Real-Time Level-of-Detail Rendering with ReSTIR

YU-CHEN WANG, University of California, Irvine, USA
MARKUS KETTUNEN, NVIDIA, Finland
DAQI LIN, NVIDIA, USA
CHRIS WYMAN, NVIDIA, USA
LIFAN WU, NVIDIA, USA
SHUANG ZHAO, University of Illinois Urbana-Champaign, USA

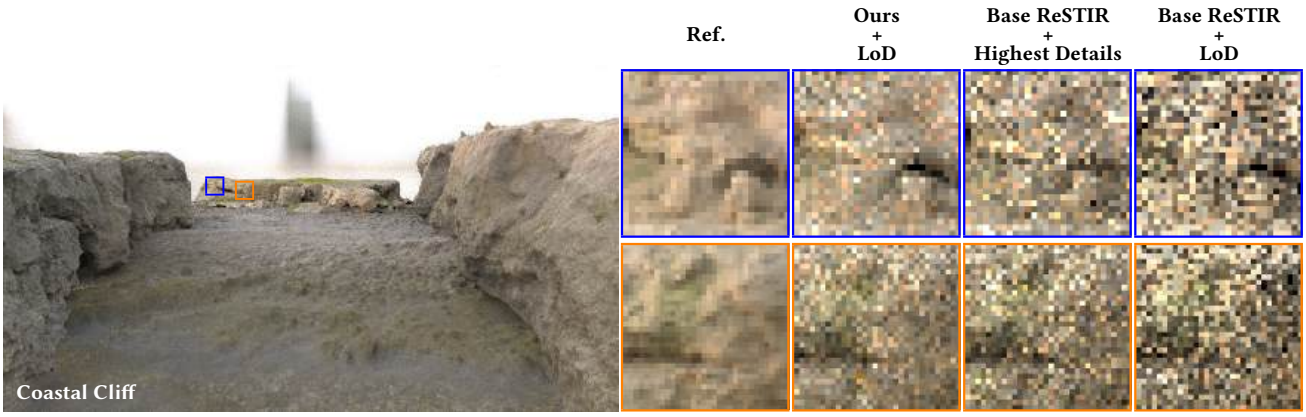


Fig. 1. Our method enables robust sample reuse across dynamic geometry level-of-detail (LoD) changes. We compare our approach on Coastal Cliff against state-of-the-art ReSTIR path tracing [Liu et al. 2025] applied to both dynamic LoD geometry and fixed highest-detail geometry in equal time (50ms). The camera moves forward toward the distant cliff in the center. Prefiltering the geometry by choosing the proper LoD reduces high-frequency variation and normally lowers noise, but when the cliff switches LoD, the topology change stops prior ReSTIR methods from reusing temporally, which results in *increased* noise (RelMSE: Baseline+Highest detail 0.104, Baseline+LoD 0.126). Our method maintains valid reuse across LoD switches and reduces noise (RelMSE: 0.092).

Rendering complex scenes in real time remains challenging due to strict performance constraints. Geometric level of detail (LoD) is widely used to reduce cost by replacing high-frequency geometry with prefiltered representations. ReSTIR significantly improves real-time rendering quality by reusing samples across space and time; however, its effectiveness degrades in the presence of complex geometry, where high-frequency detail reduces reuse efficiency. Moreover, prior ReSTIR methods require the same mesh topology across frames, causing sample reuse to break when switching LoD.

In this paper, we introduce a surface point mapping that enables sample reuse across frames containing meshes with different topologies. Building on our method, we enable LoD-aware ReSTIR by maintaining valid spatiotemporal reuse under geometry LoD changes. Our approach restores reuse efficiency in LoD scenes and significantly improves rendering quality compared to prior ReSTIR methods.

CCS Concepts: • **Computing methodologies** → **Ray tracing**.

Authors' Contact Information: Yu-Chen Wang, yuchew23@uci.edu, University of California, Irvine, USA; Markus Kettunen, mkettunen@nvidia.com, NVIDIA, Finland; Daqi Lin, daqil@nvidia.com, NVIDIA, USA; Chris Wyman, chris.wyman@acm.org, NVIDIA, USA; Lifan Wu, lifanw@nvidia.com, NVIDIA, USA; Shuang Zhao, shzhao@illinois.edu, University of Illinois Urbana-Champaign, USA.

SIGGRAPH Conference Papers '26, Los Angeles, CA, USA

ACM ISBN 979-8-4007-2554-8/2026/07
https://doi.org/10.1145/3799902.3811100

Additional Key Words and Phrases: real-time rendering, ReSTIR, level of detail, light transport simulation

ACM Reference Format:
Yu-Chen Wang, Markus Kettunen, Daqi Lin, Chris Wyman, Lifan Wu, and Shuang Zhao. 2026. Real-Time Level-of-Detail Rendering with ReSTIR. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '26), July 19–23, 2026, Los Angeles, CA, USA*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3799902.3811100

1 Introduction

Real-time rendering has many important applications including video games, virtual reality, and 3D visualization. Recent GPU and algorithmic improvements have enabled real-time complex light transport including soft shadows and global illumination. Today's increasingly complex scenes feature high-frequency geometry; this gives rise to the *multi-scale rendering* problem, as the required *level-of-detail* (LoD) varies by distance to the camera. Without dynamic LoD to appropriately prefilter, rendering distant models with too much detail leads to increased noise and aliasing; see Figure 2.

Recently, Bitterli et al. [2020] introduce *reservoir-based spatiotemporal importance resampling* (ReSTIR), which greatly improves real-time rendering quality by reusing samples across time and space; but overly high-frequency geometry can then lead to rich sub-pixel

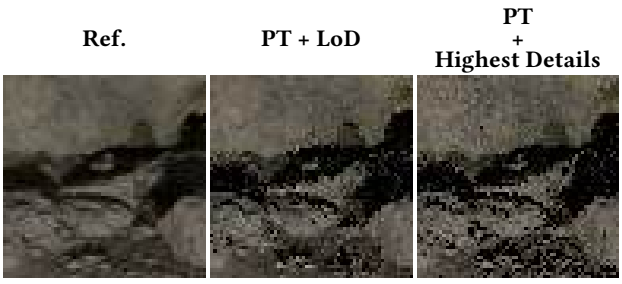


Fig. 2. A rock rendered with path tracing using 1 spp. Using a prefiltered mesh LoD allows path tracing to converge better.

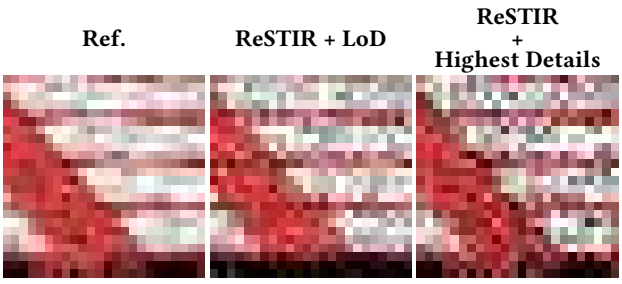


Fig. 3. A piece of cloth rendered with ReSTIR. The base geometry is made from a 1K displacement map. High-frequency details make spatiotemporal reuse challenging, slowing convergence. Using prefiltered levels of detail, ReSTIR is better able to reuse samples, producing a cleaner result.

detail, which makes spatiotemporal sample reuse challenging and increases noise. Choosing the right level of detail for the pixel footprint cuts these excess high frequencies and greatly improves rendering quality by ReSTIR path tracing; see Figure 3.

Unfortunately, current ReSTIR methods [Lin et al. 2022; Liu et al. 2025] do not support dynamic geometric LoD: reusing samples across different LoD levels is an unsolved problem. In ReSTIR, light paths are reused across different pixels and frames by *shift mappings*, but existing shift mappings require the same mesh topology across frames. When the LoD level of an object switches, the mesh topology changes, and the shift mapping fails. ReSTIR loses temporal history and restarts from zero with independent sampling.

In this paper, we enable ReSTIR to render scenes with dynamic geometric LoD by defining reuse across the mesh topology changes induced by geometric LoD switching. We do this by constructing a partial bijection[1] between the different-LoD surfaces by passing through the (non-invertible) UV map. This significantly improves rendering quality. Concretely, we make the following contributions:

(1) Extend shift mappings to handle topology-changing geometries via a new vertex mapping between geometries with different topologies.
(2) Introduce LoD-aware ReSTIR path tracing by adapting our vertex mapping to ReSTIR (Section 4.4).

We demonstrate the effectiveness of our method by comparing it with the state-of-the-art techniques in Figure 7 and Figure 8. We also provide several ablation studies to evaluate the benefits of our method in Figure 5 and Figure 6.

2 Related Work

2.1 Level-of-Detail Rendering

Level-of-detail is a common feature in many modern rendering systems. Garland and Heckbert [1997] and Kobbelt et al. [1998] simplify meshes while keeping their silhouettes and shapes. To render more complex geometric details, Dupuy et al. [2013]; Olano and Baker [2010]; Wu et al. [2019] simplify geometries and adjust materials to preserve appearance. Bako et al. [2023] leverage deep neural networks to prefilter large complex scenes.

LoD methods can also be specialized for specific appearances. For example, Zhu et al. [2024] introduce a LoD appearance model for cloth rendering and Huang et al. [2025] develop a LoD rendering framework for hair. These methods are largely orthogonal to ours; we merely assume an object has a fixed geometry during a single frame, for all rays, and that all levels of detail are UV mapped consistently.

2.2 Path Reuse and Shift Mappings

Bekaert et al. [2002] reuse paths within a block of pixels by connecting each path's primary hit to each path's secondary hit. In the context of Metropolis Light Transport [Veach and Guibas 1997] in the gradient-domain, Lehtinen et al. [2013] introduce shift mappings to deterministically map paths between pixels, with Jacobian determinants compensating for the density change. They formalize the reconnection shift and the manifold perturbation shift. Kettunen et al. [2015] introduce a shift mapping for their gradient-domain path tracing that, based on copying half-vectors, works with specular surfaces. Manzi et al. [2016] shift paths between frames for temporal gradients by replaying random numbers. Bauszat et al. [2017] propose the first primal-domain path reuse algorithm based on shift mappings. Hua et al. [2019] later join random replay with the reconnection shift for spatial shifts, and Lin et al.'s [2022] GPU-optimized hybrid shift further improves it. We build on ReSTIR, which builds on shift mappings applied in the primal-domain.

2.3 ReSTIR

Talbot et al. [2005] introduce resampled importance sampling (RIS), which merges a set of candidate samples into one or more better-distributed samples by resampling proportionally to a target function. Bitterli et al. [2020] present ReSTIR by leveraging RIS with weighted reservoir sampling [Chao 1982] to reuse light samples across pixels and frames for real-time direct illumination. Lin et al. [2022] present *generalized resampled importance sampling* (GRIS), which gives a mathematical foundation to ReSTIR and enables reuse of *light paths* across pixels and frames with shift mappings. Their method is the basis for modern ReSTIR-based path tracing, which we build on.

[1] A partial bijection from sets A and B is a bijective function $f : A' \to B'$, where $A' \subseteq A$ and $B' \subseteq B$.

Zhang et al. [2024] improve temporal reuse in pixels with high-frequency details by using fractional motion vectors at pixel centers. Liu et al. [2025] further robustify temporal reuse by *splatting* prior-frame paths onto current-frame reservoirs. We build on Liu et al.'s work by extending it to handle geometry LoD switches.

3 Preliminaries

In this section, we review some basic concepts of ReSTIR. We refer the reader to the course notes by Wyman et al. [2023] for details.

3.1 Resampled Importance Sampling

Given candidates $X_1, \ldots, X_M$ from a single domain Ω (e.g., a pixel) and their PDFs $p_1, \ldots, p_M$, resampled importance sampling (RIS) [Talbot et al. 2005] aims to provide a sample Y distributed approximately proportionally to a given target function $\hat{p}$ by weighted resampling. It first picks index s proportionally to *resampling weights*

$$w_i = m_i(X_i)\hat{p}(X_i)\frac{1}{p_i(X_i)}, \tag{1}$$

where m_i are *resampling MIS weights* that sum to 1 for each fixed x. The output $Y = X_s$ of RIS allows unbiased estimation of an integral $\int_\Omega f(x)\ \mathrm{d}x$ with the formula $f(Y)\frac{1}{\hat{p}(Y)}\sum_{i=1}^{M} w_i$. Choosing $\hat{p} = f$ yields zero-variance integration in the limit $M \to \infty$.

While ReSTIR is iterated application of resampling, the output cannot be fed to another standard RIS, since the PDF p_Y of the output is an intractable multidimensional integral. RIS must be generalized.

3.2 Generalized Resampled Importance Sampling

Since the post-resampling PDF is intractable, Lin et al. [2022] replace $1/p_X(X)$ with random variable W_X with expectation $\mathbb{E}[W_X \mid X] = 1/p_X(X)$, an *unbiased contribution weight* (UCW). Given a formula for W_X, this allows unbiased integration with $f(X)W_X$. This makes the single-domain resampling weights of generalized resampled importance sampling (GRIS) [Lin et al. 2022] $m_i(X_i)\hat{p}(X_i)W_{X_i}$.

GRIS allows candidates to be in different domains ($X_i \in \Omega_i$), and each candidate X_i is *shift mapped* into $T_i(X_i)$ in the target domain Ω. The resampling weights are evaluated in the target domain and become

$$w_i = m_i(T_i(X_i))\hat{p}(T_i(X_i))W_{X_i}\left|\frac{\partial T_i}{\partial X_i}\right|, \tag{2}$$

where the Jacobian determinant compensates for the density change in the shift mapping. Index s is sampled proportionally to the w_i, the output is $Y = T_s(X_s)$, and it is given weight

$$W_Y = \frac{1}{\hat{p}(Y)}\sum_{i=1}^{M} w_i, \tag{3}$$

which is a UCW and allows both integration with $f(X)W_X$ and passing Y as an input to another GRIS resampling, enabling ReSTIR.

3.3 ReSTIR

ReSTIR PT [Lin et al. 2022] leverages chained GRIS to solve the rendering equation [Kajiya 1986], roughly in the sequence of *initial sampling*, *temporal reuse*, and *spatial reuse*. Each pass *merges* its inputs into one path and its UCW via GRIS resampling, and stores the result in the current pixel's *reservoir*. The reservoir stores the path X, the UCW W_X, and a confidence weight determining the sample's relative weight in resampling MIS. The details vary by variant, but are usually roughly as follows:

In initial sampling, each pixel draws one or more iid *initial candidates* and merges them into the *initial sample* with standard RIS.

The temporal reuse pass merges the initial sample with the sample from the prior-frame pixel's reservoir, found by following a backward motion vector.

The spatial reuse pass merges the current pixel's temporal reuse result with those of often 1–4 random neighboring pixels from e.g., a disk with a 30-pixel radius, potentially discarding the most dissimilar based on normals and depth.

Finally, the result of spatial reuse is evaluated as $f(X)W_X$, giving the pixel color estimate. All frames repeat the same process. The confidence weights are typically summed at each GRIS merge and clamped to a confidence cap.

3.4 Shift Mappings

In GRIS, paths are reused between pixels. Taking a path from one pixel and using it for another requires modifying the path's vertices. This defines a *shift mapping* T_i from pixel i's domain Ω_i to the target pixel's domain Ω. Shift mappings are partial bijections, i.e., they need not be defined for all of Ω_i, but they are always invertible: if $T_i(\bar{\boldsymbol{x}}) = \bar{\boldsymbol{y}}$, then $T_i^{-1}(\bar{\boldsymbol{y}}) = \bar{\boldsymbol{x}}$; failing invertibility is a common source of bias. The input of a shift mapping is called the *base path* and the output is called the *offset path* [Manzi et al. 2014].

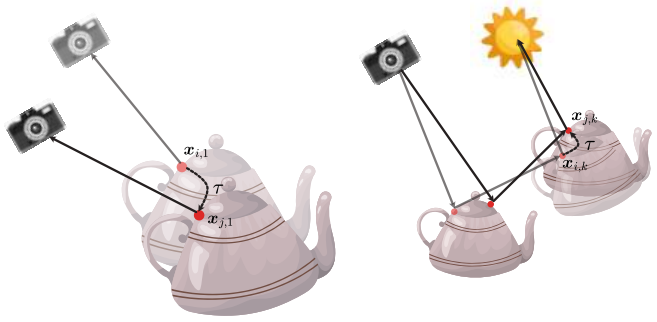
The *reconnection shift* is one of the simplest shift mappings, and works well for rough scenes. Given a primary hit y_1 in the target pixel, it maps the base path $\bar{\boldsymbol{x}} = [\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$, where $\boldsymbol{x}_0$ is the camera vertex, $\boldsymbol{x}_1$ is the primary hit, and $\boldsymbol{x}_2$ is the secondary hit, into $T_i(\bar{\boldsymbol{x}}) = [\boldsymbol{y}_0, \boldsymbol{y}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$: it connects the base path to the offset path at the first possible vertex.

Lin et al.'s [2022] GPU-optimized *hybrid shift* also reconnects at the earliest vertex, but their definition of earliest postpones the reconnection until a safe reconnection vertex is found. They use random replay until reconnection.

This is the case for spatial reuse, where the world space is shared by the base and offset paths. Temporal reuse requires mapping each vertex x_k to the offset frame. Traditionally, surface points are mapped to the next frame by looking up the triangle index and the barycentric coordinates of the base-frame surface point, and finding the point with the same triangle index and barycentric coordinates in the target frame. We call this *vertex mapping* and denote it by τ; see Figure 4.

Vertex mapping based on triangle IDs and barycentrics is not possible when mesh topology changes between frames, e.g., due to a LoD switch: LoD switches force reuse failures and reset history, yielding increased noise. This holds not only for the reconnection and hybrid shifts, but also for motion vector calculation in temporal reuse [Lin et al. 2022; Zhang et al. 2024], primary hit reconnection for depth of field [Zhang et al. 2024], and primary reprojection in reservoir splatting [Liu et al. 2025] with and without motion blur.

In Section 4 we generalize vertex mapping to changing mesh topologies by relying on consistent but not necessarily invertible mappings of surface points to a UV texture space. This enables temporal reuse for changing mesh topologies.



Primary Reprojection Path Reconnection

Fig. 4. The vertex mapping τ maps vertices between frames and is a critical component of temporal reuse. The current method of mapping by triangle id and barycentric coordinates fails and causes excessive noise at LoD changes. We fix this by building vertex correspondences based on consistent but not necessarily invertible UV mappings.

4 Our Method

We now present our method to support dynamic levels of detail (LoD) in ReSTIR; we do this by defining a new (partially) bijective vertex mapping τ between different geometries. We will no longer assume a consistent mesh topology, but rather assume consistently UV-mapped levels of detail. In Section 4.1, we introduce vertex mapping under the simplifying assumption of non-overlapping UV parametrizations. In Section 4.2, we extend the method to handle general UV parametrizations with overlaps and mirroring. Section 4.3 derives the Jacobian of our vertex mapping, which is required to compute the Jacobian of the shift mappings when adapting our method. Finally, incorporating our vertex mapping into shift mappings, we present a ReSTIR path tracing algorithm that supports dynamic LoD changes in Section 4.4.

4.1 Non-Overlapping UV Parametrizations

Let Ω_1 and Ω_2 be the base and offset path spaces with objects potentially represented in different levels of detail. The vertex mapping $\tau : \Omega_1 \to \Omega_2$ and its inverse τ^{-1} bijectively map vertices between associated surfaces $\mathcal{S}_1 \subset \Omega_1$ and $\mathcal{S}_2 \subset \Omega_2$, where $\mathcal{S}_1$ and $\mathcal{S}_2$ may have different topologies. For completeness, we allow τ and τ^{-1} to be partial bijections, i.e., not all surface points need to have an image in the other surface; this may cause the shift mapping to fail in rare occasions, which is often unavoidable.

Given the associated, UV mapped surfaces $\mathcal{S}_1$ and $\mathcal{S}_2$ with potentially different topologies, we construct the surface point correspondence using texture coordinates as a shared canonical parametrization. Let the UV mappings associated with $\mathcal{S}_1$ and $\mathcal{S}_2$ be $\Phi_1 : \mathcal{S}_1 \to \mathbb{R}^2$ and $\Phi_2 : \mathcal{S}_2 \to \mathbb{R}^2$, respectively. For any point $\boldsymbol{x} \in \mathcal{S}_1$, in the simple case that Φ_1 and Φ_2 are one-to-one (injective), i.e., the UV map is non-overlapping, we simply want $\boldsymbol{x}$ and $\tau(\boldsymbol{x})$ to have the same UV coordinates. This gives $\tau(\boldsymbol{x}) = \Phi_2^{-1}(\Phi_1(\boldsymbol{x}))$.

4.2 General UV Parametrizations

In the general case where both texture mappings can be many-to-one, we consider all points (from both surfaces) sharing the texture coordinates v:

$$\mathcal{X}_{1,v} := \{\boldsymbol{x} \in \mathcal{S}_1 \mid \Phi_1(\boldsymbol{x}) = v\}, \tag{4}$$

$$\mathcal{X}_{2,v} := \{\boldsymbol{y} \in \mathcal{S}_2 \mid \Phi_2(\boldsymbol{y}) = v\}. \tag{5}$$

To ensure τ is bijective, we seek an undirected *matching* $\mathcal{M} \subseteq \mathcal{X}_{1,v} \times \mathcal{X}_{2,v}$ between $\mathcal{X}_{1,v}$ and $\mathcal{X}_{2,v}$. In practice, we use the simple greedy Algorithm 1, which iteratively finds the closest pair of unmatched points $\boldsymbol{x}_1^* \in \mathcal{X}_{1,v}$ and $\boldsymbol{x}_2^* \in \mathcal{X}_{2,v}$ in terms of world-space[2] distance $\|\boldsymbol{x}_1^* - \boldsymbol{x}_2^*\|_2$, and adds $(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*)$ to the matching. Lastly, we set $\tau(\boldsymbol{x}_1) = \boldsymbol{x}_2$ when $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ belongs to the matching. If no such $\boldsymbol{x}_2$ exists, we consider $\tau(\boldsymbol{x}_1)$ to be nonexistent and the shift to be undefined.

As discussed above, we need the inverse of texture mappings Φ_i^{-1} on the surface $\mathcal{S}_i$ to compute the $\mathcal{X}_{i,v}$. To avoid ambiguities caused by repeated use of the same asset in a scene, our method supports object instancing by computing the vertex mapping between LoD surfaces of the same asset instance. In our system, we build an additional bounding volume hierarchy (BVH) in UV space for each LoD object as an auxiliary data structure for UV inversion. Given an instance ID, a texture space point v and a target LoD, we traverse the BVH with hardware ray tracing to find all candidate points on the associated object surface $\mathcal{S}_i$ to construct both $\mathcal{X}_{i,v}$.

ALGORITHM 1: Match the surface points between $\mathcal{S}_{1,v}$ and $\mathcal{S}_{2,v}$.

```
MatchPoints(X_{1,v}, X_{2,v})
Input: The two sets of surface points to be matched X_{1,v}, X_{2,v}.
Output: An undirectional surface point matching M.
begin
    M ← ∅
    /* Match ends when one of the sets is empty.      */
    while X_{1,v} ≠ ∅ and X_{2,v} ≠ ∅ do
        (x_1^*, x_2^*) ← argmin_{x∈X_{1,v}, y∈X_{2,v}} ‖x − y‖_2
        M ← M ∪ {(x_1^*, x_2^*)}
        X_{1,v} ← X_{1,v} \ {x_1^*}
        X_{2,v} ← X_{2,v} \ {x_2^*}
    end
    return M;
end
```

4.3 Jacobian Determinant

Given UV coordinate $v = \Phi_1(\boldsymbol{x})$ and tangent vectors $\bar{u}_i = \partial_1 \Phi_i^{-1}$ and $\bar{v}_i = \partial_2 \Phi_i^{-1}$ at v, i.e., world-space position change per unit of first and second coordinate change in the UV space, we have[3]

$$\left|\frac{\partial \tau}{\partial \boldsymbol{x}}\right| = \frac{|\bar{u}_2 \times \bar{v}_2|}{|\bar{u}_1 \times \bar{v}_1|}. \tag{6}$$

The shift mapping Jacobian is multiplied by the product of $|\partial\tau/\partial \boldsymbol{x}_i|$ for each vertex $\boldsymbol{x}_i$ reused via τ.

[2]The matching could potentially be done with object-space distances for increased robustness, but we did not find this to be an issue.

[3]The Jacobian determinant is the ratio of the areas of the differential $\mathcal{S}_1$ and $\mathcal{S}_2$ parallelograms corresponding to the same differential square in the UV space.

4.4 LoD-aware ReSTIR path tracing

With our new vertex mapping, we are now able to handle changing geometry levels of detail. We now present the details of our LoD-aware ReSTIR path tracing algorithm.

We build upon ReSTIR with reservoir splatting [Liu et al. 2025]. We assume each object is represented at a fixed level of detail in a given frame, i.e., each ray sees the same scene. While the prefiltered LoD representation could be built or chosen based on a ray footprint or depth, we opt for a simpler definition of the path space Ω. Our simple implementation assumes that each object is accompanied with one or more LoD representations chosen by an integer that may change across frames.

Next, we describe our path reservoir and our temporal and spatial reuse strategies relative to Liu et al. [2025].

Path reservoir. We use the path reservoir format of Zhang et al. [2024]. In addition, we store the object ID and the texture coordinates for the primary intersection $\boldsymbol{x}_1$ and the reconnection vertex $\boldsymbol{x}_k$.

Initial sample generation. Similarly to Lin et al. [2022], we generate initial samples with standard path tracing. We additionally record the object ID and texture coordinates for the primary intersection and reconnection vertex.

Temporal reuse. Given a prior-frame path $\bar{\boldsymbol{x}}$, we shift it to a path $\bar{\boldsymbol{y}} = T(\bar{\boldsymbol{x}})$ in the current frame following reservoir splatting [Liu et al. 2025] and the hybrid shift [Lin et al. 2022], but use our novel vertex mapping:

(1) *LoD-aware primary hit reprojection.* We map the prior-frame primary hit $\boldsymbol{x}_1$ to the current frame via $\boldsymbol{y}_1 = \tau(\boldsymbol{x}_1)$, where τ is computed on the fly following Section 4.2. This gives the primary hit of the offset path, which we reproject to the camera for splatting.
(2) *LoD-aware hybrid shift*. We shift the rest of the path vertices according to the hybrid shift, but map the reconnection vertex $\boldsymbol{x}_k$ via $\boldsymbol{y}_k = \tau(\boldsymbol{x}_k)$ on the fly, following Section 4.2.

Spatial reuse. We next reuse light paths across pixels within the same frame, following Liu et al. [2025]. Since we assume all rays see the same path space, spatial reuse remains largely unaffected: we merely ensure we cache the object ID and texture coordinates into the reservoir.

Jacobians. We multiply the Jacobian determinant $\left|\frac{\partial T}{\partial \bar{\boldsymbol{x}}}\right|$ of Liu et al. [2025] by $\left|\frac{\partial \tau}{\partial \boldsymbol{x}_i}\right|$ (Equation 6) for each τ-mapped vertex $\boldsymbol{x}_i$ to compensate for the density change by τ.

Optimizations. When mapping vertices between frames where the object's LoD level does not change, we optimize and define τ by matching the triangle ID and barycentric coordinates like before.

5 Experiments

To demonstrate the effectiveness of our vertex mapping, we compare our method with two baselines:

B.1 Base ReSTIR [Liu et al. 2025] without our vertex mapping.
B.2 Path Tracing [Kajiya 1986]. We give path tracing a higher sample count to achieve an equal-time comparison.

We render our results in 1920×1080 with Russian roulette enabled. For ReSTIR methods (ours and **B.1**), we set the confidence cap to 20, the number of spatial neighbors to 2, and the spatial reuse range to 30 pixels, mostly following Liu et al. [2025].

We implement all methods in the Falcor rendering framework [Kallweit et al. 2022]. All methods share the same interfaces for lighting, material, and geometry. Our experiments are run on a PC with an NVIDIA GeForce RTX 3090 GPU and AMD Ryzen 9 5900X CPU.

5.1 Evaluations

Figure 2 and Figure 3 establish the well-known fact that the scene should be modeled with the right precision to avoid extraneous variance.

In a dynamic real-time environment, LoD is thus often necessary, which necessarily implies LoD changes[4]. Our method's main difference to reservoir splatting is the vertex mapping τ that enables reuse past LoD changes. Outside of LoD changes, our methods provide the same results. Hence, we focus our validation efforts on LoD changes: the frames right before and soon after.

In our experiments, path tracing (**B.2**) generally suffers from noise due to lacking sample reuse. LoD switches effectively disable temporal reuse of reservoir splatting (**B.1**) at LoD changes, which drastically increases its noise for a period of time. Our method allows reuse with only mild quality loss at LoD changes, resulting in much cleaner rendering results. We show this general behavior in Figure 7. This minor quality loss is partially due to slight imprecisions in UV mapping and partially due to differences in, for example, surface normals between the LoD levels.

Figure 8 validates our method against reservoir splatting (**B.1**) and path tracing (**B.2**) in a variety of LoD scenes. Our results confirm that ReSTIR tends to be much more efficient than path tracing. However, unchecked LoD changes can locally decrease its quality. Implementing our vertex mapping (ours) recovers the ability to reuse through LoD switches, recovering ReSTIR's higher efficiency.

Dynamic LoD. Figure 1 compares our method to ReSTIR without LoD in a dynamic environment. Given our vertex mapping, dynamic LoD benefits ReSTIR—but without our vertex mapping, dynamic LoD can even be detrimental.

Invertibility and matching. In Figure 6, we study the necessity of our matching algorithm (Section 4.2) for effective reuse in the presence of a non-invertible UV mapping. Plane contains a symmetric mesh with two LoD levels, with the left and right halves sharing the same UV coordinates. In Cube, all six faces are mapped to the same UV coordinates. As shift mappings must be invertible, the vertex mapping without matching (right) gives up reuse when the same UV coordinate corresponds to multiple world-space locations, essentially disabling temporal reuse across the LoD change. Our vertex matching successfully settles the ambiguity and allows effective reuse.

Matching and performance. In the same Figure 6, we also report the full-frame performance numbers. Our matching algorithm is relatively cheap for a sensible amount of overlaps, but the extra cost can often be completely avoided by non-overlapping unwraps,

[4] Either continuous or discrete.

e.g., by utilizing a periodic texture and mapping different parts to different repeats.

Which points to map with τ? When our method detects a LoD change, it maps the corresponding vertex with our new UV mapping based τ (Section 4.4) instead of triangle index and barycentric coordinates like before. In Figure 5 we study the effect of our new τ for different vertices in Chessboard. Trying to save time by using τ only for the primary hit or only for the reconnection vertex immediately backfires with extra noise at LoD changes. It should be used for both.

5.2 Scenes

We perform our experiments in the following scenes.

Rock displays a rock mesh with four LoD levels and depth-of-field with an animated focal distance. We focus on renderer behavior when the rock changes LoD as it comes into focus.

Sponza contains several Egyptian cat statues with four LoD levels. We slowly move the camera along the hallway and capture the frame when one of the statues changes its LoD as it comes into focus.

Terrain is modeled with three LoD levels. The camera moves towards the mountain, and we capture the fourth frame after a terrain LoD level change.

Chess contains three high-poly chess pieces with three LoD levels each. We change the focal distance of the camera and capture the frame when the object in the middle changes the LoD level.

Coastal Cliff contains a high-poly cliff with three LoD levels. The camera moves forward towards the cliff in the middle, and we capture the second frame after the cliff changes its LoD level.

6 Discussion and Conclusion

Limitations. Our method assumes that scene objects are equipped with a texture mapping that is consistent across different geometry LoD representations. Objects without texture mappings, or with inconsistent texture mappings across LoD levels, are therefore not directly supported. In addition, highly distorted or degenerate texture mappings can reduce the effectiveness of the surface correspondence, degrading the quality of the rendered image.

Discussion and future work. An important takeaway from this work is that effective spatiotemporal sample reuse with ReSTIR often requires an explicit vertex mapping beyond reusing the same triangle index and barycentric coordinates as in prior work. While geometry LoD switching is a prominent example, similar challenges arise in other scenarios where surface representations vary across frames or pixels, such as fluid simulations. Designing robust vertex mappings for these cases remains an interesting direction for future research. Although our formulation does not require an object to use a single LoD level within a frame, our current implementation enforces this restriction. Extending our method to support per-ray LoD selection is an interesting direction for future work.

Conclusion. In this paper, we introduce a vertex mapping that enables ReSTIR to handle geometry topology changes induced by level-of-detail rendering. By incorporating this mapping into ReSTIR's sample reuse framework, we enable robust spatiotemporal reuse across LoD representations and significantly improve rendering quality in LoD scenes. Our results demonstrate that combining ReSTIR with geometry LoD is a promising approach for rendering complex scenes in real time.

Acknowledgments

We thank the anonymous reviewers for their helpful suggestions.

References

Steve Bako, Pradeep Sen, and Anton Kaplanyan. 2023. Deep Appearance Prefiltering. *ACM Trans. Graph.* 42, 2 (Jan. 2023), 23 pages.

Pablo Bauszat, Victor Petitjean, and Elmar Eisemann. 2017. Gradient-Domain Path Reusing. *ACM Transactions on Graphics* 36, 6 (2017), 229:1–229:9. doi:10.1145/3130800.3130886

Philippe Bekaert, Mateu Sbert, and John H Halton. 2002. Accelerating Path Tracing by Re-Using Paths. In *Eurographics Workshop on Rendering*. 125–134. doi:10.2312/EGWR.EGWR02.125-134

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020). doi:10/gg8xc7

M. T. Chao. 1982. A General Purpose Unequal Probability Sampling Plan. *Biometrika* 69, 3 (1982), 653–656.

Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, Fabrice Neyret, and Victor Ostromoukhov. 2013. Linear efficient antialiased displacement and reflectance mapping. *ACM Trans. Graph.*, Article 211 (Nov. 2013), 11 pages.

Michael Garland and Paul S. Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., USA, 209–216.

Binh-Son Hua, Adrien Gruson, Victor Petitjean, Matthias Zwicker, Derek Nowrouzezahrai, Elmar Eisemann, and Toshiya Hachisuka. 2019. A Survey on Gradient-Domain Rendering. 38, 2 (2019), 455–472. doi:10.1111/cgf.13652

T. Huang, Y. Zhou, D. Lin, J. Zhu, L. Yan, and K. Wu. 2025. Real-time Level-of-detail Strand-based Rendering. *Computer Graphics Forum* 44 (07 2025).

James T. Kajiya. 1986. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86)*. Association for Computing Machinery, New York, NY, USA, 143–150. doi:10.1145/15922.15902

Simon Kallweit, Petrik Clarberg, Craig Kolb, Tom'aš Davidovič, Kai-Hwa Yao, Theresa Foley, Yong He, Lifan Wu, Lucy Chen, Tomas Akenine-Möller, Chris Wyman, Cyril Crassin, and Nir Benty. 2022. The Falcor Rendering Framework. https://github.com/NVIDIAGameWorks/Falcor https://github.com/NVIDIAGameWorks/Falcor.

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-domain path tracing. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–13. doi:10.1145/2766997

Leif Kobbelt, Swen Campagna, and Hans-Peter Seidel. 1998. A General Framework for Mesh Decimation. In *Graphics Interface*. https://api.semanticscholar.org/CorpusID:7701157

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-domain metropolis light transport. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.

Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized resampled importance sampling: foundations of ReSTIR. *ACM Trans. Graph.*

Jeffrey Liu, Daqi Lin, Markus Kettunen, Chris Wyman, and Ravi Ramamoorthi. 2025. Reservoir Splatting for Temporal Path Resampling and Motion Blur *(SIGGRAPH Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA.

Marco Manzi, Markus Kettunen, Frédo Durand, Matthias Zwicker, and Jaakko Lehtinen. 2016. Temporal gradient-domain path tracing. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–9.

Marco Manzi, Fabrice Rousselle, Markus Kettunen, Jaakko Lehtinen, and Matthias Zwicker. 2014. Improved sampling for gradient-domain metropolis light transport. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 1–12.

Marc Olano and Dan Baker. 2010. LEAN mapping. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Washington, D.C.) *(I3D '10)*. Association for Computing Machinery, New York, NY, USA, 181–188.

Justin Talbot, David Cline, and Parris Egbert. 2005. Importance Resampling for Global Illumination. In *Eurographics Symposium on Rendering*. 139–146.

Eric Veach and Leonidas J. Guibas. 1997. Metropolis Light Transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. 65–76. doi:10.1145/258734.258775

Lifan Wu, Shuang Zhao, Ling-Qi Yan, and Ravi Ramamoorthi. 2019. Accurate appearance preserving prefiltering for rendering displacement-mapped surfaces. *ACM*

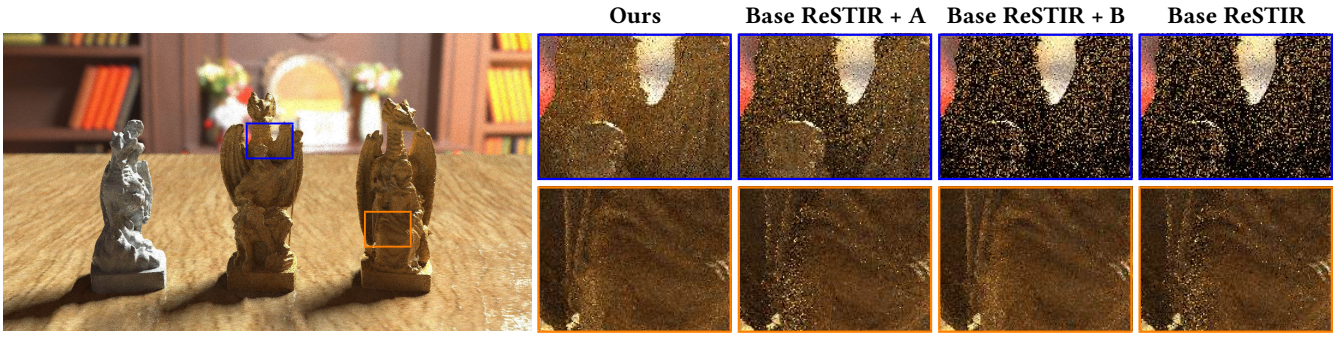


Fig. 5. *Importance of our vertex mapping at different vertices.* Our method applies our new vertex mapping τ (Section 4.4) for (A) the primary hit, and (B) the reconnection vertex, and successfully reuses across the LoD change (Ours). Implementing our new τ only for the primary hit (Base ReSTIR + A) or only for the reconnection vertex (Base ReSTIR + B) loses quality, but still improves over not using it for either (Base ReSTIR). The baseline always maps both vertices by triangle index and barycentric coordinates, which forces temporal reuse to fail at LoD changes.

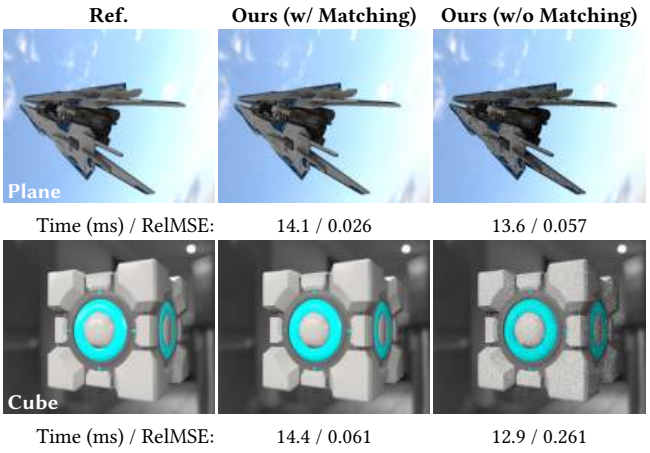


Fig. 6. *Matching and non-invertible UVs.* Our method solves ambiguities with multiple points mapping to the same UV coordinates by a world-space matching (Section 4.2). We study two scenes with UV overlaps right after a LoD change: In Plane, both halves of the symmetric spaceship share the same UV coordinates. In Cube, all faces map to the same UV coordinates. Without our matching algorithm, the UV ambiguity forces temporal shifts to fail, essentially reducing to spatial-only reuse (right). Our matching solves the UV ambiguity and allows temporal reuse through the LoD change, producing a much cleaner image.

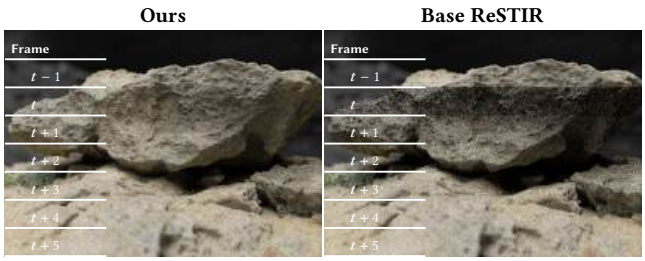


Fig. 7. *Rolling shutter capture around a LoD change at time t in Rock.* At frame $t-1$, right before the LoD change, the methods still give the same results. The LoD change at frame t effectively resets the temporal history of reservoir splatting (right), while our method with the vertex mapping only suffers a slight quality loss. After the LoD change, the quality of both methods begins to converge back to normal. Increased noise shows up as darkness due to tone mapping.

Trans. Graph. (July 2019), 14 pages.

Chris Wyman, Markus Kettunen, Daqi Lin, Benedikt Bitterli, Cem Yuksel, Wojciech Jarosz, Pawel Kozlowski, and Giovanni De Francesco. 2023. A Gentle Introduction to ReSTIR: Path Reuse in Real-time. In *ACM SIGGRAPH 2023 Courses* (Los Angeles, California).

Song Zhang, Daqi Lin, Markus Kettunen, Cem Yuksel, and Chris Wyman. 2024. Area ReSTIR: Resampling for Real-Time Defocus and Antialiasing. *ACM Trans. Graph.*, Article 98 (July 2024), 13 pages.

Junqiu Zhu, Christophe Hery, Lukas Bode, Carlos Aliaga, Adrian Jarabo, Ling-Qi Yan, and Matt Jen-Yuan Chiang. 2024. A Realistic Multi-scale Surface-based Cloth Appearance Model. In *ACM SIGGRAPH 2024 Conference Papers (SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, 10 pages.

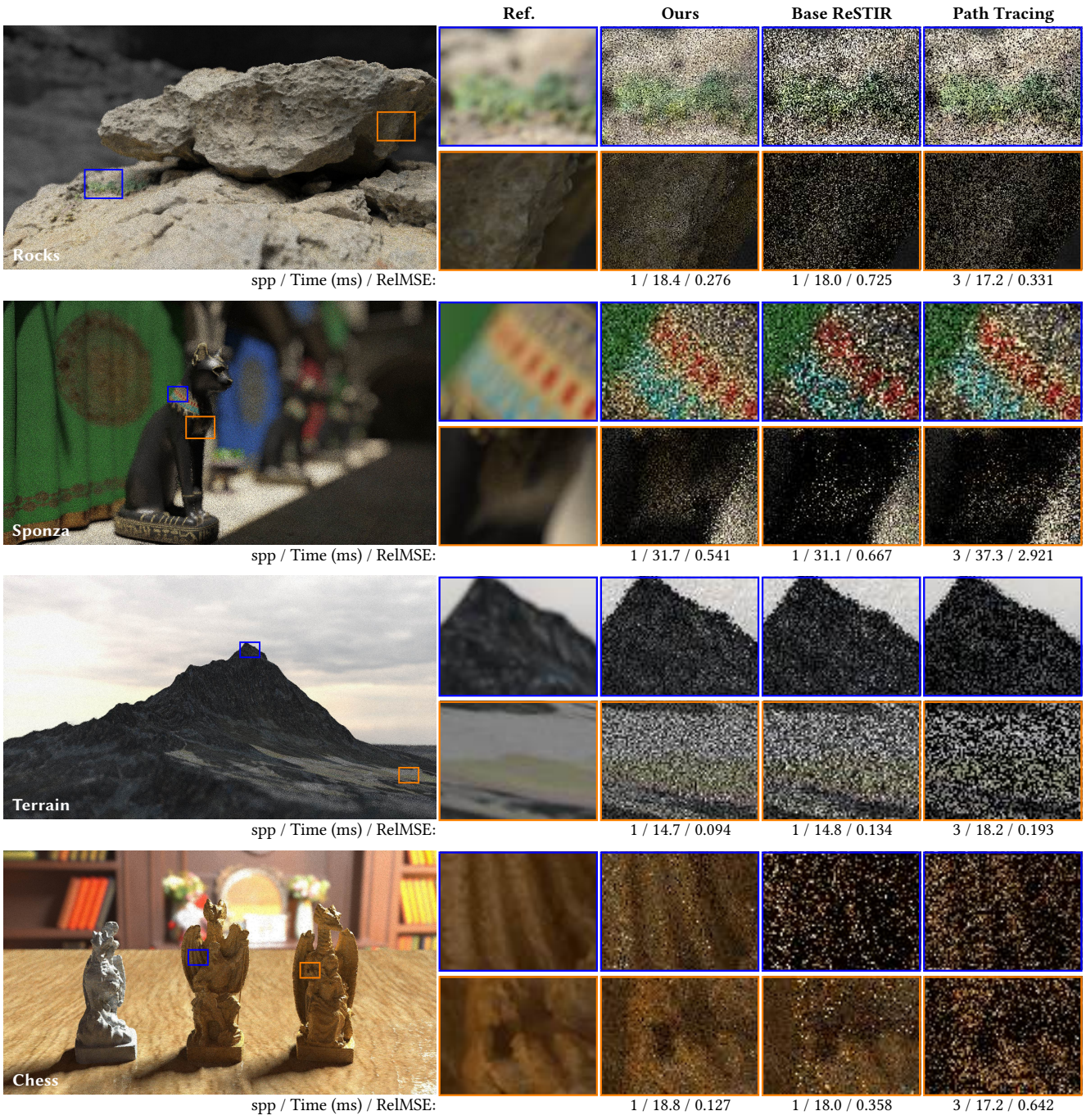


Fig. 8. Equal-time comparison between our method and baselines *at a LoD change*. Path Tracing (B.2) produces noisy results because of no sample reuse. Earlier ReSTIR methods (B.1) fail to reuse at a LoD change, essentially falling back to independent sampling for those regions. Our method enables sample reuse at LoD changes by mapping vertices between LoD levels.