Supplemental Material: Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder

CHAKRAVARTY R. ALLA CHAITANYA, NVIDIA and McGill University
ANTON S. KAPLANYAN, NVIDIA
CHRISTOPH SCHIED, NVIDIA and Karlsruhe Institute of Technology
MARCO SALVI, NVIDIA
AARON LEFOHN, NVIDIA
DEREK NOWROUZEZAHRAI, McGill University
TIMO AILA, NVIDIA

ACM Reference format:
Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Supplemental Material: Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder. *ACM Trans. Graph.* 36, 4, Article 98 (July 2017), 2 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073601

1 TRAINING DATA MANAGEMENT

Each image in a training sequence consists of multiple buffers: noisy 1spp image in HDR linear color space, linear normalized depth from the camera, shading normals in view space, and an effective material roughness.

We have found that it is important to have high precision for colors and depth, while normals and roughness can be stored with a standard 8 bits/channel precision. In order to be able to transfer the data set from a rendering farm to the training cluster, we use the following packing: HDR color is packed together with depth (in alpha channel) into a single EXR file with 16bits/channel half-precision floating point, while normals and roughness are packed as RGB channels of a compressed PNG file.

Once the data is on the training cluster, the training process needs to read random crops from these deep images. With a naïve reading from EXR and PNG, around 80−85% of training time is spent on image decompression. Therefore, we introduced an additional preprocessing step, where we go through the whole training set, decompress all images and store them as flat Python's `numpy` arrays 16-bit half-precision and 8-bit fixed point blobs correspondingly. This process needs to be done only once for each training set, and takes 30-40 minutes for a set of 3 sequences with 1000 images each. After this preprocess, during training we `memmap` and read only parts corresponding to a selected random crop. This lead to 6× speed up of the training process and GPU utilization grew to 98 – 99%, meaning that the process it GPU-bottlenecked.

2 ADDITIONAL RESULTS WITH 1SPP

In Figure 1 we provide additional results with four more scenes also used in [Bitterli et al. 2016], as well as the San Miguel scene with a different illumination setup with sharp shadows. The network was never trained on them.

Filter	San Miguel		Red Room		Horse Room		Living Room	
	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM
AAF	0.092	0.749	0.167	0.178	0.188	0.367	0.221	0.221
EAW	0.147	0.591	0.041	0.942	0.108	0.801	0.067	0.850
SBF	0.097	0.595	0.052	0.629	0.118	0.485	0.100	0.376
Our	**0.084**	**0.778**	**0.029**	**0.961**	**0.074**	**0.870**	**0.046**	**0.897**

Table 1. Error statistics for still images from Figure 1.

REFERENCES

Bitterli, B., Rousselle, F., Moon, B., Iglesias-Guitián, J. A., Adler, D., Mitchell, K., Jarosz, W., and Novák, J. 2016. Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings. *Comp. Graph. Forum* 35, 4 (2016).

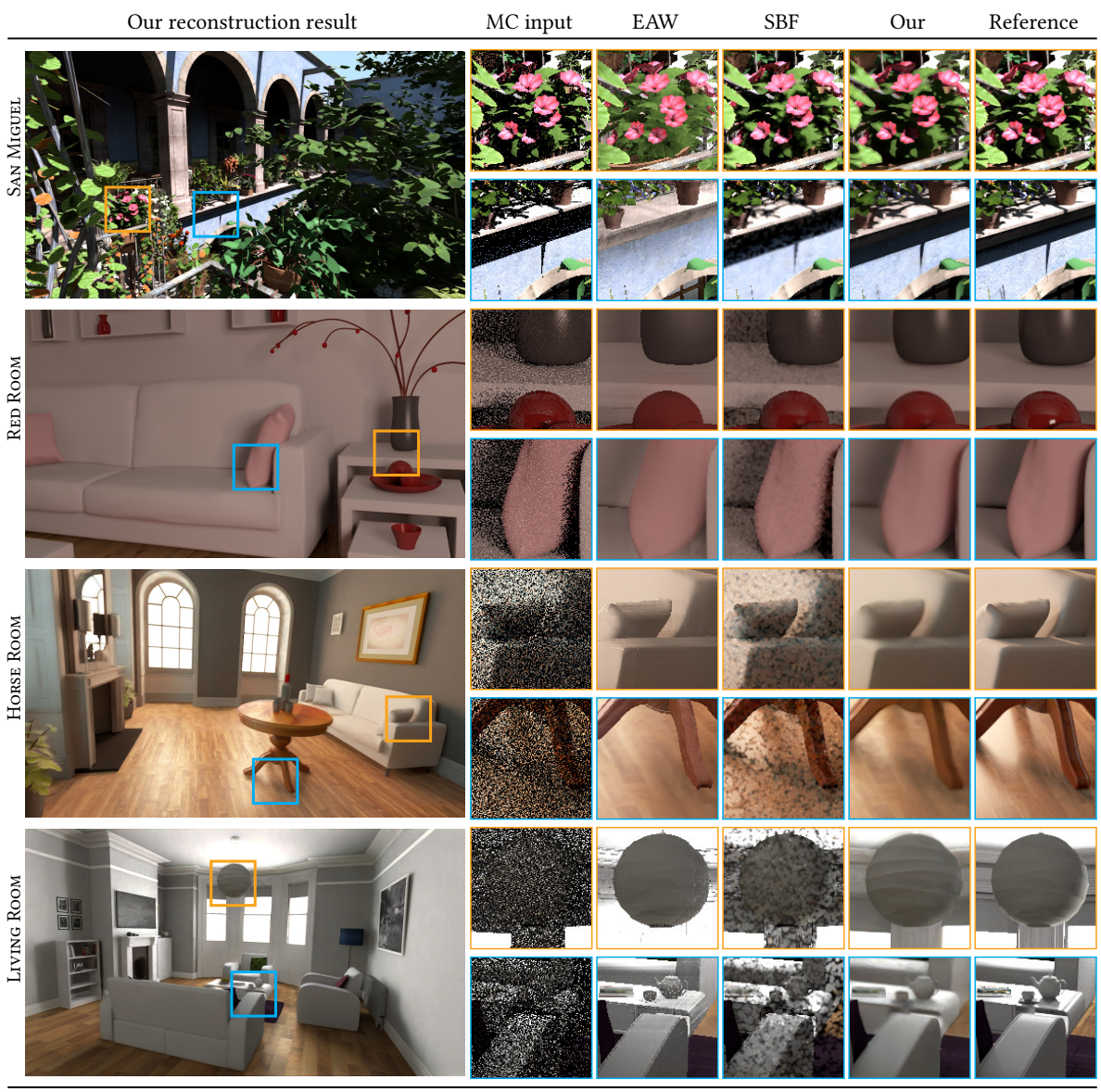

Fig. 1. Closeups of 1-bounce global illumination results for 1 spp input (MC), axis-aligned filter (AAF), À-Trous wavelet filter (EAW), SURE-based filter (SBF), and our result. Statistics are in Table 1.