Pacific Graphics 2023
R. Chaine, Z. Deng, and M. H. Kim
(Guest Editors)

Volume 42 (2023), Number 7

World-Space Spatiotemporal Path Resampling for Path Tracing

Hangyu Zhang[1] and Beibei Wang[†2,3]

[1]Xi'an Jiaotong University, China
[2]Nankai University, China
[3]Nanjing University of Science and Technology, China

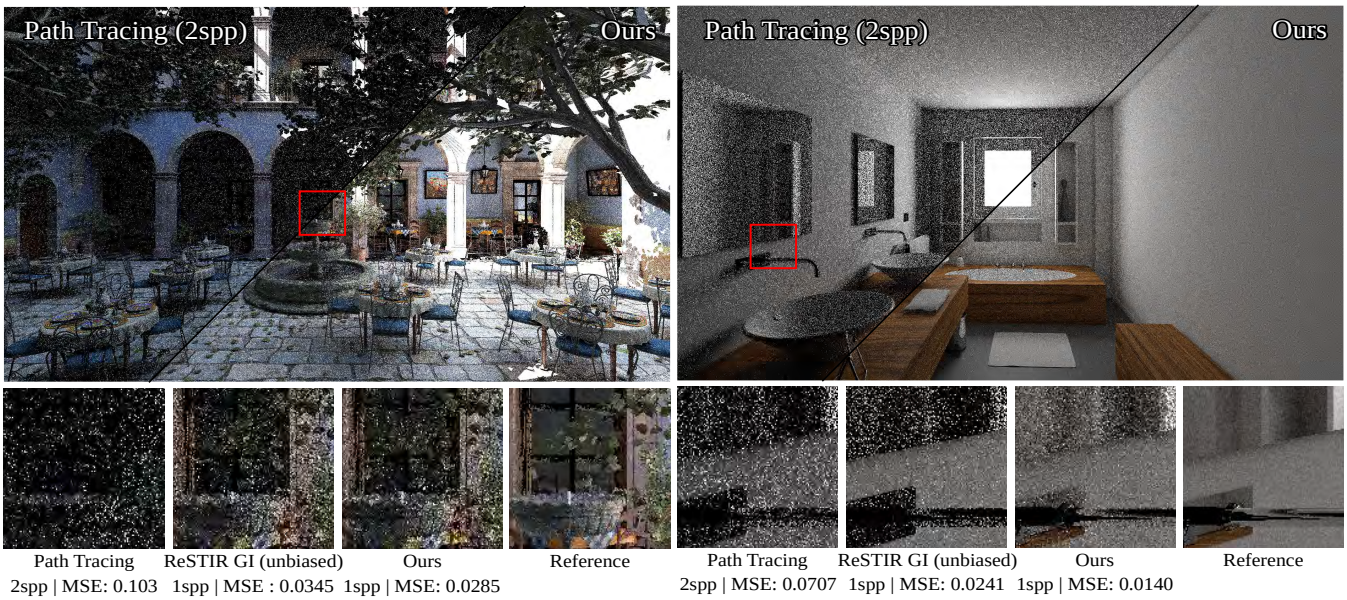


Figure 1: *Comparison between path tracing, ReSTIR GI [OLK*21] and our method with equal time on two scenes (*SAN MIGUEL *and* BATHROOM*) with complex indirect lighting. The mean squared errors (MSE) are shown under the images. The image resolution is 1920 × 1080. Our method outperforms the other methods both visually and quantitatively.*

Abstract

With the advent of hardware-accelerated ray tracing, more and more real-time rendering applications tend to render images with ray-traced global illumination (GI). However, the low sample counts at real-time framerates bring enormous challenges to existing path sampling methods. Recent work (ReSTIR GI) samples indirect illumination effectively with a dramatic bias reduction. However, as a screen-space based path resampling approach, it can only reuse the path at the first bounce and brings subtle benefits for complex scenes. To this end, we propose a world-space based spatiotemporal path resampling approach. Our approach caches more path samples into a world-space grid, which allows reusing sub-path starting from non-primary path vertices. Furthermore, we introduce a practical normal-aware hash grid construction approach, providing more efficient candidate samples for path resampling. Eventually, our method achieves improvements ranging from 16.6% to 41.9% in terms of mean squared errors (MSE) compared against the previous method with only 4.4% ∼ 8.4% extra time cost.

CCS Concepts

• ***Computing methodologies*** → ***Rendering; Ray tracing;***

† Corresponding author

1. Introduction

Monte Carlo path tracing has been popular for offline rendering for years. In recent years, with the development of hardware-accelerated ray tracing [EHNB18, Tak20], path tracing can also be used for real-time rendering by tracing a few rays per pixel. However, insufficient sampling leads to intractable noisy images. Hence, it is essential to design effective sampling strategies so that abundant scene lighting can be simulated at low sampling rates.

Bitterli et al. [BWP*20] introduce the reservoir-based streaming resampled importance sampling (RIS) to direct illumination (DI), called *ReSTIR DI*. The resampling cost is amortized among frames and neighbors in the screen space, improving direct illumination at a low time cost. Later, ReSTIR GI [OLK*21] extends ReSTIR for global illumination (GI), by resampling the path samples for visible points in the temporal and spatial domains, resulting in obvious noise reduction. However, the improvements are subtle in several scenarios, including high-frequency geometry variation, distant scene lighting, and glossy materials, due to resampling with screen-space buffers. World-space ReSTIR [Boi21] caches light sample reservoirs in the world space, producing noisy results due to the lack of path resampling. ReSTIR PT [LKB*22] generalizes the RIS theory with robust multiple importance sampling (MIS) weight and incorporates other shift mappings to handle complicated scene lighting, resulting in remarkable quality improvement at the cost of expensive time overhead due to the hybrid shift mapping.

In our paper, we introduce a practical world-space path resampling into ReSTIR GI. More specifically, we generalize the reused paths starting from the primary vertices to paths of any length, i.e., an entire path or a path suffix starting from non-primary vertices, and then use a world-space hash grid to organize these path sample reservoirs. Our approach can immediately be used for glossy materials, by resampling the subsequent points of the glossy materials. Thanks to world-space resampling, our approach can effectively reuse samples in more complicated scenes and distant scenes, as well as resampling indirect lighting for glossy materials. As a result, our method 12 Thanks to the efficiency of the grid structure, it only takes extra less than 10% overhead than ReSTIR GI to obtain such an improvement. Compared to ReSTIR PT, our method is lightweight, and can be used for real-time applications, while ReSTIR PT produces higher quality at the cost of three times slower than our method.

We review the related work in the next section and briefly introduce the ReSTIR theory in Section 3. We present our resampling approach in Section 4 and discuss the implementation details in Section 5. Then we show our results in Section 6 and conclude in Section 7.

2. Previous work

In this section, we briefly review related works, including path reuse [BSH02, BPE17], path guiding [MGN17, DGJ*20, HEV*16], resampling and mutation strategies [TCE05, KMA*15] and path space filtering [KDB14, BFK18] and particularly focus on the path resampling and world-space reuse techniques.

Resampling methods. Talbot et al. [TCE05] propose resampled importance sampling, which allows sampling from a sub-optimal distribution and then refines the distribution to a target distribution. Bitterli et al. [BWP*20] combine RIS with reservoir sampling [LC09] for direct illumination. Their approach draws samples from light sources for each reservoir and reuses samples between similar reservoirs in spatial and temporal domains. Boksansky et al. [BJW21] propose a uniform grid structure for ReSTIR, which allows multiple reservoirs within each voxel, leading to higher quality than the original RIS. Rez et al. [Rez21] employ a hash table to organize light samples in the path space and retrieve potential candidates from the table. Their method shows improvements over ReSTIR, especially in areas with low effective sample counts.

Later, Ouyang et al. [OLK*21] extend ReSTIR to global illumination by adapting the reservoir resampling to path sampling. Their method shows dramatic noise reduction compared to previous path sampling methods. Boissé [Boi21] applies spatial hashing and reuses light samples at secondary vertices to reduce noise. However, his method shows subtle benefits for handling indirect lighting due to a lack of path sampling, in particular when the primary vertex is diffuse. Relying only on BSDF sampling to generate the secondary vertices is insufficient, even if the direct lighting is resampled at the subsequent vertex.

Compared with these approaches, we cache and reuse an entire path or a path suffix starting from non-primary vertices, allowing for resampling vertices at further bounces.

Lin et al. [LKB*22] introduce a generalized RIS theory, by extending resampling to samples from an arbitrary domain. Under this theoretical framework, their approach can handle more complicated scenarios with glossy and specular materials. Our approach can also benefit from the influential theory to reuse more complex path samples in the world space with other powerful shift mappings. However, it will bring a huge overhead and deviate from our goal – a lightweight GI approach.

Several approaches couple reservoir resampling with other GI approaches, like dynamic diffuse global illumination (DDGI). DDGI resampling [MMK*21] combines ReSTIR with DDGI [MGNM19]. They produce less noisy results with a unified resampling scheme than ReSTIR GI and less bias than DDGI.

World-space reuse methods. Screen-space filtering approaches [SZR*15] denoise the rendered results in the screen space. Path-space filtering [KDB14] extends their work by averaging path contributions in the path space, providing a more accurate solution of dis-occlusions during temporal filtering. Path-space filtering dramatically improves visual quality, although searching for nearby paths is expensive. Fast path-space filtering [BFK18] replaces the costly neighborhood searching with a single query in a hash table, by separating a scene into cells and storing paths in a hash map. Path-space filtering is suitable for real-time rendering by leveraging the massively parallel structure of modern GPUs. Inspired by their work, many approaches introduce world-space reuse via spatial hashing, like Deng et al. [DHC*21] and Gautron et al. [Gau20]. In our paper, we introduce the spatial hashing idea into path resampling.

3. Background and motivation

At the core of global illumination is rendering equation [Kaj86]:

$$L(\mathbf{x},\omega_o) = \int_\Omega L_i(\mathbf{x},\omega_i)\rho(\omega_o,\omega_i)\langle\cos\theta_i\rangle d\omega_i \tag{1}$$

where L is the radiance at $\mathbf{x}$ with direction ω_o, Ω is the hemisphere of directions around the surface normal, L_i is the incoming radiance, ρ is the bidirectional scattering distribution function (BSDF) and $\langle\cos\theta\rangle$ is the cosine of the angle between the direction ω_i and the surface normal with negative values clamped to zero.

Then rendering equation can be computed using Monte Carlo estimator, by tracing N directions ω_j with the probability density function (PDF) $p(\omega_j)$:

$$\hat{L} = \frac{1}{N}\sum_{j=1}^{N}\frac{L_i(\mathbf{x},\omega_j)\rho(\omega_o,\omega_j)\langle\cos\theta_j\rangle}{p(\omega_j)}. \tag{2}$$

In real-time rendering, the sample count N should be strict to one or at most two for performance reasons. Hence, a PDF p which closely matches the integrand is required to improve the estimation quality.

3.1. Resampled Importance Sampling

A perfect PDF that matches the integrand is impossible. Talbot et al. [TCE05] proposed resampled importance sampling (RIS), which is effective in sampling complex functions. The basic idea is sampling the complex function with a known *source* PDF p to generate M samples $y = y_1, y_2, ..., y_M$ and then sampling the M candidates to get sample z using the *target* PDF $\hat{p}$ with probability:

$$p(z|y) = \frac{w(z)}{\sum_{i=1}^{M} w(y_i)}, w(y) = \frac{\hat{p}(y)}{p(y)}. \tag{3}$$

The distribution of z matches $\hat{p}$ more closely as the sample count M increases. Introducing z in Equation 2 results in a unbiased RIS estimator:

$$\hat{L} = \frac{f(z)}{\hat{p}(z)}\frac{1}{M}\sum_{j=1}^{M}\frac{\hat{p}(y_j)}{p(y_j)}, \tag{4}$$

where f is short for all the terms in the integration for clarity.

Note that RIS allows using unnormalized target PDF $\hat{p}$, and avoids sampling the complex function directly.

3.2. Weighted Reservoir Sampling

The original RIS [TCE05] allows resampling a large number of samples to get high-quality distribution. However, it requires storing all candidates before resampling, which brings heavy storage overhead and is unsuitable for GPU. Thus, Bitterli et al. [BWP*20] reformulate RIS using weighted reservoir sampling [LC09] (WRS), transforming it into a streaming algorithm. They use it for direct illumination with many lights. The resampling weight $W(z)$ for sample z is defined as:

$$W(z) = \frac{1}{\hat{p}(z)M}\sum_{j=1}^{M} w_j. \tag{5}$$

Thus, Equation 4 can be rewritten as:

$$\hat{L} = f(z)W(z). \tag{6}$$

3.3. ReSTIR GI

ReSTIR GI [OLK*21] resamples indirect lighting paths in the screen space. Since our method is built on top of their method, we define some notations and briefly review their approach.

A path generated with path tracing consists of a sequence of vertices in the scene, defined as $\bar{x} = \mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_n$, where $\mathbf{x}_0$ is the camera point and $\mathbf{x}_n$ is a point on the light source. The first shading point $\mathbf{x}_1$ along the path is called a *visible point*, visible from the camera. The second vertex $\mathbf{x}_2$ adjacent to the visible point along the path is called a *sample point*. The sub-path from the sample point to the light source forms a *path sample*.

At the core of ReSTIR GI is resample of the path samples for the visible points, together with a spatial and temporal reusing. For that, the path samples, as well as the spatiotemporal reservoirs, are stored in screen-space buffers, including positions and normals for the visible and sample points – $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{n}_1$, $\mathbf{n}_2$, and the outgoing radiance L_2 and PDF p for each path. During rendering, they resample the path samples in the spatial and temporal reservoirs and reconnect the visible point with the path sample.

For spatial reuse, shift mappings are needed to map paths between different integration domains. ReSTIR GI uses a method called *reconnection shift mapping* by only reconnecting vertex with high roughness along paths. This shift mapping converts the PDF of the sample from one domain to another:

$$p_i'(y) = p(T_i(y))\left|\frac{\partial T_i}{\partial y}\right|, \tag{7}$$

where y is a sample from originate domain Ω_i, T_i is the shift mapping maps this sample from domain Ω_i to Ω, p is the PDF weight of mapped samples in Ω and $\left|\frac{\partial T_i}{\partial y}\right|$ is the Jacobian determinant for reconnection shift mapping.

3.4. Motivation

ReSTIR GI resamples the path samples in the screen space, which brings several difficulties. Firstly, the path samples are limited to the ones starting from sample point, preventing resampling the paths from further bounces. Secondly, the spatial coherence defined in the screen space breaks for discontinuous areas (e.g., the regions in the distant view or with large normal variation), leading to a significant decrease in resampling quality. Finally, ReSTIR GI becomes less effective for paths with highly glossy surfaces, due to the reconnection shift mapping. Other shift mapping functions [LKB*22] can address this issue, by introducing exhaustive time cost. Another lightweight option is to reuse the subsequent paths after the glossy surfaces, which could alleviate the resampling quality degradation, although is less accurate than the introducing shift mapping functions. Unfortunately, this lightweight option is infeasible for ReSTIR GI due to the fact of screen-space resample.

To tackle these issues, we introduce world-space reuse into ReSTIR GI. This way, all the paths starting from the non-primary vertices can also be resampled, allowing for more path sample candidates. It also indicates that the subsequent paths after glossy surfaces can also be resampled. Moreover, a world-space structure can better leverage the spatial coherence of samples in the path space.

4. World-Space ReSTIR GI

Our world-space resampling approach mainly includes three steps, as shown in Figure 2. The path samples are generated and recorded in a sample generation step (Section 4.1). The next, they are organized into a hash grid for query efficiency (Section 4.2). Then the spatialtemporal resampling is performed at each visible point for indirect illumination (Section 4.3).

4.1. Sample Generation

In the sample generation step, we perform Monte Carlo path tracing starting from the camera and record path samples for resampling. The main difference from ReSTIR GI is that we extend the path samples to more bounces.

We shoot rays from the camera, which intersect with the scene, resulting in shading points. Starting from these shading points, we perform a regular path tracing – sample a direction at each point with BSDF sampling, and trace the ray in the sampled direction to get the next intersection, until reaching the light source. The vertices along each sampled path with a small roughness (say, smaller than 0.2) are called a specular vertex. We treat the first non-specular vertex as the *visible point*, denoted as $\mathbf{x}_1$, and then we rename all the vertices after the visible point *generalized sample point*, which are a generalization of the sample points in ReSTIR GI to arbitrary bounces.

After getting such a sampled path, we generate the path samples for resampling. We generalize the resample from sample points to all the generalized sample points. The path that connects a generalized sample point to a point on the light source is called a *generalized path sample*. For each generalized path sample starting from $\mathbf{x}_i$ (i>1), we record the following data: positions $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, normals $\mathbf{n_i}$ and $\mathbf{n_{i+1}}$, radiance L_{i+1} and source PDF p_i. The radiance of each generalized path sample is computed with the next event estimation (NEE) and MIS, while the source PDF is the BSDF. These generalized path samples are stored in a screen-space buffer and will be further organized into a hash grid later. The data structure details are shown in Section 5.1 and in Alg. 1.

4.2. Hash Grid Construction

After generating the generalized path samples, we organize them with a world-space hash grid for efficient query and compact storage. One key question is which property should be used for the hash key. Using the position (of the first vertex) of generalized path samples as the hash key is simple but becomes less efficient for generalized path samples with large normal variations. Therefore, we propose a normal-aware hash grid construction approach, considering both the position and the normal. Furthermore, we find that a uniform-sized hash grid decreases the resampling quality for regions at distant view due to over-small cells. Therefore, we introduce a cascaded hash grid, where the cells are dense around the camera and get sparse when distant from the camera.

Normal-aware hash function. The position-only hashing grid [Boi21] is constructed with an 1D hashing function $\mathcal{H}_1(\mathbf{x})$ which maps a world-space point to a hash index:

$$\mathcal{H}(\mathbf{x}) = \mathcal{H}_1(d + \mathcal{H}_1(\frac{\mathbf{x}_x}{d} + \mathcal{H}_1(\frac{\mathbf{x}_y}{d} + \mathcal{H}_1(\frac{\mathbf{x}_z}{d})))), \tag{8}$$

where $\mathbf{x} = (\mathbf{x}_x, \mathbf{x}_y, \mathbf{x}_z)$ is a world-space position, $\mathcal{H}_1()$ is a hash function defined on a given integer and d is the cell size for the hash grid.

Based on the position-only hashing grid, we introduce our normal-aware hash function:

$$\mathcal{H}(\mathbf{x}, n) = \mathcal{H}_1(\text{quantized}(n) + \mathcal{H}_1(d + \mathcal{H}_1(\frac{\mathbf{x}_x}{d} + \mathcal{H}_1(\frac{\mathbf{x}_y}{d} + \mathcal{H}_1(\frac{\mathbf{x}_z}{d}))))), \tag{9}$$

where quantized(n) is a quantize function on an input normal $n = (n_x, n_y, n_z)$. It maps continuous coordinates on a unit sphere to several discrete regions with unique indices:

$$\text{quantized}(n) = \chi(n_x) << (2 * d_n) \mid \chi(n_y) << d_n \mid \chi(n_z). \tag{10}$$

Where $\chi()$ maps any real number between 0 and 1 to integers ranging from 0 to any arbitrary normal quantized step d_n, which establishes the count of discrete regions of the normal space and is set as 2 in practice.

Cascaded hash grid. Then we construct a cascaded hash grid, by setting the cell size d as an adaptive value. First, a continuous world-space cell size d_c is computed based on the view distance d_v:

$$d_c = d_v \tan(\max(\frac{\text{fov}}{R_x}, \frac{\text{fov} \cdot R_x}{R_y^2}) d_c^s), \tag{11}$$

where tan is the tangent function, $R = (R_x, R_y)$ is the image resolution, d_c^s is the projected cell size in the screen space [Boi21]. Then, we adjust d_c with a minimum cell size $d_{\min}$, resulting in the adaptive cell size: $d = d_{\min} \cdot 2^{\lfloor \log \frac{d_c}{d_{\min}} \rfloor}$.

Now, we have a well-performed hash function that provides the identical hash key for similar samples. However, the hash conflict is inevitable as a nature property of spatial hashing. Thus, we use the commonly-used linear probing and a second hash index as well to avoid hash conflicts. At each hash cell, we store the indices of the generalized path samples, where the actual data are stored in an image buffer. This strategy is designed to optimize storage.

The implementation details and the settings of variables are shown in Section 5.2. Thanks to our normal-aware hash function, we can assign similar samples in the world space to a single unique cell index, leading to a more reasonable scene separation, as shown in Figure 3.

4.3. Spatiotemporal Path Resampling

In this section, we show how to use our world-space hash grid for spatiotemporal resampling. The constructed hash grid can be used for spatial resampling immediately. However, it can not benefit the temporal resampling due to the coarse representation of the hash

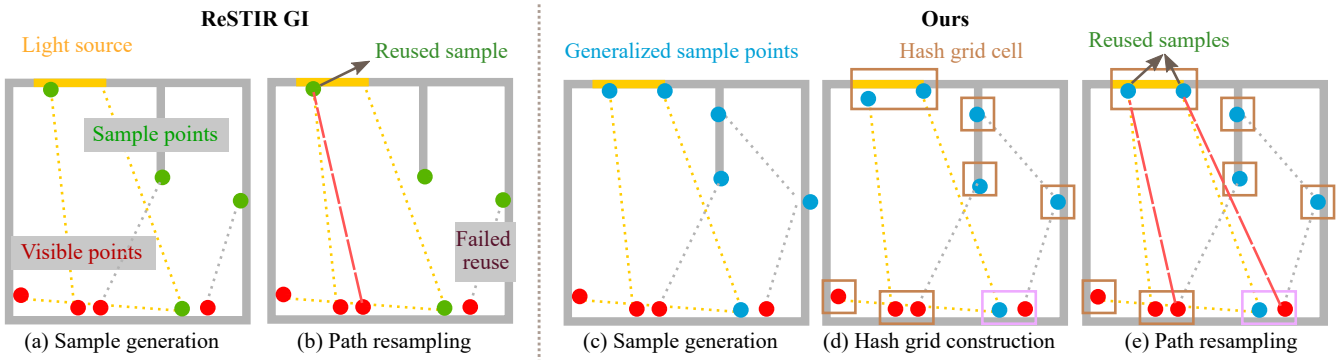


Figure 2: *Overview of ReSTIR GI and our method. In ReSTIR GI, only sample points next to visible points can be reconnected for reuse. A path sample starting from the visible point is recorded for resampling (a). During resampling, a path sample is queried in the screen space by checking the similarity of the visible points (b). Our method generalizes ReSTIR GI by including more vertices along the path, which form the* ***generalized sample points*** *(c). Starting from both visible points or generalized sample points, more path samples are generated, called* ***generalized path samples****. These generalized path samples are recorded in a hash grid for efficient query (d). During resampling, generalized path samples are resampled for each visible point, resulting in more reusing than the original ReSTIR GI. As shown in the lower right corner of the images (marked with light purple), the visible point's reservoir can reuse the sub-path starting at the generalized sample point, while the reservoir fails to reuse in ReSTIR GI (e).*

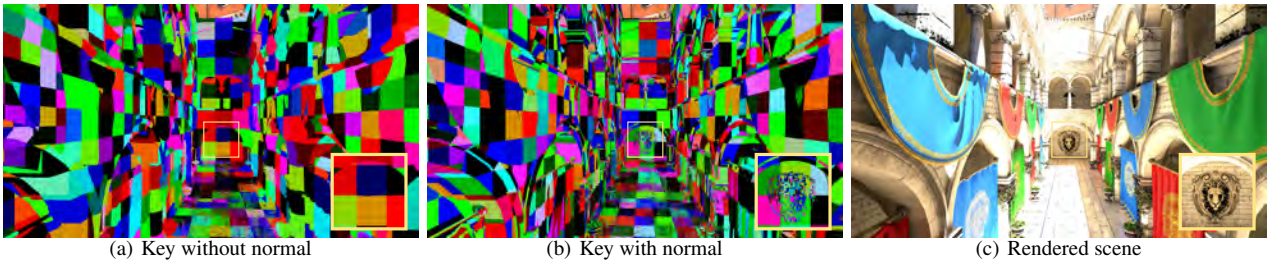


Figure 3: *Hash grid visualization w/o (left) and w/ (middle) normal included in the hash key for the* SPONZA *scene (right). Different cells are shown in different colors. The hash key consisting of normals separates the scene into smaller cells with both similar normal and position, leading to a more reasonable separation.*

grid. Increasing the resolution of the hash grid will resolve this issue, but it will affect the spatial reuse quality. Therefore, we perform the screen-space temporal resampling and then apply a world-space spatial resampling.

For temporal resampling, we follow the same way as ReSTIR GI [OLK*21]. For each pixel, we read the path samples directly from the generalized path sample buffer and randomly update the temporal reservoir with the samples, where the temporal reservoir is located by reprojection.

Then, we perform the spatial resampling. Given the position and normal of a visible point, we first use our normal-aware hash function to locate the hash cell, which has the indices of generalized path samples, and get the generalized path samples from the screen space buffers. Then, we validate the generalized path sample by checking normal similarity and visibility. We check the normal similarity by comparing the angle between the path sample's normal and the visible point's normal with a threshold (e.g., 15°). Any sample that fails to meet this condition is discarded. As for the visibility, we trace a ray from the visible point to confirm that the sample is visible from the visible point. For non-visible samples, the resampling weight is set as 0. Next, we transform the resampling weight of the generalized path sample to the visible point's domain using the contribution function described in Section 3.3. Finally, we update the spatial reservoir using this sample with the transformed resampling weight. After reusing all the generalized path samples in the hash grid cell, we correct the resampling weight of the spatial reservoir considering the contributed sample count $|Z|$, similar to ReSTIR GI [OLK*21]. The resampling details are shown in Section 5.3.

After the spatiotemporal resampling, the indirect illumination at the visible point is finally estimated as the product of the resampling weight $W(z)$, the cosine weight BSDF value, and the sample's outgoing radiance.

We provide a comparison between the world-space and screen-space spatial resampling in Figure 4, by showing the roughly reuse searching areas in a red block. Our approach (world-space resampling) yields more appropriate reuse candidates for complex scenarios, considering both the world-position similarity and the nor-

Figure 4: *Visualization of the resampling candidate region for a specific pixel (red dot) on the* BISTRO *and* GALLERY *scenes. The resampled region is shown in the red block around the query pixel. Compared to ReSTIR GI, our method considers the similarity of position and normal, leading to a more reasonable resample region and less noisy rendered results.*

Algorithm 1: Initial Sampling

for *each pixel q* **do**
  Get visible point $\mathbf{x}_1$ and normal $\mathbf{n}_1$ from G-buffer
  `// Find a rough visible point`
  while *current visible point roughness smaller than threshold* **do**
    Sample a direction and trace ray to find next point $\mathbf{x}$'
    $\mathbf{x}_1 \rightarrow \mathbf{x}$'
    $\mathbf{n}_1 \rightarrow \mathbf{n}$'
  end
  Use path tracing to generate the rest generalized sample points $\mathbf{x}_2$, $\mathbf{x}_3$, ...
  for *each generalized sample point i* ($i \geq 2$)$\in$ *path* **do**
    Store $\mathbf{x}_i$, $\mathbf{n}_i$, source PDF p_i and outgoing radiance L_i during path tracing
  end
  for *each point i* ($i \geq 1$) $\in$ *path* **do**
    InitialSampleBuffer[q][i] $\leftarrow$ Reservoir($\mathbf{x}_i$,$\mathbf{n}_i$,$\mathbf{x}_{i+1}$,$\mathbf{n}_{i+1}$,L_{i+1},p_i)
  end
end

mal similarity. Although our method's available region may be smaller than that of ReSTIR GI due to its adaptive size, the sample candidates within the region have higher resampling weights, indicating that the samples are more effective for reuse. Furthermore, our method also caches path samples at further bounces, which further enriches the candidates, leading to higher rendering quality.

5. Implementation

In this section, we will show some implementation details (data structure and algorithms) and practical choices.

5.1. Sample Generation

We keep three image-sized buffers to store sample reservoirs for different bounces, which is set as two in practice:

- `Initial sample buffer`: a buffer of initial reservoirs of generalized path samples.
- `Temporal reservoir buffer`: a buffer stores reservoirs that accept samples from previous frames.
- `Spatial reservoir buffer`: a buffer stores reservoirs that accept samples from neighbor reservoirs.

Algorithm 1 shows the pseudo-code of the sample generation step. Note that although the reservoirs are stored in screen-space buffers, their indices are stored in the hash grid to save memory bandwidth. This indicates that these reservoirs can also be queried by world-space positions and normals, rather than the screen space pixels only. We also keep another screen-space buffer for glossy surfaces, recording the accumulated BSDF value, path radiance, and path length leading up to the visible point.

5.2. Hash Grid Construction

Hash function. We use double hashing (using two hash functions) with two GPU-friendly hash functions, where *pcg*32() is our primary hash function and *jenkinsHash*() (a 32 bits version) is our secondary hash function. Both functions only require a few bit operations.

Hash grid data structure. The indices of the generalized path samples are organized in the world-space hash grid. Since each cell might have several generalized path samples, the indices of these generalized path samples per cell are stored in a sample index buffer successively. We locate them with a cell offset index for a cell and an in-cell index for each sample, which requires recording each cell's generalized path sample count and the cell offset index in two other buffers. Furthermore, the secondary hash key for each cell should be stored to identify whether the reservoirs are located in the cell. To summarize, we have the following buffers:

- `Sample index buffer`: a buffer stores the index of reservoirs in each cell.
- `Sample count buffer`: a buffer stores the count of reservoirs in each cell.
- `Cell offset buffer`: a buffer stores the offset index of cells in hash grid.
- `Checksum buffer`: a buffer stores the secondary hash key for each cell.

All the buffers for the hash grid are allocated with a fixed size corresponding to the maximum cell count, set as 3.2 M in practice. The sample index buffer is then set to the same size as the initial sample buffer.

Algorithm 2: Spatial Resampling

for *each pixel q* **do**
  Reservoir $\mathcal{R}_{sp} \leftarrow$ TemporalReservoirBuffer[q]
  cellIndex $\leftarrow$ FindCell($\mathbf{x}_{sp}$,$\mathbf{n}_{sp}$, *Checksum buffer*)
  $cell_{\text{begin}} \leftarrow$ CellOffsetBuffer[*cellIndex*]
  count $\leftarrow$ SampleCountBuffer[*cellIndex*]
  increment $\leftarrow$ (*count* + *maxIterations* -1) / *maxIterations*
  offset $\leftarrow$ Rand(0,*increment*-1)
  `// Z stores selected samples`
  $\mathbf{Z} \leftarrow \mathcal{R}_{sp}$
  for $i \leftarrow 0$ **to** $count - 1$ **by** *increment* **do**
    $index \leftarrow cell_{\text{begin}} + (i + offset) \% count$
    storageIndex $\leftarrow$ SampleIndexBuffer[*index*]
    $\mathcal{R}_n \leftarrow$ temporalReservoirBuffer[*storageIndex*]
    Compute normal similarity between $\mathbf{n}_{sp}$ and $\mathbf{n}_n$
    if *similarity is lower than threshold* **then**
      continue
    Adjust $\hat{p}_{x_n}$ with Jacobian determinant $J_{\mathcal{R}_n \to \mathcal{R}_{sp}}$
    if $\mathcal{R}_n$*'s sample is not visible for* $\mathcal{R}_{sp}$ **then**
      $\hat{p}_{x_n} = 0$
    Merge($\mathcal{R}_{sp}$,$\mathcal{R}_n$,$\hat{p}_{x_n}$)
    $Z \leftarrow Z \cup \mathcal{R}_n$
  end
  Bias correction through samples in $\mathbf{Z}$
  SpatialReservoirBuffer[q] $\leftarrow \mathcal{R}_{sp}$
end

Minimum cell size and projected cell size. The minimum cell size is used to adjust the hash grid size, and it affects performance significantly: a smaller size leads to insufficient samples for effective resampling. On the contrary, a bigger size causes more invalid samples for resampling. Thus, we set the minimum cell size considering the actual scene size. We get the bounding box of the scene and scale it with 1%. The minimum cell size is set as the minimum scaled bounding box length. The projected cell size is another user-defined parameter affecting the actual size. We set it to 10% of the image resolution, which is identical to ReSTIR GI.

5.3. Resampling

Temporal resampling. We simply keep the last frame's model, view, and projection (MVP) matrix for temporal reprojection. When the reprojection fails, the temporal resampling is canceled, but the spatial resampling is still performed.

Spatial resampling. We perform spatial sampling with a maximum iteration count, set as three, following ReSTIR GI. We generate a random path sample in the first iteration, and then choose the next samples with a fixed stride, as shown in Algorithm 2. Then, to prevent bias caused by reusing correlated samples, we obtain the spatial reservoir from previous frame's spatial reservoir buffer and retrieve neighbor reservoirs from previous frame's temporal buffer, likewise, we use the hash grid from the last frame. Moreover, reusing samples within the identical cell may lead to block artifacts. To address this issue, we introduce jittering to the position component of the hash key with a random vector proportional to the cell size, similar to previous work [BJW21, Boi21].

Reservoir storage and memory cost. Multiple-bounce reservoirs bring huge storage overhead. To avoid the enormous storage, we store the reservoirs in a packed form: using half-precision float for the outgoing radiance; compressing the position and normal in a single *float4* vector; compressing the sample count N and sample age in a four-byte unsigned integer. As a result, only 64 bytes are required for each reservoir. With a twice image-sized initial sample buffer, a double-buffering temporal reservoir buffer, a double-buffering spatial reservoir buffer, and a hash structure for the current and last frame. At a resolution of 1920×1080, the total memory cost is 933 MB, where reservoirs cost about 759 MB, and the hash structure costs 174 MB.

6. Result

We have implemented our method inside *Falcor* [KCK*22]. We compare against path tracing (PT), ReSTIR GI [OLK*21], ReSTIR PT [LKB*22], and a converged PT as references. For all these methods, we enable NEE and MIS. Additionally, ReSTIR DI is used for direct illumination. We use the implementation of the unbiased version provided by Lin et al. [LKB*22] for ReSTIR GI and ReSTIR PT. All timings in this section are measured on a GeForce RTX 3080 Laptop GPU. The time cost of our method includes all the passes, where the resampling pass costs more than 70% of the total time. We use MSE to measure the difference between each method and the ground truth. The resolutions for all the results are set as 1920 × 1080 resolution.

6.1. Comparison against previous work

In this section, we compare our results against the previous work with equal time or equal spp.

In Figure 5, we compare path tracing, ReSTIR GI and our method with roughly equal time on various scenes ranging from small indoor scenes to large open scenes with complex lighting and geometries. By comparison, both our method and ReSTIR GI provide much less noise than path tracing, while our results are less noisy than ReSTIR GI. Our approach provides a reduction ranging from 2.13× to 14.71× compared to path tracing, which demonstrates a 16.6%∼33.1% improvement over ReSTIR GI with only an extra 2 ∼ 4 ms time cost (4.41% ∼ 8.36%). Thanks to the world-space hash grid and more path sample reuse, our method shows considerable improvements over the screen-space approach [OLK*21] on areas at distance or with large normal or depth variation.

In Figure 6, we compare ReSTIR GI, our method, and ReSTIR PT for several scenes with high glossy materials with equal sampling rates. Both our method and ReSTIR PT show less noise than ReSTIR GI, which fails for glossy materials. The main reason is that, when choosing to perform resampling at the visible point on the path, the screen-space buffer fails to guarantee spatial coherence in path space for those paths starting from a highly glossy vertex: even if two glossy vertices have the same lobe shape, their successive vertices on the path may diverge significantly due to sampling and scene complexity. On the other hand, our world-space

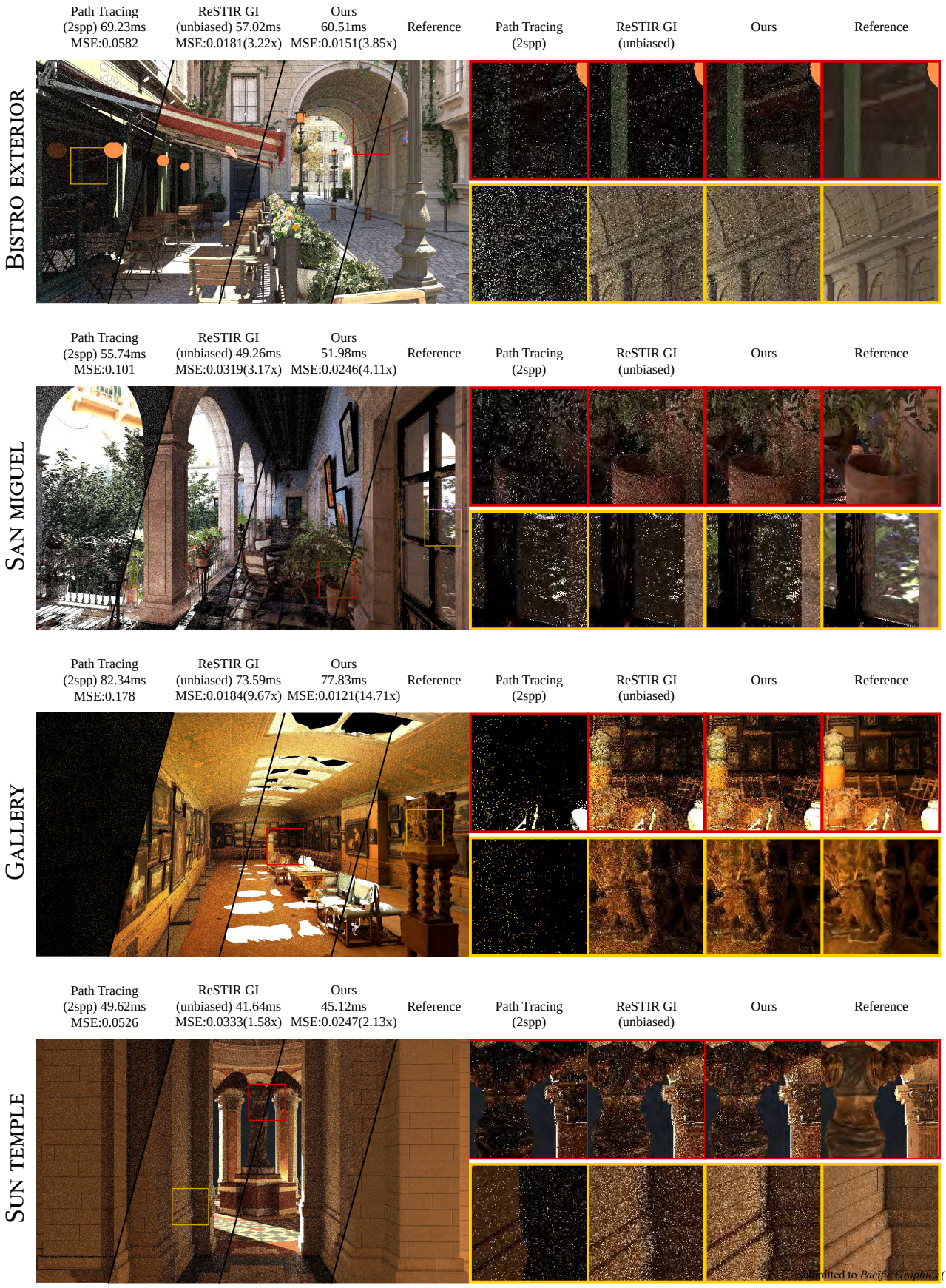


Figure 5: *Comparison among path tracing, ReSTIR GI and our method with roughly equal time on various kinds of scenes. The sampling rate is set as 2 for path tracing, and set as 1 for our method and ReSTIR GI to ensure roughly equal time. Our method exhibits a 16.6% to 33.1% improvement in terms of MSE compared to ReSTIR GI at a low extra cost.*

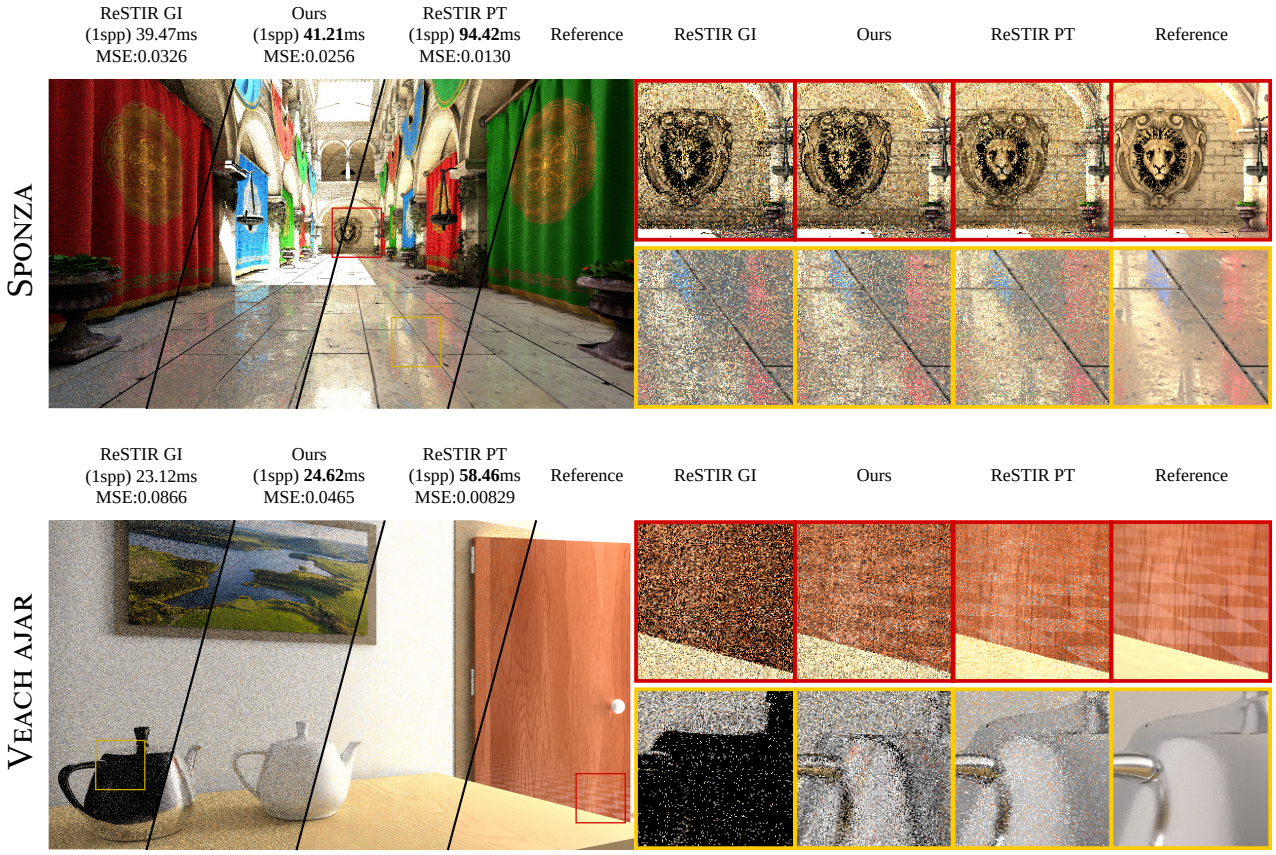


Figure 6: *Equal sampling rate comparison (1 spp) among ReSTIR GI, ReSTIR PT, and our method on the* SPONZA *and* VEACH AJAR *scenes. Our method is able to handle glossy materials, which is difficult for ReSTIR GI. Compared to ReSTIR PT, their method can provide the highest quality, but it is almost three times slower than our method and becomes difficult for real-time rendering. Our method trades the lightweight or high-performance with less accuracy.*

structure caches samples with similar normal and position in one cell, ensuring path space coherence for further bounces. This makes reconnection shift mapping still capable of achieving its effectiveness. Regarding the time cost, our method only shows a minor time cost, compared to ReSTIR GI, while ReSTIR PT introduces a heavy time overhead (almost three times slower than ours). Therefore, our method can outperform ReSTIR GI on both diffuse and glossy materials with a little time cost. Compared to ReSTIR PT, our method is much lightweight, with some quality degradation.

We also compare our method (biased) with biased ReSTIR GI and biased ReSTIR PT in Figure 7. For ReSTIR GI and our method, we disable the bias correction strategy. For ReSTIR PT, we use the constant MIS weight (a biased MIS weight). Our method produces less noisy results than ReSTIR GI, with only a 7% time overhead. Compared to ReSTIR PT, our method is twice as fast at one spp and serves as a more practical choice for real-time rendering, at the cost of lower quality.

6.2. Converged results and parameter analysis

We analyze our method's convergence via comparisons against the ground truth and then analyze the effects of the main parameters.

Converged results. In Figure 8, we compare our converged results with ground truth on several scenes. We find that our converged results are identical to the ground truth. As discussed in generalized RIS [LKB*22], resampled importance sampling requires generalized resampling MIS; otherwise, even an unbiased algorithm might not converge to the ground-truth images. In the same way as ReSTIR GI, our method also employs additional measures to ensure convergence, including M-capping and sample validation (ways to restrict the sample's lifetime in reservoirs) [OLK*21].

Parameter analysis. Figure 9 shows the rendering error (MSE) as a function of the minimum cell size on several scenes. We also provide a visual comparison of varying minimum cell sizes in Figure 10. We find that the scale factor around 1% of scenes shows the best convergence: a smaller size makes cells lack sample availability and reduces resampling quality consequently; a bigger

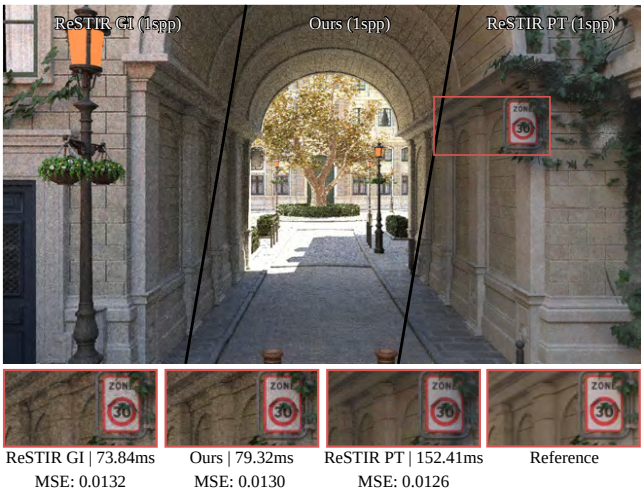


Figure 7: *Comparison among our method (biased), biased ReSTIR GI and biased ReSTIR PT with equal sampling rate (1 spp) on the* BISTRO *scene.*

size shows higher reservoir capacity, but introduces more samples which contribute little to refine the final weight. Thus, we choose 1% of the scene bounding box as the minimum cell size.

We validate our temporal reuse in Figure 11, by comparing spatiotemporal and spatial reuse only. By comparison, the temporal reuse can greatly increase the image quality.

6.3. Limitations and Discussion

We have identified several limitations of our approach as follows.

Hash grid construction time cost. Despite hash grid construction taking only $2 \sim 4$ ms, we need to rebuild it at each frame. Actually, it is not necessary to rebuild the hash grid at such a high frequency. Furthermore, rebuilding may eliminate samples capturing abundant lighting. Thus, updating the hash grid rather than reconstructing should be considered for future work.

Screen-space temporal resampling. We use a screen-space temporal resampling, due to the coarse representation of the hash grid. A finer representation of the hash grid could address this issue but will affect the spatial reuse quality. A potential solution is a multi-scale hash grid representation: the fine scale for temporal reuse, and the coarse scale for spatial reuse, which is left for future work.

More complex light transport. Our approach is ineffective in resampling caustic paths due to the reconnection shift mapping. In this case, caustic paths may be reconnected to other vertices in an identical cell, leading to artifacts, as shown in Figure 12. Besides, our approach resamples the subsequent vertex after the glossy surfaces rather than resampling at glossy vertices directly.

Dynamic lighting. Like ReSTIR PT, our temporal resample is amortized among 30 frames without any extra strategies to handle the dynamic lighting. When the light sources in the scene have drastic changes, the quality of temporal resampling degrades. ReSTIR GI [OLK*21] uses an SVGF-liked sample validation mechanism, which can be employed to improve the resampling quality further.

7. Conclusion

We have presented a practical world-space path resampling approach, which allows resampling multiple bounces path samples compared to the screen-space resampling approaches. Together with a normal-aware hash grid construction, our method shows noticeable improvements in areas with complex geometries or large normal variations, as well as for glossy indirect lighting. Our method outperforms ReSTIR GI up to 41.9% in terms of MSE while only requiring a minor time cost for each frame.

Acknowledgments

We thank the reviewers for the valuable comments. This work has been partially supported by the National Key R&D Program of China under grant No. 2022ZD0116305, National Natural Science Foundation of China under grant No. 62172220.

References

[BFK18] BINDER N., FRICKE S., KELLER A.: Fast path space filtering by jittered spatial hashing. In *ACM SIGGRAPH 2018 Talks* (New York, NY, USA, 2018), SIGGRAPH '18, Association for Computing Machinery. URL: https://doi.org/10.1145/3214745.3214806, doi:10.1145/3214745.3214806. 2

[BJW21] BOKSANSKY J., JUKARAINEN P., WYMAN C.: *Rendering Many Lights with Grid-Based Reservoirs*. Apress, Berkeley, CA, 2021, pp. 351–365. URL: https://doi.org/10.1007/978-1-4842-7185-8_23, doi:10.1007/978-1-4842-7185-8_23. 2, 7

[Boi21] BOISSÉ G.: World-space spatiotemporal reservoir reuse for ray-traced global illumination. In *SIGGRAPH Asia 2021 Technical Communications* (New York, NY, USA, 2021), SA '21 Technical Communications, Association for Computing Machinery. URL: https://doi.org/10.1145/3478512.3488613, doi:10.1145/3478512.3488613. 2, 4, 7

[BPE17] BAUSZAT P., PETITJEAN V., EISEMANN E.: Gradient-domain path reusing. *ACM Trans. Graph. 36*, 6 (nov 2017). URL: https://doi.org/10.1145/3130800.3130886, doi:10.1145/3130800.3130886. 2

[BSH02] BEKAERT P., SBERT M., HALTON J.: Accelerating path tracing by re-using paths. In *Proceedings of the 13th Eurographics Workshop on Rendering* (Goslar, DEU, 2002), EGRW '02, Eurographics Association, p. 125–134. 2

[BWP*20] BITTERLI B., WYMAN C., PHARR M., SHIRLEY P., LEFOHN A., JAROSZ W.: Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Trans. Graph. 39*, 4 (aug 2020). URL: https://doi.org/10.1145/3386569.3392481, doi:10.1145/3386569.3392481. 2, 3

[DGJ*20] DIOLATZIS S., GRUSON A., JAKOB W., NOWROUZEZAHRAI D., DRETTAKIS G.: Practical product path guiding using linearly transformed cosines. *Computer Graphics Forum 39*, 4 (2020), 23–33. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14051, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14051, doi:https://doi.org/10.1111/cgf.14051. 2

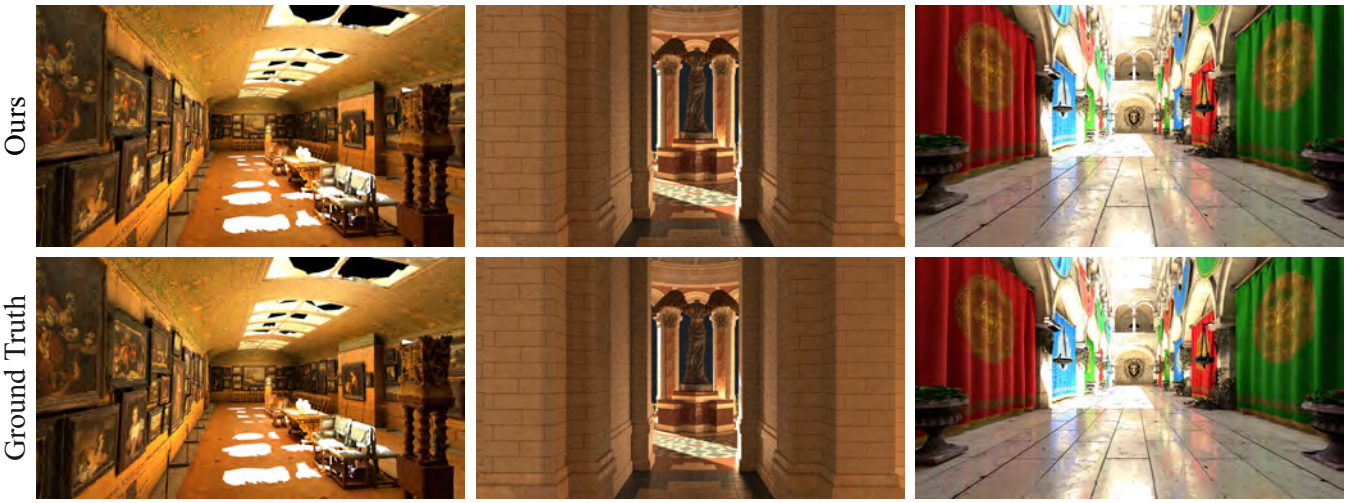


Figure 8: *Comparison between our method's converged results (accumulated with over 20,000 frames) and ground truth over several scenes. Our converged results are identical to the references.*

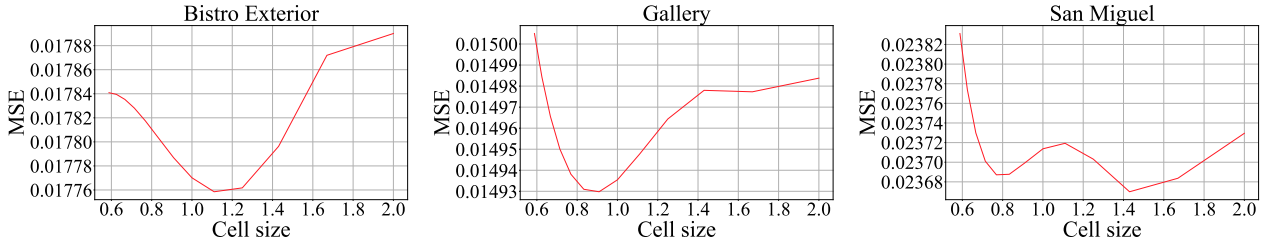
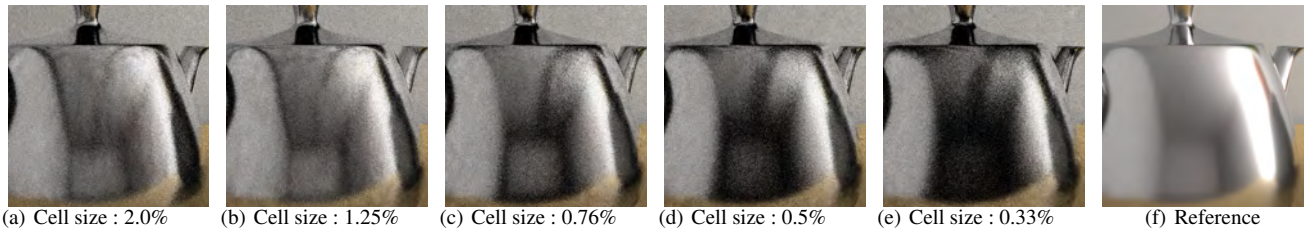


Figure 9: *Mean square error of our method in terms of the minimum cell size (range from 0.67% to 2.00% scaled scene bounding box size) in a variety of scenes. The error is computed for indirect illumination only. We choose 1.0% scaled bounding box size for the minimum cell size in practice.*



(a) Cell size : 2.0% MSE: 0.0603
(b) Cell size : 1.25% MSE: 0.0549
(c) Cell size : 0.76% MSE: 0.0685
(d) Cell size : 0.5% MSE: 0.0764
(e) Cell size : 0.33% MSE: 0.0896
(f) Reference

Figure 10: *Rendered results with different minimum cell sizes on* VEACH JAR *scene. The resampling quality declines as the minimum cell size is reduced. This occurs due to a lack of effective samples.*

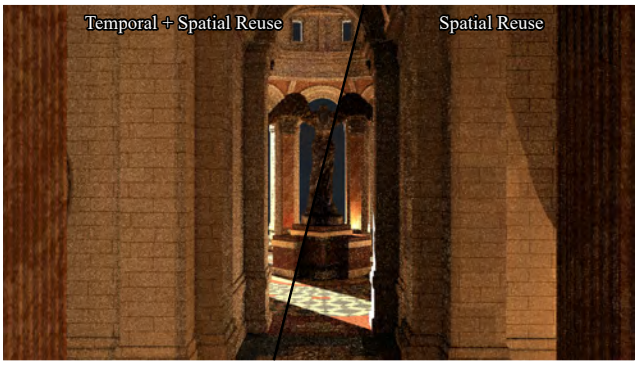
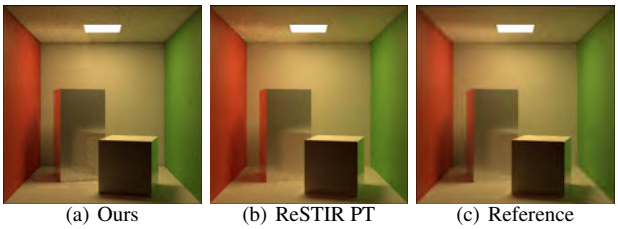


Figure 11: *Comparison between spatiotemporal reuse (left) and spatial reuse only (right) on the* SUN TEMPLE *scene.*



(a) Ours (b) ReSTIR PT (c) Reference

Figure 12: *Failure case. Our method shows artifacts for caustic paths.*

[DHC*21] DENG X., HAŠAN M., CARR N., XU Z., MARSCHNER S.: Path graphs: Iterative path space filtering. *ACM Trans. Graph. 40*, 6 (dec 2021). URL: https://doi.org/10.1145/3478513.3480547, doi:10.1145/3478513.3480547. 2

[EHNB18] EMMETT K., HENRY M., NICK S., BRANDON B.: Nvidia turing architecture in-depth, Sept 2018. URL: https://developer.nvidia.com/blog/nvidia-turing-architecture-in-depth/. 2

[Gau20] GAUTRON P.: Real-time ray-traced ambient occlusion of complex scenes using spatial hashing. In *ACM SIGGRAPH 2020 Talks* (New York, NY, USA, 2020), SIGGRAPH '20, Association for Computing Machinery. URL: https://doi.org/10.1145/3388767.3407375, doi:10.1145/3388767.3407375. 2

[HEV*16] HERHOLZ S., ELEK O., VORBA J., LENSCH H., KŘIVÁNEK J.: Product importance sampling for light transport path guiding. *Computer Graphics Forum 35*, 4 (2016), 67–77. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12950, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12950, doi:https://doi.org/10.1111/cgf.12950. 2

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, Association for Computing Machinery, p. 143–150. URL: https://doi.org/10.1145/15922.15902, doi:10.1145/15922.15902. 3

[KCK*22] KALLWEIT S., CLARBERG P., KOLB C., DAVIDOVIČ T., YAO K.-H., FOLEY T., HE Y., WU L., CHEN L., AKENINE-MÖLLER T., WYMAN C., CRASSIN C., BENTY N.: The Falcor rendering framework, 8 2022. URL: https://github.com/NVIDIAGameWorks/Falcor. 7

[KDB14] KELLER A., DAHM K., BINDER N.: Path space filtering. In *ACM SIGGRAPH 2014 Talks* (New York, NY, USA, 2014), SIGGRAPH '14, Association for Computing Machinery. URL: https://doi.org/10.1145/2614106.2614149, doi:10.1145/2614106.2614149. 2

[KMA*15] KETTUNEN M., MANZI M., AITTALA M., LEHTINEN J., DURAND F., ZWICKER M.: Gradient-domain path tracing. *ACM Trans. Graph. 34*, 4 (jul 2015). URL: https://doi.org/10.1145/2766997, doi:10.1145/2766997. 2

[LC09] LIANG F., CHEON S.: Monte carlo dynamically weighted importance sampling for spatial models with intractable normalizing constants. *Journal of Physics: Conference Series 197*, 1 (dec 2009), 012004. URL: https://dx.doi.org/10.1088/1742-6596/197/1/012004, doi:10.1088/1742-6596/197/1/012004. 2, 3

[LKB*22] LIN D., KETTUNEN M., BITTERLI B., PANTALEONI J., YUKSEL C., WYMAN C.: Generalized resampled importance sampling: Foundations of restir. *ACM Trans. Graph. 41*, 4 (jul 2022). URL: https://doi.org/10.1145/3528223.3530158, doi:10.1145/3528223.3530158. 2, 3, 7, 9

[MGN17] MÜLLER T., GROSS M., NOVÁK J.: Practical path guiding for efficient light-transport simulation. *Computer Graphics Forum 36*, 4 (2017), 91–100. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13227, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13227, doi:https://doi.org/10.1111/cgf.13227. 2

[MGNM19] MAJERCIK Z., GUERTIN J.-P., NOWROUZEZAHRAI D., MCGUIRE M.: Dynamic diffuse global illumination with ray-traced irradiance fields. *Journal of Computer Graphics Techniques (JCGT) 8*, 2 (June 2019), 1–30. URL: http://jcgt.org/published/0008/02/01/. 2

[MMK*21] MAJERCIK Z., MUELLER T., KELLER A., NOWROUZEZAHRAI D., MCGUIRE M.: Dynamic diffuse global illumination resampling. In *ACM SIGGRAPH 2021 Talks* (New York, NY, USA, 2021), SIGGRAPH '21, Association for Computing Machinery. URL: https://doi.org/10.1145/3450623.3464635, doi:10.1145/3450623.3464635. 2

[OLK*21] OUYANG Y., LIU S., KETTUNEN M., PHARR M., PANTALEONI J.: Restir gi: Path resampling for real-time path tracing. *Computer Graphics Forum 40*, 8 (2021), 17–29. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14378, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14378, doi:https://doi.org/10.1111/cgf.14378. 1, 2, 3, 5, 7, 9, 10

[Rez21] REZA M.: *Efficient Sample Reusage in Path Space for Real-Time Light Transport*. Master's thesis, Norwegain University of Science and Technology, June 2021. https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2976461. 2

[SZR*15] SEN P., ZWICKER M., ROUSSELLE F., YOON S.-E., KALANTARI N. K.: Denoising your monte carlo renders: Recent advances in image-space adaptive sampling and reconstruction. In *ACM SIGGRAPH 2015 Courses* (New York, NY, USA, 2015), SIGGRAPH '15, Association for Computing Machinery. URL: https://doi.org/10.1145/2776880.2792740, doi:10.1145/2776880.2792740. 2

[Tak20] TAKAHIRO H.: Hardware-accelerated ray tracing in amd radeon^{tm} prorender 2.0, Nov 2020. URL: https://gpuopen.com/learn/radeon-prorender-2-0/. 2

[TCE05] TALBOT J., CLINE D., EGBERT P.: Importance Resampling for Global Illumination. In *Eurographics Symposium on Rendering (2005)* (2005), Bala K., Dutre P., (Eds.), The Eurographics Association. doi:10.2312/EGWR/EGSR05/139-146. 2, 3