / Pixar).*

1. Introduction

Path tracing is the prevalent rendering algorithm in the animated movies and visual effects industry, thanks to its simplicity and ability to render physically plausible lighting effects. However, we must simulate millions of light paths before producing one final image, and error manifests as noise during rendering. In fact, it can take tens or even hundreds of CPU hours on a modern computer to render a plausible frame in a recent animated movie. Movie production and the VFX industry rely on image-based denoising algorithms to ameliorate the rendering cost, which suppresses the noise due to rendering by reusing information in the neighborhood of the pixels both spatially and temporally.

The temporal reuse of information from neighboring frames in a sequence is a critical part of a reliable and high-quality denoiser suitable for use for final-quality frames in production. First, the use of neighboring frames expands the neighborhood of each pixel, allowing the denoiser to use a higher number of similarly distributed neighbors to reduce noise. Second, the context from neighboring frames helps suppress flickering artifacts and preserves the temporal coherence of the denoised frame sequence compared to denoising each frame in isolation.

The single-frame compositional denoiser [ZMV*21] decomposes the noisy input image into multiple easier-to-denoise components, improving the quality of still-image denoising. A typical choice of 4 components delivers a 20% to 25% reduction in the rendering budget to match the denoising quality of a previous state-of-the-art kernel-predicting neural denoiser, which we denote KPAL in the description below [VRM*18]. However, the compositional denoiser does not support temporal denoising of frame sequences, leading to artifacts. Extending the compositional denoiser to use temporal information while denoising frame sequences presents practical challenges. A naive extension can significantly increase memory and runtime costs which become obstacles to the adoption in production environments.

The original single-frame approach processes each frame in a sequence independently. It decomposes the frame into learned com-

DOI: 10.2312/sr.20231142

ponents, which are then denoised through kernel prediction and combined into the final denoised image. Instead of discarding the learned components for previous frames when denoising the next frame, our method stores them in memory for later reuse, significantly reducing computation costs. As we progress through the sequence of frames, the per-frame learned components in the temporal window are regrouped into component temporal groups centered around the frame to be denoised. The component temporal groups are then passed to a temporal denoising module for context and kernel application with a multi-scale architecture similar to KPAL [VRM*18], which outputs one denoised component of the frame to be denoised.

Furthermore, as storing the learned components in memory with full 32-bit precision can exceed the memory capacity of current hardware used in production, we propose an 8-bit quantization scheme of the intermediate results, which halves the peak memory consumption. Our proposed quantization is carefully crafted to avoid introducing any quantization artifacts.

In summary, we propose a novel extension for processing frame sequences that provides context from neighboring frames while avoiding redundant computation and a novel quantization approach that reduces the memory overhead without introducing artifacts. Our method achieves a significant improvement in denoising quality over KPAL and ameliorates temporal denoising artifacts from the single-frame approach. This enables the use of the compositional denoiser [ZMV*21] in production settings. We would like to note that all the denoising time and RAM consumption measurements in this report are based on a multi-thread CPU implementation of the denoising methods, as CPUs form the majority of computation power in today's render farms. Nevertheless, our optimizations, particularly the quantization to reduce RAM consumption, could be translated to the GPU version of the temporal compositional denoiser.

2. Method and implementation

In this section, we describe our three steps towards making the compositional denoiser suitable for the production environment. We start with introducing the temporal extension (Section 2.1) that produces improved denoising quality but requires significant computation costs. Then, to ameliorate the computation costs, we propose the caching and reuse of learned components (Section 2.2) which greatly reduced the denoising time at the cost of increased RAM consumption. Finally, we describe the 8-bit quantization scheme (Section 2.3) that halves the peak RAM consumption and makes the method more compatible with typical production environments. We additionally provide implementation details in Section 2.4.

2.1. Temporal extension of the compositional denoiser

Similar to the single-frame compositional denoiser [ZMV*21], our temporal compositional denoiser operates in two steps, decomposition and denoising. Its core architecture consists of a decomposition module, which decomposes and encodes a frame into 2 learned components, and a kernel-predicting denoising module, which takes a temporal group of frames (including the frame to be denoised, 3 previous frames, and 3 next frames) and corresponding auxiliary features and outputs the denoised version of the frame to be denoised. Both modules contain a U-Net [RFB15] as the backbone.

Figure 2 summarizes the workflow of the temporal compositional denoiser. In the decomposition step, we run the decomposition module a total of 3 times hierarchically to produce 4 learned components for each frame independently, where each component consists of the component color image, the component mask, and the component learned features (see [ZMV*21] for more details on the decomposition step). The per-frame components are then grouped into 4 separate temporal component groups, each processed by the denoising module. The denoised components are summed up to yield the final denoised frame.

2.2. Reusing learned components across frames

The compositional denoiser has a higher run time than the KPAL denoiser [VRM*18] because of repeatedly running the decomposition and denoising modules. This discrepancy is further amplified for temporal denoising since the learned components for each frame are needed for all frames in the temporal neighborhood. Our temporal compositional denoiser allows caching and reusing per-frame learned components for sequence denoising, significantly reducing the overhead.

The key to enabling the reuse of components is delaying motion compensation (warping) until after decomposition, which is also shown in the middle of Figure 2, as the circled "Regroup & Warp" label. Motion compensation aligns the neighboring frames with the frame to be denoised by following the renderer's motion vectors or optical flow. and it needs to be recomputed whenever that frame changes. In our method, this motion compensation step is performed *after* decomposition, and each component group is processed individually using the same set of motion vectors. While it is also possible to run the motion compensation *before* decomposition, such an implementation would be more costly as it prevents the reuse of learned components because the decomposition of a neighboring frame will be dependent on the current frame to be denoised.

With a 7-frame temporal window, caching and reusing the per-frame components save 60% to 70% of the total denoising time for a long sequence, compared to a naive implementation, which improves the practicality of our temporal compositional denoiser.

We measured the denoising time on CPU for a 7-frame sequence of 1920×804 frames and all 3 passes (diffuse, specular and alpha) on an AMD Ryzen 7 5800X 8-Core Processor with 16 threads. The optimized implementation required 360 seconds which through caching we can reduce to 105 seconds. For reference, the corresponding denoising time for KPAL is 21 seconds. Though the compositional denoiser is slower than the baseline, the savings in the render budget outweigh the additional denoising cost.

2.3. 8-bit quantization for memory consumption optimization

When learned components are cached, the temporal compositional denoiser has a much higher RAM consumption compared to KPAL

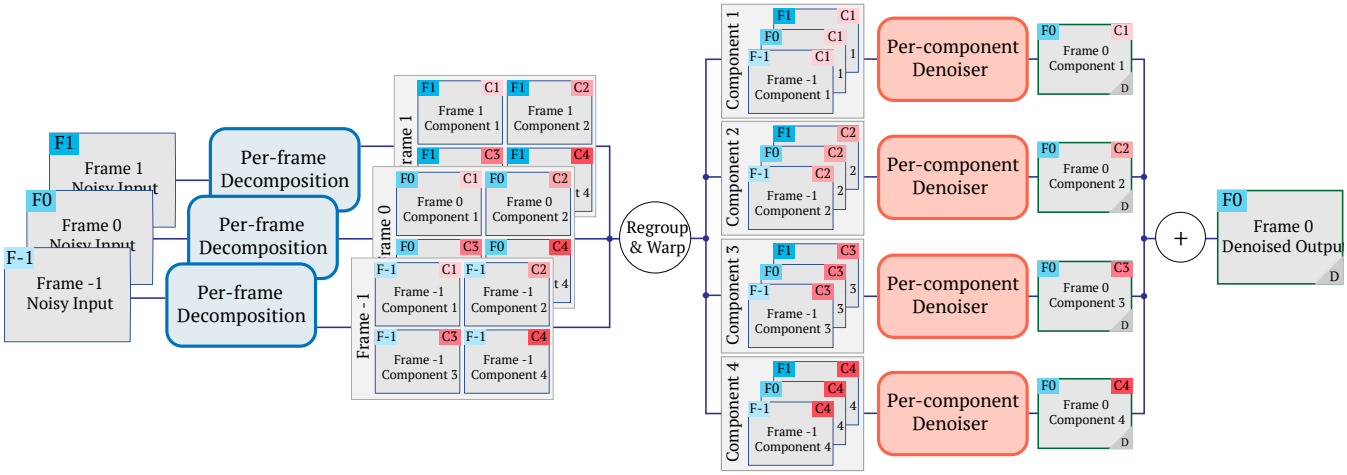

Figure 2: *Temporal compositional denoiser pipeline with per-frame decomposition and per-component denoising, using a 3-frame sequence, indexed by* -1*,* 0 *and* 1*, as an example. The sharp rectangles represent data (images and/or feature maps), the round-corner rectangles represent trainable modules, and circles represent non-trainable operations. Labels on the top-left and top-right corners of the rectangles correspond to frame index and component index, respectively, and bottom-right shaded triangle with "D" distinguishes denoised images from input ones.*

[VRM*18]. Additionally, the predicted per-component denoising kernel weights need to be stored for efficiently denoising arbitrary output variables (AOVs) such as per-light noisy images. To reduce RAM consumption, we introduce an 8-bit quantization on these intermediate tensors which are originally stored as 32-bit floats.

The quantization operates on each *channel* of the tensors, whose shape can be denoted as $H \times W \times C$, where the three dimensions represent the height, width, and channels, respectively. To quantize one channel (a matrix of shape $H \times W$), we store its minimum and maximum values in 32-bit floats, normalize the channel to $[-1, 1]$ using the minimum and maximum values, and quantize the resulting normalized tensor to 8-bit integers. Moreover, for the denoising kernel weights and the per-component noisy images, a gamma curve $f(x) = x^{\frac{1}{2.5}}$ is applied before the normalization to compress the dynamic range and reduce the impact of outliers on quantization results.

Our quantization reduces the peak RAM consumption of the temporal compositional denoiser by more than 50%. For instance, when denoising a 7-frame sequence of three-channel 2560×1440 images, the peak RAM consumption dropped from 124.9 GiB to 59.8 GiB after quantization. These savings enable the application of our denoiser on images at this widely-used resolution on machines with 64 GiB memory, a common case for today's workstations. For reference, KPAL requires 40.8 GiB of memory. A breakdown of the memory utilization is provided in Figure 3 This optimization is crucial in making the temporal compositional denoiser practical in the current production environment, since for final-quality renderings, an increase in the denoiser's run time can be more tolerated than one in its RAM usage, because the latter is a hard constraint depending on the machine specifications at each studio.

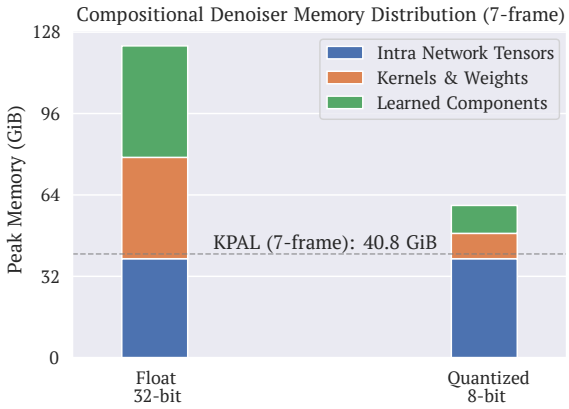

Figure 3: *Breakdown of peak memory usage when denoising a* 2560×1440 *7-frame sequence in* 3 *passes (diffuse, specular and alpha) with our temporal compositional denoiser. The memory of kernels, multi-scale reconstruction weights and learned components is reduced by* $4\times$ *yielding a total reduction of more than* 50%*.*

2.4. Implementation details

We train and evaluate our model on renderings from WDAS (Hyperion renderer [BAC*18]) and Pixar Animation Studios (RenderMan [CFS*18]), and compare our method with the KPAL denoiser [VRM*18] as the baseline. In addition to the 7-frame versions, we also train the original single-frame compositional denoiser and the baseline kernel-predicting denoiser to evaluate the difference in the compositional denoiser's improvements. The compositional denoisers use 4 learned components. The multi-frame denoisers use 4 scales in their U-Nets and multi-scale reconstruction modules, and the single-frame denoisers use 5 scales.

We use SMAPE [VRM*18] as the training loss, and the denoisers are trained to process the diffuse and specular components of each image separately. Note that for denoising different components ("passes"), the same denoiser model is run multiple times with the different component in the place for "color" input; this is the same for all four methods listed above. We train the methods with the Adam optimizer [KB14] until 2.88M iterations, performing learning rate decay with a factor of 10 three times, at 2.09M, 2.36M and 2.62M iterations, respectively. The training is performed on an NVIDIA RTX 2080ti GPU and requires approximately 30 days to complete. The methods are trained on 128×128 patches, with the baselines using batch size 4 and the compositional denoisers batch size 1 due to their increased RAM consumption.

3. Results and discussion

We compare our temporal compositional denoiser with the KPAL baseline [VRM*18] on a test set of 34 production-level scenes at various noise levels. Results show that, in the multi-frame scenario, the compositional denoiser shows improvement on almost all evaluation images over the KPAL baseline, according to the SMAPE criterion, and on average provides a 5% reduction of SMAPE and 6% of DSSIM (1-SSIM [WBSS04]) error. Similar improvements are found in the comparison between the single-frame compositional and KPAL denoisers. Notably, the 8-bit quantization reduces the RAM consumption without producing any noticeable artifacts on our evaluation dataset.

In Figure 1, we showcase the denoising result improvement from the temporal compositional denoiser over the single-frame compositional denoiser and the temporal baseline KPAL. We could observe that the compositional denoiser preserves the most details among the three methods, and the improvement is especially visible when compared with KPAL which does not use learned decomposition.

To measure the impact of the temporal compositional denoiser on rendering time, we estimate the *sampling budget savings* by our method over the KPAL baseline [VRM*18], similar to Figure 8 by Zhang et al. [ZMV*21]. More specifically, we calculate the estimated input sample count for our method to reach the same SMAPE error as the baseline. Starting from a set of available sample count levels for a scene, we use linear interpolation in the log-log space to create a smoother spp–SMAPE curve. Then, given the SMAPE value at a certain sample count for the baseline, we use the spp–SMAPE curve from our method to find the sample count at which our method reaches that error value. The difference between these two sample count levels provides an estimate on our method's capability of reducing render time. When averaged across the available sample count levels and the scenes in our test set, the temporal compositional denoiser delivers roughly a 20%–25% reduction in sampling budget to reach the same denoising quality of the temporal KPAL method, similar to the improvement on single-frame denoising.

4. Summary and conclusion

In this report, we described our method for extending the compositional denoiser [ZMV*21] to process frame sequences while also meeting practical requirements from production. For the neural network architecture, we use per-frame decomposition and extend the denoising module to handle frame sequences, which shows improved results over the temporal neural denoiser but is computationally intensive. We reuse the per-frame learned components by caching them in memory and recomputing motion compensation whenever the denoised frame changes, saving around two-thirds of the computation cost when denoising a long sequence. Finally, to combat the RAM consumption issue resulting from caching, we apply an 8-bit quantization on the cached tensors, which halves the peak RAM consumption and makes this high-quality denoiser practical for use in production environments.

References

[BAC*18] Burley B., Adler D., Chiang M. J.-Y., Driskill H., Habel R., Kelly P., Kutz P., Li Y. K., Teece D.: The design and evolution of disney's hyperion renderer. *ACM Trans. Graph. 37*, 3 (July 2018). URL: https://doi.org/10.1145/3182159, doi:10.1145/3182159. 3

[CFS*18] Christensen P., Fong J., Shade J., Wooten W., Schubert B., Kensler A., Friedman S., Kilpatrick C., Ramshaw C., Bannister M., et al.: Renderman: An advanced path-tracing architecture for movie rendering. *ACM Transactions on Graphics (TOG) 37*, 3 (2018), 1–21. 3

[KB14] Kingma D. P., Ba J.: Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014). arXiv:1412.6980. 4

[RFB15] Ronneberger O., Fischer P., Brox T.: U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention - 18th International Conf.* (2015), Springer, pp. 234–241. 2

[VRM*18] Vogels T., Rousselle F., McWilliams B., Röthlin G., Harvill A., Adler D., Meyer M., Novák J.: Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018) 37*, 4 (2018), 124:1–124:15. doi:10.1145/3197517.3201388. 1, 2, 3, 4

[WBSS04] Wang Z., Bovik A., Sheikh H., Simoncelli E.: Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (April 2004), 600–612. 4

[ZMV*21] Zhang X., Manzi M., Vogels T., Dahlberg H., Gross M., Papas M.: Deep Compositional Denoising for High-quality Monte Carlo Rendering. *Computer Graphics Forum 40*, 4 (2021), 1–13. doi:10.1111/cgf.14337. 1, 2, 4