Fast Volume Rendering with Spatiotemporal Reservoir Resampling

DAQI LIN, University of Utah, USA
CHRIS WYMAN, NVIDIA, USA
CEM YUKSEL, University of Utah, USA

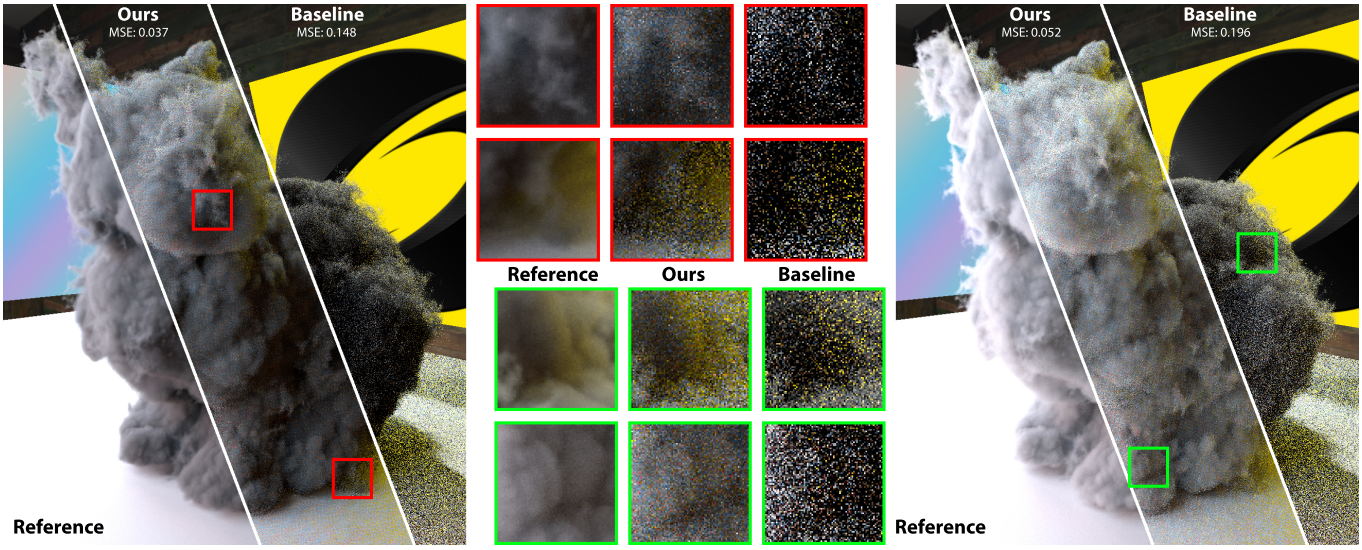


Fig. 1. *A volumetric bunny illuminated by a complex environment map and emissive logos. We compare our new volumetric ReSTIR with offline references and an equal-time baseline (combining decomposition tracking [Kutz et al. 2017] and residual ratio tracking [Novák et al. 2014]). We show our work with (left) single scattering in 55 ms and (right) three-bounce multiple scattering in 142 ms.*

Volume rendering under complex, dynamic lighting is challenging, especially if targeting real-time. To address this challenge, we extend a recent direct illumination sampling technique, spatiotemporal reservoir resampling, to multi-dimensional path space for volumetric media.

By fully evaluating just a single path sample per pixel, our volumetric path tracer shows unprecedented convergence. To achieve this, we properly estimate the chosen sample's probability via approximate perfect importance sampling with spatiotemporal resampling. A key observation is recognizing that applying cheaper, biased techniques to approximate scattering along candidate paths (during resampling) does not add bias when shading. This allows us to combine transmittance evaluation techniques: cheap approximations where evaluations must occur many times for reuse, and unbiased methods for final, per-pixel evaluation.

With this reformulation, we achieve low-noise, interactive volumetric path tracing with arbitrary dynamic lighting, including volumetric emission, and maintain interactive performance even on high-resolution volumes. When paired with denoising, our low-noise sampling helps preserve smaller-scale volumetric details.

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

Additional Key Words and Phrases: ray tracing, volume rendering, resampled importance sampling, reservoir sampling, ReSTIR

ACM Reference Format:
Daqi Lin, Chris Wyman, and Cem Yuksel. 2021. Fast Volume Rendering with Spatiotemporal Reservoir Resampling. *ACM Trans. Graph.* 40, 6, Article 279 (December 2021), 18 pages. https://doi.org/10.1145/3478513.3480499

Authors' addresses: Daqi Lin, University of Utah, USA; Chris Wyman, NVIDIA, USA; Cem Yuksel, University of Utah, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
0730-0301/2021/12-ART279 $15.00
https://doi.org/10.1145/3478513.3480499

1 INTRODUCTION

Smoke, fire, clouds, and other participating media are vital in virtual scenes; modern movies, games, and simulations rely heavily on media for realism and ambiance. But real-time rendering of participating media is challenging. Even traditional raster pipelines have separate order-independent transparency passes [Wyman 2016] and data structures for volume lighting [Kaplanyan and Dachsbacher 2010]. With real-time ray tracing [Kilgariff et al. 2018] and more complex ray-traced lighting [Majercik et al. 2019], integrating dynamic ray-traced media will be vital for achieving a uniform look.

In this paper, we introduce an effective path sampling solution for real-time volume rendering with multiple scattering and volumetric emission. To do this, we generalize *resampled importance sampling* [Talbot et al. 2005] and *spatiotemporal reservoir resampling* [Bitterli

et al. 2020] to path integrals. These have proven effective for sampling direct illumination on surfaces. We generalize them to path space, providing importance sampling that closely approximates our integrand: a path integral formulation of the volume rendering equation. We also present numerous optimizations to minimize computation and memory overhead. As a result, we can estimate the multi-dimensional volume rendering integral while shading just one path per pixel, enabling real-time volume rendering under arbitrary scene illumination, including environment maps, area lights, and volumetric emission.

Our technical contributions include:

- A generalization of resampled importance sampling (Section 3) and spatiotemporal reservoir resampling (Section 4) to complex path integrals,
- An efficient importance sampling estimator for the volumetric path integral (Section 3.3), including multiple scattering and volumetric light emission,
- A temporal reprojection (Section 4.4) and practical velocity resampling method for robust temporal reuse (Section 5.2),
- Optimized path space transmittance estimates (Section 5.1). These only affect importance sampling, not final path throughput, allowing use of efficient biased estimates without biasing the results (e.g., sampling lower resolution volumes).

Our renderer runs interactively, reusing carefully-chosen paths to evaluate the volume rendering equation. While we can produce unbiased renderings (with static volumes and dynamic lighting), by allowing a little bias (Section 4.4) we can reduce sampling variance and handle more dynamism. Our work significantly lowers variance compared to state-of-the-art real-time path sampling (Figure 1).

Section 2 reviews prior work, summarizing resampled importance sampling (RIS) and spatiotemporal reservoir resampling (ReSTIR). In Section 3, we develop an RIS estimator to efficiently sample the volumetric path integral. In Section 4, we modify ReSTIR's iterative resampling to efficiently reuse spatiotemporal volumetric path samples. We provide key implementation details in Section 5.

2 BACKGROUND

The volume rendering equation represents incident radiance L at point $\mathbf{x}_0$ from direction $\boldsymbol{\omega}_o$ and integrates the outgoing radiance through volumetric media L^m and the surface or light behind it, L^s

$$L(\mathbf{x}_0, \boldsymbol{\omega}_o) = \int_0^{z_s} T(\mathbf{x}_0 \leftrightarrow \mathbf{x})\, \sigma_t(\mathbf{x})\, L^m(\mathbf{x}, \boldsymbol{\omega}_o)\, dz + T(\mathbf{x}_0 \leftrightarrow \mathbf{x}_s)\, L^s(\mathbf{x}_s \rightarrow \mathbf{x}_0)\,, \tag{1}$$

where $\mathbf{x} = \mathbf{x}_0 - z\boldsymbol{\omega}_o$ is a point along direction $\boldsymbol{\omega}_o$ towards $\mathbf{x}_0$, $\sigma_t(\mathbf{x})$ is the extinction coefficient at $\mathbf{x}$, and the transmittance function $T(\mathbf{x}_0 \leftrightarrow \mathbf{x})$ represents visibility between $\mathbf{x}_0$ and $\mathbf{x}$

$$T(\mathbf{x}_0, \mathbf{x}) = e^{-\int_0^z \sigma_t(\mathbf{x}_0 - y\boldsymbol{\omega}_o) dy}, \tag{2}$$

and $\mathbf{x}_s = \mathbf{x}_0 - z_s\boldsymbol{\omega}_o$ is the corresponding surface along the ray. L^m includes volumetric emission L_e^m and in-scattering

$$L^m(\mathbf{x}, \boldsymbol{\omega}_o) = \frac{\sigma_a(\mathbf{x})}{\sigma_t(\mathbf{x})} L_e^m(\mathbf{x}, \boldsymbol{\omega}_o) + \frac{\sigma_s(\mathbf{x})}{\sigma_t(\mathbf{x})} \int_S \rho(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}_o) L(\mathbf{x}, \boldsymbol{\omega}) d\boldsymbol{\omega}, \tag{3}$$

where σ_a and σ_s are absorption and scattering coefficients with $\sigma_t(\mathbf{x}) = \sigma_a(\mathbf{x}) + \sigma_s(\mathbf{x})$, S is the sphere of all directions, and $\rho(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}_o)$ is the media's phase function.

With multiple scattering, computing $L(\mathbf{x}, \boldsymbol{\omega})$ inside the integral from Equation 3 via Equation 1 is costly, particularly for real-time rendering. For simple single scattering, with no volumetric emission and a few point or directional lights, volumetric shadow mapping can achieve real-time rendering performance [Delalandre et al. 2011; Gautron et al. 2013; Jansen and Bavoil 2010; Kim and Neumann 2001; Salvi et al. 2010; Yuksel and Keyser 2008]. But these are inefficient for more general lighting conditions and multiple scattering. Modern games use "froxel" representations [Hillaire 2015] to align volumes with the view frustum to minimize memory incoherence during traversal, but otherwise behave similarly.

2.1 Monte Carlo Sampling for Volume Rendering

Path tracing and Monte Carlo sampling provide a more general solution for integrating the volume rendering equations 1, 2, and 3.

Importance sampling distance z (Equation 1) in a homogeneous volume (constant σ_t) with a probability distribution function (PDF) proportional to $\sigma_t T(\mathbf{x}_0, \mathbf{x})$ is trivial; the resulting cumulative distribution function (CDF) $1 - e^{-z\sigma_t}$ is easily inverted. The resulting z values are called the *free-flight distance*. In heterogeneous volumes, *regular tracking* [Sutton et al. 1999] represents $\sigma_t(\mathbf{x})$ with piecewise simple functions with analytically invertible CDFs, allowing tracking of each piece separately to find scattering events via an exact PDF. In general heterogeneous volumes, *ray marching* finds these events with an approximated PDF [Novák et al. 2018] , which introduces bias to the result. *Null-collision* methods avoid bias by introducing fictitious media to simplify the CDF. For example, delta tracking [Raab et al. 2008; Woodcock et al. 1965] uses piecewise constant majorant $\bar{\sigma}$ (for $\bar{\sigma} \geq \sigma_t$) and determines null collisions via a secondary Monte Carlo process, but the sampling PDF is generally not available in closed form.

Recently, Miller et al. [2019] introduces a special path space formulation including null scattering to obtain analytical PDFs. But null-collisions can reduce performance in highly uneven volumetric densities, when long chains of short null collisions occur. Acceleration structures (e.g., super-voxels [Szirmay-Kalos et al. 2011] and kd-trees [Yue et al. 2011]) partition space with separate, tighter majorants to improve performance. Decomposition tracking [Kutz et al. 2017] reduces overhead by splitting media into a constant density control volume and a residual volume, where tracking occur separately and the minimum distance is taken. Kutz et al. [2017] introduces weighted delta tracking [Galtier et al. 2013] to allow non-tight upper bounds in the residual volume.

Distance sampling techniques also apply to transmittance estimation. For example, delta tracking gives a per-sample binary transmittance decision, based on if a real collision occurs. *Ratio tracking* modifies delta tracking, replacing stochastic termination with its expectation, giving non-binary transmittance and improving convergence speed. *Residual ratio tracking* uses fewer steps for ratio tracking by separating residual and control components of the extinction function [Novák et al. 2014] and applies in various contexts [d'Eon and Novák 2021; Szirmay-Kalos et al. 2017]. Recent

next-flight estimators [Novák et al. 2018] improve delta and ratio tracking efficiency if the fictitious media has a lower density.

New integral formulations using power-series expansion improve transmittance estimation via sample stratification [Georgiev et al. 2019]. Kettunen et al. [2021] propose unbiased ray marching that corrects biased methods with occasional higher order terms, leveraging ray marching efficiency to compute low-noise transmittance.

Importance sampling $\boldsymbol{\omega}$ in the phase function ρ [Henyey and Greenstein 1941] is similar to any bidirectional scattering distribution function. However, explicitly sampling light sources with *next event estimation* (NEE) often increases efficiency. NEE can be combined with other sampling via *multiple importance sampling* (MIS). Miller et al. [2019] use MIS to integrate in path space using previously unknown PDFs. To efficiently integrate volumetric emission, Simon et al. [2017] introduce forward next event estimation (FNEE) that samples solid angle to perform line integration.

All these methods consider local sampling, rather than importance sampling full paths. This limits quality, increasing samples needed to converge and reducing their real-time appeal. Recent denoisers can filter noisy samples to a final image. But sample counts must be sufficient to achieve post-filter temporal stability. Thus, improving sample quality is vital, even with state-of-the-art denoising.

2.2 Bidirectional Volume Rendering

Bidirectional methods consider entire paths to improve sampling quality. Equiangular sampling [Kulla and Fajardo 2012] jointly samples a light vertex and the penultimate path vertex receiving inscattered light. Joint importance sampling [Georgiev et al. 2013] allows double scattering with a joint distribution. Zero-variance random walks further extend joint sampling, building random walks with near-zero variance by considering all terms in the path integral. In some scenarios, this applies to subsurface scattering [Křivánek and d'Eon 2014; Meng et al. 2016] and can improve path guiding [Herholz et al. 2019] in general participating media.

Other bidirectional techniques estimate photon density using volumetric photon maps [Jensen and Christensen 1998]. Density estimation queries can be improved using beams [Jarosz et al. 2008]. Higher dimensional primitives, such as photon beams [Jarosz et al. 2011], photon planes and volumes [Bitterli and Jarosz 2017], and photon surfaces [Deng et al. 2019] can significantly reduce variance.

Bidirectional techniques based on virtual lights [Novák et al. 2012a,b] also benefit from higher dimensional light representations. Combining different kinds of density estimation with path tracing [Křivánek et al. 2014] often provides a robust framework to optimally sample across various scenes.

However, bidirectional methods often use complex data structures with costly generation and maintenance phases. This adds often insurmountable engineering complexity in real-time contexts.

2.3 Path Reuse

Photon- and VPL-based bidirectional methods reuse subpaths, but reuse can occur between pixels [Bekaert et al. 2002]. This can dramatically reduce variance, though it imposes fairly high storage and computation costs; paths must explicitly remain in memory to benefit others, and visibility rays are required to connect neighbors.

More recent work reuses paths via finite differences in the gradient domain [Lehtinen et al. 2013], which can also apply to path tracing [Kettunen et al. 2015] and volume rendering [Gruson et al. 2018]. Similar to the spatiotemporal resampling we use in this paper, these techniques also leverage correlations between pixels.

However, spatiotemporal resampling reuses samples indirectly to improve PDFs, rather than explicitly reusing paths for shading. In this sense, it is more akin to path guiding [Vorba et al. 2019], if done in a feed-forward, streaming fashion.

2.4 Resampled Importance Sampling (RIS)

Our volume sampling builds on *resampled importance sampling* (RIS) [Talbot et al. 2005], Given function $f(x)$ defined over domain $x \in D$, RIS provides an importance sampling estimator for the integral:

$$I = \int_D f(x)\, dx\ . \tag{4}$$

Let $\hat{p}$ be a *target* PDF without a practical sampling algorithm. RIS generates $M \geq 1$ *candidate samples* $\mathbb{x} = \{x_1, \ldots, x_M\}$ using a (suboptimal) *source* PDF p. Then, it randomly selects a sample x_r, for $r \in \{1, \ldots, M\}$, using discrete probabilities

$$p(x_r|\mathbb{x}) = \frac{w(x_r)}{\sum_{j=1}^{M} w(x_j)} \qquad \text{with} \qquad w(x) = \frac{\hat{p}(x)}{p(x)}\ . \tag{5}$$

The resulting 1-sample RIS estimator can be written as

$$\langle I \rangle_{\text{ris}}^{1,M} = E_{\hat{p}}(x_r) \left(\frac{1}{M} \sum_{j=1}^{M} w(x_j) \right) \quad \text{with} \quad E_{\hat{p}}(x_r) = \frac{f(x_r)}{\hat{p}(x_r)}\ . \tag{6}$$

The parenthetical term corrects for differences between the actual probability used to sample x_r and the desired PDF $\hat{p}(x_r)$. This gives an unbiased estimate if $p(x)$ and $\hat{p}(x)$ are non-zero for all x with non-zero $f(x)$. As $M \to \infty$, the distribution of x_r approaches $\hat{p}$.

RIS is particularly effective if $\hat{p}$ closely approximates f and generation and evaluation of candidate samples x_j and $w(x_j)$ are cheap. In Talbot et al. [2005], x is a point on a light source, $p(x)$ is the light sampling PDF. $\hat{p}(x)$ is unshadowed reflected radiance, including BSDF, geometry term, and incident radiance. This reasonably approximates the integrand, without expensive visibility queries. When directly lighting opaque surfaces, this improves sampling quality over standard importance sampling.

2.5 Spatiotemporal Reservoir Resampling (ReSTIR)

Spatiotemporal reservoir resampling (ReSTIR) [Bitterli et al. 2020] transforms RIS into a streaming algorithm, avoiding storage of most candidate samples by using weighted reservoir sampling [Chao 1982]. It is designed for direct illumination sampling from many lights for real-time rendering. For each pixel, ReSTIR maintains a reservoir that stores a sample x_r selected from the previous m candidates. Each new candidate x_{m+1} is selected with probability

$$p(x_{m+1}|\mathbb{x} \cup \{x_{m+1}\}) = \frac{w(x_{m+1})}{\sum_{j=1}^{m+1} w(x_j)}\ . \tag{7}$$

This can be seen as *streaming* candidate samples into a reservoir. Since the reservoir stores only the selected sample and a running

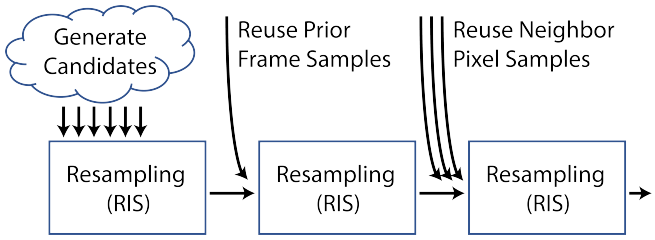


Fig. 2. *Extending ReSTIR [Bitterli et al. 2020] for participating media entails updating each component of this ReSTIR pipeline: defining candidate generation in path space (see Section 3.2), RIS estimators to efficiently evaluate high-dimensional integrals (Section 3.3), and spatial and temporal sample reuse in this domain (Section 4).*

sum of weights $\sum_{j=1}^{m+1} w(x_j)$, many candidate samples M can be considered without additional storage, improving the sampling quality.

ReSTIR also enables spatiotemporal reuse by combining the reservoirs of nearby pixels and the previous frame, exponentially increasing the *effective* candidate sample count (Figure 2). While rendering the first frame, each pixel q allocates a reservoir and streams M newly generated candidates to select a sample x_q. Then, the reservoirs of a random subset of nearby pixels are combined. Let q' represent a pixel near q. Their two reservoirs cannot simply be combined, unless the target PDFs $\hat{p}_q(x)$ and $\hat{p}_{q'}(x)$ for both pixels are identical. Generally, this is not the case and $\hat{p}_q(x) \neq \hat{p}_{q'}(x)$. Therefore, ReSTIR includes a correction factor $w_{q'\to q}$ for using the selected sample $x_{q'}$ in reservoir q' for pixel q, defined as

$$w_{q'\to q} = \frac{\hat{p}_q(x_{q'})}{\hat{p}_{q'}(x_{q'})} w^{\text{sum}}_{q'}, \quad \text{where} \quad w^{\text{sum}}_{q'} = \sum_{j=1}^{M} \frac{\hat{p}_{q'}(x_j)}{p_{q'}(x_j)}. \tag{8}$$

Note that $w^{\text{sum}}_{q'}$ is the running sum in the reservoir from initial candidate generation. For multiple iterations of reuse (chained RIS passes), $w^{\text{sum}}_{q'}$ becomes the running sum from the prior RIS pass.

The resulting estimator combining N neighboring reservoirs from pixels $q_1, \ldots, q_N$ can be written as

$$\langle I \rangle^{N,1,M}_{\text{ReSTIR}} = E_{\hat{p}_q}(x_r) \left(\frac{1}{M_q} \sum_{i=0}^{N} w_{q_i\to q} \right), \tag{9}$$

where x_r is the sample selected from one of the $N+1$ reservoirs, $M_q = (N+1)M$ is the total *effective* candidate sample count for pixel q, and we define $q_0 \equiv q$. Yet, this leads to a biased estimator, because $\hat{p}_{q'}(x)$ can be zero for x with non-zero $p_q(x)$. For correcting this bias, Bitterli et al. [2020] propose replacing the $1/M_q$ term in Equation 9 with the MIS weight of the selected sample

$$\frac{\hat{p}_r(x_r)}{M \sum_{i=0}^{N} \hat{p}_{q_i}(x_r)}, \tag{10}$$

where $\hat{p}_r$ denotes the PDF of the reservoir that produced x_r.

This MIS weight is stochastic (it depends on the chosen sample). Although cheaper to evaluate than the deterministic MIS weight proposed by Talbot [2005], it introduces noise. The deterministic Talbot MIS can be used by multiplying the weights of samples in Equation 9 with an additional term:

$$w^{\text{new}}_{q_i\to q} = w_{q_i\to q} \cdot \frac{M_q \hat{p}_{q_i}(x_{q_i})}{M \sum_{s=0}^{N} \hat{p}_{q_s}(x_{q_i})}, \tag{11}$$

Additionally, ReSTIR allows temporal reuse by passing the final selected sample x_r forward for reuse next frame. Combining spatial and temporal reuse, the *effective* candidate sample count M_q grows exponentially. To prevent unbounded influence of temporal samples, a user-defined *temporal limiting factor* Q is used to enforce $M_q \leq QM$. When M_q exceeds this limit, the running sum is scaled by QM/M_q and then M_q is updated as $M_q \leftarrow QM$.

But the benefit of spatiotemporal reuse is not indefinite. If $\hat{p}_{q'}$ from neighbor reservoir q' substantially differs from $\hat{p}_q$, spatially reusing q' can negatively impact sampling quality instead of improving it. Thus, applying heuristics to selectively reject reservoirs and using high quality MIS to reweight samples can substantially reduce variance [Bitterli 2021; Wyman and Panteleev 2021].

ReSTIR is highly effective in estimating direct illumination on opaque surfaces from many lights [Bitterli et al. 2020]. It chains RIS passes spatiotemporally to quickly accumulate many samples. Additionally, successive RIS passes can use higher quality $\hat{p}$ to improve sampling quality at relatively low cost. While $\hat{p}$ typically contains unshadowed reflected radiance, ReSTIR injects visibility into $\hat{p}$ at a lower frequency (known as *visibility reuse*).

Boksansky et al. [2021] store reservoirs in world space so a path tracer can efficiently perform NEE on secondary path vertices. Concurrent work by Ouyang et al. [2021] extends screen space ReSTIR for surface global illumination. This can be viewed as a special case of our method, which handles both surface and volume transport and interreflections between them (see Figure 1, right).

We extend the concepts in ReSTIR to real-time volume rendering; to achieve that, we solve various challenges when resampling (Section 3) and reusing samples (Section 4) in volumetric path space.

3 RIS FOR VOLUME RENDERING

We target extending ReSTIR [Bitterli et al. 2020] to volumetric path tracing. As ReSTIR builds on RIS [Talbot et al. 2005], we begin by developing an RIS estimator for the volume rendering equation.

Volume rendering involves higher-dimensional integrals than the direct surface illumination in Bitterli et al. [2020] and Talbot et al. [2005]. Visibility alone forms an integral along primary rays. Thus, we cannot just sample light positions; we must sample entire paths.

In this section, we provide a path integral representation of the volume rendering equation (Section 3.1), describe how we can generate candidate paths (Section 3.2), and explain how to estimate the volume rendering equation using RIS (Section 3.3).

3.1 Path Integral Representation of Volume Rendering

Let $\boldsymbol{\lambda}$ denote a path. We can write the volume rendering equation (Equation 1) as a path integral

$$L(\mathbf{x}_0, \boldsymbol{\omega}_o) = \int_{\Lambda} F(\boldsymbol{\lambda})\, d\boldsymbol{\lambda}, \tag{12}$$

where Λ is the set of all paths and $F(\boldsymbol{\lambda})$ is the incident radiance through the path $\boldsymbol{\lambda}$.

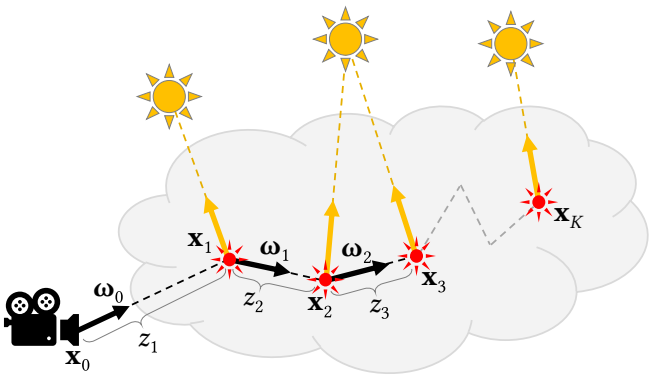


Fig. 3. *A random walk with K vertices generates $2K$ candidate paths for later reuse: K are scattering paths that terminate at a light (yellow) with next event estimation and the remaining K are emission paths terminating in the media due to volumetric emission (red).*

Consider a path $\boldsymbol{\lambda}$ with k scattering events, forming $k+2$ vertices $\mathbf{x}_0, \ldots, \mathbf{x}_{k+1}$, with $\mathbf{x}_0$ a camera vertex and $\mathbf{x}_{k+1}$ a *light vertex*. A light vertex can be a point on a light surface or inside emissive media. For brevity, the formulations below assume intermediate vertices $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are in the medium, but this can easily be extended to points on surfaces (e.g., see Figures 1 and 18).

Let $z_i = |\mathbf{x}_{i+1} - \mathbf{x}_i|$ be the distance between consecutive path vertices and $\boldsymbol{\omega}_i = (\mathbf{x}_{i+1} - \mathbf{x}_i)/z_i$ be the direction towards the next path vertex. Then, we can write the incident radiance as

$$F(\boldsymbol{\lambda}) = \Gamma_s(\boldsymbol{\lambda}, k)\, T(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1})\, G(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1})\, L(\mathbf{x}_{k+1} \to \mathbf{x}_k), \quad (13)$$

where geometry term $G_i = G(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)$ is 1 using solid angle measure and $1/z_i^2$ using volume measure, and Γ_s represents path throughput:

$$\Gamma_s(\boldsymbol{\lambda}, k) = \prod_{i=1}^{k} T(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)\, \sigma_s(\mathbf{x}_i)\, G_i\, \rho_i\,, \quad (14)$$

with $\rho_i = \rho(\mathbf{x}_i, -\boldsymbol{\omega}_{i-1}, \boldsymbol{\omega}_i)$ the phase function and $L(\mathbf{x}_{k+1} \to \mathbf{x}_k)$ the emitted radiance at $\mathbf{x}_{k+1}$ towards $\mathbf{x}_k$. Note, for a path with $k = 0$ (i.e. no scattering events) we take $\Gamma_s(\boldsymbol{\lambda}, k) = 1$.

The emitted radiance L can come from either a light sample at $\mathbf{x}_{k+1}$, if $\boldsymbol{\lambda}$ is a *scattering path*, or volumetric emission at $\mathbf{x}_{k+1}$, if $\boldsymbol{\lambda}$ is an *emission path*. More specifically, we can write

$$L(\mathbf{x}_{k+1} \to \mathbf{x}_k) = \begin{cases} L^s(\mathbf{x}_{k+1} \to \mathbf{x}_k) & \text{if scattering path,} \\ \sigma_a(\mathbf{x}_{k+1})\, L_e^m(\mathbf{x}_{k+1} \to \mathbf{x}_k) & \text{if emission path.} \end{cases} \quad (15)$$

We use L^s to represent radiance from the light. The notation assumes the source is an emissive surface, but it can easily be extended to other lights. For example, for an environment map $\mathbf{x}_{k+1}$ is an infinitely distant vertex and L^s is the radiance along direction $\mathbf{x}_{k+1} \to \mathbf{x}_k$.

3.2 Generating Path Samples

To use an RIS estimator for the path integral formulation of volume rendering (Equation 12), we must generate numerous random paths $\boldsymbol{\lambda}$ with PDF $p(\boldsymbol{\lambda})$.

Our path generation approach is similar to path tracing with next event estimation, as shown in Figure 3. We start with a ray from $\mathbf{x}_0$ towards $\boldsymbol{\omega}_0 = -\boldsymbol{\omega}_o$. At each step, we first pick a random distance z_i along our ray with PDF $p(z_i|\mathbf{x}_{i-1}, \boldsymbol{\omega}_{i-1})$. This specifies the next path vertex $\mathbf{x}_i = \mathbf{x}_{i-1} + z_i\boldsymbol{\omega}_{i-1}$. Then, for each scattering event, we pick a scattering direction $\boldsymbol{\omega}_i$ with a PDF $p(\boldsymbol{\omega}_i|\mathbf{x}_i)$. We repeat this step to generate a random walk of a desired length.

As shown in Figure 3, each vertex $\mathbf{x}_i$ on our random walk spawns two candidate paths to feed our resampling. The first is a scattering path, using next event estimation to sample a light for $\mathbf{x}_{i+1}$. If our media emits light, we generate an emission path ending at $\mathbf{x}_i$. Both scattering and emission paths end at a light: either on a surface or in the media. Emission at intermediate vertices is ignored, as it is accounted for on shorter paths (i.e. spawned at vertex $\mathbf{x}_j$, for $j < i$).

With this procedure, the PDF of a scattering or emission path with k scattering events and $k+2$ vertices can be written as

$$p(\boldsymbol{\lambda}) = \prod_{i=1}^{k'} p(z_i|\mathbf{x}_{i-1}, \boldsymbol{\omega}_{i-1})\, G_i\, p(\boldsymbol{\omega}_i|\mathbf{x}_i)\,, \quad (16)$$

where $k' = k$ for scattering paths and $k+1$ for emission paths. Here, the PDF of sampling a direction is

$$p(\boldsymbol{\omega}_i|\mathbf{x}_i) = \begin{cases} \rho_i & \text{if } i < k', \\ p_{\mathrm{NEE}}(\boldsymbol{\omega}_k|\mathbf{x}_k) & \text{if } i = k' \text{ and scattering path,} \\ 1 & \text{if } i = k' \text{ and emission path.} \end{cases} \quad (17)$$

where $p_{\mathrm{NEE}}(\boldsymbol{\omega}_k|\mathbf{x}_k)$ represents the light sampling PDF used for next event estimation.

The PDF of sampling a distance z_i along a ray from $\mathbf{x}_{i-1}$ towards $\boldsymbol{\omega}_{i-1}$ depends on the sampling method used. When using RIS without ReSTIR, delta tracking [Woodcock et al. 1965] is a convenient choice for this task. Delta tracking has a very desirable PDF

$$p(z_i|\mathbf{x}_{i-1}, \boldsymbol{\omega}_{i-1}) = T(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)\, \sigma_t(\mathbf{x}_i)\,. \quad (18)$$

A problem with this PDF is transmission T is either not available in closed form or expensive to compute, and it must be reevaluated repeatedly (as part of $p(\boldsymbol{\lambda})$) when resampling via RIS. Fortunately, we can select target PDF $\hat{p}(\boldsymbol{\lambda})$ to cancel terms in $p(\boldsymbol{\lambda})$, avoiding explicit evaluation of T in $p(\boldsymbol{\lambda})$. But when spatiotemporally reusing samples, this cancellation is no longer possible; thus, we must replace delta tracking with another sampling method, as discussed in Section 4.

A random walk up to (a user-defined maximum of) K steps generates up to K scattering paths with $0 < k \leq K$ and K emission paths with $0 \leq k < K$, as shown in Figure 3. But a random walk may terminate early, if it exits the media prior to the K^{th} scattering event. Let n be the total number of paths generated by a random walk (i.e. $n \leq 2K$). If sampling one of these n paths uniformly, the joint sampling PDF $\bar{p}(\boldsymbol{\lambda})$ can be written relative to the PDF of the random walk $p(\boldsymbol{\lambda})$ as

$$\bar{p}(\boldsymbol{\lambda}) = \frac{1}{n}\, p(\boldsymbol{\lambda})\,. \quad (19)$$

To generate more candidate paths, we can perform additional random walks starting from $\mathbf{x}_0$.

3.3 RIS Estimation of Volume Rendering

We generate paths using M random walks. Each random walk j produces n_j paths. Uniformly selecting one of the n_j paths as a sample for the path integral leads to high variance. Instead, we use RIS to select one path from each random walk to estimate the path integral, treating the n_j paths as stratified samples with source PDF $\bar{p}(\boldsymbol{\lambda}_j^i)$ for each path $i \in \{1, ..., n_j\}$ and using a target PDF $\hat{p}(\boldsymbol{\lambda}_j^i)$. If $M = 1$, then this RIS estimator can be written as

$$\langle L(\mathbf{x}_0, \boldsymbol{\omega}_o)\rangle_{\text{ris}}^{1,1} = E_{\hat{p}}(\boldsymbol{\lambda}_j^r)\frac{1}{n_j}\sum_{i=1}^{n_j}\frac{\hat{p}(\boldsymbol{\lambda}_j^i)}{\bar{p}(\boldsymbol{\lambda}_j^i)} = E_{\hat{p}}(\boldsymbol{\lambda}_j^r)\sum_{i=1}^{n_j}\frac{\hat{p}(\boldsymbol{\lambda}_j^i)}{p(\boldsymbol{\lambda}_j^i)} \quad (20)$$

where $E_{\hat{p}}(\boldsymbol{\lambda}_j^r) = F(\boldsymbol{\lambda}_j^r)/\hat{p}(\boldsymbol{\lambda}_j^r)$ and $\boldsymbol{\lambda}_j^r$ represents the selected path. Notice the $1/n_j$ factor cancels the same value inside $\bar{p}(\boldsymbol{\lambda}_j^i)$. Our supplemental document has a more rigorous derivation. Given M random walks, we again resample to select one of the M paths $\boldsymbol{\lambda}_1^r, ..., \boldsymbol{\lambda}_M^r$ (note index r varies with j). As each candidate path $\boldsymbol{\lambda}_j^r$ comes from a prior RIS step, they must be weighted appropriately (by the running sum from the prior RIS round). Thus, the RIS estimator to select one path out of all M random walks is

$$\langle L(\mathbf{x}_0, \boldsymbol{\omega}_o)\rangle_{\text{ris}}^{1,M} = E_{\hat{p}}(\boldsymbol{\lambda}_r)\left(\frac{1}{M}\sum_{j=1}^{M}\sum_{i=1}^{n_j} w(\boldsymbol{\lambda}_j^i)\right). \quad (21)$$

We simplify the notation to $\boldsymbol{\lambda}_r$ (instead of $\boldsymbol{\lambda}_{r'}^r$) to represent the final path sample from this round of RIS. Here both $E_{\hat{p}}(\boldsymbol{\lambda}_r) = F(\boldsymbol{\lambda}_r)/\hat{p}(\boldsymbol{\lambda}_r)$ and $w(\boldsymbol{\lambda}_j^i) = \hat{p}(\boldsymbol{\lambda}_j^i)/p(\boldsymbol{\lambda}_j^i)$ depend on chosen target PDF $\hat{p}(\boldsymbol{\lambda})$.

Ideally, $\hat{p}(\boldsymbol{\lambda})$ closely matches $F(\boldsymbol{\lambda})$ but is cheaper to compute, as $\hat{p}(\boldsymbol{\lambda})$ gets evaluated for each candidate path sample. Therefore, we use $\tilde{F}(\boldsymbol{\lambda})$, a cheaper approximation of $F(\boldsymbol{\lambda})$, for paths with next event estimation

$$\hat{p}(\boldsymbol{\lambda}) = \begin{cases} \tilde{F}(\boldsymbol{\lambda}) & \text{if scattering path,} \\ F(\boldsymbol{\lambda}) & \text{if emission path.} \end{cases} \quad (22)$$

This approximation comes from simply using a cheaper transmittance estimate $\tilde{T}$ for light samples (discussed in Section 5.1):

$$\tilde{F}(\boldsymbol{\lambda}) = \Gamma_s(\boldsymbol{\lambda})\,\tilde{T}(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1})\,G(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1})\,L(\mathbf{x}_{k+1} \to \mathbf{x}_k). \quad (23)$$

This particular definition of $\hat{p}(\boldsymbol{\lambda})$ includes the same transmittance terms T as $p(\boldsymbol{\lambda})$. Thus, all T terms cancel when computing $w(\boldsymbol{\lambda}) = \hat{p}(\boldsymbol{\lambda})/p(\boldsymbol{\lambda})$, such that

$$w(\boldsymbol{\lambda}) = W_k(\boldsymbol{\lambda})\prod_{i=1}^{k}\frac{\sigma_s(\mathbf{x}_i)}{\sigma_t(\mathbf{x}_i)}, \quad (24)$$

where

$$W_k(\boldsymbol{\lambda}) = \begin{cases} \frac{\rho_k\,\tilde{T}(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1})\,G_{k+1}\,L^s(\mathbf{x}_{k+1} \to \mathbf{x}_k)}{p_{\text{NEE}}(\boldsymbol{\omega}_k|\mathbf{x}_k)} & \text{if scattering path,} \\ \frac{\sigma_a(\mathbf{x}_{k+1})}{\sigma_t(\mathbf{x}_{k+1})}\,L_e^m(\mathbf{x}_{k+1} \to \mathbf{x}_k) & \text{if emission path.} \end{cases} \quad (25)$$

This particular choice for $\hat{p}(\boldsymbol{\lambda}_r)$ also simplifies the computation of the $E_{\hat{p}}(\boldsymbol{\lambda}_r) = F(\boldsymbol{\lambda}_r)/\hat{p}(\boldsymbol{\lambda}_r)$ term, such that

$$E_{\hat{p}}(\boldsymbol{\lambda}_r) = \begin{cases} \frac{T(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1})}{\tilde{T}(\mathbf{x}_k \leftrightarrow \mathbf{x}_{k+1})} & \text{if scattering path,} \\ 1 & \text{if emission path.} \end{cases} \quad (26)$$

Note the only remaining T term is needed to compute direct illumination for the one chosen path sample $\boldsymbol{\lambda}_r$ in $E_{\hat{p}}(\boldsymbol{\lambda}_r)$. This T term appears in no PDFs, only final shading, so we can afford unbiased estimates or even analytical methods. We discuss our choices for evaluating and estimating T in Section 5.1.

All source PDFs, however, cannot use stochastic estimation or biased approximation. While resampling allows arbitrarily *defining* $\hat{p}(\boldsymbol{\lambda})$ (including approximations), approximating source PDF $p(\boldsymbol{\lambda})$ after choosing sample $\boldsymbol{\lambda}$ introduces estimation error. By carefully choosing $\hat{p}(\boldsymbol{\lambda})$, we avoid expensive T terms in our PDFs and allows building an efficient RIS estimator for volume rendering with multiple scattering and volumetric emission.

This RIS estimator can be performed in a streaming manner using weighted reservoir sampling [Chao 1982], such that only the one selected sample per pixel is stored, instead of explicitly storing all M candidate path samples. We provide implementation details in our supplemental document.

4 SPATIOTEMPORAL REUSE

Quality of the RIS estimator in Equation 21 depends on the candidate sample count M. By reusing a pixel's candidate samples when evaluating neighbor pixels, we can substantially increase the effective per-pixel candidate sample count with minimal overhead. Similar to the direct illumination sampling in ReSTIR [Bitterli et al. 2020], we leverage streaming RIS and screen-space spatiotemporal reuse, storing intermediates in per-pixel reservoirs.

We generate M candidate samples $\boldsymbol{\lambda}_1^r, ..., \boldsymbol{\lambda}_M^r$ for each pixel. Each pixel selects one candidate $\boldsymbol{\lambda}_r$ using RIS and weighted reservoir sampling. We then consider the samples selected in nearby reservoirs and from the prior frame. Combining these reservoirs, we pick one sample per pixel to evaluate.

While our reuse follows the pattern of Bitterli et al. [2020], key changes are needed to reuse volumetric path samples. Generating candidate paths for spatiotemporal reuse requires sampling a closed-form PDF, which requires updating the candidate generation process in Section 3.3, as we describe in Section 4.1. We discuss reusing paths between reservoirs in Section 4.2; unlike reusing direct light samples in ReSTIR, paths can be mapped between reservoirs in different ways with varying trade-offs. We introduce changes for spatial reuse in Section 4.3, refining the transmittance estimation and removing bias via different MIS weighting. Finally, in Section 4.4 we introduce a new stochastic reprojection for temporally reusing volumetric path samples, as surface motion vectors fail in volumes. Combined spatiotemporal reuse dramatically increases effective sample count, giving better quality than either reuse alone, as shown in Figure 4.

Importantly, our work combines transmittance estimates of varying quality, as we need a closed-form PDF for distance sampling, an efficient way to resample transmittance spatiotemporally, and unbiased transmittance for final shading (see Sections 4.1 and 4.3). We exploit resampling to iteratively refine transmittance, doing more expensive computations at lower frequency (see Figure 5).

4.1 Generating Candidate Samples for Reuse

When reusing samples $\boldsymbol{\lambda}$ from neighbor pixels or prior frames, we must explicitly compute (i.e. *resample*) the target PDF $\hat{p}(\boldsymbol{\lambda})$ at the

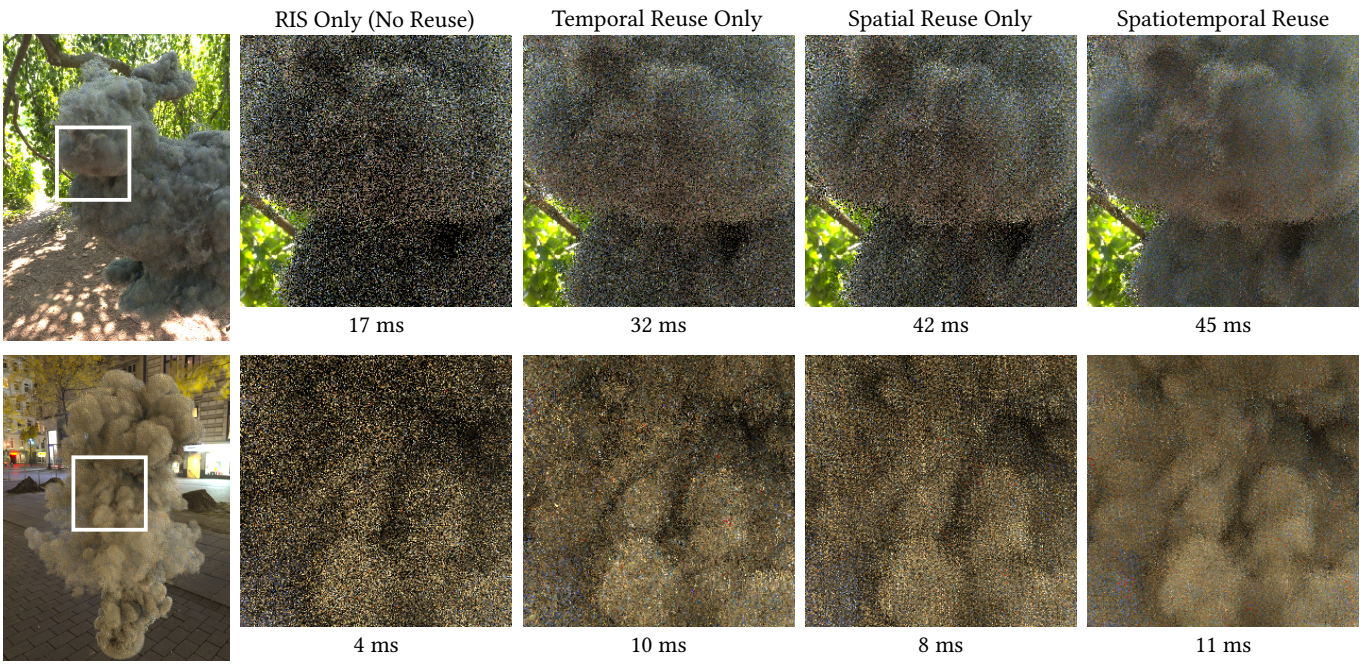


Fig. 4. *Both spatial and temporal reuse improve quality significantly. (Left) scenes with spatiotemporal reuse, and (right) insets comparing quality and performance without reuse, with only spatial or temporal reuse alone, and with spatiotemporal reuse. (Top) the Bunny Cloud scene uses a quickly rotating environment map and (bottom) the Plume scene has dynamic volume data. See the supplementary video for animated results.*

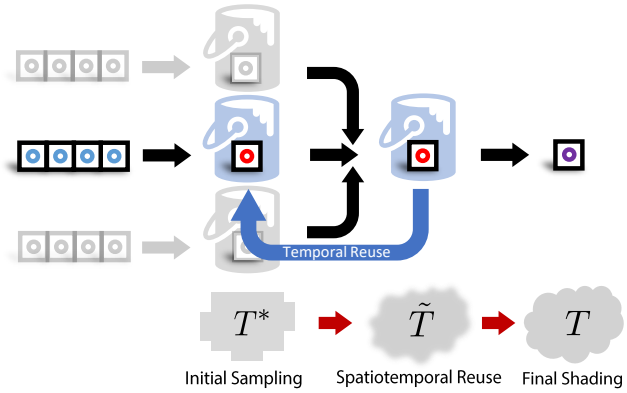


Fig. 5. *During sample reuse, transmittance gets refined from an initial piecewise constant approximation T^* to trilinear interpolation for ray marching $\tilde{T}$ to an analytical evaluation for shading T. But we always compute transmittance for NEE via ray marching (except final shading). Buckets visualize reservoirs, with initial candidates marked in blue, reservoir samples in red, and final shaded samples in purple.*

current pixel and frame. Thus, using a target $\hat{p}(\boldsymbol{\lambda})$ containing expensive transmittance terms T (as in Section 3) makes spatiotemporal reuse computationally infeasible. But not including T in $\hat{p}(\boldsymbol{\lambda})$ means cancellation will not occur (in Equation 24) while computing $w(\boldsymbol{\lambda}) = \hat{p}(\boldsymbol{\lambda})/p(\boldsymbol{\lambda})$, requiring explicit transmittance computation for each candidate path (in $p(\boldsymbol{\lambda})$). To avoid this expense, we avoid using T terms in both $p(\boldsymbol{\lambda})$ and $\hat{p}(\boldsymbol{\lambda})$, replacing delta tracking (during candidate path generation) with an alternative distance sampling method with a closed-form PDF that is cheap to evaluate.

We use regular tracking [Sutton et al. 1999]. Regular tracking may not outperform delta tracking, but it can be accelerated using a piecewise-constant approximation of the volume. For voxelized volumes, all points $\mathbf{x}$ within voxel v with constant density $\sigma^*_{t,v}$ get $\sigma^*_t(\mathbf{x}) = \sigma^*_{t,v}$. Let T^* denote transmittance between two points in this piecewise-constant volume. The PDF for regular tracking with piecewise-constant volume is then

$$p(z_i|\mathbf{x}_{i-1}, \boldsymbol{\omega}_{i-1}) = T^*(\mathbf{x}_{i-1}, \mathbf{x}_i)\, \sigma^*_t(\mathbf{x}_i)\ . \tag{27}$$

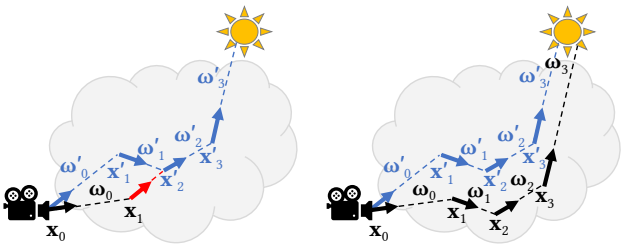
Here, the transmittance term can be written as

$$T^*(\mathbf{x}_{i-1}, \mathbf{x}_i) = \prod_v e^{-\sigma^*_{t,v} d_{i,v}}\ , \tag{28}$$

where $d_{i,v}$ is the length of line segment $\overline{\mathbf{x}_{i-1}\mathbf{x}_i}$ inside voxel v (thus, $d_{i,v} = 0$ for voxels that do not intersect $\overline{\mathbf{x}_{i-1}\mathbf{x}_i}$).

We use this piecewise-constant volume only for importance sampling, i.e., to generate candidates $\boldsymbol{\lambda}^i_j$ and evaluate their PDFs $p(\boldsymbol{\lambda}^i_j)$ and $\hat{p}(\boldsymbol{\lambda}^i_j)$. When computing final path throughput, $F(\boldsymbol{\lambda}_r)$, for the one chosen sample $\boldsymbol{\lambda}_r$ per pixel, we use the more expensive, unbiased transmittance function T and density σ_t.

Care is required when generating candidates from the piecewise-constant volume. If σ_t is non-zero *anywhere* inside voxel i, $\sigma^*_{t,i}$ for the voxel *must* be non-zero to avoid bias. For example, we cannot trilinearly interpolate a voxel grid for T and use nearest sampling for T^*. Nearest sampling can return zero some places where trilinear sampling gives non-zero density, which would introduce bias. To



(a) Vertex Reuse (b) Direction Reuse

Fig. 6. *Path $\boldsymbol{\lambda}$ (black) is created from a neighbor pixel's path $\boldsymbol{\lambda}'$ (blue). Vertex $\mathbf{x}_1$ is the same distance along primary ray $\boldsymbol{\omega}_0$ (as $\mathbf{x}'_1$ along $\boldsymbol{\omega}'_0$). Vertex reuse connects $\mathbf{x}_1$ to $\mathbf{x}'_2$ (red segment) to form the rest of $\boldsymbol{\lambda}$. Direction reuse takes $\boldsymbol{\omega}_i = \boldsymbol{\omega}'_i$ and $z_i = z'_i$ along the remaining path.*

avoid this, we use nearest sampling for T^*, except in zero-density voxels where we return the average density of their neighbors.

For further acceleration, we can use a lower resolution piecewise-constant volume for path generation. Our results (Section 5.1) show that defining this piecewise-constant volume at lower resolution than the original volume substantially improves performance with only minor quality impacts.

4.2 Path Reuse

To reuse paths we must create a path $\boldsymbol{\lambda}$ with vertices $\mathbf{x}_0, \ldots, \mathbf{x}_{k+1}$ in pixel q based on a path $\boldsymbol{\lambda}'$ from a different pixel q' with vertices $\mathbf{x}'_0, \ldots, \mathbf{x}'_{k+1}$. As both $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ start at the same camera position, we get $\mathbf{x}_0 = \mathbf{x}'_0$. However, the same is not true for the other vertices. The pixels may have different primary ray directions $\boldsymbol{\omega}_0 \neq \boldsymbol{\omega}'_0$, so the next vertex $\mathbf{x}_1 = \mathbf{x}_0 + z_1\boldsymbol{\omega}_0$ must be different as well (i.e. $\mathbf{x}_1 \neq \mathbf{x}'_1$). We can, however, use the same distance along the primary ray for both paths, such that $z_1 = z'_1$.

For the following vertices, we consider two options (Figure 6):

- **Vertex reuse** by simply setting $\mathbf{x}_i = \mathbf{x}'_i$ for $i \geq 2$, or
- **Direction reuse** by taking $\boldsymbol{\omega}_i = \boldsymbol{\omega}'_i$ and $z_i = z'_i$.

Vertex reuse reduces computation, as we need not recompute $T^*(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})$ for $i \geq 1$. However, it includes an unbounded geometry term $G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) = 1/|\mathbf{x}_2 - \mathbf{x}_1|^2$ that introduces fireflies. In Bitterli et al. [2020], singularities occur around corners and edges, but in volumes they can occur anywhere.

Direction reuse must compute $T^*(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1})$, but avoids these artifacts. It is also possible to combine both approaches by reusing directions for a desired number of scattering events then switching to vertex reuse; this reduces the probability of fireflies and bounds the cost. Our experiments show reduced noise for a slight cost increase with direction reuse (see comparison in the supplemental document), so results in the paper all rely on direction reuse. While long very paths may be initially generated, they are unlikely to be selected and reused via RIS as they carry less energy. This helps bounds the average cost of direction reuse.

As for the last vertex $\mathbf{x}'_{k+1}$, if on a light surface or in emissive media, we take $\mathbf{x}_{k+1} = \mathbf{x}'_{k+1}$. If our reused path samples the environment map, we take $\boldsymbol{\omega}_k = \boldsymbol{\omega}'_k$.

Vertex reuse in the presence of surfaces is straightforward, by simply adding surface vertices to the path. For direction reuse, if $\mathbf{x}'_i$ lies on a surface, we put $\mathbf{x}_i$ onto the closest surface along the ray starting at $\mathbf{x}_{i-1}$ with direction $\boldsymbol{\omega}_{i-1}$. Hence, if the scene does not contain a volume, only reflection directions are reused.

4.3 Spatial Reuse

After generating M per-pixel candidates, each pixel q's reservoir has selected a path $\boldsymbol{\lambda}_q$ (i.e. $\boldsymbol{\lambda}_r$ for each q). Next, we spatially reuse from neighbor pixel reservoirs, using RIS to combine the neighbor reservoirs with the current pixel's. For each neighbor q', this involves computing correction factor $w_{q'\to q}$ (per Equation 8), using

$$w_{q'\to q} = \frac{\hat{p}_q(\boldsymbol{\lambda}_{q'})}{\hat{p}_{q'}(\boldsymbol{\lambda}_{q'})} w^{\text{sum}}_{q'}, \quad \text{where} \quad w^{\text{sum}}_{q'} = \sum_{j=1}^{M} \sum_{i=1}^{n_j} \frac{\hat{p}_{q'}(\boldsymbol{\lambda}^i_j)}{p_{q'}(\boldsymbol{\lambda}^i_j)}. \tag{29}$$

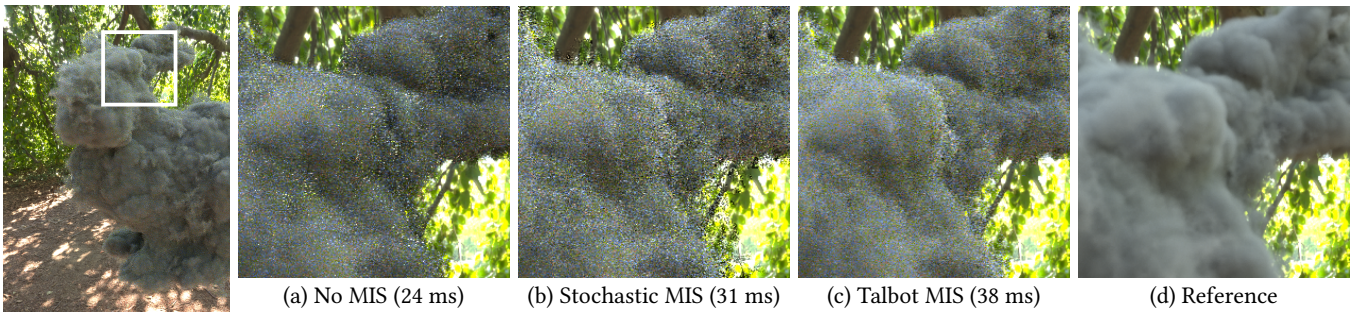
Here, the same path sample $\boldsymbol{\lambda}_{q'}$ contains different vertices in pixels q and q' due to the reuse of z_1 along different rays and later direction reuse. Therefore, when computing $\hat{p}_q/\hat{p}_{q'}$ in $w_{q'\to q}$, not all transmittance terms in the PDFs cancel, as they are evaluated for different pixels (none of them cancel with direction reuse and only some of them cancel with vertex reuse).

As noted in Section 4.1, as a byproduct of distance sampling we used a piecewise-constant volume to compute transmittance T^*. However, when recomputing transmittance values during reuse, there is no computational need for such simplification (cancellation of terms generally cannot happen between neighboring pixels). Plus, the piecewise-constant sampling enlarges non-zero density regions and leads to suboptimal sampling quality.

Instead of keeping the lower quality transmittance estimate T^*, during resampling we can update target function $\hat{p}_q$ to use higher quality transmittance than the input $\hat{p}_{q'}$ values. For resampling $\hat{p}_q$ during reuse, we compute a new transmittance estimate, $\tilde{T}$, using ray-marching with trilinearly filtered densities to improve subsequent sample quality. Because this (biased) ray marching is only used during importance sampling, and not for final throughput in $F(\boldsymbol{\lambda})$, it does not bias the rendering. An additional advantage of ray marching is the ability to tune step size, depending on our resampling budget.

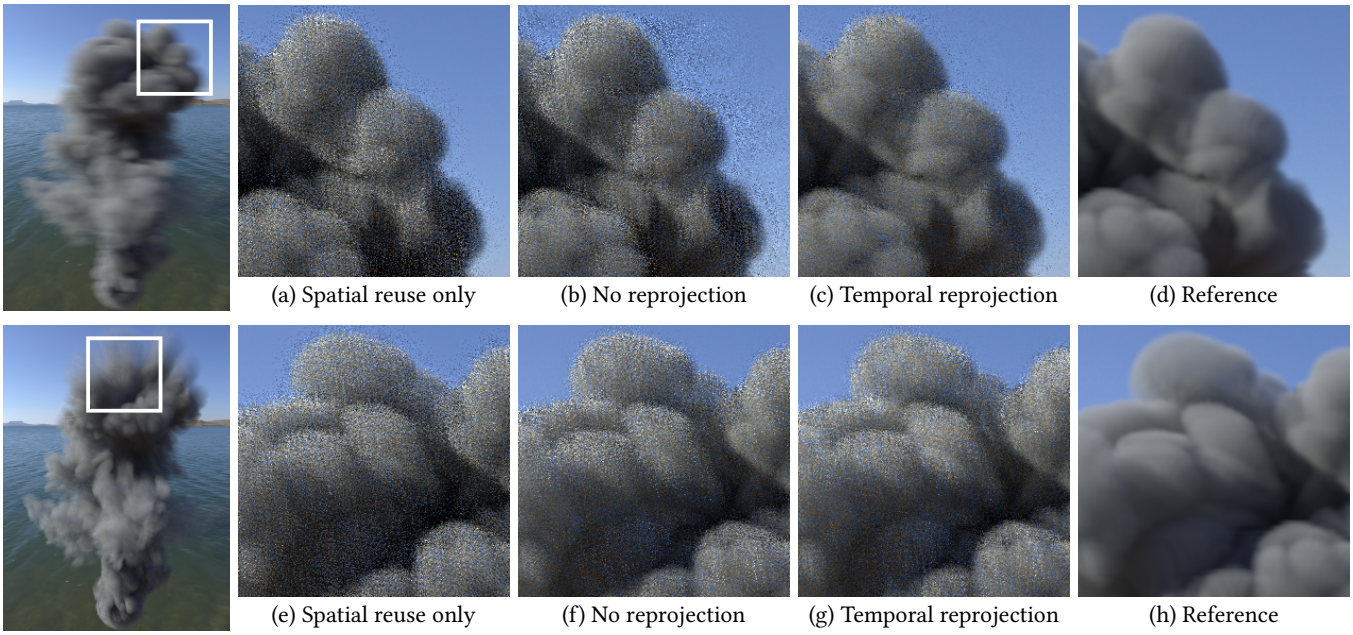
Thus, $\hat{p}_q \neq \hat{p}_{q'}$ even for $q = q'$, simply because we use an updated target function for $\hat{p}_q$ (with $\tilde{T}$ instead of T^*) for $\hat{p}_q$. This improved transmittance estimate behaves similar to Bitterli et al.'s [2020] *visibility reuse*.

We must also consider that some valid path samples $\boldsymbol{\lambda}$ for pixel q may never be sampled by a neighboring pixel q', i.e., $p_{q'}(\boldsymbol{\lambda})$ may be zero for some $\boldsymbol{\lambda}$ with non-zero $p_q(\boldsymbol{\lambda})$. Simply ignoring this introduces sampling bias, excessively darkening the results. Bitterli et al. [2020] correct this via stochastic MIS weighting (i.e. Equation 10). Although faster than the deterministic MIS (Equation 11) weighting introduced by Talbot [2005], we found stochastic MIS excessively noisy in volumes, as shown in Figure 7. Heuristically rejecting spatial neighbors based on features like surface normal or depth is effective for reducing the noise on surfaces [Wyman and Panteleev 2021], but for volumes such features are stochastic, making heuristics-based rejection challenging. Instead, we use Talbot



(a) No MIS (24 ms) (b) Stochastic MIS (31 ms) (c) Talbot MIS (38 ms) (d) Reference

Fig. 7. *Sample reuse without fireflies requires MIS to appropriately weight samples. Bitterli et al. [2020] introduced a cheaper $O(N)$ stochastic MIS, though for our volume formulation the more expensive Talbot [2005] MIS works better.*



(a) Spatial reuse only (b) No reprojection (c) Temporal reprojection (d) Reference

(e) Spatial reuse only (f) No reprojection (g) Temporal reprojection (h) Reference

Fig. 8. *Under camera motion (top) or volume deformation (bottom), temporal reprojection helps identify good samples for reuse. At left, we show references using motion blur to illustrate the magnitude of per-frame motion. At right, we show insets from a single frame without motion blur. (a,e) Only spatial reuse within the current frame. (b,f) Without temporal reprojection we reuse from inappropriate prior frame locations, causing halos and masking the noise reduction from temporal reuse. (c,g) Our novel temporal reprojection reduces the haloing and generally reduces noise. See the supplemental video for full comparison.*

MIS for spatial reuse; while it has quadratic cost, this is acceptable when using a small number of spatial neighbors.

4.4 Temporal Reuse

Temporal reuse significantly improves sample quality by incorporating knowledge from prior frames. The challenge for such reuse is finding relevant samples by *temporally reprojecting* prior frames, including changes from camera motion and volumetric deformation.

But temporal reprojection is ill-defined for volume rendering. The media in any pixel may move in many directions, so no single "correct" motion vector can tell us what prior-frame data should be reused.

We approach the problem probabilistically. With temporal reprojection we seek motion vectors that select, with high probability, prior frame reservoirs containing useful data. For example, if most media in a pixel has one motion vector, reusing a reservoir corresponding to that motion likely reduces variance best (e.g., preferentially sampling motion from denser media in a pixel instead of following a closer, wispy cloud's motion).

To that end, we use the motion vector at $\mathbf{x}_1$, the first vertex on pixel q's selected path $\boldsymbol{\lambda}_q$ (prior to spatial reuse). To compute the motion vector, we treat $\mathbf{x}_1$ as a particle such that its previous frame's position is determined by the velocity field of the volume. This randomizes the choice of motion vector, allowing any visible media to (potentially) contribute motion, using the target PDF $\hat{p}(z_1|\mathbf{x}_0, \boldsymbol{\omega}_0)$.

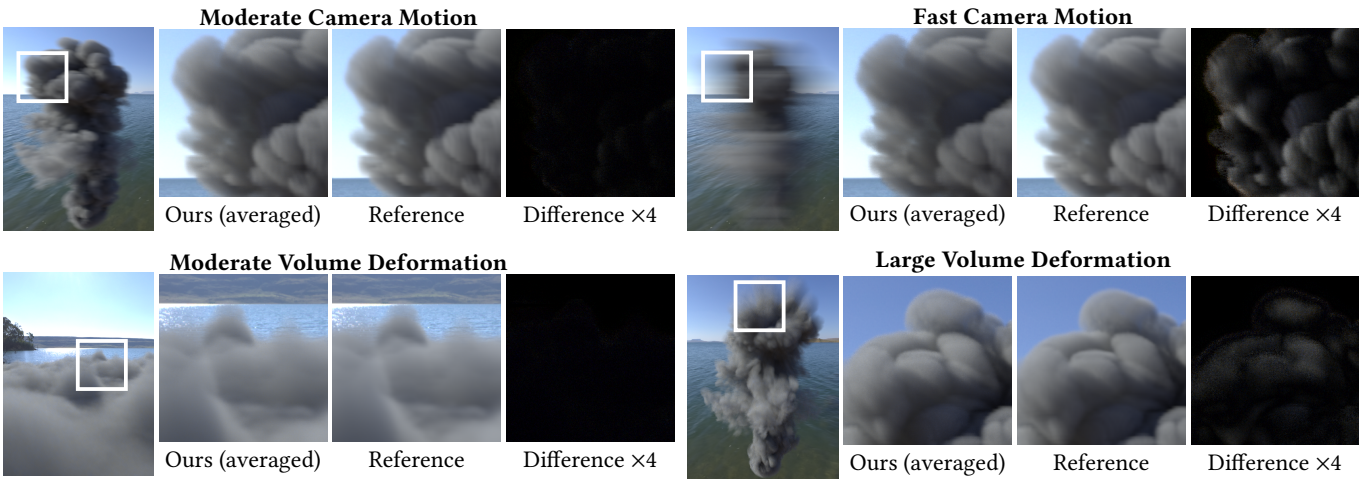


Fig. 9. *Bias in temporal reuse with (top) camera motion and (bottom) volume deformation, comparing the results of our method averaged over 256 recomputations of the same frame (to produce nearly-converged images) to reference images. (Left) with moderate motion/deformation bias is imperceptible, but (right) faster camera motion or larger volume deformation increases this bias. Note that we use a slowly deforming fog as the example for moderate volume deformation, which is different from other images. The full images on the left are rendered with motion blur to illustrate the magnitude of the camera motion or volume deformation. The insets do not include motion blur.*

Media with higher $\hat{p}(z_1|\mathbf{x}_0, \boldsymbol{\omega}_0)$ has a higher probability to provide the motion vector, which is reasonable as it contributes more pixel radiance.

Figure 8 shows examples of camera animation and volumetric deformation with and without temporal reprojection. Notice that temporal reprojection can help reduce the noise substantially.

An important limitation of this temporal reuse and reprojection is it remains unbiased only for static volumes and camera. Under camera motion or volume deformation, the chosen temporal reservoir depends on z_1 (i.e. the first scatter event). This turns the target PDFs in RIS into conditional PDFs (conditioned on z_1), introducing a slight bias during reuse if treated as a marginalized PDF.

Bias increases with larger camera motion or volume deformation (see Figure 9). But the bias is generally hard to perceive. Figure 9 shows examples with fast camera motion and large volume deformation, but only slight darkening/brightening happens. Lowering Q, the temporal limiting factor (see Section 2.5), reduces bias, and the bias disappears entirely a few frames after motion ends.

Note that dynamic lighting does not add bias. Instead, sudden lighting changes effectively lower the PDF for temporal candidates, increasing variance near lighting discontinuities.

5 IMPLEMENTATION DETAILS

The above volumetric sampling techniques can be implemented in various ways. In this section, we provide the details of our prototype.

Our implementation has four passes, similar to the flow in Figure 2. First, we generate initial candidate paths for each pixel and pick one, via RIS, to share with neighbors. Second, we reproject to find a temporal neighbor for reuse and again resample. Third, we perform spatial resampling. Finally, we evaluate the selected path sample for shading. We visualize our pipeline in Figure 5, and provide pseudocode in the supplemental document. Our reservoir stores full paths as a list of $(z_i, \boldsymbol{\omega}_i)$ tuples. Memory costs are bounded by the allowed number of scattering events, K. But supporting infinite bounces is possible by switching to vertex reuse after a few bounces and caching incident radiance of the remaining path.

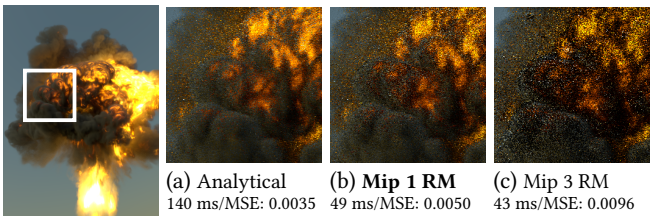
5.1 Optimizing Transmittance Computation

Transmittance plays a vital role in volumetric resampling, as it contributes significant cost to target function $\hat{p}$ and must be evaluated between every two path vertices. Prior work [Bitterli et al. 2020] notes resampling efficiency is maximized when choosing a target function $\hat{p}$ that closely approximates integrand f but is much cheaper to evaluate.

When computing transmittance $\tilde{T}$ in $\hat{p}$ for spatiotemporal reuse, we ray march a coarser volume (i.e. Mip 1, the original volume downsampled by half). Our step size for ray marching is the diagonal of a Mip 1 voxel. We lossily compress the downsampled volume with DirectX's BC4 block compression format, which compresses density values to 4 bits, giving a total 64:1 compression from the original; this greatly reduces sampling bandwidth, improving performance.

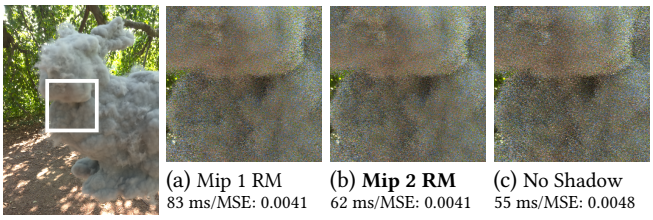
Directly rendering such volumes causes overblurring, but we use it just for importance sampling. Figure 10 compares ray marching our downsampled volume with analytical transmittance computations in the original volume. The downsampled volume slightly reduces sampling quality, but significantly improves performance. But coarsening can go too far; Figure 10c uses a Mip 3 volume for importance sampling. While cost drops further, sampling quality decreases significantly.

However, Figure 11 shows initial candidate paths can ray march a Mip 2 volume to estimate the transmittance on NEE segments (i.e. for volumetric shadows) without affecting sampling quality. This shows the benefit of incrementally injecting higher quality



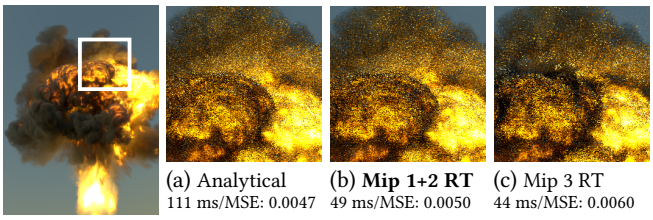
(a) Analytical 140 ms/MSE: 0.0035 (b) **Mip 1 RM** 49 ms/MSE: 0.0050 (c) Mip 3 RM 43 ms/MSE: 0.0096

Fig. 10. *When resampling, approximating transmittance by ray marching though coarser volumes (Mip 1 RM) greatly lowers cost, in exchange for a little noise (compared to analytical transmittance). But lowering resolution too far (e.g., Mip 3) adds more noise without much speedup.*



(a) Mip 1 RM 83 ms/MSE: 0.0041 (b) **Mip 2 RM** 62 ms/MSE: 0.0041 (c) No Shadow 55 ms/MSE: 0.0048

Fig. 11. *As Bitterli et al. [2020] found, injecting higher fidelity visibility incrementally during resampling improves quality without the cost to compute it everywhere. Using (c) no transmittance for NEE segments increases noise due to poorer sample quality. Adding transmittance reduces noise, but (b) even very crude approximations (e.g., ray marching a $1/4^3$ sized volume) provide most of the benefits.*



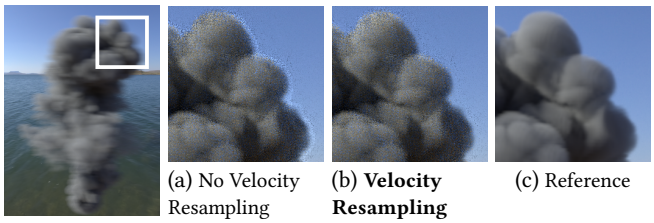
(a) Analytical 111 ms/MSE: 0.0047 (b) **Mip 1+2 RT** 49 ms/MSE: 0.0050 (c) Mip 3 RT 44 ms/MSE: 0.0060

Fig. 12. *During candidate path generation, computing transmittance analytically is costly. We use regular tracking (RT) through coarser volumes to reduce cost with little impact to sample quality (using Mip 1 for primary and Mip 2 for indirect rays). Coarsening too far (e.g., Mip 3) noticeably reduces quality without much performance win.*

transmittance into target function $\hat{p}$ over multiple rounds of RIS, rather than always using the highest quality.

We also sample distances from downsampled volumes when generating initial path candidates. Figure 12 shows that mixing Mip 1 for sampling primary path segments and Mip 2 volume for indirect path segments yields similar quality as analytical regular tracking in the original volume, but with much higher performance. Again, coarsening too far significantly skews the sampling distribution, reducing quality (see Figure 12c).

In the integrand F, we analytically compute transmittance T by traversing the voxels using piecewise-trilinear regular tracking [Szirmay-Kalos et al. 2011], giving a closed-form, unbiased transmittance. This traversal is expensive, fetching 8 density values per step to evaluate a cubic polynomial. However, we only do this for one path—the one selected for final shading. Should such a closed form solution be infeasible, we can switch to ratio tracking [Novák et al. 2014] for this final transmittance estimate; a large majorant should be used to minimize noise by forcing a smaller average step.



(a) No Velocity Resampling (b) **Velocity Resampling** (c) Reference

Fig. 13. *Temporal reprojection with and without velocity resampling, shown for a dynamic camera. (a) Overusing background motion along silhouettes causes brightening bias around the edge. (b) Velocity resampling fixes this.*

5.2 Velocity Resampling

Our temporal reprojection (Section 4.4) uses the motion vector of vertex $\mathbf{x}_1$ on each pixel's selected path $\boldsymbol{\lambda}_q$. Generally this works well, but near volume silhouettes, $\boldsymbol{\lambda}_q$ may not have vertices in the media, placing $\mathbf{x}_1$ on the background. This often moves differently than the volume, giving a halo along edges (see Figure 13).

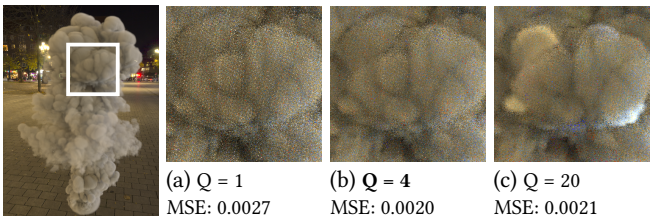
We address this with *velocity resampling*. For any $\mathbf{x}_1$ not in the volume, we generate a new distance z corresponding to a particle in the volume proportional to the free flight distance $p(z) = \sigma_t(\mathbf{x}'_1)T(\mathbf{x}_0 \leftrightarrow \mathbf{x}'_1)$ with $\mathbf{x}'_1 = \mathbf{x}_0 + z\boldsymbol{\omega}_0$, and use the motion vector for this sample. As z must be sampled in the volume, $p(z)$ is unnormalized. We approximate distribution $p(z)$ by assigning each voxel a weight proportional to its average $p(z)$, importance sampling along the ray, and uniformly picking a point within the selected voxel.

This approach increases the probability of picking a point in the media, providing better temporal reprojection. This reduces noise and bias from suboptimal reprojections, as temporal reservoirs are more likely to provide relevant paths that reduce variance during reuse.

5.3 Parameters of Spatiotemporal Reuse

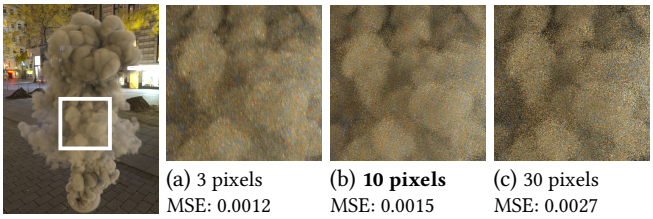
Resampling has various parameters impacting quality and performance. First, we use $M = 4$ random walks to generate initial candidates. In scattering paths, this produces fewer light samples than the 32 light samples used by Bitterli et al. [2020], but our path generation is more expensive, so we trade fewer initial samples for more spatiotemporal reuse, which maximizes the sampling efficiency in a given budget.

We typically use $Q = 4$ as the temporal limiting factor, controlling the maximum prior frame contribution. Larger Q accumulates more samples, but increases the chance of reusing stale temporal reservoirs, which can cause fireflies under large lighting changes (see Figure 14c).



(a) Q = 1 MSE: 0.0027 (b) **Q = 4** MSE: 0.0020 (c) Q = 20 MSE: 0.0021

Fig. 14. *The temporal limiting factor Q controls reuse behavior. With $Q = 1$, temporal samples get low relative weight. With $Q = 20$, older frames have more aggregate impact; this reduces overall noise, but very old stale samples can get disproportionally weighted under quickly changing illumination, causing fireflies. We use $Q = 4$.*



(a) 3 pixels MSE: 0.0012 (b) **10 pixels** MSE: 0.0015 (c) 30 pixels MSE: 0.0027

Fig. 15. *Comparing spatial reuse over different radii. A smaller radius reduces MSE, but correlation between nearby pixels becomes visually apparent. Increasing the radius reduces visual correlation, but also increases error. We use a 10 pixel radius, balancing these considerations.*

However, in scenes combining volumes and surfaces under complex illumination, we use larger Q to accumulate more effective samples to reduce noise. The value chosen depends on which issue is more problematic. In scenes containing surfaces, we use $Q = 10$.

During spatial reuse, we use a low-discrepancy sequence to sample 3 random neighbors within a 10 pixel radius. This achieves a balance between correlation artifacts and error, as shown in Figure 15. We use direction reuse by default.

6 RESULTS

We built our algorithm in the Falcor real-time rendering framework [Benty et al. 2020] using GVDB [Hoetzlein 2016] to load and access VDB assets [Museth 2013]. We captured results on an NVIDIA GeForce RTX 3090. Performance numbers include initial candidate generation, spatiotemporal reuse, and final shading. Scenes with surfaces use inline ray tracing to allow handling the surface visibility together with volumes in one compute shader.

We report error metrics and timing for 1920×1080 images, averaging over 256 frames (after a warmup) to smooth variations. We use HDR light probes and polygonal scenes with many emissive triangles to create realistic lighting environments.

Unless noted, all figures show static scenes and cameras and are fully unbiased. We do not leverage this to simplify, cancel, or otherwise reduce computation; thus, the timings are equivalent under animation. We show scenes with dynamic cameras, volumes, and lighting in the supplemental video.

We compare our results with a fast implementation of decomposition tracking [Kutz et al. 2017] (to sample free flight distances) and residual ratio tracking [Novák et al. 2014] (for estimating transmittance in NEE). We call this our *baseline*. Both decomposition and residual ratio tracking use super-voxels [Szirmay-Kalos et al. 2011] with 8× the original voxel size to store local minimum, maximum, and average density values; these control volume densities as described by Novák et al. [2014] and Kutz et al. [2017]. For GVDB, we use 8×8×8 voxel bricks [Hoetzlein 2016]. This enables storing super-voxel density bounds in brick headers, allowing efficient fetches during VDB traversal. This maximizes our baseline's performance.

Our video includes examples using the NVIDIA OptiX 7.3 temporal denoiser [NVIDIA 2017], though recent work [Hofmann et al. 2021] may provide even better denoising quality. Denoised comparisons can be found in our supplemental document and video.

6.1 Single and Multiple Scattering Results

We show our method in six scenes: the *Bunny Cloud* in two lighting configurations (Figures 1 and 16), the *Disney Cloud* (Figure 16), an *Explosion* (Figure 17), an animated *Plume* (Figure 17), the *Amazon Bistro* with a smoke plume (Figure 18), and the *Emerald Square* with fog (Figure 18). These cover various uses: emissive lights, complex environment lighting, volume self-emission, dynamic media, and volume-surface interactions. We use isotropic scattering ($g = 0$), unless otherwise stated. In all cases, our method significantly improves over the optimized baseline.

In real-time contexts, K provides an important performance knob, defining the allowable scattering events. Due to costs in dense voxel grids (e.g., the bunny), we limit evaluation in most scenes to $K = 3$. For $K > 3$, we apply Russian roulette after $\mathbf{x}_3$ to stochastically terminate a random walk according to the albedo of the scattering point. We do the same for terminating paths with the baseline. To produce equal-time comparison, we choose the number of samples per pixel (spp) for the baseline method to make the render time match our method. With each sample, the baseline method performs a random walk up to K scattering events and it performs NEE at each bounce, just like our initial path candidate generation. Note we only evaluate 1 spp per frame, in a Monte Carlo sense, but we generate and reuse many samples as part of importance sampling.

Figures 16 and 17-top explicitly compare performance and quality in three scenes with environment lighting using varying number of maximum scattering events ($K = 1$, 3, and 7). We sample the environment proportional to texel intensity. In all cases, our approach significantly reduces error compared to equal-time baseline renderings.

In the *Explosion* scene with $K = 2$ and $K = 4$ scattering events, our method significantly reduces the emission sampling noise compared to the baseline. This provides better scattering quality (emissive voxels lighting other parts of the volume) as well.

For the *Plume* scene in Figure 17, our method significantly outperforms the baseline with both single scattering and multiple scattering up to 7.

We show results of volume single scattering from light sources and surface direct lighting in the *Bistro* with over 20k emissive triangles and *Emerald Square* with around 90k emissives (Figure 18), our work enables volumes to benefit from RIS and ReSTIR, similar to surfaces [Bitterli et al. 2020]. Here, the baseline uses a light

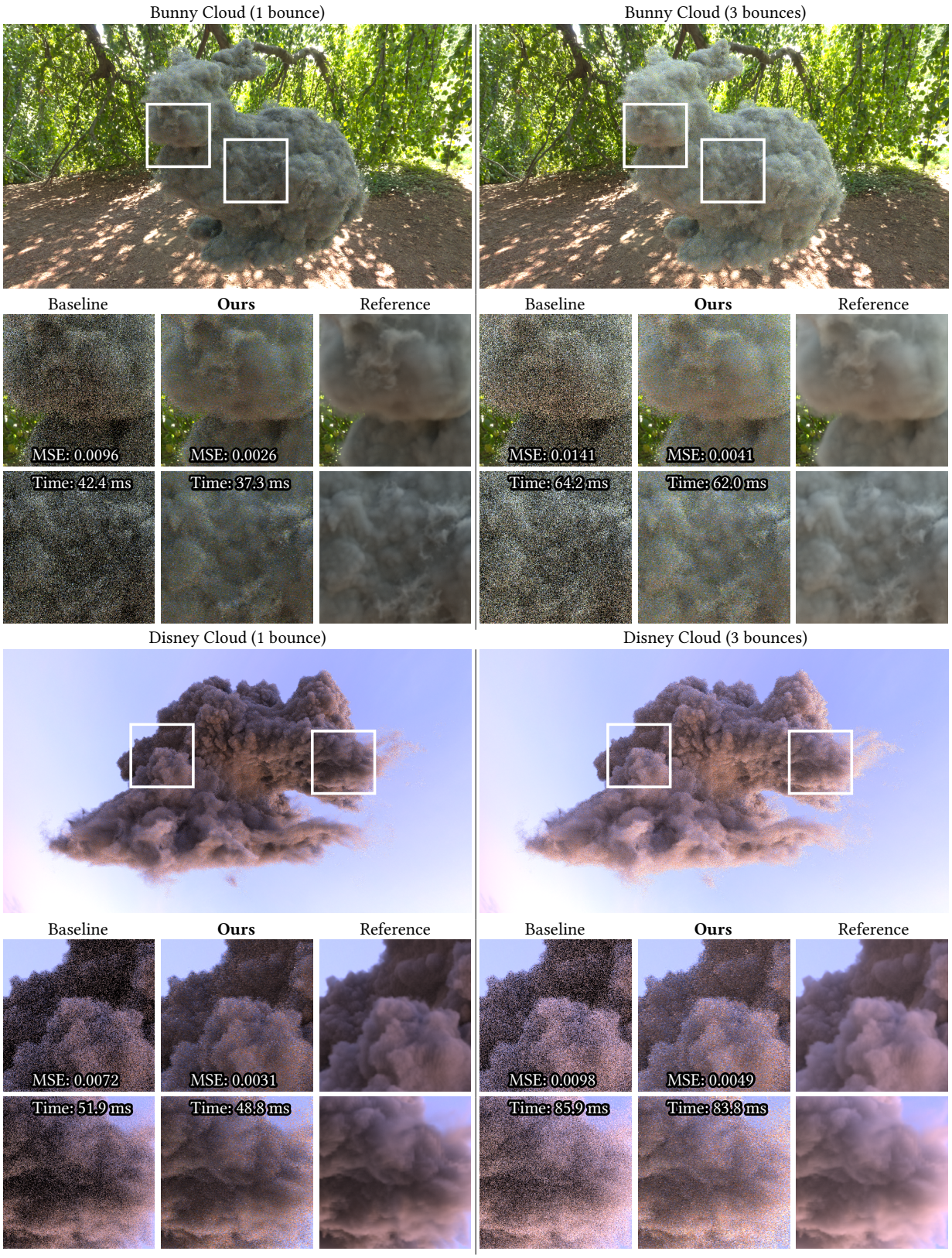


Fig. 16. *The* Bunny Cloud *and* Disney Cloud *scenes, with roughly equal-time comparisons between the baseline and our method.*



Fig. 17. *The* Plume *and* Explosion *scenes, with roughly equal-time comparisons between the baseline and our method.*

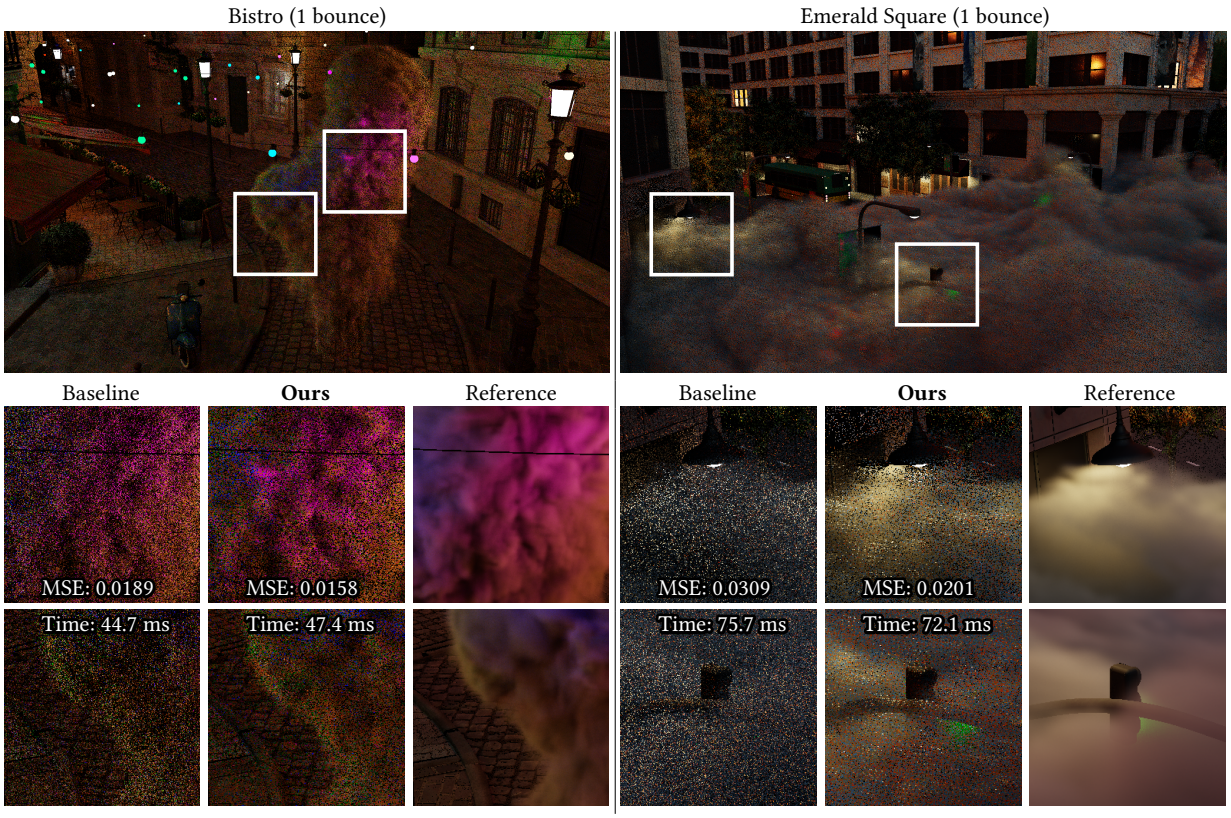


Fig. 18. *The* Bistro *and* Emerald Square *scenes, with roughly equal-time comparisons to the baseline.*

BVH [Moreau et al. 2019] for light sampling, while our method samples sources proportional to power. We show results with multiple scattering and multiple-bounce surface-volume interreflection in Figure 1. Note that this significantly increases render time, especially using multiple scattering. Profiler results reveal this stems from thread divergence between ray tracing and volume tracking.

In all scenes, our method gives lower MSE than the baseline with approximately equal or less time.

6.2 Participating Media with Different Densities

Our algorithm estimates transmittance using ray marching. Since path segments are longer in low density volumes, our algorithm accesses more data for the same model size.

In Figure 19, we scale the density of the Disney Cloud to 0.2× and 3× of the original for comparison. Our method's cost grows 47% from 83.8 ms to 123.1 ms. In comparison, the baseline method requires less time per sample, as decomposition tracking and residual ratio tracking take larger average flight distances between null collisions (due to smaller majorant).

At the lowest density, our method has slightly higher MSE than the baseline. Looking closely shows this is caused by color noise, a limitation of ReSTIR [Bitterli et al. 2020] caused by using the same scalar PDF to importance sample all color channels. We still achieve lower MSE for monochromatic images. At low densities, more background samples are produced. When the background color differs strongly from the scattered light, color noise is amplified. This is not bias; aggregating more frames reduces color noise (Figure 20) and converges to the reference. Note that increasing the initial candidate count M does not reducing color noise, as they all contribute our scalar target PDF, producing one integrand f whose chroma is randomized.

Conversely, increasing media density speeds our algorithm and makes each baseline sample more expensive. Note that we still have color noise, but its influence is smaller than the remaining variance in the overall integral.

6.3 Participating Media with High Anisotropy

To validate robustness with highly anisotropic phase functions, we change the Henyey-Greenstein asymmetry parameter g to 0.8 in the Bunny Cloud, causing strong forward-scattering (Figure 21 top). Our baseline, using only NEE, is outperformed by null scattering [Miller et al. 2019], which combines residual ratio tracking in NEE with the radiance of escaped ray (from free-flight sampling) using MIS; this is infeasible without null scattering. But our method still outperforms null scattering, despite only using NEE for light samples.

But null scattering does not always outperform decomposition tracking. For an isotropic phase function ($g = 0$) and complex lighting (Figure 21 bottom), the MIS in null scattering may not yield better quality and it adds cost to each sample. As null scattering generates fewer samples per pixel than our baseline for equal time, the

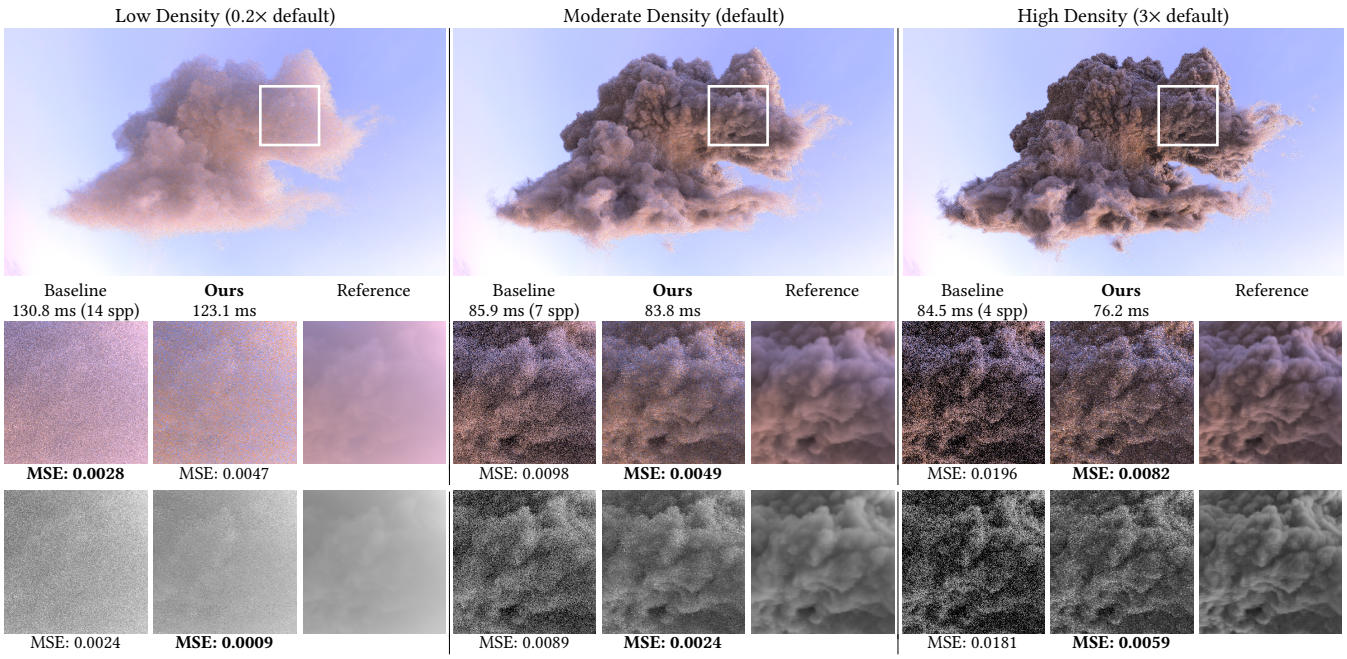
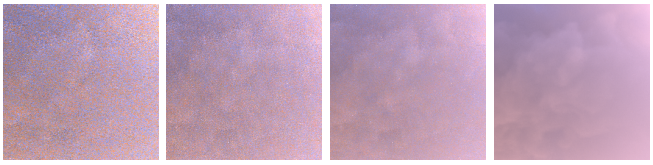


Fig. 19. *Comparing our method and the baseline in the same volume with different density multipliers. Notice that lower density makes the execution time of our method longer, while it reduces the time for each sample of baseline. We provide both color and monochromatic images to show the impact of color noise. Images shown with 3-bounce multiple scattering.*



(a) 1 frame (b) 4 frames (c) 16 frames (d) Reference

Fig. 20. *Color noise decreases when accumulating multiple frames.*

sampling efficiency becomes lower. Here our method significantly outperforms both methods, despite having some color noise.

Note we could use MIS to sample candidate lights using the MIS weights for RIS [Talbot 2005]. Since our initial path generation operates with closed-form PDFs, we do not rely on the null scattering formulation to compute MIS weights. However, this adds overhead to candidate generation. To discover when MIS is most effective requires further investigation.

6.4 Longer Time Convergence

To compare how error evolves with time, we accumulate frames of both our method and baseline, comparing errors from 100 ms to 10 seconds (Figure 22). The Bunny Cloud, Explosion, and Emerald Square scenes are selected as representative of high albedo scattering, emissive volumes, and mixed scenes with complex lighting.

The plots show our method consistently produces less error than the baseline. Note that allowing more scattering events slows convergence in both methods. Scenes mixing volumes and surfaces under complex lighting are also more challenging. The general observation is that while producing lower error than the baseline, our longer term convergence speed slows down for multiple scattering and complex surface scenes. Interestingly, our long term convergence still shows clear advantage over the baseline for emissive volumes with multiple scattering. Overall, our method is superior in 1–10s time range, suggesting our approach may be applicable to previsualization for offline rendering.

7 CONCLUSION

We introduced a sampling solution extending resampled importance sampling [Talbot 2005] and ReSTIR [Bitterli et al. 2020] to path space, enabling real-time rendering of heterogeneous volumes in complex lighting environments. Resampling exposes new user-defined target PDFs in each reuse step. By adjusting these target PDFs, even with biased or approximate distributions, we can dramatically improve the distribution quality used to select our final pixel samples.

Beyond the prior resampling work, our approach extends resampling to multi-bounce paths on surfaces and in volumes, mixes path samples of varying lengths, and shows how transmittance estimates of increasing fidelity can be injected over multiple resampling steps. We jointly sample multiple dimensions during resampling; free-flight distances and scattering directions are mixed together, unlike prior work [Bitterli et al. 2020] that exclusively considers directions. We demonstrate an efficient GPU implementation that outperforms state-of-the-art.

Our work inherits some limitations of prior resampling [Bitterli et al. 2020] techniques. For instance, we exploit coherence between

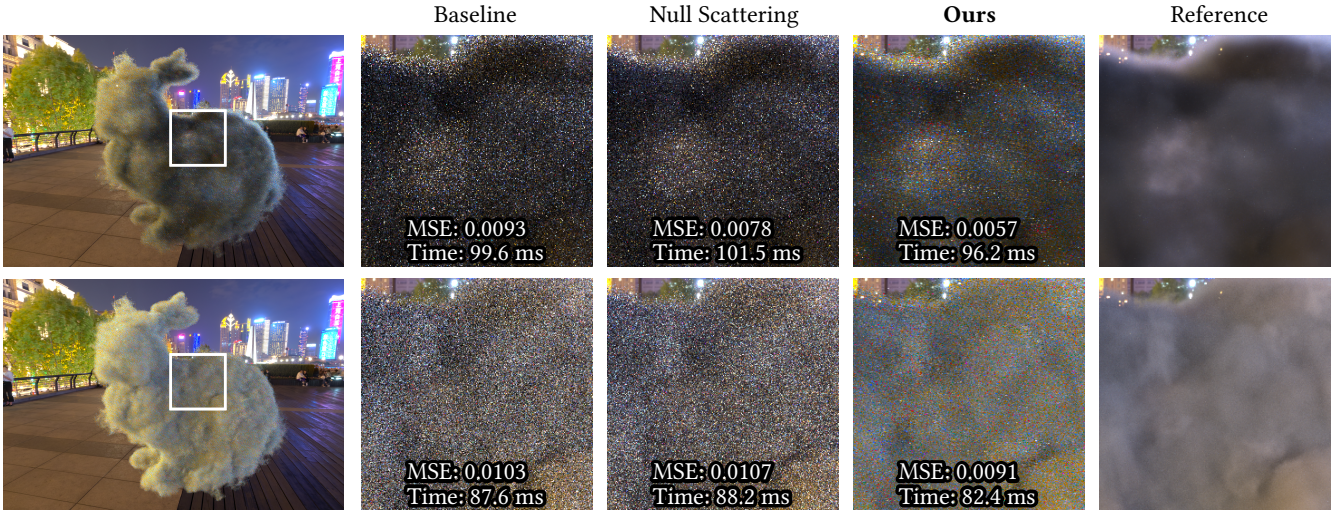


Fig. 21. *Comparing our baseline (Kutz et al. [2017] plus Novák et al. [2014]), a null scattering integration [Miller et al. 2019], and our method on a (top) highly anisotropic Bunny Cloud (Henyey-Greenstein scattering coefficient $g = 0.8$). (Bottom) For comparison, we show an isotropic Bunny Cloud under identical lighting. Images shown with 3-bounce multiple scattering.*

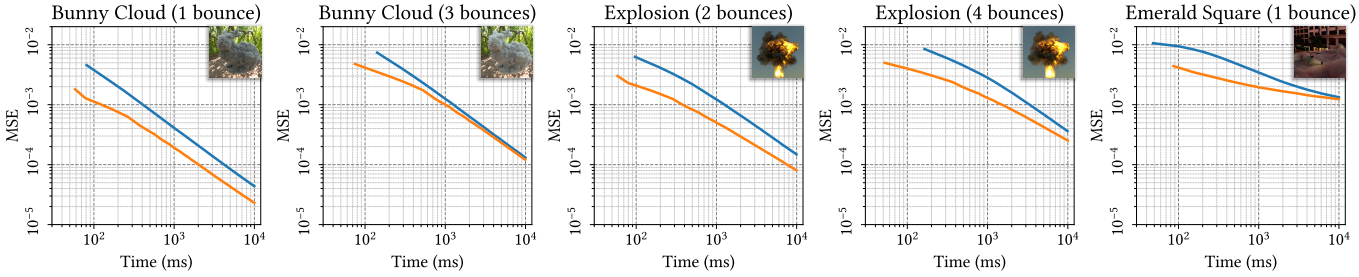


Fig. 22. *Comparing convergence between the baseline and our algorithm using log-log plots showing MSE vs. render time from 0 to 10 seconds.*

samples and perform poorly where no coherence exists. Specifically, high frequency variations (e.g., of lighting, density, motion) limit coherence across boundaries, increasing nearby variance.

Additionally, our work uses scalar target functions. This samples chroma channels identically, leaving color noise (e.g., Figure 19). Such noise is usually minor, except in scenes with different, highly-saturated lights. Using separate target functions per channel avoids this issue, but at substantial cost. Exploring efficient sampling to reduce color noise is interesting future work.

For sampling emission, a line integral which effectively combines our method with the FNEE method [Simon et al. 2017] may more efficiently collect radiance.

Another issue is the relative high cost for initial candidates and target function evaluation inside media, compared to Bitterli et al. [2020]. Coarse volumes reduce cost at the expense of quality, less accurately approximating target functions. Future work may explore adaptive representations or ray marching to speed computations while minimizing quality loss. Such improvements will accelerate our work, enabling fast, many-bounce global lighting in the presence of complex lighting, volumes, and surfaces.

As in most modern real-time renderers, our volume rendering system provides input to a denoiser. But spatiotemporal reuse can introduce correlations in the noise, a characteristic not handled well by existing denoisers (e.g., in OptiX [NVIDIA 2017]). Finding additional ways to decorrelate the noise or better adapt the denoiser are interesting future directions.

REFERENCES

Philippe Bekaert, Mateu Sbert, and John H Halton. 2002. Accelerating Path Tracing by Re-Using Paths. In *Rendering Techniques*. 125–134.

Nir Benty, Kai-Hwa Yao, Petrik Clarberg, Lucy Chen, Simon Kallweit, Tim Foley, Matthew Oakes, Conor Lavelle, and Chris Wyman. 2020. The Falcor Rendering Framework. https://github.com/NVIDIAGameWorks/Falcor

Benedikt Bitterli. 2021. *Correlations and reuse for fast and accurate physically based light transport.* Ph.D. Dissertation. Dartmouth College.

Benedikt Bitterli and Wojciech Jarosz. 2017. Beyond points and beams: Higher-dimensional photon samples for volumetric light transport. *ACM Trans. Graph.* 36, 4 (2017), 1–12.

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Trans. Graph.* 39, 4 (2020), 148:1–148:17.

Jakub Boksansky, Paula Jukarainen, and Chris Wyman. 2021. Rendering Many Lights with Grid-Based Reservoirs. In *Ray Tracing Gems II*. Springer, 351–365.

Min-Te Chao. 1982. A general purpose unequal probability sampling plan. *Biometrika* 69, 3 (1982), 653–656.

Cyril Delalandre, Pascal Gautron, Jean-Eudes Marvie, and Guillaume François. 2011. Transmittance function mapping. In *Symposium on Interactive 3D Graphics and Games*. 31–38.

Xi Deng, Shaojie Jiao, Benedikt Bitterli, and Wojciech Jarosz. 2019. Photon surfaces for robust, unbiased volumetric density estimation. *ACM Trans. Graph.* 38, 4 (2019), 46.

Eugene d'Eon and Jan Novák. 2021. Zero-variance Transmittance Estimation. In *Eurographics Symposium on Rendering - DL-only Track*.

Mathieu Galtier, Stephane Blanco, Cyril Caliot, Christophe Coustet, Jérémi Dauchet, Mouna El Hafi, Vincent Eymet, Richard Fournier, Jacques Gautrais, Anais Khuong, et al. 2013. Integral formulation of null-collision Monte Carlo algorithms. *Journal of Quantitative Spectroscopy and Radiative Transfer* 125 (2013), 57–68.

Pascal Gautron, Cyril Delalandre, Jean-Eudes Marvie, and Pascal Lecocq. 2013. Boundary-aware extinction mapping. *Computer Graphics Forum* 32, 7 (2013), 305–314.

Iliyan Georgiev, Jaroslav Krivanek, Toshiya Hachisuka, Derek Nowrouzezahrai, and Wojciech Jarosz. 2013. Joint importance sampling of low-order volumetric scattering. *ACM Trans. Graph.* 32, 6 (2013), 164–1.

Iliyan Georgiev, Zackary Misso, Toshiya Hachisuka, Derek Nowrouzezahrai, Jaroslav Křivánek, and Wojciech Jarosz. 2019. Integral formulations of volumetric transmittance. *ACM Trans. Graph.* 38, 6 (2019), 1–17.

Adrien Gruson, Binh-Son Hua, Nicolas Vibert, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2018. Gradient-domain volumetric photon density estimation. *ACM Trans. Graph.* 37, 4 (2018), 1–13.

Louis G Henyey and Jesse L Greenstein. 1941. Diffuse radiation in the galaxy. *Astrophysical Journal* 93 (1941), 70–83. https://doi.org/10.1086/144246

Sebastian Herholz, Yangyang Zhao, Oskar Elek, Derek Nowrouzezahrai, Hendrik PA Lensch, and Jaroslav Křivánek. 2019. Volume path guiding based on zero-variance random walk theory. *ACM Trans. Graph.* 38, 3 (2019), 1–19.

Sébastien Hillaire. 2015. Physically-based and Unified Volumetric Rendering in Frostbite. In *SIGGRAPH Courses: Advances in Real-Time Rendering in Games*.

Rama Karl Hoetzlein. 2016. GVDB: Raytracing Sparse Voxel Database Structures on the GPU. In *High Performance Graphics*. 109–117.

Nikolai Hofmann, Jon Hasselgren, Petrik Clarberg, and Jacob Munkberg. 2021. Interactive Path Tracing and Reconstruction of Sparse Volumes. *Proc. ACM Comput. Graph. Interact. Tech.* 4, 1, Article 5 (April 2021), 19 pages.

Jon Jansen and Louis Bavoil. 2010. Fourier opacity mapping. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. 165–172.

Wojciech Jarosz, Derek Nowrouzezahrai, Iman Sadeghi, and Henrik Wann Jensen. 2011. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Trans. Graph.* 30, 1 (2011), 1–19.

Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. 2008. The beam radiance estimate for volumetric photon mapping. In *ACM SIGGRAPH 2008 classes*. 1–112.

Henrik Wann Jensen and Per H Christensen. 1998. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 311–320.

Anton Kaplanyan and Carsten Dachsbacher. 2010. Cascaded Light Propagation Volumes for Real-Time Indirect Illumination. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 99–107.

Markus Kettunen, Eugene D'Eon, Jacopo Pantaleoni, and Jan Novák. 2021. An Unbiased Ray-Marching Transmittance Estimator. *ACM Trans. Graph.* 40, 4, Article 137 (July 2021), 20 pages.

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-Domain Path Tracing. *ACM Trans. Graph.* 34, 4, Article 123 (July 2015), 13 pages.

Emmett Kilgariff, Henry Moreton, Nick Stam, and Brandon Bell. 2018. NVIDIA Turing Architecture In-Depth. https://developer.nvidia.com/blog/nvidia-turing-architecture-in-depth/. [Online; accessed 19-January-2021].

Tae-Yong Kim and Ulrich Neumann. 2001. Opacity shadow maps. In *Rendering Techniques 2001*. Springer, 177–182.

Jaroslav Křivánek and Eugene d'Eon. 2014. A zero-variance-based sampling scheme for Monte Carlo subsurface scattering. In *ACM SIGGRAPH 2014 Talks*. 1–1.

Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. 2014. Unifying points, beams, and paths in volumetric light transport simulation. *ACM Trans. Graph.* 33, 4 (2014), 1–13.

Christopher Kulla and Marcos Fajardo. 2012. Importance sampling techniques for path tracing in participating media. *Computer graphics forum* 31, 4 (2012), 1519–1528.

Peter Kutz, Ralf Habel, Yining Karl Li, and Jan Novák. 2017. Spectral and decomposition tracking for rendering heterogeneous volumes. *ACM Trans. Graph.* 36, 4 (2017), 1–16.

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-Domain Metropolis Light Transport. 32, 4, Article 95 (July 2013), 12 pages.

Zander Majercik, Jean-Philippe Guertin, Derek Nowrouzezahrai, and Morgan McGuire. 2019. Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields. *Journal of Computer Graphics Techniques (JCGT)* 8, 2 (5 June 2019), 1–30.

Johannes Meng, Johannes Hanika, and Carsten Dachsbacher. 2016. Improving the Dwivedi sampling scheme. *Computer Graphics Forum* 35, 4 (2016), 37–44.

Bailey Miller, Iliyan Georgiev, and Wojciech Jarosz. 2019. A null-scattering path integral formulation of light transport. *ACM Trans. Graph.* 38, 4 (2019), 1–13.

Pierre Moreau, Matt Pharr, Petrik Clarberg, M Steinberger, and T Foley. 2019. Dynamic Many-Light Sampling for Real-Time Ray Tracing. In *High Performance Graphics (Short Papers)*. 21–26.

Ken Museth. 2013. VDB: High-Resolution Sparse Volumes with Dynamic Topology. *ACM Trans. Graph.* 32, 3, Article 27 (2013), 22 pages.

Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz. 2012a. Progressive virtual beam lights. *Computer Graphics Forum* 31, 4 (2012), 1407–1413.

Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz. 2012b. Virtual ray lights for rendering scenes with participating media. *ACM Trans. Graph.* 31, 4 (2012), 1–11.

Jan Novák, Andrew Selle, and Wojciech Jarosz. 2014. Residual ratio tracking for estimating attenuation in participating media. *ACM Trans. Graph.* 33, 6 (2014), 179–1.

Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. 2018. Monte Carlo methods for volumetric light transport simulation. *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)* 37, 2 (May 2018).

NVIDIA. 2017. NVIDIA® OptiX™ AI-Accelerated Denoiser. https://developer.nvidia.com/optix-denoiser

Yaobin Ouyang, Shiqiu Liu, Markus Kettunen, Matt Pharr, and Jacopo Pantaleoni. 2021. ReSTIR GI: Path Resampling for Real-Time Path Tracing. *Computer Graphics Forum* (2021).

Matthias Raab, Daniel Seibert, and Alexander Keller. 2008. Unbiased global illumination with participating media. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*. Springer, 591–605.

Marco Salvi, Kiril Vidimče, Andrew Lauritzen, and Aaron Lefohn. 2010. Adaptive volumetric shadow maps. *Computer Graphics Forum* 29, 4 (2010), 1289–1296.

Florian Simon, Johannes Hanika, Tobias Zirr, and Carsten Dachsbacher. 2017. Line Integration for Rendering Heterogeneous Emissive Volumes. *Computer Graphics Forum* 36 (July 2017), 101–110.

TM Sutton, FB Brown, FG Bischoff, DB MacMillan, CL Ellis, JT Ward, CT Ballinger, DJ Kelly, and L Schindler. 1999. *The physical models and statistical procedures used in the RACER Monte Carlo code*. Technical Report. Knolls Atomic Power Lab.

László Szirmay-Kalos, Iliyan Georgiev, Milán Magdics, Balazs Molnar, and David Legrady. 2017. Unbiased Light Transport Estimators for Inhomogeneous Participating Media. *Computer Graphics Forum* 36 (May 2017), 9–19.

László Szirmay-Kalos, Balázs Tóth, and Milán Magdics. 2011. Free Path Sampling in High Resolution Inhomogeneous Participating Media. *Computer Graphics Forum* 30 (March 2011), 85–97.

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

Justin F Talbot. 2005. *Importance resampling for global illumination*. Master's thesis. Brigham Young University.

Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. 2019. Path Guiding in Production. In *ACM SIGGRAPH 2019 Courses* (Los Angeles, California). ACM, New York, NY, USA, Article 18, 77 pages.

E Woodcock, T Murphy, P Hemmings, and S Longworth. 1965. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Proc. Conf. Applications of Computing Methods to Reactor Problems*, Vol. 557.

Chris Wyman. 2016. Exploring and Expanding the Continuum of OIT Algorithms. In *High Performance Graphics*. 1–11.

Chris Wyman and Alexey Panteleev. 2021. Rearchitecting Spatiotemporal Resampling for Production. In *High-Performance Graphics - Symposium Papers*.

Yonghao Yue, Kei Iwasaki, B. Chen, Y. Dobashi, and T. Nishita. 2011. Toward Optimal Space Partitioning for Unbiased, Adaptive Free Path Sampling of Inhomogeneous Participating Media. *Computer Graphics Forum* 30 (2011).

Cem Yuksel and John Keyser. 2008. Deep Opacity Maps. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2008)* 27, 2 (2008), 675–680.