A Better Light Candidate Generation Algorithm for ReSTIR Ray Tracing Using an Acceleration Structure to Identify Relevant Lights

Rafayel Gardishyan[1]

Supervisors: Christoph Peters[1]**, Michael Weinmann**[1]**, Elmar Eisemann**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 20, 2025

Name of the student: Rafayel Gardishyan
Final project course: CSE3000 Research Project
Thesis committee: Michael Weinmann, Christoph Peters, Georgios Smaragdakis, Elmar Eisemann

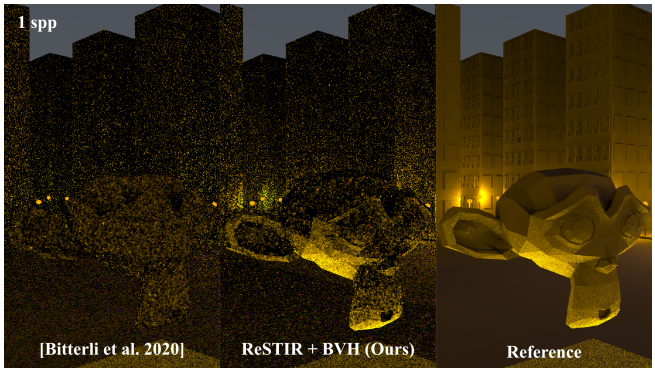


Fig. 1. Render of the Night Cityscape scene. Comparison of the first frame between the original ReSTIR paper, our method and a reference render. Rendered with $M = 23$, 1 sample per pixel

Abstract

The efficient rendering of scenes with many light sources remains one of the most challenging problems in real-time ray tracing. As the complexity of virtual environments continues to increase, with some scenes containing thousands of light sources, traditional Monte Carlo methods struggle to achieve acceptable noise levels under real-time constraints. This paper introduces a novel approach that combines Reservoir-based Spatiotemporal Importance Resampling (ReSTIR) with a specialised bounding volume hierarchy (BVH) structure to enhance light candidate generation for scenes with many light sources. The BVH-assisted candidate generation is tested on multiple scenes, resulting in a significant decrease in image noise levels measured with the root mean squared metric (RMSE), along with improved visual quality from the first frame onward, especially in scenes with numerous light sources illuminating local areas.

1. Introduction

Real-time rendering of scenes illuminated by many light sources presents a significant computational challenge in computer graphics. While ray tracing provides physically accurate lighting solutions, the computational cost of evaluating contributions from numerous light sources makes achieving real-time performance difficult. In production environments, tracing even a single ray per pixel may only be feasible on high-end hardware, making the efficient use of each sample critical for maintaining interactive frame rates. Reservoir-based Spatio-temporal Importance Resampling (ReSTIR), introduced by [Bitterli et al. 2020], represents a big advance in handling complex lighting environments efficiently. Their original algorithm demonstrates remarkable efficiency improvements, demonstrating similar visual fidelity, while offering a speed-up by a factor of $6\times$-$60\times$ compared to Resampled Importance Sampling (RIS) when rendering direct lighting. ReSTIR works by repeatedly resampling a set of candidate light samples and applying both spatial and temporal resampling to leverage information from samples for similar shading points, effectively multiplying the number of samples considered per pixel, without the need for additional sampling.

The original ReSTIR algorithm demonstrated the power of spatio-temporal resampling for direct lighting but did not focus on optimising candidate generation. Recent extensions like ReSTIR GI [Ouyang et al. 2021] and ReSTIR PT [Lin et al. 2022] have expanded the technique to handle global illumination, but still face challenges with initial candidate selection in many light scenarios. Meanwhile, dedicated research on many-light sampling [Moreau et al. 2019; Boksansky et al. 2021; Shah et al. 2023] has explored specialized data structures without fully leveraging the benefits of reservoir-based resampling.

The paper by [Moreau et al. 2019] has explored hierarchical data structures to accelerate direct lighting calculations. For instance, dynamic many-light importance sampling approaches utilizing bounding volume hierarchies (BVHs) have shown promise in efficiently identifying relevant light sources [Moreau et al. 2019]. However, these approaches have a high cost in both candidate generation and maintenance in dynamic scenes due to the frequent updates required for the acceleration structures, potentially offsetting their performance benefits [Moreau et al. 2019].

Despite these advances, the initial candidate generation phase of ReSTIR, often chosen to be a uniform random selection, remains a bottleneck when dealing with vast numbers of locally illuminating lights. The world-space properties of the shading points and the light sources are not considered, which can lead to suboptimal convergence rates and reduced rendering quality when real-time conditions strictly constrain sample budgets. [Boksansky et al. 2021].

Building on these foundations, we propose an integration of ReSTIR with a specialised BVH, containing light sources and their respective volumes of influence. While the light samples are evaluated and reused by the ReSTIR algorithm, the light candidates are generated by performing point queries for the shading point to identify relevant light sources. We will investigate whether it is possible to integrate the BVH candidate selection into the ReSTIR pipeline, and compare the performance of the said candidate generation method on the aspects of convergence speed, render time and memory usage against that of the original ReSTIR with the goal to speed-up convergence of many light renders like the scene demonstrated in Figure 1

The remainder of this paper will describe the problem, the proposed algorithm, the experiment methodology and the conclusion in detail.

2. Problem Statement & Background

The rendering equation, as can be seen in Equation (1) expresses the outgoing radiance L_o at point y in direction ω_0 as an integral over all incoming directions ω_i. In real-time ray tracing—especially in scenes with dozens or hundreds of streetlamps—the brute-force evaluation of this integral is expensive. Instead, Monte Carlo integration approximates it by randomly sampling incoming directions. To keep variance low, we employ importance sampling: we bias our samples toward directions (and thus light sources) that contribute

most through the bidirectional reflectance distribution function (BRDF) f_r and the geometry term $\cos\theta_i$. Choosing the right subset of lights for sampling is important in complex environments since any light might potentially be relevant to the shading without a beforehand evaluation of f_r.

$$L_o(y,\omega_0) = \int_\Omega f_r(y,\omega_i \to \omega_0) L_i(y,\omega_i) \cos\theta_i \, d\omega_i \quad (1)$$

2.1. Light Sampling

Uniform sampling, which selects one light at random per shading point and progressively averages the pixel colour after many iterations, often results in high variance and thus noisy images. This is because lights in the distance, which are less visible, are equally likely to be sampled as nearby ones. An improved technique for sampling is Importance Sampling (IS), which aims to reduce variance by sampling N samples from an easier to sample distribution. We choose a distribution $p \propto f_r(\omega)\cos\theta$ and calculate the Monte Carlo estimate as described in Equation (2). In practice, instead of a direction, a point on the light is sampled, which is then converted to a direction to calculate the rendering equation. For readability purposes, we will denote L_o as $\int_X f(x)\,dx$, where $f(x) \equiv f_r(\omega(x))L_i(\omega(x))\cos\theta$ after dropping the viewing direction ω_0 and the shading point y, and introducing the conversion function $\omega(x)$ from a point x on the light to a direction ω.

$$\langle L \rangle_{\text{IS}} = \frac{1}{N}\sum_i^N \frac{f(x_i)}{p(x_i)} \approx L \text{ [Bitterli et al. 2020]} \quad (2)$$

Perfect IS would sample from the actual distribution, which is hard to normalise, thus sample from. Because of that an easier to sample, but yet similar distribution is often used to obtain good results, for example by evaluating the solid angle and skipping the expensive visibility test. Still, IS has to evaluate the probabilities for all lights to be able to sample correctly.

2.2. Resampled Importance Sampling

Resampled Importance Sampling (RIS) is a technique introduced to reduce the cost of IS by numerical sampling from a distribution that is difficult to sample from analytically [Talbot et al. 2005]. RIS works by selecting M candidates from the total light area L with an easy-to-evaluate function, which we call the *source probability distribution function* p. This can even be a uniform distribution. Then we select the actual sample x_i from the selected M samples using the original, more expensive to sample, *target probability density function (PDF)* $\hat{p}$. The target PDF can be the BRDF, but also a mixture of visibility, geometry term and light contributions. It is important to note that the expensive to sample from target PDF is evaluated for M samples, where $M \ll L$, but is not sampled from. The M samples are given a weight, w_i like in Equation (3), which defines the distribution the final sample is drawn from. A sample $y \equiv x_i$ is selected with RIS using the estimator in Equation (4)

$$p(x_i|\mathbf{x}) = \frac{w(x_i)}{\sum_j^M w(x_j)} \text{ where } w(x) = \frac{\hat{p}(x)}{p(x)} \quad (3)$$

$$\langle L \rangle_{\text{RIS}}^M = \frac{f(x)}{\hat{p}(y)} \cdot \left(\frac{1}{M}\sum_j^M w(x_j) \right) \text{ [Bitterli et al. 2020]} \quad (4)$$

2.3. ReSTIR

Reservoir Spatio-temporal Importance Sampling (ReSTIR) combines RIS with reservoir sampling and introduces spatio-temporal reuse [Bitterli et al. 2020].

A reservoir is a data structure that holds one or more samples selected from a larger set, while also keeping track of the number of candidates for which the reservoir was updated with (M) and the total weight of those candidates (w_{sum}) [Wyman 2021]. This reservoir can be used effectively for RIS, as we only choose one sample from a stream of samples. The reservoirs are updated according to Algorithm 1.

Algorithm 1 Weighted Reservoir Sampling [Wyman 2021]

Global:
reservoir R,
total weight $w_{\text{sum}} \leftarrow 0$,
sample count $M \leftarrow 0$

Function `update`(S_i, w_i)**:**
- $w_{\text{sum}} \leftarrow w_{\text{sum}} + w_i$
- $M \leftarrow M + 1$
- $\xi \leftarrow$ rand() $\in [0,1]$
- **if** $\xi < w_i / w_{sum}$ **then**
  - $R \leftarrow S_i$

Function `combine`(R_1, w_1, M_1, R_2, w_2, M_2)**:**
- $w_{\text{sum}} \leftarrow w_1 + w_2$
- $M_c \leftarrow M_1 + M_2$
- $\xi \leftarrow$ rand() $\in [0,1]$
- **if** $\xi < w_1 / w_{sum}$ **then**
  - $R_c \leftarrow R_1$
- **else**
  - $R_c \leftarrow R_2$
- **return** R_c, w_{sum}, M_c

ReSTIR keeps a fixed-size reservoir of light samples for every pixel and updates the reservoirs during the RIS process. Then it applies temporal reuse by merging the reservoir of the pixel from the previous frame, and spatial reuse by merging the reservoirs of the surrounding pixels. This reuse effectively multiplies the number of samples considered per pixel without the need to sample more.

The original ReSTIR algorithm, however, needs multiple frames to find relevant light sources when used on a scene with many lights, especially when there is a multitude of light sources affecting a small, local region. To accelerate the light selection and steer towards faster convergence, we suggest

building a bounding-box volume hierarchy (BVH) acceleration structure to hold the light sources within their respective volume of influence. This structure will then be used to cull irrelevant light sources for any shading point. This effectively sets the source distribution $p(x) = 0$ for all x whose influence volume does not contain the shading point. To avoid bias that could result from this, we suggest interleaving the samples from the acceleration structure with uniformly drawn samples.

3. Proposed Algorithm

In the original ReSTIR paper [Bitterli et al. 2020], samples are generated uniformly over the area of emitters with a source distribution p, as shown in Algorithm 2. In contrast, we aim to sample emitters in a scene-aware fashion, prioritising those lights that are most likely to contribute to the shading point.

Our approach relies on an acceleration structure—in our case, a BVH built over each light's "influence volume"—which is constructed once during scene setup. At render time, we perform a point query in this BVH for every shading point, extracting only the lights whose influence regions contain that point. We then restrict sampling to this reduced subset. To keep the unbiased property, we interleave one uniformly distributed emitter sample among every k world-oriented samples, ensuring that every light source has a non-zero chance of being selected.

Algorithm 2 ReSTIR Initial Candidate Generation [Bitterli et al. 2020]

Input: $m \geq 1$
Output: Reservoir r
$r \leftarrow$ empty reservoir
for $i \leftarrow 1$ **to** m **do**
  generate $x_i \sim p_{\text{uniform}}$; `// Sample on light`
  r.update $(x_i, \hat{p}(x_i)/p_{\text{uniform}}(x_i))$

3.1. BVH Construction

For the construction of the BVH for the lights, we need to determine the influence radius, or rather volume, of the light.

In our renderer, we have chosen to only use triangular lights for the sake of simplicity of modelling scenes. A simple approach would be to use the luminance value of the light source as a factor in determining a sphere of influence.

The sizing factor S_r is a constant which has to be determined for every scene individually. A list of primitives can then be generated using the build_primitives function of Algorithm 3. These bounding boxes can be intersected using any BVH construction and intersection library, which supports sphere intersections and point queries.

A more sophisticated algorithm to construct the bounding boxes takes into consideration the visibility of a certain light source for a shading point. This method is more costly during construction, but might lead to even faster convergence. To achieve this, we suggest using an AABB instead of a sphere.

Algorithm 3 Simple BVH Construction

class LightPrimitive
  Data: points: Vec3[], position: Vec3, radius: float
  function set_position
    position $\leftarrow \sum_{p \in \text{points}} p/3$
  function get_aabb()
    min_vec $\leftarrow$ position - **Vec3**(radius)
    max_vec $\leftarrow$ position + **Vec3**(radius)
    return AABB(min_vec, max_vec)

function build_primitives*(lights)*
  Input : lights: list of Light, $[l_1, \ldots, l_n]$
  Output: primitives: list of (LightPrimitive, AABB)
  primitives $\leftarrow$ empty list
  foreach $l \in$ *lights* **do**
    pos $\leftarrow l$.position
    r $\leftarrow S_r \sqrt{l.\text{intensity} \cdot l.\text{area}}$
    prim $\leftarrow$ **new** LightPrimitive(pos, r)
    box $\leftarrow$ prim.get_aabb()
    primitives.append((prim, box))
  return primitives

During the construction, we start with a zero-sized AABB. We then cast i probe rays in different directions to determine the distance to the farthest shading point in every direction with for every ray $t_{max} = \text{radius}$. We adjust the size of the AABBs min and max after every intersection to enlarge the influence volume of the light, if any ray gets farther than the existing AABB. By taking the farthest values from the light source position, we ensure that no shading point that could have chosen this source as its sample gets excluded. This procedure is outlined in Algorithm 4.

Combining AABBs

If the light sources are not imported as objects but as separate triangles, the AABB construction method will generate many AABBs at a small distance from each other. This might result in suboptimal render times, specifically in the performance of point queries.

To mitigate this problem, we suggest combining light source AABBs with a clustering algorithm. In our results, we used a simple clustering algorithm based on proximity described in Algorithm 5.

Optimising the BVH scaling

The earlier-mentioned parameter S_r is different for every scene and needs to be adapted based on the scene scale and light intensity. To automate the parameter tuning of our algorithm, we propose to scale the AABBs automatically. There are multiple algorithms possible for this purpose, but for simplicity's sake, we have chosen to go with a naive algorithm, based on the min and max overlap count of the AABBs, with a quadratic running time. This algorithm is described in Algorithm 6. Used values are $\text{MAX\_OVERLAP} = 5$ and $\text{MIN\_OVERLAP} = 1$ (Boxes don't have to overlap, as the self-overlap is counted).

Algorithm 4 Visibility-Aware AABB Construction

```
class LightPrimitive
    Data: position: Vec3, radius: float
    function get_aabb(scene)
        Input  : lights: list of triangular lights,
                 scene: acceleration structure for ray intersect
        Output: prims: list of visibility-aware primitives
        min_vec, max_vec ← position
        sampled_bounds ← initialize empty bounds
        for j ← 1 to i do
            point ← sample a random point on the area of the
              light triangle
            ray ← generate ray from point in random direc-
              tion
            Offset origin slightly outward depending on ray
              direction
            if not scene.intersect(ray, max_t=radius) then
                set the ray t to radius
            expand sampled_bounds to include hit point
        min_vec, max_vec ← sampled_bounds
```

Algorithm 5 Combining Neighbouring AABBs

```
function combine_aabbs()
    Input  : primitives: list of bounding-boxed primitives,
             max_distance: distance threshold
    Output: combined_groups: list of merged primitive groups
    combined_groups ← empty list
    visited ← flag array for each primitive, all false
    foreach primitive p in primitives do
        if visited[p] then
            Continue
        group ← initialize group containing p
        visited[p] ← true
        foreach primitive q in primitives not yet visited do
            if distance(p, q) ≤ radius then
                add q to group
                visited[q] ← true
        merged_box ← merge_bounds(group)
        merged_ids ← collect_ids(group)
        combined_groups.append((merged_box,
          merged_ids))
    return combined_groups
```

Algorithm 6 Optimize AABB's scales

```
function OptimizePrims
    Input: combined_prims
    count bounding-box overlaps
    scaling ← 1.0
    while max_overlap > MAX_OVERLAP or
      min_overlap < MIN_OVERLAP do
        adjust S_r (smaller if big overlap, bigger else)
        scale all boxes about centers
        count bounding-box overlaps
```

3.2. Candidate Generation

To generate candidates from the BVH, we perform a point query for the shading point to identify light sources that are relevant to that shading point. From the light sources returned by the BVH, we choose m samples uniformly at random. Additionally, we want each light to have a non-zero chance of being selected as a sample to avoid bias. We achieve this by interleaving the samples generated by the BVH with samples generated uniformly at random. For every $k-1$ BVH samples, we pick 1 sample from a uniform distribution.

The PDF for the uniform distribution p_{uniform} (ReSTIR) is calculated as shown in Equation (5). Let

- N_L = total number of light sources in the scene
- A = area of the sampled light source
- r^2 = squared distance from the shading point to the light source
- $\cos\theta$ = cosine angle between the light direction and the normal of the light source

$$p_{\text{uniform}}(x) = \frac{r^2}{\cos\theta}\frac{1}{AN_L} \tag{5}$$

for the area of the light source A, a sampled point on the light x, the angle between the triangle normal and the light direction θ and the shading point.

For candidate generation using the BVH, we need to modify this source distribution function to reflect our generation mechanism. Because we are now sampling from two different distributions, we need to combine those distributions. We apply Multiple Importance Sampling (MIS) [Veach et al. 1995] to do this. Let

- b = the number of BVH-based samples
- u = the number of samples generated by uniform
- N_f = number of light sources found by the BVH
- $g = 1$ if the sample was found by the BVH, 0 otherwise

We define both area-measure PDFs p_1 and p_2 as

$$p_1 = \frac{g}{A\,N_f}, \quad p_2 = \frac{1}{A\,N_L} \tag{6}$$

The mixed area measure PDF is then defined by

$$p_A = \frac{b\,p_1 + u\,p_2}{b+u} = \frac{1}{b+u}\left(b\frac{g}{A\,N_f} + u\frac{1}{A\,N_L}\right) \tag{7}$$

Finally, converting from area measure to solid-angle measure introduces the geometry term $\frac{r^2}{cos\theta}$ with the resulting MIS density being defined as in Equation (8).

$$p_{\text{MIS}}(x) = p_A \cdot \frac{r^2}{\cos\theta} = \frac{r^2}{\cos\theta}\frac{1}{b+u}\left(\frac{b\,g}{A\,N_f} + \frac{u}{A\,N_L}\right) \tag{8}$$

This PDF is then used to calculate RIS weight w_i for a sample i as $w_i = \frac{\hat{p}(x_i)}{p_{\text{MIS}}(x_i)}$. In the case that the BVH has no intersections ($N_f = 0$) we fall back to uniform sampling, which results in $b = 0$ and $u = m$, effectively setting p_{MIS} to the same value as p_{uniform} would have been. The full sampling function is outlined in Algorithm 7.

Algorithm 7 BVH-Aware ReSTIR Initial Candidate Generation

Input: $m \geq 1$, $k > 1$, shading point y
Output: Reservoir r
$r \leftarrow$ empty reservoir
$b, u \leftarrow 0, 0$
$F \leftarrow$ lights overlapping y via BVH query
$N_f \leftarrow |F|$
for $i \leftarrow 1$ **to** m ***in parallel*** **do**
  if $N_f = 0$ ***or*** $i \mod k = 0$ **then**
    $l \leftarrow$ uniformly sample from all lights
    $u \leftarrow u + 1$
    $g \leftarrow 0$
  else
    $l \leftarrow$ uniformly sample from F
    $b \leftarrow b + 1$
    $g \leftarrow 1$
  $x_i \leftarrow$ sample point on light l
  $A \leftarrow$ area(l),
  $r^2 \leftarrow$ distance²(y, x_i),
  $\cos\theta \leftarrow$ geometry term
  $p_{\text{MIS}}(x_i) \leftarrow p_A \cdot \frac{r^2}{\cos\theta}$ // Equation (8)
  r.update$(x_i, \hat{p}(x_i)/p_{\text{MIS}}(x_i))$

4. Responsible Research

To ensure the integrity of our research, we took measures to make our research open, reproducible and correct. All the code, including the rendering engine and the models, is available on a Git repository. This ensures that our results are reproducible by peers and can be verified. Also, the openness of our approach ensures that curious-minded developers can integrate the method into their rendering pipeline.

Furthermore, the experiments are designed to be able to evaluate the performance of our method in different environments, like scenes with few or many lights and with different scales of geometry.

5. Experimental Setup

This section describes the experimental design employed to evaluate our BVH light candidate generation. In the following sections, we will discuss the experimental setup, including the rendering engine used, the scenes rendered and the comparison method. Afterwards, the results will be presented.

5.1. Rendering Engine

To evaluate our method's performance, we developed a minimal 3D rendering engine. This engine is designed to render direct illumination and supports only diffuse surfaces, providing a controlled environment for our experiments. It utilises the Intel® Embree ray tracing library [Wald et al. 2014]. The engine incorporates various light sampling techniques, including Uniform Importance Sampling, RIS, ReSTIR, and our custom implementation using a light BVH. The source code for the engine is publicly available on GitHub[1].

5.2. Scenes

We selected four synthetic scenes, created in the Blender 3D modelling software, to test the algorithmic performance across different scales and light–source configurations. Each scene employs only features supported by our renderer. The chosen scenarios are:

- **Monkey [Blender Foundation 2002]**: The Blender testing monkey, Suzanne, illuminated by only one light source.
- **Colourful Mess:** A compact arrangement of multiple objects illuminated by many coloured lights.
- **Sponza [Crytek 2001]:** A scene with a large number of triangles, illuminated by one light source.
- **Night Cityscape:** A city model with a large number of triangles, illuminated by many light posts and party lights.

The models are also included in the Git repository, alongside the code.

5.3. Experiments

For each scene, we have generated the following:

1. A *RIS* reference render [Talbot et al. 2005].
2. A *ReSTIR* reference render [Bitterli et al. 2020].
3. A render using our *BVH–based* sampling method.
4. A *ground–truth* render, produced via Uniform Importance Sampling with a high sample count (4096 samples per pixel).

All scenes are rendered for 20 frames, each having one sample per pixel. We evaluated image fidelity visually and by measuring the root mean squared error (RMSE) relative to the ground–truth render (lower values indicate higher accuracy). In addition, we recorded the per–frame rendering time as well as the preprocessing time required to construct the BVH data structure. To test the effect of the visibility-aware BVH construction, we render the images with both techniques.

[1] https://github.com/RafayelGardishyan/BVH-based-Light-Candidate-Generation

6. Results

The visual results, particularly from the "Night Cityscape" scene depicted in Figure 2, demonstrate the better performance of our method in low-light, locally illuminated environments. Figure 2 illustrates a side-by-side comparison where our approach, rendered with $m = 32$ and only 1 sample per pixel, side by side with the RIS and the ReSTIR techniques, both rendered with the same parameters. It is visually apparent that our method produces a higher-quality image with noticeably reduced noise, especially in areas close to light sources. For instance, the illumination caused by the light posts on the streets (highlighted by red and green insets) is more accurately resolved by our technique. These challenging points within the scene converge more rapidly and exhibit greater fidelity compared to the ReSTIR's uniform candidate generation approach.

The yellow inset, where the green party light has a greater influence on the building wall, demonstrates the ability of our method to identify the best light source nearby to multiple light sources, where ReSTIR struggles to do so.

Furthermore, our technique shows the ability to identify dim light sources that end up contributing to a shading point better than uniform candidate generation, because of the world-oriented approach. A good example of this is found in the blue inset of Figure 2, where the light under the monkey is detected by our algorithm. Our technique performs well, also for other scenes. The comparisons of these scenes can be found in Appendix A.

6.1. Effect of the radius sizing factor

During experimentation, we noticed that the BVH-guided candidate generation has a worse performance when many lights are selected by the point query. This affects both render times and image quality, which becomes more similar to that of ReSTIR. Changing the S_r value during BVH construction helps mitigate this issue. The algorithm detailed in Algorithm 6 works quite well for this purpose, and thus we compare the results utilising that algorithm.

6.2. Real-Time Convergence

The sped-up convergence of our proposed method is further substantiated in Figure 3, which showcases the progressive rendering of the "Night Cityscape" scene. This figure compares our technique (right column, rendered with $M = 32, 1$ sample per pixel) against the Uniform Candidate generation approach (left column, [Bitterli et al. 2020]) across frames 1 through 5 without temporal accumulation.

As shown in Figure 3, our method consistently produces images with lower noise levels from the initial frames. This rapid convergence is particularly useful for real-time applications where immediate visual feedback is crucial. Specifically, our approach effectively illuminates challenging areas, such as the party lights reflecting on building walls or the light illuminating the monkey. In contrast, the uniform candidate selection implementation exhibits more pronounced noise and slower convergence in these complex regions under the single sample per pixel constraint. The uniform candidate generation of ReSTIR did not successfully find the green party light in the initial frames, highlighting the efficiency of our BVH-based sampling.

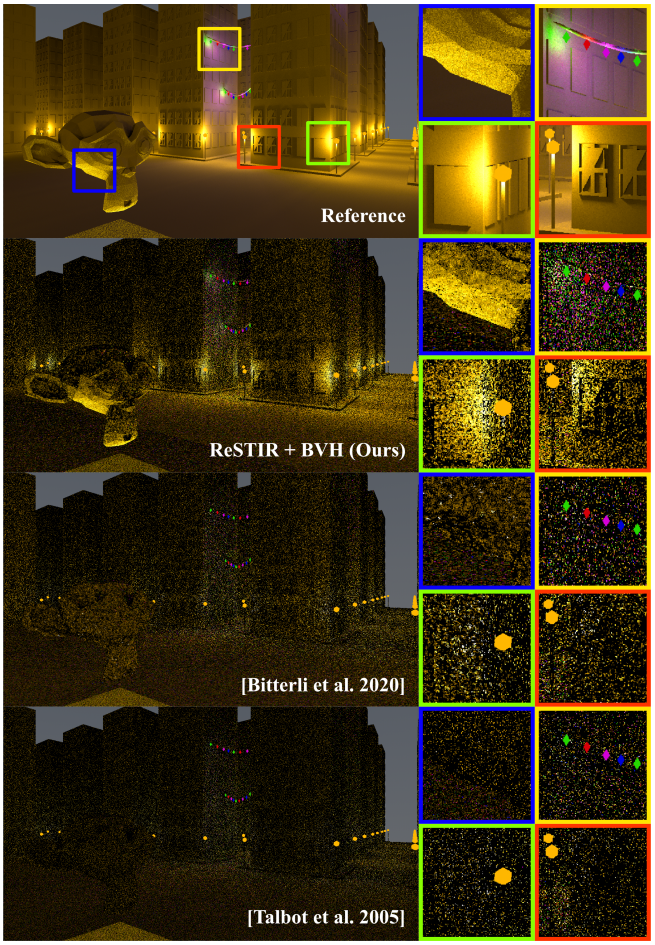


Fig. 2. The Night Cityscape scene. First frame comparison. Rendered with $M = 32$, $k = 8$ neighbours for spatial reuse in a radius of 5, 1 sample per pixel.

6.3. Error compared to reference

Figure 4 shows that our method achieves lower RMSE values during the first few frames of the rendering process. As more frames are accumulated, ReSTIR's error converges toward that of our approach. Across different scenes, the RMSE varies, mostly noticeable in the fact that more complex scenes start with a higher error and need more time to converge (Night Cityscape and Sponza); nonetheless, our method's error remains closely aligned with ReSTIR's performance, thus we see the results as trustworthy. For scenes with large, more global illumination, the performance seems comparable (Monkey).

In scenes containing only a few light sources, our method and ReSTIR produce almost identical RMSE curves. In scenes with many light sources, however, our approach has a consistently lower error throughout rendering. This can also be observed in the renders in Figure 5.

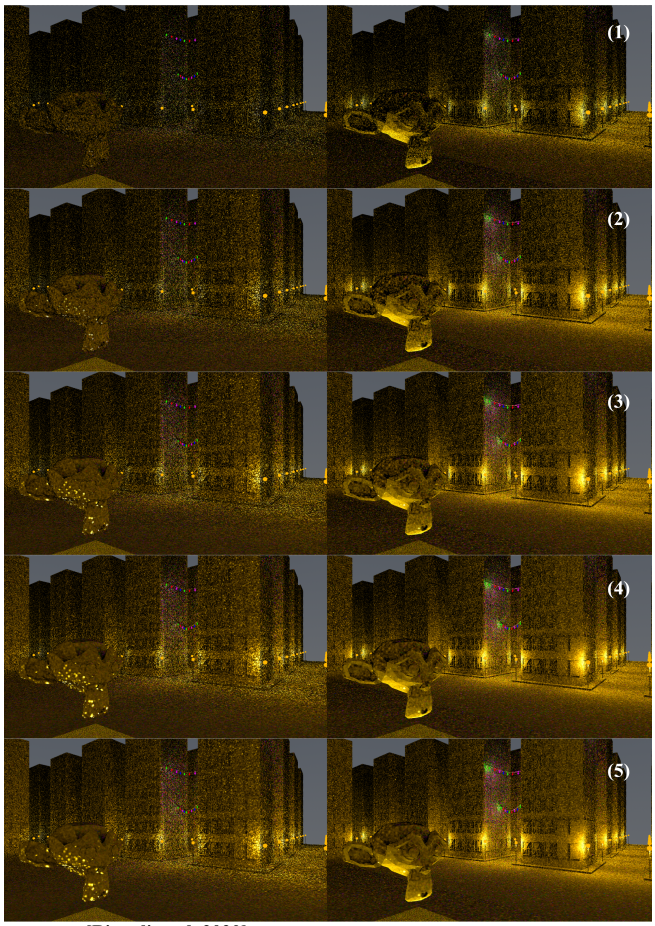


Fig. 3. The Night Cityscape scene. Frames 1–5, without accumulation. Left: ReSTIR [Bitterli et al. 2020], right: Ours. Rendered with $M = 32$, $k = 8$ neighbours for spatial reuse in a radius of 5, 1 sample per pixel

6.4. Render times & resource usage

When rendering with tuned S_r, the light BVH has a small overhead. Figure 6 shows that the render times are on average 12% slower than for uniform candidate generation in ReSTIR. This increase can be attributed to the unoptimized implementation of the rendering engine, as well as the overhead of the point queries and the expensive "find" operation, to check if the BVH found the light source.

When building the visibility-aware BVH, the BVH construction time has a significant impact on render times when the scene changes. On the other hand, the simple BVH construction method has a small set-up time and needs rebuilding less often. Table 1 shows the additional overhead of rebuilding the BVH. During experimentation, we found that the performance of visibility-aware BVH construction is not better than the simple strategy using the radius. This is detailed in Appendix B.

Furthermore, the handling of the light BVH has a low RAM consumption, because of the small amount of bounding boxes and the target to limit the overlap between the bounding boxes, resulting in negligible extra memory utilised to store the found lights. When used with improper values, however, memory consumption could become problematic. We recorded an extra 4% of memory usage when rendering with suboptimal parameters.

Model	BVH Simple	BVH Visibility Aware
Big City	0.3595 ms	2159.285 ms
Colorful Mess	0.4875 ms	2286.675 ms
Monkey	1.0×10^{-3} ms	0.7895 ms
Sponza	2.0×10^{-3} ms	1.5125 ms

Table 1: Median build time per mode and model

7. Discussion

Our results demonstrate that BVH-based light candidate generation offers benefits over the traditional uniform sampling strategy, particularly in complex lighting environments containing lots of local illumination. The improvements are most pronounced in scenes with many light sources, such as the "Night Cityscape," where uniform candidate selection struggles to identify significant contributors to illumination in early frames. In contrast, our BVH-based method efficiently narrows down the candidate lights, improving convergence speed and perceptual quality with a small per-frame computational cost increase.

Across most tested scenes, our method consistently outperformed ReSTIR in the early frames, excluding the Monkey scene, as evidenced by lower RMSE values and visually cleaner renders. This improvement is largely attributable to the spatial structure of the BVH, which allows the sampler to prioritise lights based on proximity, rather than relying solely on a uniform distribution or resampling over previous results. In real-time contexts, where initial frame quality is critical, this faster convergence is a compelling advantage.

Moreover, the qualitative results show that our method captures contributions from dim nearby light sources, but needs more time to find distant light sources that contribute to the image. This is apparent in Figure 3, where our BVH-based sampling has an equal amount of noise as ReSTIR in darker areas, where there are no light sources in close proximity.

In scenes with a low count of light sources, the performance of our method matches that of uniform light selection, which ensures that our method improves upon the established uniform selection.

However, our method has some limitations. The computational overhead is of the same magnitude as the difference between RIS and ReSTIR, but is still an additional slowing factor in the render time. Additionally, while BVH preprocessing introduces an upfront computational cost, most scenes include static lighting, which justifies the computational overhead to construct the light BVH. More so, when visibility data is not accounted for, the overhead is minimal. Because our experiments show that the performance of the BVH does

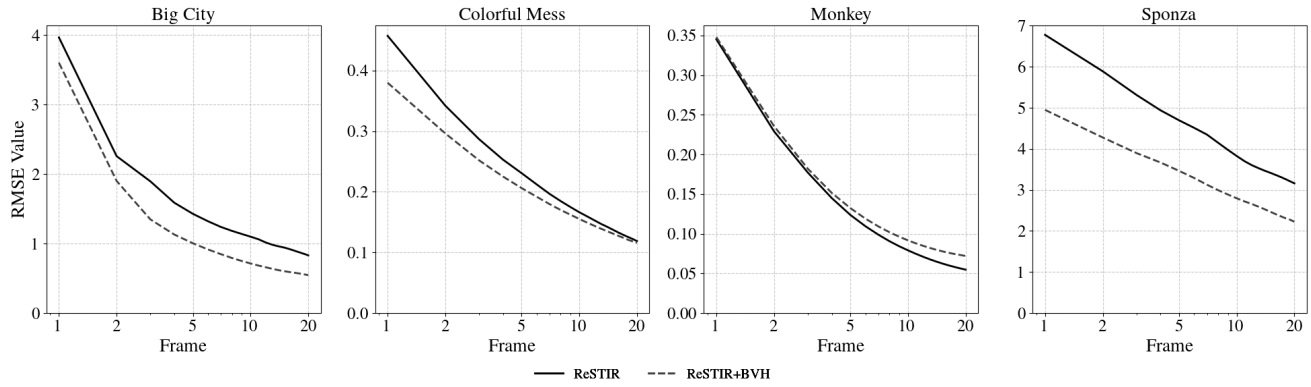


Fig. 4. Evolution of error (RMSE) in our scenes over frames rendered (accumulated). We compare ReSTIR [Bitterli et al. 2020] and our approach for 20 frames. Rendered with $M = 32$, $k = 8$ neighbours for spatial reuse in a radius of 5

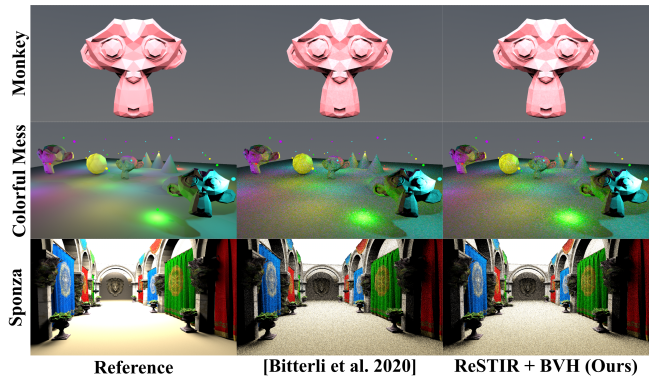


Fig. 5. Comparison of different scenes. Reference is rendered with Uniform Importance Sampling, with 4096 samples per pixel. ReSTIR and Ours rendered with $M = 32$, $k = 8$ neighbours for spatial reuse in a radius of 5, 1 sample per pixel. Accumulated 20 frames

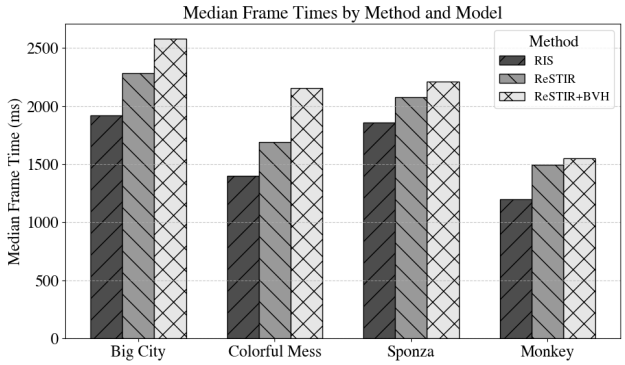


Fig. 6. Median frame times across different scenes. $N = 20$ for each method. Frames are rendered with a resolution of 1920×1080 pixels. Rendered with $M = 32$, $k = 8$ neighbours for spatial reuse in a radius of 5, 1 sample per pixel.

not improve with visibility awareness, we conclude that this additional computation is too expensive to include.

Because our rendering engine is CPU-based and not optimised, we measured performance per frame rendered and not in time. The rendering and BVH candidate generation can be accelerated when rendering on a GPU and will result in faster frame times.

Overall, our findings suggest that incorporating a world-space-oriented acceleration structure like a BVH into the light candidate generation process enhances the effectiveness of ReSTIR-based rendering. This structured approach is an alternative to uniform sampling and improves sample reuse by generating better samples from the first frames on.

8. Conclusions and Future Work

Our research addressed the challenge of rendering scenes with many local lights in Monte Carlo ray tracers. By integrating a better candidate generation, which effectively is a better sampling distribution, into the ReSTIR rendering pipeline, we have demonstrated an easy and effective technique to achieve images with less noise from the first frames. By interleaving the samples with uniformly generated candidates, we avoided bias.

Our experiments show that this combined approach significantly reduces noise and improves visual image quality from the first frame on, especially in complex scenes with lots of light sources. Our work contributes to the accessibility and efficiency of real-time ray tracing, even for larger scenes.

The algorithm suggested by this paper can be improved on, for example, in the automatic determination of the S_r parameter or experimenting with differently shaped bounding volumes. Furthermore, several promising directions for future research are identified. For example, dynamic scene support, possibly with a top and bottom-level BVH, would be a good addition to the algorithm. An implementation of the technique on a GPU-based rendering engine will also help prove the suitability of the technique to existing applications. A more challenging topic would be the integration of the technique with global illumination.

These extensions are a challenging follow-up to our research and would help push the boundaries of real-time ray tracing.

References

Bitterli, Benedikt et al. (Aug. 2020). "Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting". In: *ACM Transactions on Graphics* 39.4. DOI: 10.1145/3386569.3392481 (cit. on pp. 1–3, 5–8).

Blender Foundation (2002). *Blender Suzanne Model*. https://www.blender.org. The "Suzanne" monkey head is Blender's standard test model (cit. on p. 5).

Boksansky, Jakub, Paula Jukarainen, and Chris Wyman (Jan. 2021). *Rendering Many Lights with Grid-Based Reservoirs*, pp. 351–365. DOI: 10.1007/978-1-4842-7185-8_23 (cit. on p. 1).

Crytek (2001). *Sponza Atrium 3D Model*. https://www.crytek.com/sponza. Original model by Marko Dabrović, improved and released by Frank Meinl via Crytek (cit. on p. 5).

Lin, Daqi et al. (July 2022). "Generalized resampled importance sampling". In: *ACM Transactions on Graphics* 41.4, pp. 1–23. DOI: 10.1145/3528223.3530158. URL: https://doi.org/10.1145/3528223.3530158 (cit. on p. 1).

Moreau, Pierre and Petrik Clarberg (Jan. 2019). *Importance Sampling of many lights on the GPU*, pp. 255–283. DOI: 10.1007/978-1-4842-4427-2_18 (cit. on p. 1).

Ouyang, Y. et al. (Nov. 2021). "ReSTIR GI: Path Resampling for Real-Time Path Tracing". In: *Computer Graphics Forum* 40.8, pp. 17–29. DOI: 10.1111/cgf.14378. URL: https://doi.org/10.1111/cgf.14378 (cit. on p. 1).

Shah, Ishaan Nikhil, Aakash Kt, and P. J. Narayanan (Nov. 2023). "Combining Resampled Importance and Projected Solid Angle Samplings for Many Area Light Rendering". In: vol. 35, pp. 1–4. DOI: 10.1145/3610543.3626165. URL: https://doi.org/10.1145/3610543.3626165 (cit. on p. 1).

Talbot, Justin F., David Cline, and Parris Egbert (June 2005). "Importance resampling for global illumination". In: *Eurographics Symposium on Rendering Techniques*, pp. 139–146. URL: https://dl.acm.org/doi/10.5555/2383654.2383674 (cit. on pp. 2, 5).

Veach, Eric and Leonidas J. Guibas (1995). "Optimally combining sampling techniques for Monte Carlo rendering". In: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '95. New York, NY, USA: Association for Computing Machinery, pp. 419–428. ISBN: 0897917014. DOI: 10.1145/218380.218498. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/218380.218498 (cit. on p. 4).

Wald, Ingo et al. (July 2014). "Embree". In: *ACM Transactions on Graphics* 33.4, pp. 1–8. DOI: 10.1145/2601097.2601199. URL: https://doi.org/10.1145/2601097.2601199 (cit. on p. 5).

Wyman, Chris (Jan. 2021). *Weighted reservoir sampling: randomly sampling streams*, pp. 345–349. DOI: 10.1007/978-1-4842-7185-8_22 (cit. on p. 2).

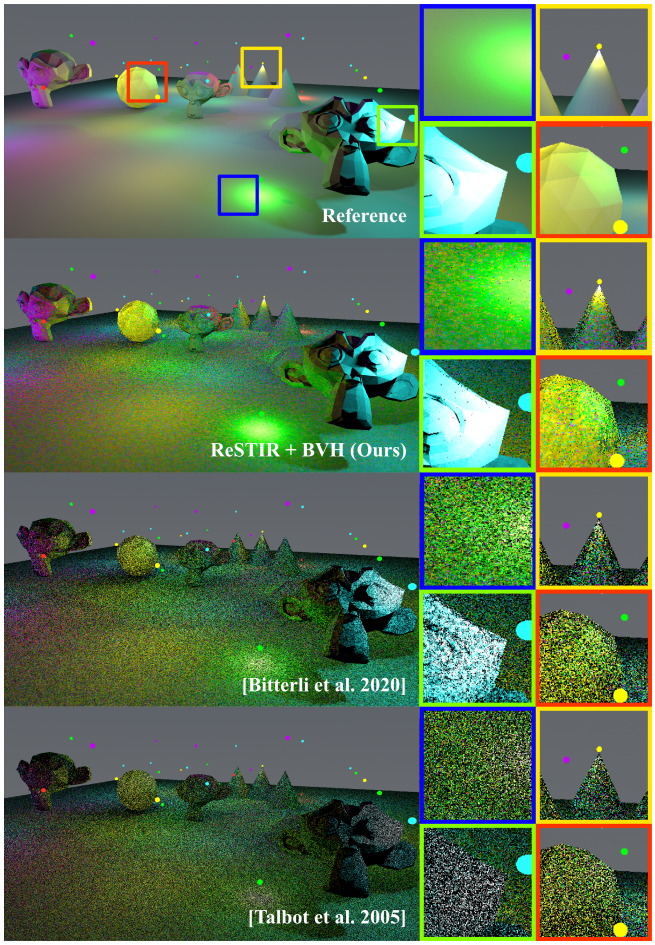


Fig. 7. The Colorful Mess scene. First frame comparison. Rendered with $M = 32$, $k = 8$ neighbours for spatial reuse in a radius of 5, 1 sample per pixel.

A. More Scene Comparisons

This appendix provides a more in-depth comparison of the Bounding Volume Hierarchy (BVH) based candidate generation algorithm against the state-of-the-art ReSTIR algorithm across a variety of scenes. The comparisons focus on the crucial first frame of rendering, which is of high relevance in real-time rendering.

As shown in Figure 7, the BVH-based candidate generation algorithm extends its performance beyond the "Night Cityscape" scene, but also for other scenes with many light sources, like the "Colorful Mess" scene. By culling away irrelevant light sources, the algorithm identifies the most prevalent light sources much quicker. The visual improvements are visible in the insets:

- The blue inset demonstrates better convergence of the light, especially in the gradient from bright to less bright.
- The yellow and red insets show an earlier convergence of subtle lighting.
- The green inset shows reduced noise levels on brightly lit surfaces, even with only one sample per pixel being rendered.

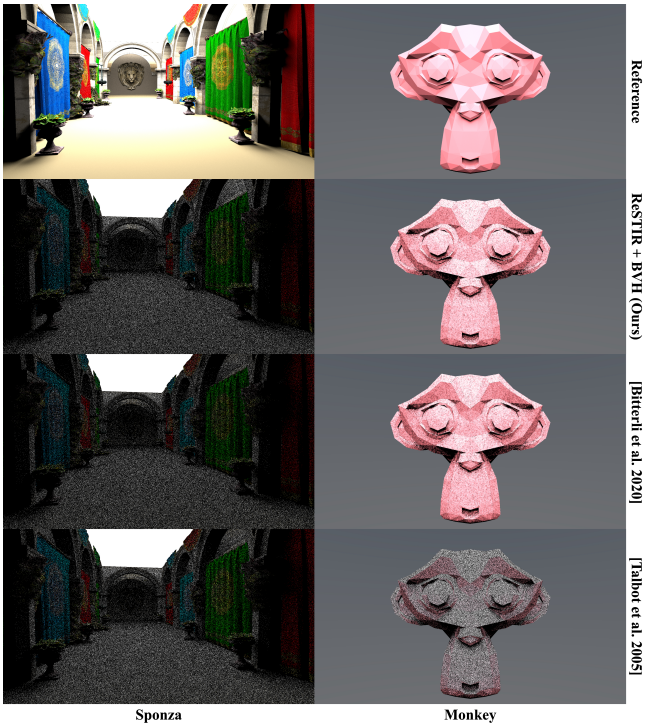


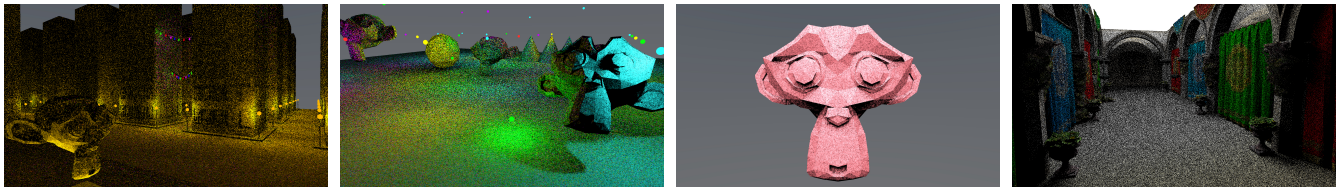
Fig. 8. The Monkey and Sponza scenes. First frame comparison. Rendered with $M = 32$, $k = 8$ neighbours for spatial reuse in a radius of 5, 1 sample per pixel.

In scenes with very few light sources, the improvements by the BVH-based candidate generation are negligible. We do measure, however, a similar performance as uniform candidate generation, which is expected, as our method is targeting scenes with a multitude of light sources. Consequently, Figure 8 shows the visual similarities between the different methods for scenes with a low emissive triangle count. For the "Monkey" scene, both uniform and BVH-based candidate generation show a better performance than plain RIS. For the "Sponza" scene, this difference is negligible.

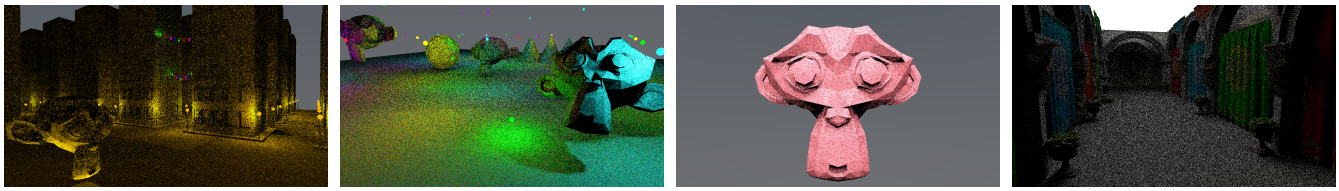
B. Visibility Aware BVH-Construction

This appendix provides additional details on the two BVH construction strategies evaluated in our work. The *simple* radius-based algorithm constructs the BVH purely according to a radius heuristic (described in Algorithm 3). This method is favourable due to its low construction cost and simple nature. The *visibility-aware* variant (described in Algorithm 4) tries to more closely align the light AABBs with the actual visibility. To achieve this, our visibility-aware method shoots rays to shrink the AABB parts that are not visible from any shading point.

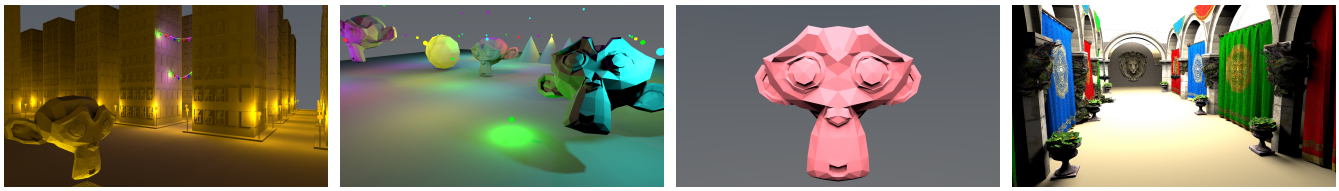
As shown in Figure 9, both BVH construction methods result in similar visual quality. Furthermore, the simple BVH construction method, for some scenes, results in a lower RMSE error (Figure 10).



Simple BVH construction strategy



Visibility-aware BVH construction strategy



Night Cityscape | Colorful Mess | Monkey | Sponza

Reference

Fig. 9. First frame comparison of all scenes using the simple and visibility-aware BVH construction strategies. Rendered with $M = 32$, $k = 8$ neighbours for spatial reuse in a radius of 5, 1 sample per pixel.

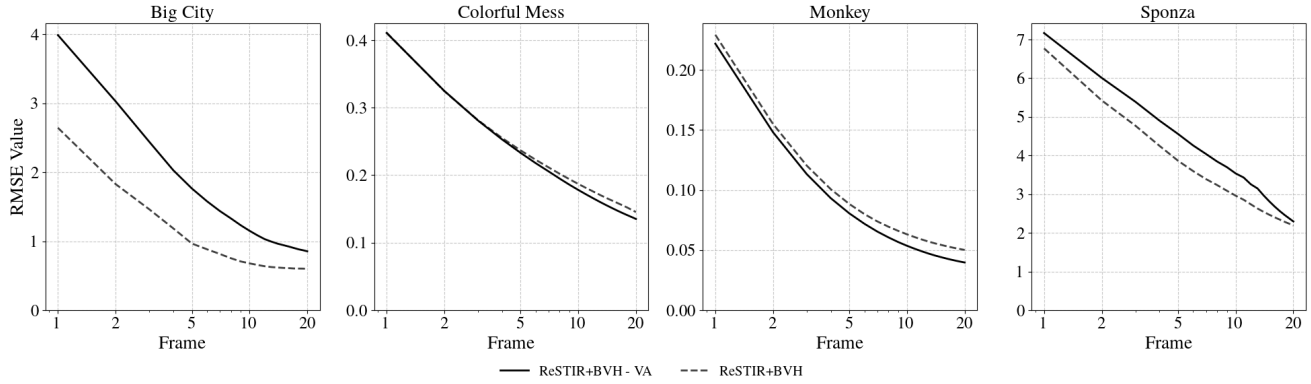


Fig. 10. Evolution of error (RMSE) in our scenes over frames rendered (accumulated). We compare simple BVH construction (ReSTIR+BVH) and visibility-aware BVH construction (ReSTIR+BVH - VA) for 20 frames. Rendered with $M = 32$, $k = 8$ neighbours for spatial reuse in a radius of 5.