Tiled Reservoir Sampling for Many-Light Rendering

Yusuke Tokuyoshi
yusuke.tokuyoshi@amd.com
Advanced Micro Devices, Inc.

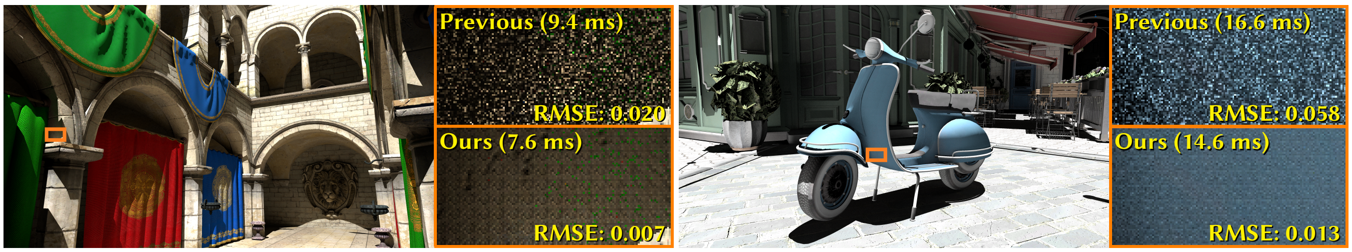


Figure 1: Dynamic global illumination using reservoir sampling and temporal reuse for 1M VPLs. Our tiled reservoir sampling for initial candidates reduces the variance caused by VPLs close to surfaces. The computation time is the total of the reservoir sampling pass and the reuse pass (3840×2160 pixels, AMD Radeon™ RX 6900 XT GPU).

ABSTRACT

While *reservoir-based spatiotemporal importance resampling* (ReSTIR) is a powerful technique for real-time many-light rendering, initial candidate samples are often generated ignoring the distance between the shading point and the light source when they are close. For dynamic lights close to surfaces such as *virtual point lights* (VPLs), it is inefficient even reusing samples spatially and temporally by ReSTIR. To take into account the distance from the light sources for sampling initial candidates, we introduce a *tiled reservoir sampling* technique that combines tile-based *stochastic light culling* and reservoir sampling. This initial candidate sampling is unbiased and performs efficiently on the GPU.

1 INTRODUCTION

While *virtual point lights* (VPLs) [Keller 1997] have often been used to approximate indirect illumination, improving the quality of dynamic scenes is still challenging. The visibility of VPLs can be tested by tracing shadow rays on the GPU, just as this is done for direct illumination. However, to achieve real-time frame rates, the number of rays must be limited to a small number, such as one ray per pixel. Therefore, we need an efficient method to sample an important VPL for each pixel from many (e.g., millions of) dynamic VPLs.

For real-time direct illumination, efficient light sampling methods have been developed such as *reservoir-based spatiotemporal importance resampling* (ReSTIR) [Bitterli et al. 2020]. ReSTIR resamples a light from candidate samples that have been reused spatially and temporally based on *weighted reservoir sampling* [Chao 1982]. Although this spatiotemporal reuse increases the number of candidate samples from tens to thousands [Wyman and Panteleev 2021], initial candidates are often generated ignoring the distance between the shading point and the light source when they are close. Since the contribution from each light is inversely proportional to the distance squared, initial candidate sampling that ignores the distance is inefficient, especially for lights close to surfaces. For dynamic VPLs, this problem becomes more significant (see Fig. 1), because the VPLs are on surfaces and have spatially varying normals due to normal maps.

In this report, we introduce a *tiled reservoir sampling* method that uses tile-based *stochastic light culling* [Tokuyoshi and Harada 2017] in initial candidate generation for reservoir sampling. Stochastic light culling is an unbiased culling technique based on *Russian roulette* [Arvo and Kirk 1990], and it efficiently rejects distant lights by using random light ranges in an existing tiled culling framework [Stewart 2015]. By using stochastic light culling, we sample candidates considering the distance from each light. This approach reduces the variance for lights close to surfaces such as VPLs. This report shows improvements in VPL-based dynamic indirect illumination as a challenging case, but our method can also be applied to both direct and indirect illumination.

2 TILED RESERVOIR SAMPLING

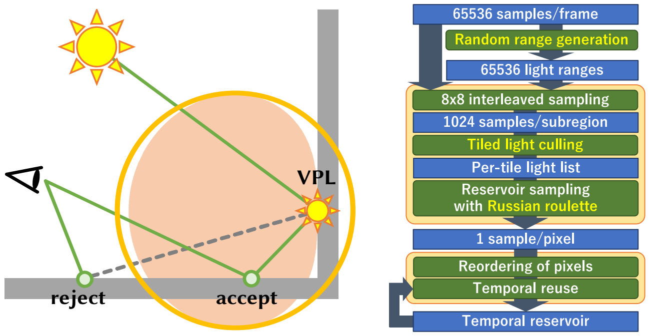
2.1 Stochastic Light Culling for Candidates

While a previous ReSTIR implementation [Wyman and Panteleev 2021] uniformly sampled 32 candidates per pixel from a subset of lights for each tile, we now generate candidates by rejecting unimportant lights from the light subset. This rejection is performed using Russian roulette accelerated with tile-based stochastic light culling. For our Russian roulette, the probability of acceptance $P(x) \in [0, 1]$ is proportional to the radiant intensity $I \in [0, \infty)$ divided by the distance squared $l^2 \in [0, \infty)$ for each light vertex (e.g., VPL) x as follows:

$$P(x) = \min\left(\frac{I}{\delta l^2}, 1\right),$$

where $\delta \in (0, \infty)$ is a user-specified parameter to control the variance (for parameter setting, please refer to the original work [Tokuyoshi and Harada 2016]). In this report, we use $\delta = 0.002$ for our experiments. By generating a single random number for each light, the acceptance range is mapped to a distance from the light (see Fig. 2a).

email: yusuke.tokuyoshi@amd.com
Advanced Micro Devices, Inc. Technical Report, No. 21-11-ecdc, November 2021.



(a) Stochastic light culling **(b) Our sampling pipeline**

Figure 2: Stochastic light culling (a) restricts the range of influence of each light (e.g., VPL) based on Russian roulette. We cull the initial candidate lights using the bounding spheres of these random light ranges. In our sampling pipeline (b), the processes for this culling are highlighted in yellow. Each orange box is implemented with a single-pass compute shader.

Therefore, by using this random range for each light, we cull lights (which would be rejected by Russian roulette) before performing Russian roulette.

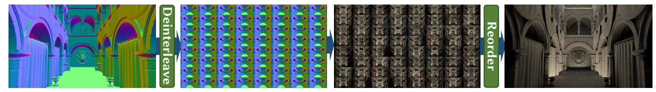
2.2 Tiled Culling Implementation

Our culling implementation (see Fig. 2b) is a combination of compute-based tiled culling [Stewart 2015] and interleaved sampling [Segovia et al. 2006]. This tiled culling stores indices of accepted lights (with false positives) in a light list allocated in a local data share for each tile. To save the data size, we use a 16-bit integer type for this light list. Thus, our culling implementation limits the number of lights to 65536. To handle more than 65536 lights, we first split lights into groups of 65536 lights, and then randomly select one group for every frame in a round-robin manner. For the selected 65536 lights, random light ranges and their bounding spheres are generated before the culling process. Then, during the culling process, these 65536 lights are further split into 1024 lights per pixel by 8×8 interleaved sampling. This interleaved sampling avoids a positive correlation between adjacent pixels. To perform tiled culling with interleaved sampling, pixels are deinterleaved into 8×8 subregions (see Fig. 3b) so that pixels have the same light subset (i.e., 1024 lights) within each subregion. Finally, the light subsets are culled for each tile using their bounding spheres. This implementation assumes diffuse VPLs for simplicity, but we can also support glossy VPLs by extending the bounding spheres to ellipsoids. For details on constructing bounding spheres and bounding ellipsoids for VPLs, please refer to Tokuyoshi and Harada [2017].

2.3 Reservoir Sampling with Russian Roulette

Once the culling process is completed, we perform Russian roulette and reservoir sampling for the light list. The integration of lighting using reservoir sampling with one reservoir is written as follows:

$$\int_{\Omega} f(x)\mathrm{d}\mu(x) \approx \frac{f(y)}{\hat{p}(y)} w,$$



(a) G-buffer **(b) 8×8 subregions** **(c) Tiled reservoir sampling** **(d) Reservoirs**

Figure 3: Interleaved sampling via deinterleaving. Our tiled reservoir sampling is executed for deinterleaved pixels to exploit the pixel coherence.

where $f(x) \in [0, \infty)$ is the contribution from a light vertex x, y is the light vertex resampled from candidates according to their weights, $\hat{p}(y) \in [0, \infty)$ is the target PDF (proportional to the light contribution without the visibility), and $w \in [0, \infty)$ is the accumulated candidate weight that is the normalization factor for the PDF. For our initial candidates, this accumulated weight is given by

$$w = \frac{1}{M} \sum_{i \in L} \underbrace{\frac{\hat{p}(x_i)}{p(x_i)} \frac{H\left(P(x_i) - \xi_i\right)}{P(x_i)}}_{\text{weight for each candidate}},$$

where $M \in \mathbb{N}$ is the candidate count, i is the light index in the light list L, $p(x_i) \in [0, \infty)$ is the source PDF, $\xi_i \in [0, 1)$ is the per-light uniform random number, and $H\left(P(x_i) - \xi_i\right)$ is the Heaviside function: 1 if $\xi_i < P(x_i)$, otherwise 0. Although our initial candidate count is $M = 1024$, the computational complexity of this sampling is $O(|L|)$. Also, light data access is coherent in a tile. Thus, our sampling is inexpensive while taking into account the light intensity and the distance using Russian roulette.

2.4 Reordering of Reservoir Pixels

Since our tiled reservoir sampling is executed for deinterleaved pixels (see Fig. 3c), the resulting reservoirs should be reordered to their respective original pixels (see Fig. 3d). Therefore, we store the reservoirs in a buffer at the end of the reservoir sampling pass, and then gather them at the beginning of the reuse pass. This reuse pass updates y and w in each reservoir as in the existing ReSTIR.

3 RESULTS

Here we show the results rendered using a combination of our tiled reservoir sampling and temporal reuse for 1M single-bounce VPLs on an AMD Radeon™ RX 6900 XT GPU. The tile size is 16×16 pixels for our tiled reservoir sampling. The image quality is evaluated with the root-mean-squared error (RMSE) metric. For temporal reuse in this experiment, we reuse the shadow ray visibility both for temporal resampling and shading [Wyman and Panteleev 2021] to perform one ray per pixel. We omit spatial reuse (which requires additional rays) for simplicity. Although temporal reuse can produce correlation artifacts during motion, we reduce this correlation by using stochastic rounding for temporal reprojection (see Appendix A for details).

Fig. 4 shows the comparison between the previous uniform candidate sampling and our tiled reservoir sampling. The previous method samples 32 candidates per pixel from 1024 lights per tile (i.e., $M = 32$ without Russian roulette), while our method culls distant lights from 1024 lights per tile (i.e., $M = 1024$ with Russian roulette). For the previous method, we set the tile size to 8×8

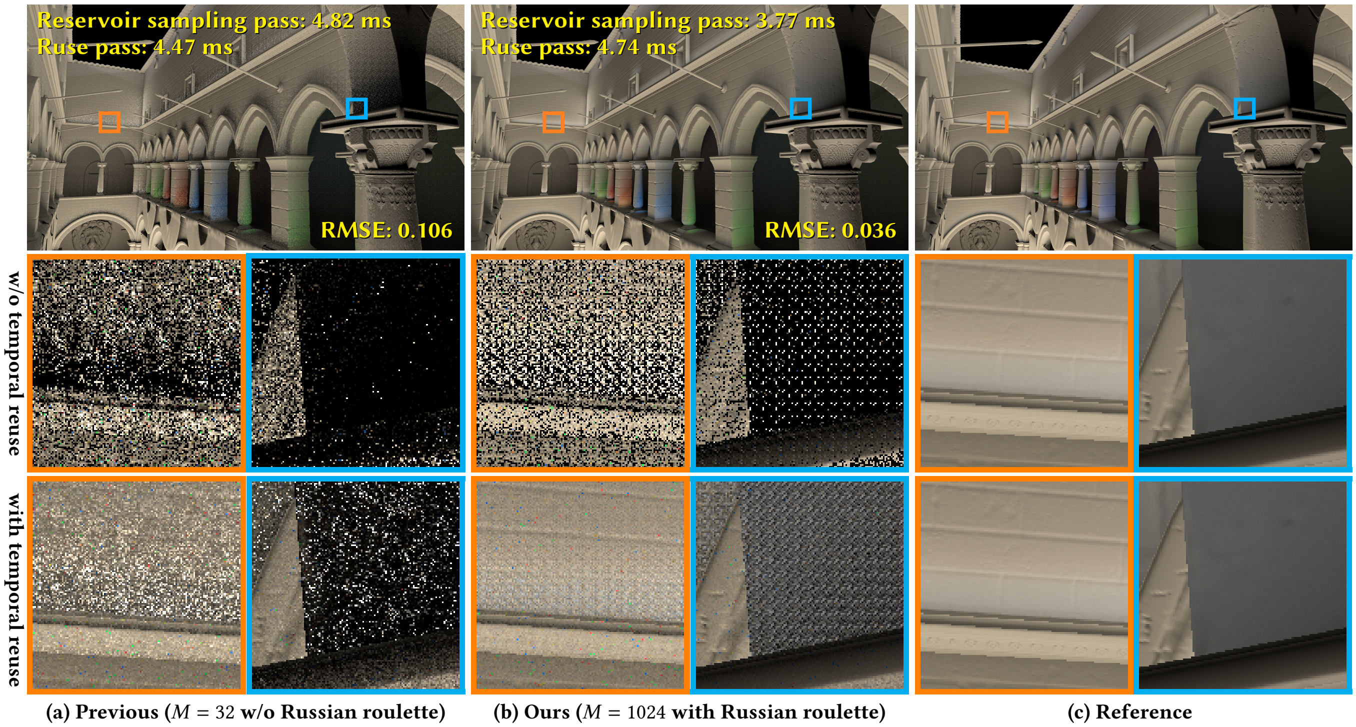


Figure 4: Indirect illumination (3840×2160 pixels) using reservoir sampling. (a) The use of 32 uniformly sampled initial candidates produces a significant variance for VPLs. (b) Our method efficiently reduces the variance by stochastically culling distant VPLs from 1024 candidates. The computation time for the reuse pass includes shadow ray tracing for visibility reuse.

pixels to reduce a positive correlation between adjacent pixels, as recommended in Wyman and Panteleev [2021]. For this scene, our sampling produces a smaller error with less computation time than the previous method. Although our method uses a larger tile size (i.e., 16×16 pixels) than the previous method and performs coherent sampling in the tile, a positive correlation between adjacent pixels is avoided thanks to interleaved sampling. The correlated variance due to our coherent sampling is visible as an interleaved pattern, and this variance is reduced by combining the reservoir sampling with temporal reuse. Although the interleaved pattern remains slightly, it can be further reduced by a postprocess denoiser.

4 FUTURE WORK

In this report, we have presented a technique to sample a light from a light subset (i.e., 1024 lights for each tile), considering the distance from each light. However, the light subset is still generated without taking this distance into account. For future work, we would like to investigate importance sampling methods to generate the light subset, such as grid-based reservoirs [Boksansky et al. 2021].

REFERENCES

James Arvo and David Kirk. 1990. Particle Transport and Image Synthesis. *SIGGRAPH Comput. Graph.* 24, 4 (1990), 63–66. https://doi.org/10.1145/97879.97886

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting. *ACM Trans. Graph.* 39, 4, Article 148 (2020), 17 pages. https://doi.org/10.1145/3386569.3392481

Jakub Boksansky, Paula Jukarainen, and Chris Wyman. 2021. *Rendering Many Lights with Grid-Based Reservoirs.* Apress, 351–365. https://doi.org/10.1007/978-1-4842-7185-8_23

Min-Te Chao. 1982. A General Purpose Unequal Probability Sampling Plan. *Biometrika* 69, 3 (1982), 653–656. https://doi.org/10.1093/biomet/69.3.653

Alexander Keller. 1997. Instant Radiosity. In *SIGGRAPH '97*. 49–56. https://doi.org/10.1145/258734.258769

Benjamin Segovia, Jean-Claude Iehl, Richard Mitanchey, and Bernard Péroche. 2006. Non-Interleaved Deferred Shading of Interleaved Sample Patterns. In *GH '06*. 53–60.

Jason Stewart. 2015. Compute-Based Tiled Culling. In *GPU Pro 6: Advanced Rendering Techniques*. A K Peters/CRC Press, 435–458.

Yusuke Tokuyoshi and Takahiro Harada. 2016. Stochastic Light Culling. *JCGT* 5, 1 (2016), 35–60.

Yusuke Tokuyoshi and Takahiro Harada. 2017. Stochastic Light Culling for VPLs on GGX Microsurfaces. *Comput. Graph. Forum* 36, 4 (2017), 55–63. https://doi.org/10.1111/cgf.13224

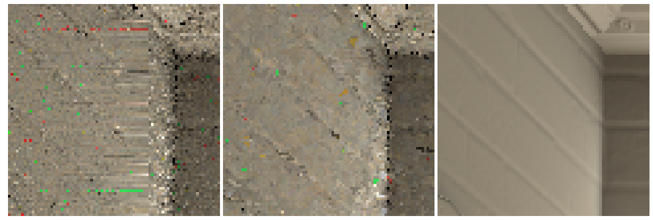
Chris Wyman and Alexey Panteleev. 2021. Rearchitecting Spatiotemporal Resampling for Production. In *HPG '21*.

A STOCHASTIC ROUNDING FOR TEMPORAL REPROJECTION

In the process of temporal reuse, reservoirs in the previous frame are looked up by using temporal reprojection. This reprojection should be performed without texture filtering, because reservoirs have discrete samples. However, the use of the nearest neighboring reservoir given by rounding off a reprojected pixel position produces correlation artifacts between adjacent pixels during motion (see Fig. 5). Although combining with spatial reuse can reduce these artifacts [Wyman and Panteleev 2021], we reduce the correlation by using stochastic rounding for reprojection. Given $\mathbf{s} \in \mathbb{Z}^2$ as the current pixel position, we randomly reproject it to the previous frame as follows:

$$\mathbf{s}' = \lfloor \mathbf{s} + \mathbf{v} + \boldsymbol{\xi} \rfloor,$$

where $\mathbf{v} \in \mathbb{R}^2$ is the motion vector, and $\boldsymbol{\xi} \in [0, 1)^2$ is a 2D uniform random number.



(a) Deterministic Rounding **(b) Stochastic Rounding** **(c) Reference**

Figure 5: (a) Temporal reprojection with deterministic rounding produces correlation artifacts that swim during motion. (b) We reduce the artifacts by using stochastic rounding.