SUPPLEMENTARY DOCUMENT

This document provides further details and results for our paper "A Statistical Approach to Monte Carlo Denoising" (https://doi.org/10.1145/3680528.3687591). Moreover, we provide additional materials (e.g., source code) for our paper at https://www.cg.tuwien.ac.at/StatMC. In particular, we provide our result images in lossless PFM format at https://doi.org/10.48436/p3ert-wkm13.

A WELCH'S T-TEST AND OPTIMAL PAIR-WISE SYMMETRIC WEIGHTS

In Eq. (13) we define a binary membership function based on the optimal pair-wise symmetric weight, where two estimators are combined only if $s = 1 - w^*_{ij} > \gamma$. Here, we show that this test is equivalent to Welch's t-test, which is defined as

$$t = \frac{\left|\hat{\theta}_i - \hat{\theta}_j\right|}{\sqrt{\sigma_i^2/n_i + \sigma_j^2/n_j}} < \gamma_w. \tag{S.1}$$

Squaring and multiplying by the denominator, we find

$$(\sigma_i^2/n_i + \sigma_j^2/n_j)t^2 = (\hat{\theta}_i - \hat{\theta}_j)^2. \tag{S.2}$$

Substituting the ground-truth values with estimates (i.e., $\mathrm{Var}(\hat{\theta}_i) = \sigma_i^2/n_i$) in Eq. (12), and expanding our test, results in

$$s = 1 - \frac{2(\hat{\theta}_i - \hat{\theta}_j)^2 + \sigma_i^2/n_i + \sigma_j^2/n_j}{2\left((\hat{\theta}_i - \hat{\theta}_j)^2 + \sigma_i^2/n_i + \sigma_j^2/n_j\right)}. \tag{S.3}$$

Substitution of the left-hand side of Eq. (S.2) for $(\hat{\theta}_i - \hat{\theta}_j)^2$ yields

$$s = 1 - \frac{2(\sigma_i^2/n_i + \sigma_j^2/n_j)t^2 + (\sigma_i^2/n_i + \sigma_j^2/n_j)}{2(\sigma_i^2/n_i + \sigma_j^2/n_j)t^2 + 2(\sigma_i^2/n_i + \sigma_j^2/n_j)}. \tag{S.4}$$

Finally, simplifying the fraction on the right-hand side allows us to reformulate our test variable s as a function of Welch's t-statistic:

$$s = 1 - \frac{t^2 + 1/2}{t^2 + 1}, \tag{S.5}$$

or after rearranging terms, express t based on the optimal weight

$$t = \sqrt{\frac{1}{2s} - 1} = \sqrt{\frac{1}{2(1 - w^*_{ij})} - 1}. \tag{S.6}$$

Similarly, we can also relate the critical value of Welch's t-test to our threshold for the optimal weight, meaning that testing $t < \gamma_w$ is equivalent to testing $1 - w^*_{ij} > \gamma$ for

$$\gamma = \frac{1}{2(\gamma_w^2 + 1)}. \tag{S.7}$$

This result follows directly from substitution of the previous expression for t in $t < \gamma_w$, i.e. $\sqrt{1/(2s) - 1} < \gamma_w \rightarrow 1/(2s) < \gamma_w^2 + 1 \rightarrow 1/(2(\gamma_w^2 + 1)) < s$.

B SPEEDING-UP PAIR-WISE TESTS

In our implementation, we square and rearrange terms in Eq. (S.1) as follows to speed up the evaluation of our membership functions:

$$\begin{aligned} (\hat{\theta}_i - \hat{\theta}_j)^2 &< \gamma_w^2(\sigma_i^2/n_i + \sigma_j^2/n_j), \\ \hat{\theta}_i^2 + \hat{\theta}_j{}^2 - 2\hat{\theta}_i\hat{\theta}_j &< \gamma_w^2\sigma_i^2/n_i + \gamma_w^2\sigma_j^2/n_j, \\ \hat{\theta}_i^2 - \gamma_w^2\sigma_i^2/n_i &< 2\hat{\theta}_i\hat{\theta}_j - (\hat{\theta}_j^2 - \gamma_w^2\sigma_j^2/n_j). \end{aligned} \tag{S.8}$$

In this way, we avoid first fetching both estimator variances and then computing the square root of their sum in each membership-function evaluation. Instead we precompute the expression $\hat{\theta}_i^2 - \gamma_w^2\sigma_i^2/n_i$ for each estimator independently. For evaluating the membership function, we simply fetch these precomputed values as well as the estimator means and only calculate one additional product and one difference before performing the comparison. As there are many more pair-wise evaluations than individual estimators, this precomputation generally results in a significant speedup; in our case, approx. 2.5× for our default kernel radius of 20 pixels.

C APPROXIMATE-CONTRIBUTION RUSSIAN ROULETTE

C.1 Choosing the Number of Tracked Bounces

Our ACRR tracks and denoises average incoming radiance for different bounce indices, as described in §7.1. We have experimented with tracking different numbers of bounces. Fig. S1 shows that tracking more than about four bounces leads to diminishing returns: For the same number of samples per pixel, the overall error is lower later during the rendering; however, this decrease is outweighed by the additional cost of tracking and denoising the additional bounces, resulting in inferior efficiency overall. Tracking less than about four bounces results in lower computation times, which are, however, outweighed by the resulting increase in error. According to our experiments, four seems to be a good choice for the number of tracked bounces, as used for our results. We note that this choice pertains to our rendering system only, as the implementation of our method in other systems might lead to execution times that suggest a different choice. Further note that the termination probability for bounces that are not tracked is calculated using the average-radiance approximation of the last tracked bounce.

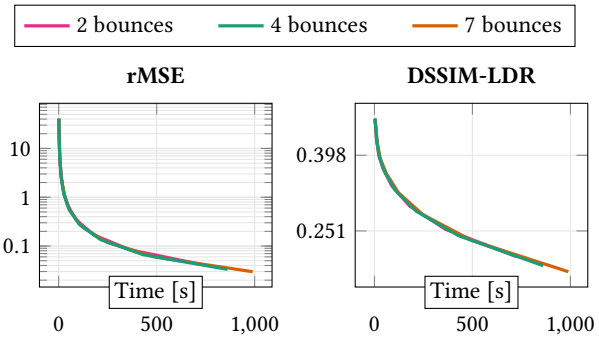


Fig. S1. Plots illustrating the trade-off between computation time and error when choosing the number of tracked bounces for our approximate-contribution Russian roulette (ACRR). We have found a sweet spot at around four tracked bounces. We used the Veach Ajar scene and a maximum number of 2048 SPP.

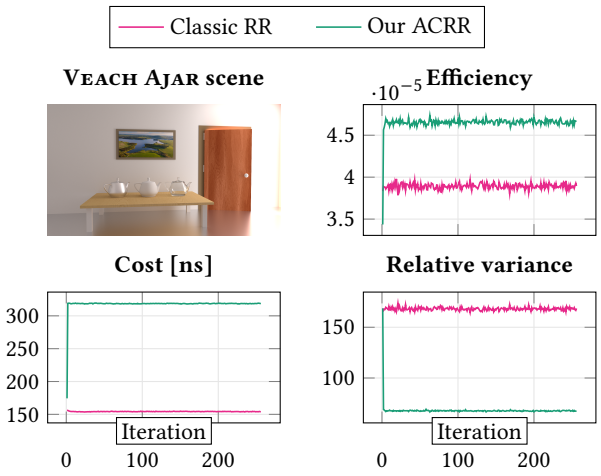


Fig. S2. Average efficiency, cost, and relative variance of classic Russian roulette (RR) versus our approximate-contribution variant for the Veach Ajar scene. For this scene, our method has 1.20× higher efficiency as cost increases by about 2.07× and relative variance is reduced by about 2.48×.

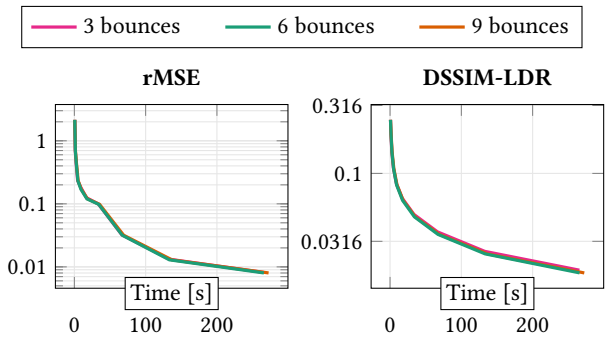
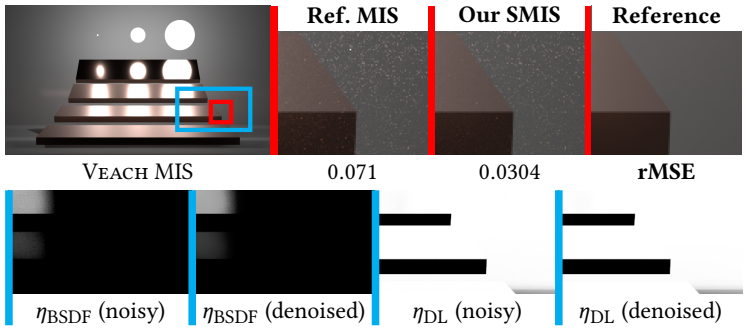


Fig. S3. Plots illustrating the trade-off between computation time and error when choosing the number of tracked bounces for our selective multiple importance sampling (SMIS). We have found a sweet spot at around six tracked bounces. We used the Veach MIS scene and a maximum number of 2048 SPP.



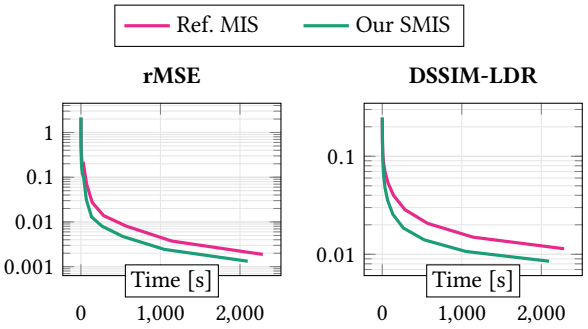


Fig. S4. Comparison of our SMIS to standard multiple importance sampling [Veach 1997] ("Ref. MIS") and visualization of first-bounce SMIS win rates for BSDF sampling (η_{BSDF}) and direct-light sampling (η_{DL}) before and after denoising. Images show Ref. MIS at 512 SPP (71.13 s) and our SMIS at 543 SPP (70.71 s).

C.2 Sampling Efficiency

We can obtain further insights into the performance of our method by analyzing sampling efficiency. In Fig. S2, we provide plots for average efficiency, cost, and relative variance. We use the same approach as Rath et al. [2022] to calculate the three metrics: for each iteration, we measure the average sampling time for one sample as cost, the average relative variance of one sample (analogous to relative mean squared error (rMSE) by Rousselle et al. [2011]), and the inverse product of the two as efficiency.

Interestingly, for the Veach Ajar scene, our method essentially reaches peak efficiency—an improvement of about 1.2× on average over classic RR—after the first iteration, indicating that rough *denoised* estimates are sufficient. For the plots, we have rendered 256 iterations with 4 SPP each for a total number of 1024 SPP. To reduce noise, we have averaged the metrics over 128 independent runs. Both RR techniques start terminating with the first bounce from the camera.

D SELECTIVE MULTIPLE IMPORTANCE SAMPLING

D.1 Choosing the Number of Tracked Bounces

Similarly to ACRR, our SMIS requires tracking win rates over the first few bounces starting from the camera, as described in §7.2. Here, the number of tracked bounces also determines the trade-off between additional computation time and increased error due to decreased sampling efficiency. We have empirically evaluated a sweet spot of about six bounces, as illustrated in Fig. S3. We also show an additional comparison on the classical test scene by Veach in Fig. S4.

E ADDITIONAL RESULTS

Here we present additional results and comparisons on some of Bitterli's test scenes; Figs. S5–S8 and Figs. S10–S11. Figure S9 shows a convergence comparison for these scenes. Note that the error spike in the White Room example results from an outlier (firefly), which then takes a long time to average out.

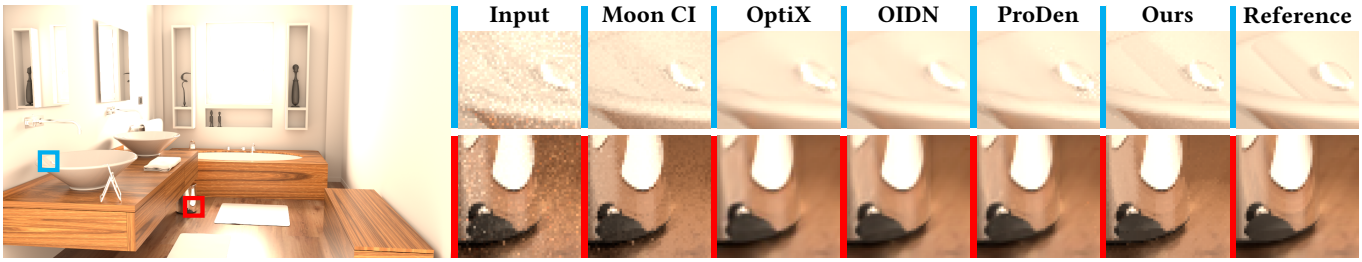


Fig. S5. Salle de Bain scene rendered with 512 SPP. Denoising time for Moon CI: 35.7 ms, OptiX: 73.0 ms, OIDN: 18.5 ms, ProDen: 1870.7 ms, ours: 27.9 ms.

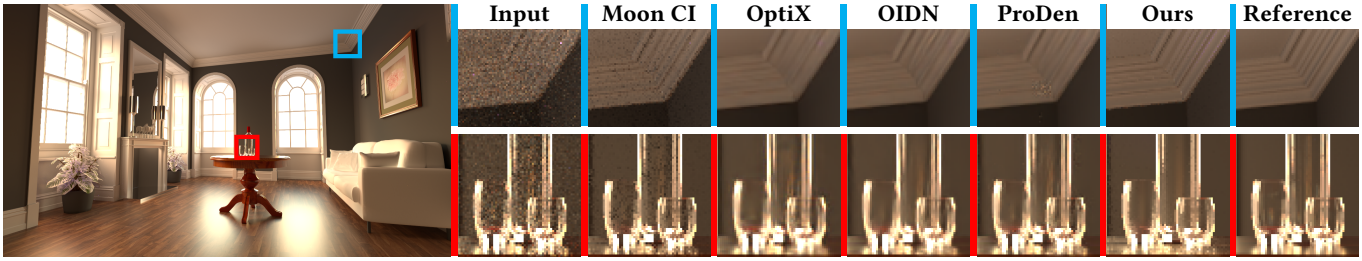


Fig. S6. Grey and White Room scene rendered with 512 SPP. Denoising time for Moon CI: 34.5 ms, OptiX: 76.0 ms, OIDN: 21.4 ms, ProDen: 1864.1 ms, ours: 27.5 ms.

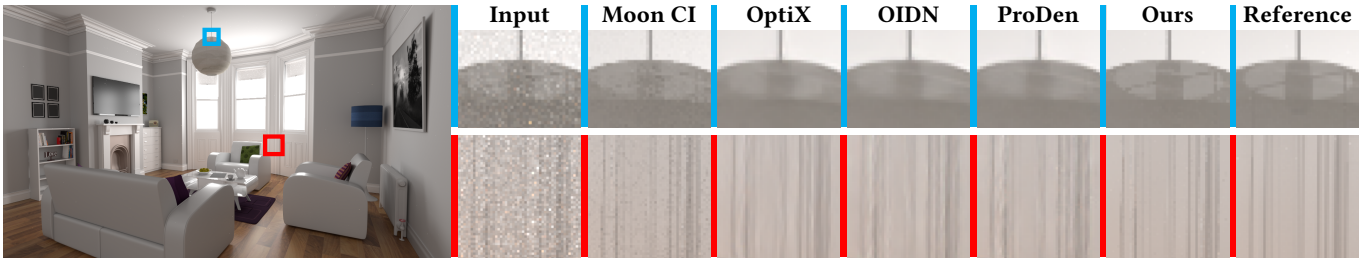


Fig. S7. White Room scene rendered with 512 SPP. Denoising time for Moon CI: 35.6 ms, OptiX: 81.4 ms, OIDN: 21.3 ms, ProDen: 1812.3 ms, ours: 28.0 ms.

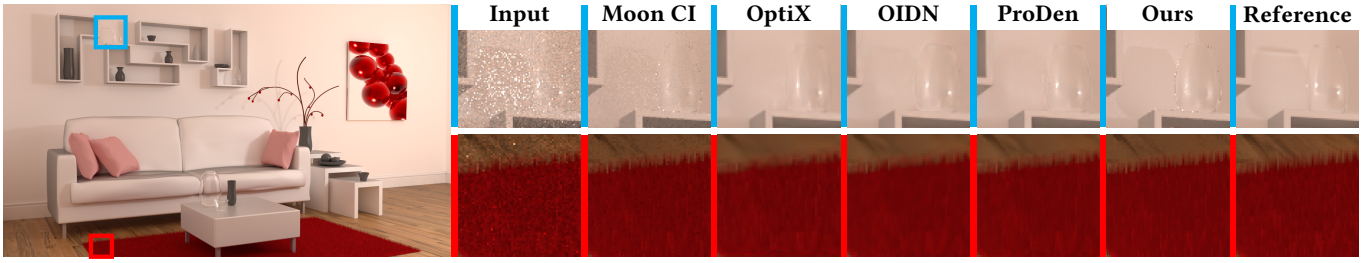


Fig. S8. Modern Living Room scene rendered with 256 SPP. Denoising time for Moon CI: 35.1 ms, OptiX: 78.2 ms, OIDN: 21.6 ms, ProDen: 1849.5 ms, ours: 27.6 ms.

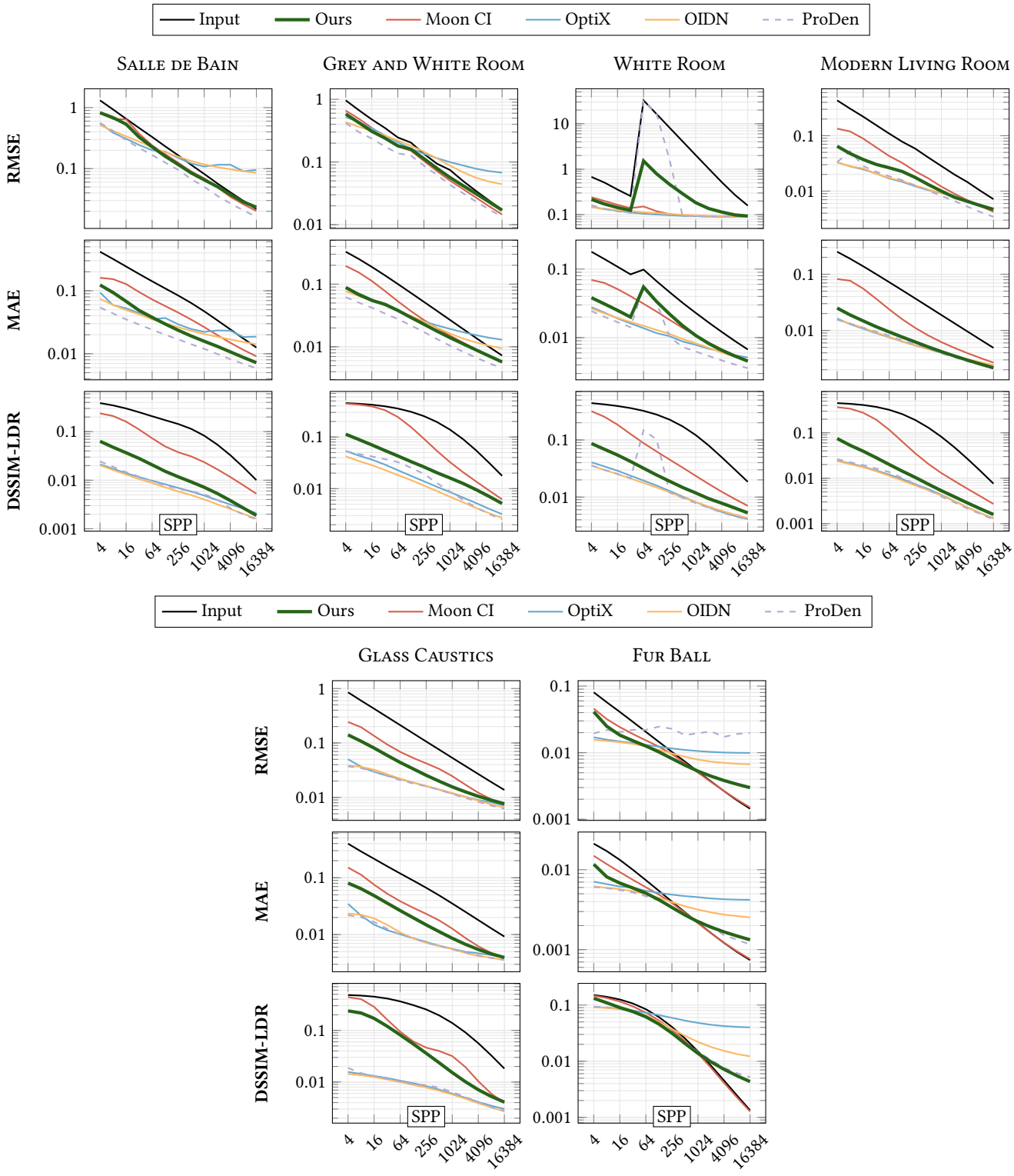


Fig. S9. Convergence plots for our denoising method, compared to the approach by Moon et al. [2014] ("Moon CI"), NVIDIA OptiX AI-Accelerated Denoiser ("OptiX"), Intel Open Image Denoise (OIDN), and progressive denoising [Firmino et al. 2022] ("ProDen"). Error metrics top-to-bottom: root mean square error (RMSE), mean absolute error (MAE), and structural dissimilarity (DSSIM) (evaluated after tone mapping to low-dynamic-range (LDR) images, because DSSIM requires a bound value range). Scenes left-to-right (top): Figs. S5–S8. Scenes left-to-right (bottom): Figs. S10 and S11.

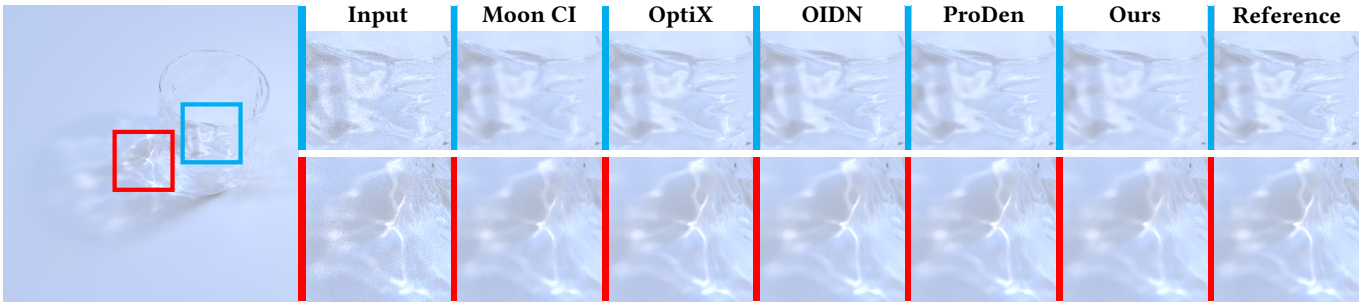


Fig. S10. Glass Caustics scene rendered with 4096 SPP. Denoising time for Moon CI: 13.4 ms, OptiX: 92.3 ms, OIDN: 21.7 ms, ProDen: 2029.5 ms, ours: 12.6 ms. For this scene, we have reduced the radius for our filter to 6 pixels (from the default 20 pixels) and increased the significance level α to 0.05 (from the default 0.005), both to mitigate over-blurring. Note that for a fair comparison, we have also reduced the filter radius for Moon CI to 6 pixels.

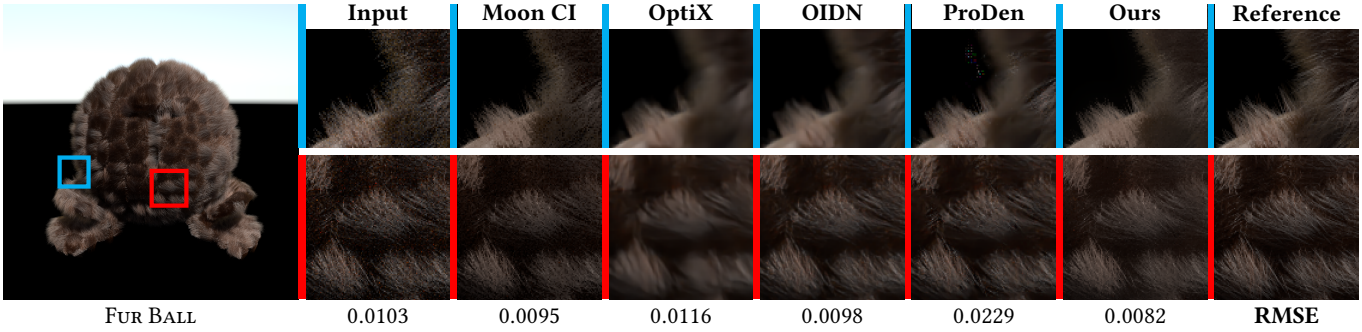


Fig. S11. Fur Ball scene rendered with 256 SPP. Denoising time for Moon CI: 42.5 ms, OptiX: 101.1 ms, OIDN: 21.6 ms, ProDen: 2251.7 ms, ours: 28.7 ms.