Measuring the efficiency of the V-Ray denoiser

Internal report

Vladimir Koylazov

Chaos Software Ltd

Document version 1.0

Thanks to Emanuele Lecchi for optimizing and rendering the scenes

Author email: vlado@chaosgroup.com

Abstract

Since version 3.4, V-Ray includes a built-in denoising tool. This white paper aims at measuring the performance of the denoiser for reducing the error of Monte Carlo sampling.

Keywords: denoising, V-Ray, Monte Carlo, noise reduction

Introduction

Denoising Monte Carlo renders has recently gained traction as a method for reducing render times. While denoising has been available for a long time as part of various packages for processing images and video sequences, it has recently become popular in the context of Monte Carlo raytracing. A renderer can provide additional information that can help to improve the results of denoising algorithm. Many render engines in commercially available tools now include denoising tools (the ART renderer in 3ds Max, Maxwell renderer, Corona, V-Ray, RenderMan, Cycles), and standalone tools are also available.

Different approaches have been developed for reducing noise in Monte Carlo renders. The earliest such attempts date to [3]. Later algorithms focused on blending traditional denoising algorithms like non-local means with additional information that can be extracted from the renderer. Most recently, there have been attempts to use deep learning for this purpose as in [4] and [5].

The V-Ray denoiser implements a non-local means filter that uses additional (feature) buffers to preserve details. For a more detailed discussion of the implementation, see [1].

Method

We selected several different use cases that represent common scenes typically rendered by V-Ray users. Each scene was rendered V-Ray's bucket sampler with 1000 maximum AA subdivs (=1,000,000 samples per pixel maximum) and different noise thresholds. Each of the resulting renders was then denoised using the V-Ray denoising tool at default settings, which is the typical use case. We also rendered a reference result with 1000 max AA subdivs and a noise threshold of 0.00025.

Assessments and Measures

To assess the difference between the raw renders and the denoised results on the one hand, and the reference image on the other, we used relative mean square difference (rMSD) over all pixels in the images. The source code for the tool that calculates rMSD is provided in Appendix B.

It is debatable if rMSE is a good metric for measuring the performance of the denoiser, because it fails to take into account human perception in the sense that even if an image has a large rMSE error compared to the ideal result, it can still be an acceptable result in practice. Nevertheless, rMSE is easy to compute and gives at least some insight into the denoising performance.

Another debatable point is that the reference images themselves are computed with Monte Carlo integration and despite the high quality, they still contain some residual noise.

Results

The results are shown in the table in Appendix A. There are several observations to be made.

The first observation is the nearly linear relationship between the rMSE and the noise threshold in the V-Ray image sampler. This is expected, as the rMSE for Monte Carlo integration generally decreases linearly with the number of samples, and the V-Ray noise threshold is based on the variance of the samples in a pixel’s area (which is more or less constant) and the number of samples for the pixel. This means that as the noise threshold is reduced to zero, the image will also converge to the true result. From the simple global rMSE estimate it is not clear if the rMSE is uniformly reduced across the entire image and it will be interesting to explore this further. However we have reasons to believe that this is the case.

Next, the efficiency of the denoiser itself seems to vary considerably from scene to scene. For some scenes with large uniform areas, the denoiser performs well. For other scenes with more detail, the rMSE error is not reduced that much. In other scenes involving large portions of the surfaces visible through reflection and refraction, the denoiser might actually increase the rMSE of the result if specific measures are not taken (namely, allowing feature render elements to propagate through glass).

Depending on the scenes, the denoiser produces rMSE reduction between 1 and 4 times, usually around 1.5-2 times. In only two tests out of all 25, the denoiser produced more than 5 times rMSE reduction, and both of them involved very specific cases of low sampling settings and a scene with the same uniformly gray material.

Finally, the rMSE of the denoised images also shows a linear dependency on the noise threshold and converges to zero as the noise threshold is reduced. This is expected because the denoiser takes into account the estimated noise level from the image sampler and the denoising effect is reduced as the image is sampled more accurately. This should be an advantage over denoising algorithms that do not take sampling information into account as it allows the V-Ray denoiser to adjust automatically to the quality of the input result without user intervention.

Future directions

We only evaluated the performance of the denoiser for still images but not for animations. For animations, the V-Ray denoiser can look at several frames at once. It would be interesting to see how the additional information available from the multiple frames affects the rMSE.

List of used scenes

We used the following scenes for our tests:

Iron Spider - scene by Andrew Averkin;

White room - scene by Chaos Software;

ArchInteriors 38, scene 1 - from the respective Evermotion pack;

ArchInteriors 39, scene 3 - from the respective Evermotion pack; shaders were additionally optimized to reduce render times;

ArchExteriors 20, scene 5 - from the respective Evermotion pack; shaders were additionally optimized to reduce render times.

References

[1] Koylazov, Vladimir (2016). Denoising Monte Carlo renders, *CG^2 Talk Code Conference 2016*, https://www.youtube.com/watch?v=UrOtyaf4Zx8

[2] Rouselle, F. et al, Adaptive Sampling and Reconstruction using Greedy Error Minimization, *ACM Transactions on Graphics, Vol. 30(6)* https://cs.umd.edu/~zwicker/publications/AdaptiveSamplingGreedyError-SIGA11.pdf

[3] Jensen, H. W., Optimizing Path Tracing using Noise Reduction Filters, *in Proceedings of WSCG 1995, pp. 134-142, 1995*

[4] Bako, S. et al, Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings, *ACM Transactions on Graphics, Vol. 36, No. 4, Article 97*

[5] Chaitanya, C. R. A. et al, Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder, *ACM Transactions on Graphics, Vol. 36, No. 4, Article 98*