Implementing ReSTIR DI in La Jolla

Xinyuan Liang

March 26, 2023

1 Introduction

This final project implements the Reservoir-based Spatiotemporal Importance Resampling (ReSTIR) in La Jolla to render multiple-light scenes. The ReSTIR is known for achieving real-time rendering in interactive lighting scenes without extra complex data structures or high-sample costs. Generally speaking, it combines Resampled Importance Sampling (RIS), Weighted Reservoir Sampling (WRS), and Spatio-temporal Reuse to provide those outstanding features: 1) approximate perfect importance sampling proportionally to a target PDF, which no practical sampling algorithm may exist (e.g. $\rho \cdot L_e \cdot G$); 2) using a constant length "reservoir" to store accepted samples; 3) reusing information from nearby samples both spatially and temporally.

This final project, however, only focuses on exploring direct illumination and spatial reuse (omitting the temporal reuse). The next section describes the mathematical background and preliminaries of ReSTIR. Section 3 dives into the experiments and implementation details. The last part presents the result.

2 Background

2.1 Resampled Importance Sampling

The reflected radiance due to direct light is calculated as $f(x) = L_e(x)\rho(x)G(x)V(x)$, where $L_e(x)$ is the emitted radiance, $\rho(x)$ is the BSDF term, $V(x)$ is the visibility between x and light, $G(x)$ is the geometry term. It's very hard to important sample proportional to the full product terms $(L_e \cdot \rho \cdot G)$, and also increasing sample overhead is unwanted. RIS is an alternative to only sample proportional to the product of the sub-optional PDF (e.g. L_e). RIS can sample arbitrary target distribution $\hat{p}(x)$ even if it's unnormalized.

To approximate the unnormalized target function $\hat{p}(x)$, it firstly generates M candidates samples proportional to $p(x)$, which is the source pdf, and randomly chooses one (index with z) from candidates with the below discrete probabilities driven by target $\hat{p}(x)$. It's unbiased if both $p(x)$ and $\hat{p}$ are positive (will discuss it later).

$$p(z|x) = \frac{w(x_z)}{\sum_{i=1}^{M} w(x_i)}$$

with

$$w(x) = \frac{\hat{p}(x)}{p(x)}$$

Therefore, the integral turns out to be

$$\int_{\Omega} f(x)dx = f(y)W$$

with

$$W = \frac{1}{\hat{p}(x)}(\frac{1}{M}\sum_{j=1}^{M} w(x_j))$$

A sample y is selected and used by the RIS estimator as if it were drawn from $\hat{p}$. The intuition of the W is to correct the fact that the true distribution of y only approximate $\hat{p}$. In other words, W can approximate the inverse PDF in standard Monte-Carlo integration.

The intuition behind the RIS is actually another Monte Carlo integration. To sample according to the unnormalized target function, RIS approxiamte the normalization:

$$p(x) = \frac{\hat{p}(x)}{\int_{\Omega} \hat{p}(x)dx}$$

and,

$$p(x) \approx \frac{\hat{p}(x)}{\frac{1}{M}\sum_j \frac{\hat{p}(x_j)}{p(x_j)}}$$

The RIS is valid and unbiased as long as it satisfies:

$$\mathbb{E}[W(x,z)] = \frac{1}{p(x_z)} = \frac{\int_{\Omega} \hat{p}(x)dx}{\hat{p}(x_z)}$$

which is just Monte Carlo integration (the $p(x_z)$ here is the normalized PDF). To summary, the RIS estimator is:

$$F = \int_{\Omega} f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}[\frac{f(x_i)}{\hat{p}(x_i)}\frac{1}{M}\sum_{j=1}^{M}\frac{\hat{p}(x_{ij})}{p(x_{ij})}]$$

2.2 Weighted Reservoir Sampling

When doing the RIS, it's not memory-efficient to store all the candidates. Therefore, WRS is needed for streaming sampling. WRS is an algorithm to sample N random elements from an unknown length of stream x1, x2, ... Each elements x_i is associated with weight $w(x_i)$ and should be selected accordingly:

$$p_i = \frac{w(x_i)}{\sum_{j=1}^{M} w(x_j)}$$

The reservoir is a fix-length array with the size of N. To keep the above invariant, the elements in the reservoir are replaced with the next sample x_{m+1} with the probability

$$\frac{w(x_{m+1})}{\sum_{j=1}^{m+1} w(x_j)}$$

2.3 Spatiotemporal Reuse

The reservoir sampling also makes it possible to reuse information from nearby samples spatially and temporally. Spatio-temporal reuse is basically feeding the result of RIS into another round of RIS. It can improve efficiency by increasing the number of candidates dramatically from the neighbor pixels.

2.3.1 Spatial Reuse

After first generating M candidates for every pixel q using RIS, each pixel selects K neighbors and combines their reservoirs. As mentioned above, the output of spatial reuse can be passed as input for the next iteration of spatial reuse. For example, taking two reservoirs into consideration, the new reservoir (combining these two reservoirs) updates its w_sum and chooses its candidate according to their weights $\hat{p_q}(r_1.y)(r_1.W)(r_1.M)$ and $\hat{p_q}(r_2.y)(r_2.W)(r_2.M)$, where $\hat{p_q}$ is the target pdf of the current pixel. The reason why the neighbor's reservoirs are assigned to the weights $\hat{p_q}(r.y)(r.W)(r.M)$ is that $\hat{p_q}(r.y)(r.W)$ approximates the standard RIS weight $\hat{p_q}(r.y)/p(r.y)$. The weight is additionally scaled by the number of candidates $r.M$ that produce $r.y$.

The mathematical insight behind the reuse is that the RIS estimator can be plugged into the RIS estimator:

$$p(x) \approx \frac{\hat{p_0}(x)}{\frac{1}{M_0}\sum_j \frac{\hat{p_0}(x_j)}{p_0(x_j)}} \approx \frac{\hat{p_0}(x)}{\frac{1}{M_0}\sum_j [\frac{\hat{p_0}(x_j)}{\hat{p_1}(x_j)} \frac{1}{M_1}\sum_k \frac{\hat{p_1}(x_{jk})}{q(x_{jk})}]}$$

2.3.2 Visibility Reuse

So far, visibility is not considered in the productions. The target $\hat{p}$ samples unshadowed path contribution well. But when taking visibility into account, it needs to evaluate the visibility of the selected sample y for each pixel's reservoir and discard any occluded samples. Without considering visibility in reuse, the noise will dominate as M grows. Notice that when visibility reuse is considered, it needs to shoot another shadow ray per pixel.

2.4 Bias

Analyzing the eliminating bias caused by reusing RIS is the most exciting and challenging part of ReSTIR.

Previously, it's proved that the expectation of W is converging to the desired $1/p(x_z)$. But this is only true when the domains of the samples are identical. When reusing, the RIS combines the candidates from neighbor pixels whose integration domain and target distribution are varied, which introduces bias (in other words, the new source pdf in the denominator for the combined reservoirs comes from different pixel's target pdf).

2.4.1 Analyze Bias

The standard RIS (Talbot et al. 2005) assumes that all candidate samples are produced according to the same source pdf. Since the samples reused from the neighbor pixel are resampled following a different target distribution, Bitterli et al.[1] allows each sample x_i coming from a potentially different source pdf $p_i(x_i)$. and proves the expected RIS weight W simplifies to:

$$\mathbb{E}[W(x,z)] = \frac{1}{p(x_z)} = \frac{|Z(y)|}{M}$$

where |Z(y)| is the number of candidates which are "effective". In other words, Z(y) is a set:

$$Z(y) = \{i|1 < i <= M \wedge p_i(y) > 0\}$$

Therefore, if both the target pdf and the source pdf are non-zero, which means all pdfs can generate sample y, then $|Z| = M$, and the W becomes the unbiased estimator of inverse RIS pdf. Otherwise, $|Z| < M$, and the results will look darker because the expected value of W is biased to be smaller. Now the unbiased RIS estimator is:

$$F = \int_\Omega f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}[\frac{f(x_i)}{\hat{p}(x_i)}\frac{1}{|Z(x_i)|}\sum_{j=1}^{M}\frac{\hat{p}(x_{ij})}{p(x_{ij})}]$$

Intuition of Bias The above paragraph explains bias from a mathematics aspect, but it's beneficial to understanding the intuition of bias source to boost up the debiasing procedure. In short, bias comes from using the sampled lights which are non-zero at the current pixel but are zero at neighbors. In Chris Wyman et al.[2] paper, an intuitive figure 1 illustrates these situations clearly: Reusing samples between pixels A and B causes bias if the target functions have zeros in different locations. The first situation is illustrated in the second figure: the sampled light a is invisible to B, which means the $\hat{p}(x)$ from B is zero. The second situation is illustrated in the third figure: A and B are on different surfaces having two different normals and hemispheres. Some samples generated by A (e.g. the light a) would never be seen by B, verse the same. In other words, the target pdf $\hat{p}(x)$ from B doesn't cover the hemisphere at A or the source pdf $p(x)$ from B doesn't cover the hemisphere at A. More examples are easy to enumerate, e.g. B's BRDF has zero-value reflectance at A. Therefore, when reusing the neighbor's samples, an appropriate M is needed. For the right-most example illustrated above, if sample a is selected, the M should be M_A and if sample c is selected, the M should be $M_A + M_B$. In the biased version ignoring the $M/|Z(x)|$, the $\frac{1}{M_i}$ turns out to be too large and darkens the results.

2.4.2 Eliminate Bias

Now comes the most exciting debiasing part. Papers from Bitterli et al.[1] and Wyman et al.[2] provide several methods for eliminating the bias. From the section, the RIS estimator replace $\frac{1}{M_i}$ with m_i:

$$F = \int_\Omega f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}[\frac{f(x_i)}{\hat{p}(x_i)}m_i\sum_{j=1}^{M}\frac{\hat{p}(x_{ij})}{p(x_{ij})}]$$

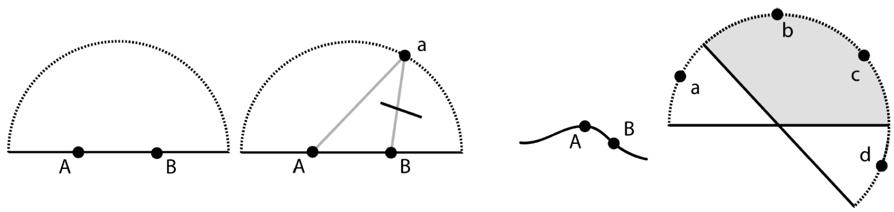


Figure 1: Bias

The RIS weight also is changed to be:

$$W(x,z) = \frac{1}{\hat{p}(x_z)}[m(x_z)\sum_{i=1}^{M} w_i(x_i)]$$

And if the $\sum_{i \in Z(y)} m(x_i) = 1$ then the estimator is unbiased:

$$\mathbb{E}[W(x,z)] = \frac{1}{p(y)} \sum_{i \in Z(y)} m(x_i) = \frac{1}{p(y)}$$

Naive method The most straightforward is simply set $m(x_z) = 1/|Z(x_z)|$, that is, instead of dividing by M(the number of candidates), dividing the number of candidates with non-zero target pdf at that location. The sum of $m(x_z)$ over samples in $Z(y)$ is always 1.

MIS method The naive approach will cause noises when the candidate pdfs are close to zero, but not exactly zero. Combining samples with multiple-importance sampling (MIS) is a better choice to mitigate the problem. The balance heuristic MIS is used in both papers:

$$m(x_z) = \frac{p_z(x_z)}{\sum_{i \in Z(y)} p_i(x_z)}$$

For clarity of notation, z is the selected sample in the set $Z(y)$, $p_z(x_z)$ is the corresponding source pdf of that sample. Note that the source pdf of samples are different now.

Combined with Heuristics The above methods reduce bias after computing the |Z(y)|, which means for every reused sample, a shadow ray needed to be generated and recalculate the PDF at the current pixel. To reduce the cost of debiasing, choosing effective samples from neighbors is important. The simple heuristics is that choosing samples from neighbors which have similar depth(e.g. $|z_i - z_j|/z_j < \epsilon$) and similar directions (e.g. $|\vec{n_i} \cdot \vec{n_j}| > \tau$). The papers choose to use 0.1 as ϵ and $cos(25°)$ as τ.

With heuristics, the Z(y) is changed to be:

$$Z(y) = \{i | 1 < i <= M \wedge p_i(y) > 0 \wedge H(i) = 1\}$$

where $H(i) \in \{0, 1\}$ based on the heruistic result.

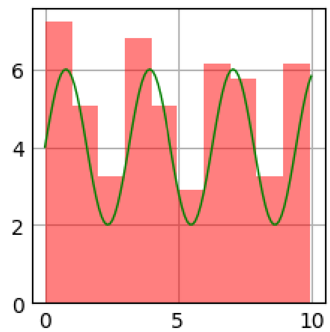


Figure 2: expt. 1

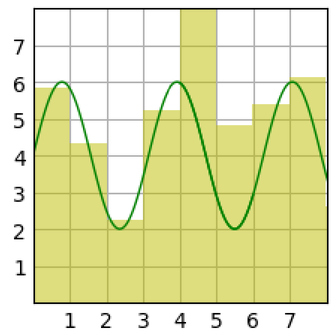


Figure 3: expt. 2

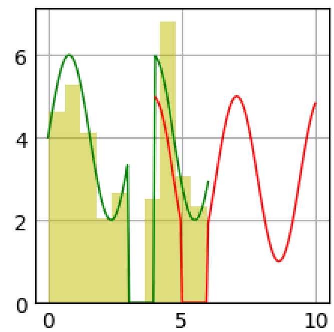


Figure 4: expt. 3

2.4.3 Convergence

Though this project doesn't consider M capping, it still needs to consider convergence guarantees when increasing the reusing window. The paper from Kettunen et al.[3] states that convergence is only guaranteed if $|R| > O(\sqrt{M})$, where $|R|$ is the number of canonical samples (the samples generated in the current pixel), and M is the number of total samples. Therefore, before reusing the neighbor reservoir, it still needs to do a regular RIS and generate a reservoir to include more samples whose domain is in the current pixel.

3 Experiments

To help understand, some experiments are reproduced as the presentation given by Kettunen et al.[3] before implementing code in La Jolla. Basically, these experiments are implementing RIS in different conditions to approximate the desired target PDF. All experiments are conducted with M = 32 (the number of candidates). To verify the correctness, the experiments compare the expected integration result of the target pdf (ground truth), the estimated result of Monte Carlo, and the estimated result of RIS. Also, the histograms show how the selected samples are distributed.

3.1 Experiment1: RIS

This is the simplest version of RIS, the same as Talbot et al. The entire domain has the same target pdf:

$$\hat{p}(x) = 4 + 2sin(2x)$$

and the candidates are generated by the same uniform pdf. The integrated domain is [0 - 10].

The histogram in figure 2 shows that RIS indeed generates samples proportional to the target pdf. The table 1 shows that the RIS generates a result close to the ground truth but with a smaller sample variance compared with Monte Carlo.

3.2 Experiment2: Reuse

The second version of RIS explores the reuse of RIS (Talbot et al 2005). The domain has the same non-zero target pdf as above, but the candidates are generated by two different source pdf: $p_1(x)$

Table 1: RIS version 1 result

method	value	std
ground truth	40.59	0
Monte Carlo	40.12	13.88
RIS	40.60	2.95

Table 2: RIS version 2 result

method	domain	value	std
ground truth	[0, 6]	24.15	0
Monte Carlo	[0, 6]	24.33	8.61
RIS	[0, 6]	24.20	1.70
ground truth	[4, 10]	23.45	0
Monte Carlo	[4, 10]	23.35	8.43
RIS	[4, 10]	23.46	1.69
ground truth	[0, 10]	40.59	0
Monte Carlo	[0, 10]	41.48	13.41
RIS	[0, 10]	40.39	2.79

only distributed in domain [0, 6] and $p_2(x)$ only distributed in domain [4, 10], which simulates the case that two adjacent pixels, A and B, may have different normals and some samples generated at A could not be generated at B. The RIS uses the naive method to eliminate bias (dropping the samples with pdf $<= 0$). The integrated domain is still [0 - 10]. The histogram in figure 3 and the table 2 show a similar result as before (adding two additional integration over the different source pdf domains just for verification).

3.3 Experiment3: Zero PDF

The third version of RIS includes two different target pdfs. The domain has two different target pdf as shown by the green and red lines. Notice that some part of the target pdf is zero. The candidates are generated by two different source pdf as the second experiment. The RIS still uses the naive method to eliminate bias. The integrated domain is now [0 - 6] (considering the first domain as the current pixel's domain). The main difference between experiment 2 and experiment 3 is that when reusing two pixels and calculating the m, experiment 3 needs to exclude the samples whose $\hat{p}(x_z)$ is zero. The histogram in figure 4 shows a similar result as before and the RIS generates a result close to the ground truth with a smaller variance.

4 Implementation

Most of the pseudocodes of algorithms are mostly the same as those in the paper Bitterli et al.[1]. Some changes are needed to port the ReSTIR into La Jolla, which are written in the C++ code style below. Also, note that lots of implementation details are not possible to be included concisely here.

Table 3: RIS version 3 result

method	domain	value	std
ground truth	[0, 6]	20.37	0
Monte Carlo	[0, 6]	19.56	11.74
RIS	[0, 6]	19.01	1.87
ground truth	[0, 6]	15.86	0
Monte Carlo	[4, 10]	15.70	11.19
RIS	[4, 10]	16.10	1.79
RIS(reuse)	[0, 6]	19.08	1.83

4.1 Streaming RIS

```
void regular_ris(const Scene& scene, Reservoir& rsv,
    pcg32_state& rng, const PathVertex& vertex, const Vector3& dir_view) {
    for (int r = 0; r < scene.options.ris_samples; r++) {
        // First, we sample a point on the light source.
        ....
        PointAndNormal point_on_light = sample_point_on_light(...);

        Real source_pdf = light_pmf(...) * pdf_point_on_light(...);

        // target_pdf = Le * f * G
        Real target_pdf = eval_target_pdf(vertex, dir_view, point_on_light);

        // w = target_pdf / source_pdf
        Real contri_W = 1.0 / source_pdf;
        Real w = target_pdf * contri_W;
        Real reservior_r = next_pcg32_real<Real>(rng);
        update_reservoir(rsv, point_on_light, w, reservior_r);
    }
    rsv.ref_vertex = vertex;
    rsv.prev_dir_view = dir_view;
    return;
}


struct Reservoir {
    Real M;
    Real w_sum;
    Real W;
    PointAndNormal y;
    PathVertex ref_vertex;
    Vector3 prev_dir_view;
};
```

```
inline bool update_reservoir(Reservoir &rsv,
    PointAndNormal x, Real w, const Real &pdf_w) {
    rsv.w_sum += w;
    rsv.M += 1;
    if (rsv.w_sum == 0) {
        return false;
    }
    if (pdf_w < (w / rsv.w_sum)) {
        rsv.y = x;
        return true;
    }
    return false;
}
```

The subsection explains how to do regular RIS and generator a new reservoir for a pixel. Note that the reservoir needs to store the sample y, M, W, w_sum, as well as the current pixel vertex and dir_view for the latter reservoirs combining.

4.2 Combine Reservoir

```
Real combine_reservoirs(const Scene &scene,
                        const std::vector<Reservoir>& reservoirs,
                        const PathVertex& vertex,
                        pcg32_state &rng,
                        const Vector3& dir_view,
                        Reservoir& new_reservoir,
                        int& selected_reservoir_id,
                        bool isMIS) {
    Real new_reservoir_M = 0;
    for(int r = 0; r < size(reservoirs); r++) {
        //* target_pdf = Le * f * G
        Real target_pdf = eval_target_pdf(vertex, dir_view, reservoirs[r].y);

        Real w = target_pdf * reservoirs[r].W * reservoirs[r].M;
        if (isMIS) {
            w = target_pdf * reservoirs[r].W;
        }

        if(update_reservoir(new_reservoir, reservoirs[r].y, w, reservoir_r) {
            selected_reservoir_id = r;
        }
        new_reservoir_M += reservoirs[r].M;
    }
    return new_reservoir_M;
}
```

The subsection briefly shows how to combine reservoirs. Note that the `eval_target_pdf` here needs to consider visibility (doesn't list here). The RIS weight used for the naive and the MIS versions is slightly different, too. Basically, the MIS version do not need to scale with the M because the RIS weight is totally represented by the MIS weight. The naive version, however, represents the RIS weight with $M(i)/\sum M(i)$.

4.3 Render

```
Image3 img(w, h);
ImageReservoir imgReservoir(w, h);
ImageReservoir prevImgReservoir(w, h);

constexpr int tile_size = 16;
int num_tiles_x = (w + tile_size - 1) / tile_size;
int num_tiles_y = (h + tile_size - 1) / tile_size;

std::vector<std::vector<pcg32_state> > rngs(num_tiles_x);
for (int rng_i = 0; rng_i < num_tiles_x; rng_i++) {
    rngs[rng_i] = std::vector<pcg32_state>(num_tiles_y);
    for (int rng_j = 0; rng_j < num_tiles_y; rng_j++) {
        // generate rng for each tile
        ...
    }
}

ProgressReporter reporter(spp);
for (int s = 0; s < spp; s++) {
    parallel_for([&](const Vector2i &tile) {
        int x0 = tile[0] * tile_size;
        int x1 = min(x0 + tile_size, w);
        int y0 = tile[1] * tile_size;
        int y1 = min(y0 + tile_size, h);
        for (int y = y0; y < y1; y++) {
            for (int x = x0; x < x1; x++) {
                std::vector<Reservoir> reservoirs;
                if(s > 0) {
                    for(int d = 0; d < 4; d++) {
                        int n_x = x + random(..)
                        int n_y = y + random(...)
                        n_rsv = prevImgReservoir(n_x, n_y)
                        reservoirs.push_back(n_rsv);
                    }
                }

                bool reuse = !(s==0);
                Reservoir rsv = init_reservoir();
```

```
                pcg32_state rng = rngs[tile[0]][tile[1]];
                Spectrum radiance = restir_path_tracing(...);

                img(x, y) += radiance / Real(spp);
                imgReservoir(x, y) = rsv;
            }
        }
    }, Vector2i(num_tiles_x, num_tiles_y));
    reporter.update(1);
    prevImgReservoir = imgReservoir;
}
```

Note that, it moves the spp iteration out of the pixel iteration. Also, it needs to generate and reuse the same rng for each tile. Otherwise, it will generate artifacts with the previous La Jolla renderer.

4.4 Path Tracing

```
...
if(first_bounce) {
    // RIS
    if(!reuse) {
        regular_ris(scene, rsv, rng);
        unbiased_reuse_m = 1.0 / rsv.M;
        point_on_light = rsv.y;
    } else {
        switch(scene.options.unbiased) {
        case static_cast<int>(Unbiased::NONE): {
            regular_ris(scene, rsv, rng);
            reservoirs.push_back(curr_pixel_rsv);

            // reuse with bias(Algorithm (4))
            new_reservoir = combine_reservoirs(...);
            point_on_light = new_reservoir.y;
            rsv = new_reservoir; // reuse
            m = 1.0 / rsv.M;
            break;
        }
        ...
}
...
//* replace 1 / pdf with W
Real p1 = light_pmf(...) * pdf_point_on_light(...);
if (!first_bounce) {
    C1 /= p1;
} else {
    if(rsv.M == 0) {
```

```
        //* empty reservior
        C1 = make_const_spectrum(0);
    } else {
        if (luminance(C1) == 0) {
            //* if target_pdf is 0, set W to 0
            rsv.W = 0;
        } else {
            rsv.W = m * (1. / luminance(C1)) * rsv.w_sum;
        }
        C1 *= rsv.W;
    }
}
...
```

Since this project only uses ReSTIR for direct illumination, only the first bounce of the path tracing needs to be changed. Also, reuse happens when spp is larger than 1. When calculating the contribution of the next event estimation in the first bounce, the C1 is multiplied with the RIS weight W instead of dividing by p1 (note that the p1 is still used for regular MIS). The remaining part is exactly the regular Monte Carlo estimator.

The project implements three unbiased RIS along with the biased one: the naive method, the MIS method, and the HMIS method which combines RIS with heuristics.

5 Result

5.1 Cornell Box

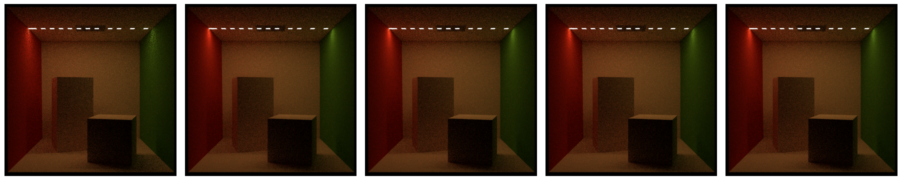
The project first renders the Cornell Box for functional tests. The light number is increased from 1 to 13. The sample number for the ReSTIR reservoir is set to 8.

Since that ReSTIR is a real-time method to generate good results with low sample cost, it's fair to start with low spp. Figure 5 [1] shows the result with just spp=2. It's obvious to see a great improvement and much lower variance with ReSTIR from parts of the result 6. The results overall look similar, which proves the correctness of the ReSTIR implementation. Figure 7 shows the results of spp=50, which are more similar to each other and proves the convergence of the ReSTIR can be maintained with spp=50 (though this project didn't do an experiment with super high spp). Figure 8 compares the results generated by biased and unbiased estimators in high spp. It shows that when spp grows, the artifacts of the biased version dominate and generate a darker result. The difference is larger at the position where the target pdf changes dramatically, e.g. the normal changes.

5.2 Many Lights

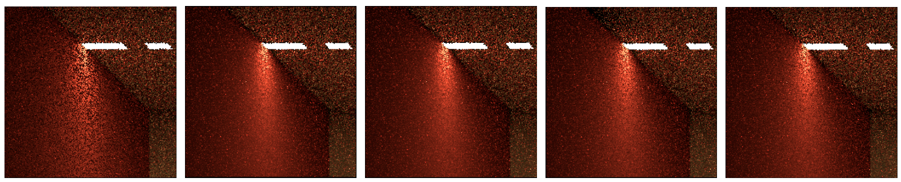
The project also renders some scenes with many lights to further show the power of ReSTIR. The scene has 400 lights in total, and most of them are above the stage (though they can't be seen from the image, they indeed exist in the scene file). Figure 9 shows a great difference between the Monte

[1] Please see the "final" folder for result images.



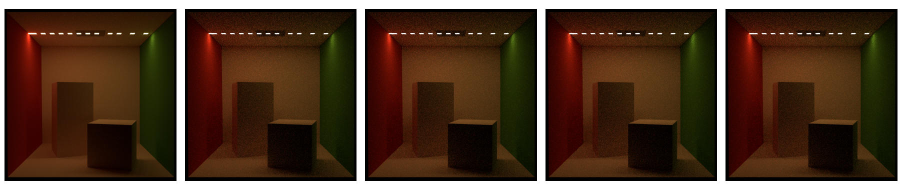
(a) mc (b) biased (c) naive (d) MIS (e) HMIS

Figure 5: Cornell box – full result (spp = 2)



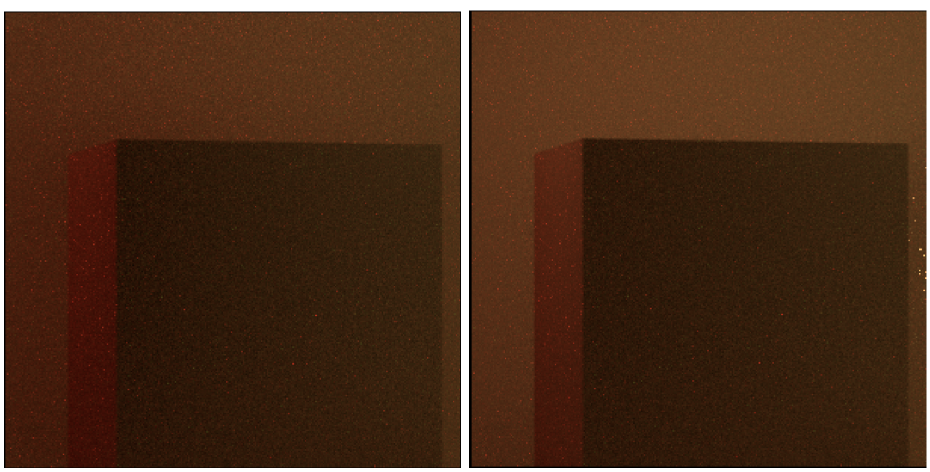
(a) mc (b) biased (c) naive (d) MIS (e) HMIS

Figure 6: Cornell box – partial result (spp = 2)



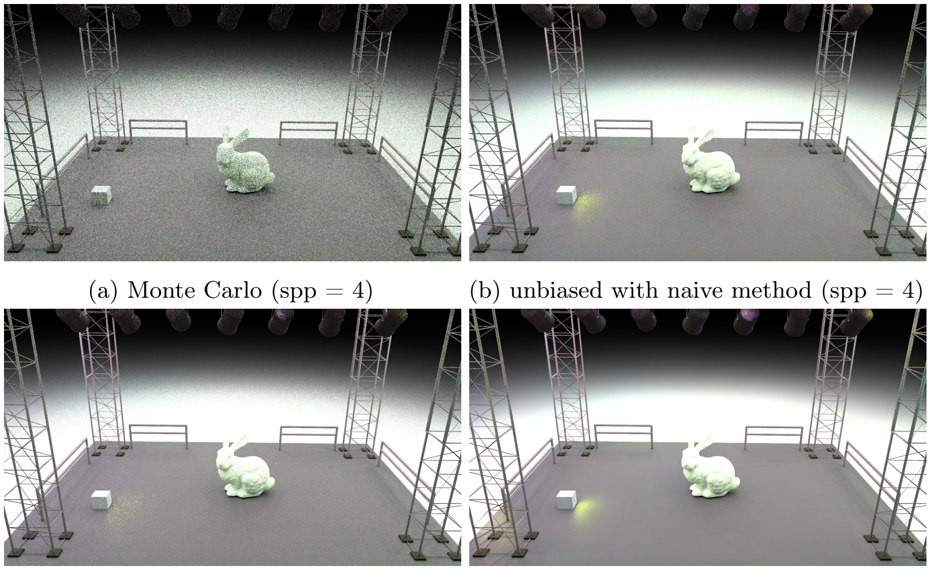
(a) mc (b) biased (c) naive (d) MIS (e) HMIS

Figure 7: Cornell box – partial result (spp = 50)



(a) biased (b) unbiased with naive method

Figure 8: Cornell box – compare biased and unbiased (spp = 50)



(a) Monte Carlo (spp = 4) (b) unbiased with naive method (spp = 4)

(c) Monte Carlo (spp = 40) (d) unbiased with naive method (spp = 40)

Figure 9: many lights (full scene)

Carlo method and ReSTIR (native version) at spp=4. Figure 10 represents the difference between these 5 methods in the same locations.

5.3 Final scene

A final scene 11 that may not have enough lights to illustrate the power of ReSTIR but looks good:)

Acknowledgments and References

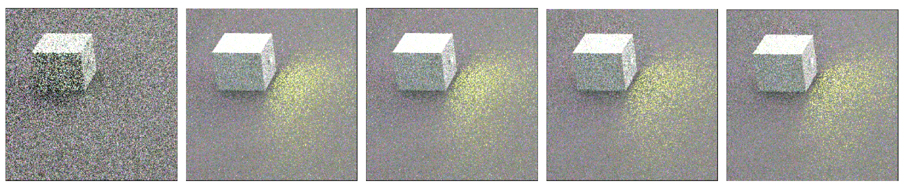
I'm extremely grateful to the teaching assistant Trevor Hedstrom. This project would not have been possible without his invaluable patience and help.

The scene is built in the blender with reference to cgtrader. References follow the acknowledgments.

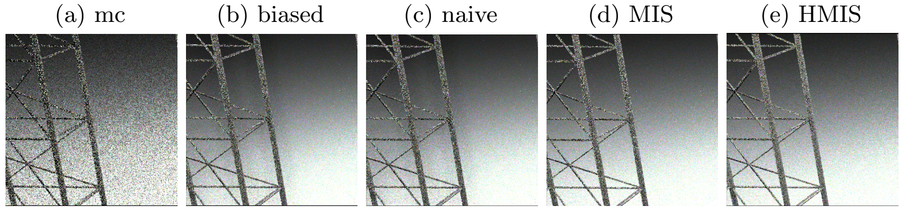
[1] Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. ACM Trans. Graph. (Proc. SIGGRAPH), 39(4):148, 2020.

[2] Chris Wyman, and Alexey Panteleev. Rearchitecting Spatiotemporal Resampling for Production. High-Performance Graphics (2021)

[3] Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, Chris Wyman. Generalized Resampled Importance Sampling: Foundations of ReSTIR. ACM Transactions on Graphics (SIGGRAPH 2022), 41, 4, 2022



(a) mc (b) biased (c) naive (d) MIS (e) HMIS



(f) mc (g) biased (h) naive (i) MIS (j) HMIS

Figure 10: many lights (spp = 4, partial scene)

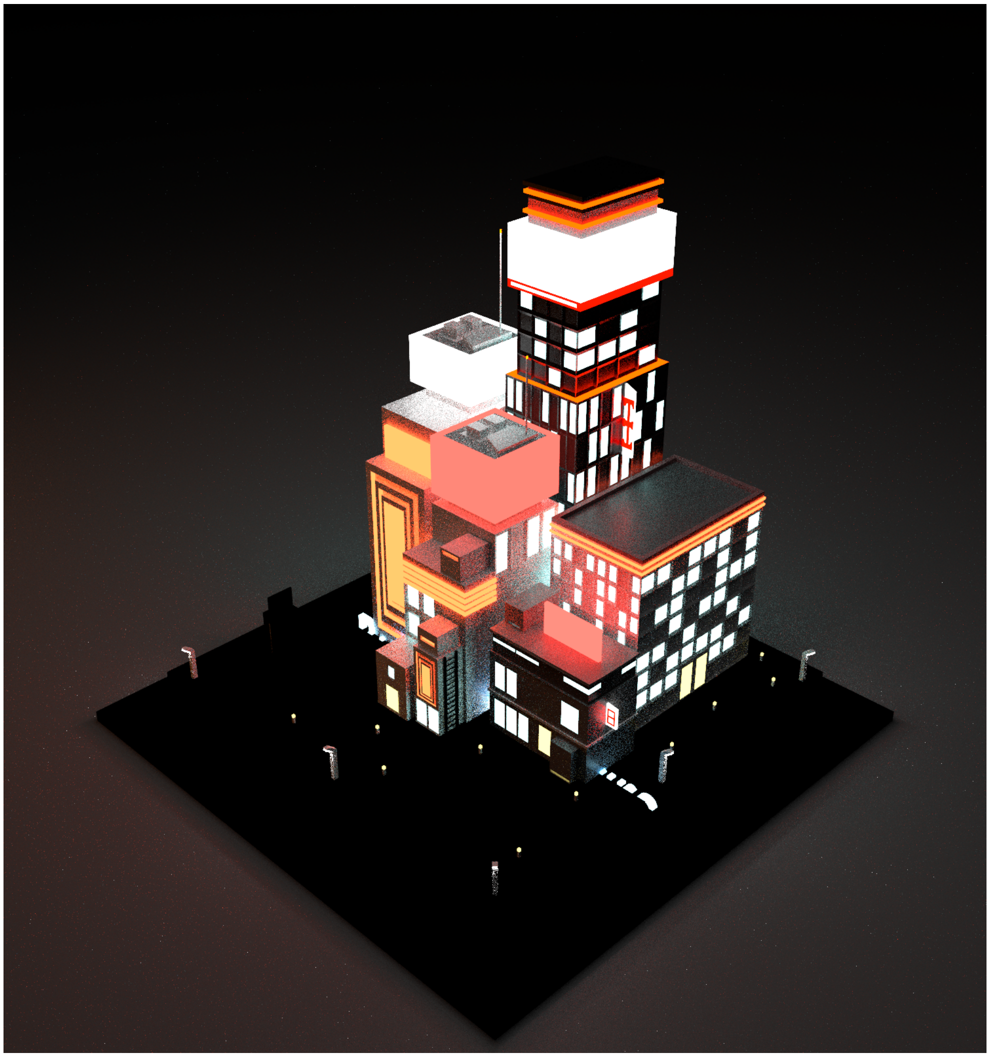


Figure 11: Cyberpunk