Bernstein Bounds for Caustics

ZHIMIN FAN, State Key Lab for Novel Software Technology, Nanjing University, China
CHEN WANG and YIMING WANG, State Key Lab for Novel Software Technology, Nanjing University, China
BOXUAN LI and YUXUAN GUO, State Key Lab for Novel Software Technology, Nanjing University, China
LING-QI YAN, University of California, Santa Barbara, United States of America
YANWEN GUO, State Key Lab for Novel Software Technology, Nanjing University, China
JIE GUO*, State Key Lab for Novel Software Technology, Nanjing University, China

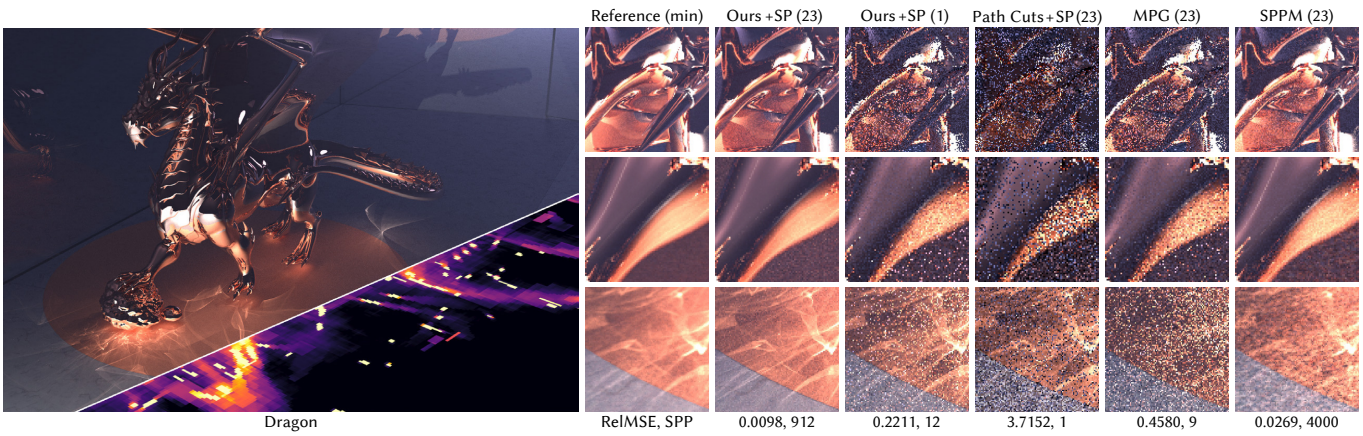

Fig. 1. Rendering sharp caustics reflected by complex geometry (0.35M triangles), where existing methods perform slowly. Consequently, even if deterministically searching for the complete set of admissible paths, they still produce high variance due to low sample rates. Our method samples triangles leveraging the bounds for caustics, leading to more converged results. We visualize the irradiance bound (in the base 10 logarithmic space) summed over tuples. All methods render single reflections only. We compare with Path Cuts [Wang et al. 2020], Specular Polynomials (SP) [Fan et al. 2024], Manifold Path Guiding (MPG) [Fan et al. 2023], and Stochastic Progressive Photon Mapping (SPPM) [Hachisuka and Jensen 2009]. Two budgets for ours focus on equal time (32 sec for precomputation with finer subdivisions, 23 min in total) and roughly equal quality comparisons (9 sec for precomputation, 1 min in total), respectively.

Systematically simulating specular light transport requires an exhaustive search for triangle tuples containing admissible paths. Given the extreme inefficiency of enumerating all combinations, we significantly reduce the search domain by stochastically sampling such tuples. The challenge is to design proper sampling probabilities that keep the noise level controllable. Our key insight is that by bounding the irradiance contributed by each triangle tuple at a given position, we can sample a subset of triangle tuples with potentially high contributions. Although low-contribution tuples are assigned a negligible probability, the overall variance remains low.

Therefore, we derive position and irradiance bounds for caustics casted by each triangle tuple, introducing a bounding property of rational functions on a Bernstein basis. When formulating position and irradiance expressions into rational functions, we handle non-rational parts through remainder variables to maintain bounding validity. Finally, we carefully design the sampling probabilities by optimizing the upper bound of the variance, expressed only using the position and irradiance bounds.

The bound-driven sampling of triangle tuples is intrinsically unbiased even without defensive sampling. It can be combined with various unbiased and biased root-finding techniques within a local triangle domain. Extensive evaluations show that our method enables the fast and reliable rendering of complex caustics effects. Yet, our method is efficient for no more than two specular vertices, where complexity grows sublinearly to the number of triangles and linearly to that of emitters, and does not consider the Fresnel and visibility terms. We also rely on parameters to control subdivisions.

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

Additional Key Words and Phrases: specular, caustics

ACM Reference Format:
Zhimin Fan, Chen Wang, Yiming Wang, Boxuan Li, Yuxuan Guo, Ling-Qi Yan, Yanwen Guo, and Jie Guo. 2025. Bernstein Bounds for Caustics. *ACM Trans. Graph.* 44, 4, Article 1 (August 2025), 15 pages. https://doi.org/10.1145/3618360

*Corresponding author.

Authors' addresses: Zhimin Fan, zhiminfan2002@gmail.com, State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China; Chen Wang, 02yimingwang@gmail.com; Yiming Wang, 02yimingwang@gmail.com, State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China; Boxuan Li, tianyuxiaoty@outlook.com; Yuxuan Guo, 213210060@seu.edu.cn, State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China; Ling-Qi Yan, lingqi@cs.ucsb.edu, University of California, Santa Barbara, Santa Barbara, United States of America; Yanwen Guo, ywguo@nju.edu.cn, State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China; Jie Guo, guojie@nju.edu.cn, State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China.

1 INTRODUCTION

High-frequency caustics have long been a core challenge in physically-based rendering. To discover the set of admissible paths that satisfy specular constraints, many specialized methods have been proposed [Hanika et al. 2015; Jakob and Marschner 2012]. Generally, all these methods involve a root-finding process to connect a pair of non-specular endpoints (e.g., a point on the light source and a diffuse shading point), but the domain they operate on differs.

While walking on the specular manifold over the whole scene enjoys great generality [Hanika et al. 2015; Jakob and Marschner 2012; Zeltner et al. 2020], its point-sampling nature makes the convergence hard to guarantee, which could sometimes produce extremely high variance in complex scenes. On the other hand, systematic approaches migrate this unbounded convergence by solving for specular paths within each tuple of triangles [Fan et al. 2024; Walter et al. 2009; Wang et al. 2020]. Yet, their overall efficiency largely depends on the selection of these triangle tuples. Currently, they rely on interval arithmetics to prune non-contributing regions, which are loose, deterministic, and without energy considerations.

Deterministic enumeration produces pixel-perfect rendering results but incurs significant computational costs. In practice, stochastic sampling is crucial for achieving an efficient and unbiased solution; however, maintaining low variance is essential to avoid noisy results. To facilitate such efficient sampling of specular triangle tuples, it is necessary to gather information on the distribution of irradiance each triangle tuple contributes to the receiver.

Conventional approaches, such as path guiding, fit energy distributions from path or photon samples. Despite their generality and flexibility, they could encounter many pitfalls. For example, point samples can easily miss certain parts so that the variance could be extremely high. Besides, online training still requires a (usually uniform) initial distribution. To our knowledge, a reliable (i.e., error-bounded) and self-contained approach for sampling specular paths that enjoys theoretically controllable variance is a clear gap.

Our key insight is that if we can acquire a conservative bound of position and irradiance of caustics cast by each triangle tuple, we can construct a stochastic estimator of the total irradiance at each shading point with controllable variance. The core challenge here is to efficiently obtain a correct bound as tightly as possible.

To this end, we develop a systematic framework to provide the bounds of positions and irradiance along specular paths within a given region. Here, the key ingredient is a rational formulation of vertex positions and Jacobians. We can easily obtain the bounds of their range by expressing these rational functions in the Bernstein polynomial basis [Garloff et al. 2012; Narkawicz et al. 2012].

Unlike conventional MC methods on point samples [Kajiya 1986; Vorba et al. 2019; Zeltner et al. 2020], our approach operates on functions within a finite interval. This may look complex, but arithmetic operations in Bernstein basis can be understood as modeling functions as Bézier surfaces and operating on their control points [Farouki 2012], which is explainable, easy to understand, and enjoys better numerical stability than monomial basis. We finally leverage our triangle sampling to develop a reliable and effective caustics rendering pipeline, which performs up to an order of magnitude faster than existing unbiased sampling approaches.

In summary, our main contribution includes:

- A Bernstein bound of position and irradiance for caustics.
- A bound-driven sampler with controllable variance.
- An efficient pipeline for rendering sharp caustics.

Our current method has some limitations. Firstly, the complexity grows rapidly as chain lengths increase. Thus, it is only feasible for one or two bounces. Besides, subdivision is required to achieve reasonably tight bounds when triangles are not small enough, which introduces some parameters. Additionally, the convergence rate of bound tightness is not yet guaranteed. In addition, we make simplifying assumptions during derivations. We ignore visibility and the Fresnel term during precomputation. Also, we compute bounds for each triangle tuple, which suffers from complexity growth sublinear to the number of triangles. Lastly, we consider a single, small light source and pure specular surfaces only. Thus, the precomputation cost scales linearly to the number of light sources.

2 RELATED WORKS

Advanced sampling techniques, such as Metropolis light transport [Veach and Guibas 1997] and path guiding [Müller et al. 2017; Vorba et al. 2019], largely accelerate the convergence rate of Monte Carlo rendering. Nevertheless, caustics produced by tiny light sources and specular surfaces still pose a challenge to these local path sampling techniques [Kajiya 1986; Veach 1998; Veach and Guibas 1995]. Therefore, productions [Droske et al. 2023] have traditionally utilized photons [Georgiev et al. 2012; Hachisuka and Jensen 2009; Jensen and Christensen 1995] or regularization [Kaplanyan and Dachsbacher 2013]. However, despite their generality, these approaches could result in unexpected bias. Thus, researchers have developed specialized methods to unbiasedly estimate those sharp caustics.

Deterministic search for specular paths. Some early approaches perform an exhaustive searching process to find nearly all admissible chains connecting given endpoints. Starting from Fermat's principle, interval arithmetic and Newton's method are leveraged to identify reflection paths on parametric surfaces [Mitchell and Hanrahan 1992]. By further introducing the Implicit Function Theorem, Chen and Arvo [2000] tackles endpoint perturbations of specular paths using high-dimensional Taylor series. In more recent works, for single-refractive specular chains on triangle meshes, Walter et al. [2009] introduce a spindle test based on interval arithmetic to prune non-contributing region. Then, they employ (interval) Newton's method to find admissible paths. While this can be extended to render multi-bounce glints [Wang et al. 2020], for caustics, online pruning would experience performance degradation [Fan et al. 2024], and the interval arithmetic bound could be extremely loose. This challenge motivates us to precompute the bounds of caustics in a tighter manner, narrowing the search domain. Furthermore, as the number of solutions increases, enumerating all solutions is already time-consuming, necessitating the use of stochastic sampling.

Stochastic sampling on specular manifolds. Random walks with a Newton solver on specular manifolds enable more robust handling of specular paths for Metropolis sampling [Jakob and Marschner 2012; Kaplanyan et al. 2014], which is then applied to a regular Monte Carlo context [Hanika et al. 2015]. By further introducing

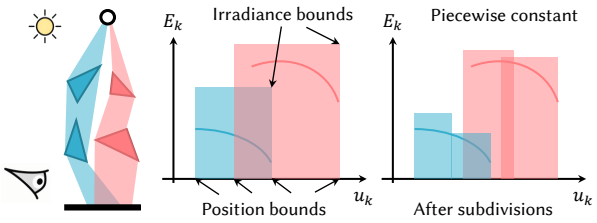

Fig. 2. **Overview of the precomputation pass.** We trace light beams passing through each triangle tuple and leverage their position and irradiance bounds to sample these triangle tuples. Note that our discussion of irradiance is primarily about its upper bound only.

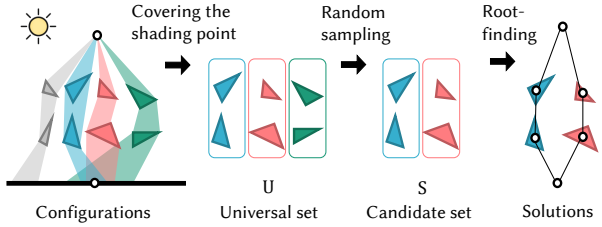

Fig. 3. **Overview of the rendering pass.** For a given shading point, the triangle tuples whose position bound covers it form a set U. We aim to sample a subset S $\subseteq$ U. For each triangle tuple $\mathcal{T} \in$ S, we run existing root-finding methods to solve for admissible paths within it.

random initialization strategies together with reciprocal estimators, unbiasedness is guaranteed [Zeltner et al. 2020], albeit at the cost of high variance. Although it is possible to eventually find all solutions, the variance could be arbitrarily high [Fan et al. 2023]. Instead, we adopt stochastic sampling of regions (triangle tuples), effectively controlling the variance introduced by random sampling.

Analyzing the contribution of glossy triangles. For single scattering, Loubet et al. [2020] derive a closed-form approximation of a glossy triangle's contribution. They compute a position bound for each triangle using samples, which is not conservative, so it relies on path tracing to keep unbiasedness, producing visible fireflies. Even worse, for pure specular surfaces, path tracing cannot find caustics at all so it leads to bias. This inspires us to seek a theoretically conservative bound, leveraging the Bernstein polynomials.

Beam tracing for caustics rendering. Our method revisits the concept of beam tracing [Heckbert and Hanrahan 1984], traditionally applied to caustics rendering approximately. Watt [1990] pioneered this manner by generating a beam of light for each specular polygon and projecting it onto a receiver, defining its irradiance as proportional to the area ratio of these polygons. Despite subsequent improvements in accuracy [Shinya et al. 1987] and performance [Iwasaki et al. 2003], they rely on heuristics that assume constant or linear variations in directions or irradiance. Even more, actual caustics do not necessarily form polygons on the receiver. These limitations motivate our more accurate modeling of caustic position and irradiance based on a rational formulation. Besides, we only use precomputation to drive stochastic sampling in rendering.

Bernstein polynomials. The Bernstein polynomial basis [Bernstein 1912] has been widely adopted in computer-aided design to model parametric curves and surfaces [Farouki 2012]. A significant advantage is that the Bernstein coefficients of a polynomial offer valuable insights into its behavior over a finite interval, leading to numerous beneficial properties. In this work, we leverage a key property of the Bernstein coefficients, which provide conservative bounds for rational functions, demonstrating better tightness (see Fig. 4) than conventional methods like interval arithmetic [Garloff et al. 2012].

3 MOTIVATION AND OVERVIEW

Monte Carlo rendering of caustics involves sampling specular chains $\bar{\boldsymbol{x}}$ comprised of specular vertices $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{k-1}$, which connects two non-specular endpoints $\boldsymbol{x}_0$ and $\boldsymbol{x}_k$. Typically, $\boldsymbol{x}_0$ lies on the light source, and $\boldsymbol{x}_k$ is a non-specular shading point. Each specular vertex $\boldsymbol{x}_i$ lies on a triangle[1] $\mathcal{T}_i$. We refer to previous works for a more gentle introduction of how this process works with a path tracer [Fan et al. 2023; Hanika et al. 2015; Zeltner et al. 2020].

Our method divides this sampling process into two separate steps: first, sampling some tuples of triangles given endpoints, and second, solving for all specular chains within each selected tuple $\mathcal{T} = (\mathcal{T}_1, ..., \mathcal{T}_{k-1})$. Note that we focus on a single light source. If multiple emitters are involved, they should be treated as part of the tuple $\mathcal{T}$. While existing works proposed ways to solve for paths within $\mathcal{T}$ [Fan et al. 2024; Walter et al. 2009; Wang et al. 2020] (the second step), the sampling of $\mathcal{T}$ (the first step) is a clear gap.

The goal of our method. We aim to sample a set of triangle tuples given endpoints according to their contributions. This requires a probability $P_{\mathcal{T}}$ (conditioned on $\boldsymbol{x}_k$) describing the chance we sample $\mathcal{T}$. We compute this probability in a precomputation pass, which is stored on non-specular surfaces for later use in the rendering pass. Precomputation takes the light source, specular caustics casters, and non-specular receivers as input but is independent of the camera. Instead of fitting from point samples that cannot preserve all information over the whole domain, we theoretically analyze the energy distribution of specular paths over finite intervals.

Overall pipeline. In the precomputation pass, we trace beams from the light source passing through each tuple of specular triangles. This process is illustrated in Fig. 2. For simplicity, we firstly consider point light sources only and a specific type of specular chain, that is, the chain length and the scattering type (reflection/refraction) at each vertex. This allows us to parameterize paths via the barycentric coordinates $\boldsymbol{u}_1$ of the first specular vertex $\boldsymbol{x}_1$. Each path intersects with the receiver $\mathcal{T}_k$ at vertex $\boldsymbol{x}_k$ with barycentric coordinates $\boldsymbol{u}_k$, which we formulate as a rational function of $\boldsymbol{u}_1$.

Given a box on the domain of $\boldsymbol{u}_1$, the **position bound** describes the range of $\boldsymbol{u}_k$ where the corresponding paths hit the receiver, which indicates whether we need to consider a triangle tuple $\mathcal{T}$ at a shading point $\boldsymbol{x}_k$ on $\mathcal{T}_k$. However, the contribution of an admissible path could be small in some instances. Fortunately, we found that a rational function can bound the irradiance contribution. Therefore, we additionally introduce the **irradiance bound** that describes the

[1]We first assume these triangles are given and defer how to incrementally find $\mathcal{T}_j$ given $\mathcal{T}_1, ..., \mathcal{T}_{j-1}$ (for any $j > 1$) to the implementation section.

range of irradiance received at $\boldsymbol{x}_k$. These bounds are stored as (axis-aligned) boxes in the space of $\boldsymbol{u}_k$ with irradiance values.

During the final rendering (Fig. 3), for a non-specular shading point (generated by, for example, path tracing), we query the boxes that cover this shading point and sample a subset of corresponding triangles according to their irradiance bound. We determine the sampling probability by optimizing an upper bound of variance.

All these bounds stem from the range bound of functions over a finite domain. Due to the availability of rational formulations, we obtain such bounds utilizing the Bernstein polynomials.

4 BOUNDS FOR SPECULAR PATHS

In this section, we aim to acquire a conservative bound of the irradiance distribution of caustics. We first introduce a property of Bernstein polynomials [Farouki 2012], which enables a piecewise constant approximation of any rational function.

4.1 Bernstein bound for rational functions

The Bernstein basis for multivariate functions $p(\boldsymbol{x})$ over $\boldsymbol{x} = (x_1, ..., x_m) \in \mathcal{U}^m = [0,1]^m$ are polynomials defined as

$$B_{\boldsymbol{i},\boldsymbol{n}}(\boldsymbol{x}) = \prod_{j=1}^{m} \binom{n_j}{i_j} x_j^{i_j} (1-x_j)^{n_j - i_j}, \tag{1}$$

where $\binom{n_j}{i_j}$ is the binomial coefficient, $\boldsymbol{n} = (n_1, ..., n_m)$ is the degree of the polynomial, $\boldsymbol{i} = (i_1, ..., i_m)$ is the index of basis, and j denotes the index of dimension, for all $j \in [1, m]$, $i_j \in [0, n_j]$. An essential property of Bernstein polynomials is that we can bound the range using its coefficients, which also generalizes to rational functions. Specifically, consider a rational function $f(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}$ expressed in Bernstein coefficients $b_{\boldsymbol{i}}(p)$ and $b_{\boldsymbol{i}}(q)$ in the same degree:

$$p(\boldsymbol{x}) = \sum_{i_1=0}^{n_1} \cdots \sum_{i_m=0}^{n_m} b_{\boldsymbol{i}}(p) B_{\boldsymbol{i},\boldsymbol{n}}(\boldsymbol{x}) \tag{2}$$

and similar for $q(\boldsymbol{x})$. Assuming that all $b_i(q)$ have the same sign and are non-zero, the minimal and maximal coefficient ratios can bound[2] the range of the function [Narkawicz et al. 2012]:

$$\min_{i_1=0}^{n_1} \ldots \min_{i_m=0}^{n_m} \frac{b_{\boldsymbol{i}}(p)}{b_{\boldsymbol{i}}(q)} = \underline{f} \le f(\boldsymbol{x}) \le \overline{f} = \max_{i_1=0}^{n_1} \ldots \max_{i_m=0}^{n_m} \frac{b_{\boldsymbol{i}}(p)}{b_{\boldsymbol{i}}(q)}. \tag{3}$$

The above $f(x)$ corresponds to the barycentric coordinates of the receiver vertex $\boldsymbol{u}_k(\boldsymbol{u}_1)$ and the irradiance $E_k(\boldsymbol{u}_1)$ received there; we will present their detailed formulation in Secs. 4.3 and 4.4. In these cases, the restrictions on the denominator q are not always satisfied. When not all $b_{\boldsymbol{i}}(q)$ have the same sign, we consider the **reciprocal** of $f(\boldsymbol{x})$ so that if all $b_{\boldsymbol{i}}(p)$ have the same sign, we can still get a valid bound, though may be separated into two intervals. If both are unsatisfied, the result bound is the universal set $(-\infty, +\infty)$, which typically occurs near grazing angles and focal points.

[2] To explain the high-level intuition that why Bernstein bounds are tighter than interval counterparts, we regard interval arithmetic as always using independent variables for different operands. For instance, suppose we are computing the bound of $f(x) = g(x)h(x)$ in $[0,1]$, with $g(x) = x$ and $h(x) = 4 - x$. Using interval arithmetic, we obtain $[0,1] \times [3,4] = [0,4]$. This is because it completely ignores the correlation, effectively treating it as $x(4-y)$ with two independent variables x and y. Instead, polynomial operations keep correlations due to the common variable x. As we obtain Bernstein coefficients $b_0 = 0, b_1 = 2, b_2 = 3$, we get a tighter bound $[0,3]$.

Table 1. List of important symbols.

Symbol	Description
$\mathcal{T}_k$	Receiver triangle
$\mathcal{T} = (\mathcal{T}_1, ..., \mathcal{T}_{k-1})$	Specular triangle tuple
$\boldsymbol{x}_0$	Position of the point light
$\overline{\boldsymbol{x}} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k-1})$	Position of the specular vertices
$\boldsymbol{x}_k$	Position of the shading point
$\boldsymbol{n}_i$	Un-normalized interpolated normal of $\boldsymbol{x}_i$
$\boldsymbol{d}_i$	Position difference of vertices $\boldsymbol{x}_{i+1}$ and $\boldsymbol{x}_i$
$\boldsymbol{u}_i = (u_i, v_i)$	Barycentric coordinate of $\boldsymbol{x}_i$
$\boldsymbol{p}_{i,0}, \boldsymbol{p}_{i,1}, \boldsymbol{p}_{i,2}$	Vertex positions
$\boldsymbol{e}_{i,1}, \boldsymbol{e}_{i,2}$	Vector of triangle edges
$\boldsymbol{n}_{i,0}, \boldsymbol{n}_{i,1}, \boldsymbol{n}_{i,2}$	Vertex normals
$E_k(\boldsymbol{u}_1)$	Path's irradiance received at $\boldsymbol{x}_k$
$E(\mathcal{T}, \boldsymbol{u}_k, \mathcal{T}_k)$	Tuple's irradiance received at $\boldsymbol{x}_k$
$\underline{f}, \overline{f}$	Lower/upper bound of function f's range

A constant bound with substantial internal variation is intrinsically loose. Fortunately, the Bernstein bound of polynomials enjoys quadratic convergence with respect to the length of the interval [Garloff 1986], so we can also improve the bound of rational functions via subdivisions [Narkawicz et al. 2012]. This motivates us to use a **piecewise constant bound** of the function, which acts as a piecewise constant approximation of the function. The number of pieces controls the balance between tightness and computational cost. As the number of pieces tends to infinity, the bound converges to the actual value of the functions. Thus, the infinite bound is a small portion after proper subdivisions, as shown in Figs. 1 and 13.

4.2 Bounds in specular light transport

Now, we apply the bounding property of Bernstein polynomials to specular light transport. Given a triangle tuple $\mathcal{T}$ and a box $\boldsymbol{U}_1 = [\underline{u}_1, \overline{u}_1] \times [\underline{v}_1, \overline{v}_1]$, we define the **position bound** $\boldsymbol{U}_k = [\underline{u}_k, \overline{u}_k] \times [\underline{v}_k, \overline{v}_k]$ as a box covering the range of $\boldsymbol{u}_k(\boldsymbol{u}_1)$ on the receiver $\mathcal{T}_k$:

$$\underline{u}_k \le \inf_{\boldsymbol{u}_1 \in \boldsymbol{U}_1} u_k(\boldsymbol{u}_1), \quad \overline{u}_k \ge \sup_{\boldsymbol{u}_1 \in \boldsymbol{U}_1} u_k(\boldsymbol{u}_1). \tag{4}$$

The same applies to $\underline{v}_k$ and $\overline{v}_k$. Similarly, the **irradiance bound** is the interval $\boldsymbol{E}_k = [\underline{E}_k, \overline{E}_k]$ that covers the range of $E_k(\boldsymbol{u}_1)$. Note that all these quantities depend on $\mathcal{T}$ and $\mathcal{T}_k$; we omit them for simplicity. We summarize important symbols in Table 1. Note that throughout this paper, we use $\underline{f}$ and $\overline{f}$ for the bound we compute. It differs from the supremum, infimum, and range of a function.

We compute these bounds leveraging the aforementioned ratios of Bernstein coefficients. For instance,

$$\underline{u}_k = \min_{i_1=0}^{n_1} \ldots \min_{i_m=0}^{n_m} \frac{b^u_{\boldsymbol{i}}(p)}{b^u_{\boldsymbol{i}}(q)}, \quad \overline{u}_k = \max_{i_1=0}^{n_1} \ldots \max_{i_m=0}^{n_m} \frac{b^u_{\boldsymbol{i}}(p)}{b^u_{\boldsymbol{i}}(q)}. \tag{5}$$

Here, $b^u_{\boldsymbol{i}}$ refers to the Bernstein coefficients of $u_k(\boldsymbol{u}_1)$. Likewise, $\underline{v}_k, \overline{v}_k$ and $\underline{E}_k, \overline{E}_k$ are computed from the Bernstein coefficients of $v_k(\boldsymbol{u}_1)$ and $E_k(\boldsymbol{u}_1)$, respectively. Nevertheless, the bounding property only works for rational functions, which requires formulating coordinates $\boldsymbol{u}_k(\boldsymbol{u}_1)$ and irradiance $E_k(\boldsymbol{u}_1)$ into rational functions while keeping the bound valid.

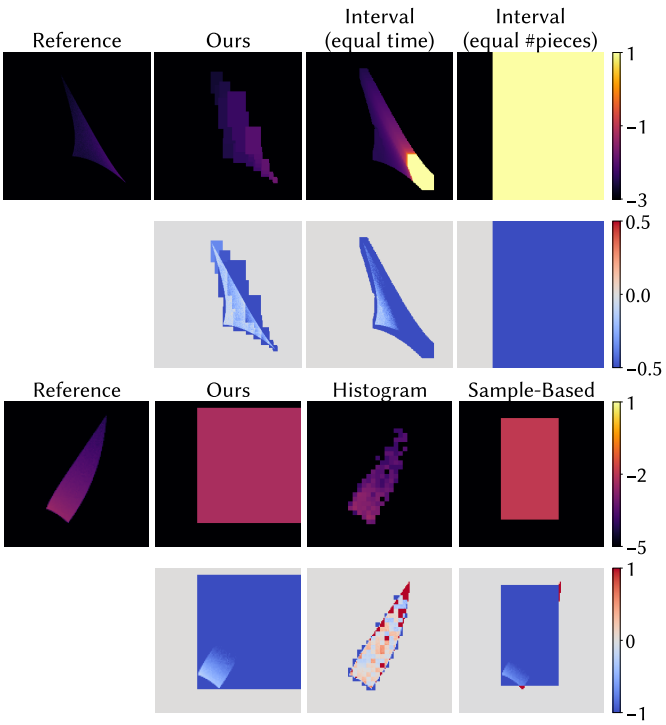

Fig. 4. **Visualizing the bound of caustics** cast by a single triangle reflector. We show the bound and its ratio with reference in the base 10 logarithmic space. Note that the axes are barycentric coordinates u_k and v_k on the **receiver**. Thus, the position bounds of different pieces may overlap. **Top:** Our bound is tighter than interval arithmetics [Wang et al. 2020] in both an equal number of pieces and equal time. Ours computes 50 pieces while interval arithmetic computes 4K pieces in equal time. **Bottom:** Utilizing 500 uniformly sampled paths (in roughly equal time) to fit irradiance distributions (Histogram) like path guiding [Jensen 1995] or use the range of samples as bounds (Sample-Based) [Loubet et al. 2020] are not conservative and could result in fireflies or bias in rendering.

4.3 Rational formulation of vertex positions

Our rational formulation of vertex positions (coordinates) builds upon rational coordinate mapping between vertices in a specular chain [Fan et al. 2024]. Here, we briefly review the necessary formulas and refer to the original paper for derivation.

Rational coordinate mapping. Formally, we use the barycentric coordinates $\boldsymbol{u}_i = (u_i, v_i)^\top$ to represent the vertex positions, normals, and differences between consecutive vertices:

$$\boldsymbol{x}_i = \boldsymbol{p}_{i,0} + u_i(\boldsymbol{p}_{i,1} - \boldsymbol{p}_{i,0}) + v_i(\boldsymbol{p}_{i,2} - \boldsymbol{p}_{i,0}), \tag{6}$$

$$\boldsymbol{n}_i = \boldsymbol{n}_{i,0} + u_i(\boldsymbol{n}_{i,1} - \boldsymbol{n}_{i,0}) + v_i(\boldsymbol{n}_{i,2} - \boldsymbol{n}_{i,0}), \tag{7}$$

$$\boldsymbol{d}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i. \tag{8}$$

Note that $\boldsymbol{n}_i$ and $\boldsymbol{d}_i$ are un-normalized. All these expressions are functions on $\boldsymbol{u}_1$. For simplicity, we omit the function's variables.

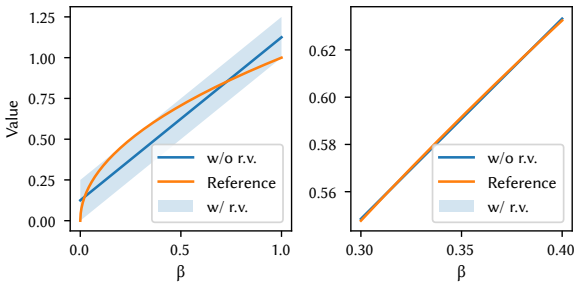

Fig. 5. **With remainder variables**, we correctly bound the range of $\sqrt{\beta}$, though looser than the proper range (left). Note that the actual range of β is smaller, so the bound could be tighter (right).

Each vertex can be represented by its preceding vertex's coordinates [Möller and Trumbore 1997]:

$$\boldsymbol{u}_{i+1} = \left(\frac{(\tilde{\boldsymbol{d}}_i \times \boldsymbol{e}_{i+1,2}) \cdot (\boldsymbol{x}_i - \boldsymbol{p}_{i+1,0})}{(\tilde{\boldsymbol{d}}_i \times \boldsymbol{e}_{i+1,2}) \cdot \boldsymbol{e}_{i+1,1}}, \frac{((\boldsymbol{x}_i - \boldsymbol{p}_{i+1,0}) \times \boldsymbol{e}_{i+1,1}) \cdot \tilde{\boldsymbol{d}}_i}{(\tilde{\boldsymbol{d}}_i \times \boldsymbol{e}_{i+1,2}) \cdot \boldsymbol{e}_{i+1,1}} \right)^\top. \tag{9}$$

Here, $\tilde{\boldsymbol{d}}_i$ is a scaled version of $\boldsymbol{d}_i$, which is determined by the scattering type at $\boldsymbol{x}_i$. We define $\boldsymbol{e}_{i,1} = \boldsymbol{p}_{i,1} - \boldsymbol{p}_{i,0}$ and similar for $\boldsymbol{e}_{i,2}$. The (un-normalized) outgoing direction at $\boldsymbol{x}_i$ is

$$\tilde{\boldsymbol{d}}_i = \begin{cases} (\boldsymbol{n}_i \cdot \boldsymbol{n}_i)\boldsymbol{d}_{i-1} - 2(\boldsymbol{d}_{i-1} \cdot \boldsymbol{n}_i)\boldsymbol{n}_i, & \text{for reflection} \\ \eta_i' \left((\boldsymbol{n}_i \cdot \boldsymbol{n}_i)\boldsymbol{d}_{i-1} - (\boldsymbol{d}_{i-1} \cdot \boldsymbol{n}_i)\boldsymbol{n}_i \right) - \sqrt{\beta_i}\boldsymbol{n}_i, & \text{for refraction} \end{cases} \tag{10}$$

with η_i' being the ratio of index of refractions and

$$\beta_i = (1 - \eta_i'^2)(\boldsymbol{n}_i \cdot \boldsymbol{n}_i)(\boldsymbol{d}_{i-1} \cdot \boldsymbol{d}_{i-1}) + \eta_i'^2 (\boldsymbol{d}_{i-1} \cdot \boldsymbol{n}_i)^2. \tag{11}$$

The square root is not rational. Specular polynomials handle it by introducing a precomputed piecewise rational approximant [Fan et al. 2024]. However, this approximation introduces error and thus could lead to an invalid (i.e., not conservative) bound.

Fortunately, we can correct it through auxiliary **remainder variables** that model the approximation error.

Remainder variables. We introduce a remainder variable ξ_i for each refractive vertex to compensate for the difference between $\sqrt{\beta_i}$ and a rational approximation $r(\beta_i)$. Now we can safely replace the original occurrence of $\sqrt{\beta_i}$ with a rational function

$$\tilde{r}(\beta_i, \xi_i) = r(\beta_i) + (1 - \xi_i)\underline{\delta}_i + \xi_i \overline{\delta}_i, \tag{12}$$

where $\underline{\delta}_i$ and $\overline{\delta}_i$ is the bound of the approximation error $\delta_i(\boldsymbol{u}_i) = \sqrt{\beta_i} - r(\beta_i)$. Note that ξ_i is independent of β_i. This introduces an extra dimension and corrects the bound, though it may become loose.

To guarantee the bound will converge to the true value as the domain of $\boldsymbol{u}_1$ gets infinitely small, we generate the approximation $r(\beta_i)$ on the fly, leveraging the range bound of β_i on the current domain. Specifically, we use a linear approximation (Fig. 5):

$$r(\beta_i) = a\beta_i + b. \tag{13}$$

Please refer to the supplemental document for the proof and the closed-form calculation of parameters a, b, $\underline{\delta}_i$, and $\overline{\delta}_i$.

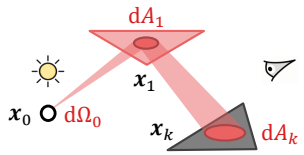

Fig. 6. **Illustration of the generalized geometric term (GGT)**. Light emitted in the differential solid angle $d\Omega_0$ passing through specular surfaces finally hits the differential area dA_k on the receiver. Note that we use $d\Omega_0$ instead of the projected solid angle $d\Omega_0^{\perp}$ because we consider point light sources that emit intensity uniformly in different directions.

Until now, by substituting Eq. (9) into Eq. (3), it is already possible to obtain a conservative position bound of specular vertices. Nevertheless, efficient sampling of triangles also requires additional information on their irradiance E_k.

4.4 Rational formulation of irradiance

Motivation. Analyzing irradiance enables us to sample triangles based on their contributions. It is particularly beneficial when specific triangles contribute minimal energy. After all, a nearly zero contribution is effectively equivalent to having no solution. An irradiance bound allows us to assign them a correspondingly low probability and focus more on paths with high contributions.

Overview. The irradiance carried by the path $\overline{\boldsymbol{x}}$ received at $\boldsymbol{x}_k$ comprises several independent factors [Jakob and Marschner 2012; Walter et al. 2009; Wang et al. 2020]:

$$E_k(\boldsymbol{u}_1) = E_k(\overline{\boldsymbol{x}}) = I_0 \frac{\mathrm{d}\Omega_0}{\mathrm{d}A_k} V(\overline{\boldsymbol{x}}) F(\overline{\boldsymbol{x}}). \tag{14}$$

We parameterize by $\boldsymbol{u}_1$ since $\overline{\boldsymbol{x}}$ is determined by $\boldsymbol{u}_1$ when considering a specific chain type. Here, I_0 is the emitter's intensity[3], $\mathrm{d}\Omega_0$ refers to the differential solid angle at the point light source $\boldsymbol{x}_0$, and $\mathrm{d}A_k$ the differential area at vertex $\boldsymbol{x}_k$. V represents the visibility term, and F is the product of Fresnel terms at each vertex. $\frac{\mathrm{d}\Omega_0}{\mathrm{d}A_k}$ is the generalized geometric term (GGT). As illustrated in Fig. 6, it describes the solid angular measure of emitted flux concentrated into a unit area on the receiver. For simplicity, we ignore F and V in the discussion since they are no greater than 1. Thus, the upper bound of irradiance is still correct. We expand $\frac{\mathrm{d}\Omega_0}{\mathrm{d}A_k}$ using the chain rule, which is similar to previous works [Kaplanyan et al. 2014]:

$$\frac{\mathrm{d}\Omega_0}{\mathrm{d}A_k} = \frac{\mathrm{d}\Omega_0}{\mathrm{d}A_1} \left|\frac{\partial \boldsymbol{x}_1}{\partial \boldsymbol{u}_1}\right| \left|\frac{\partial \boldsymbol{u}_1}{\partial \boldsymbol{u}_k}\right| \left|\frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{x}_k}\right|. \tag{15}$$

We obtain rational expressions of these four terms separately. The key part is the Jacobian $\left|\frac{\partial \boldsymbol{u}_1}{\partial \boldsymbol{u}_k}\right|$; we discuss the remaining parts in the supplemental document. Here, we provide two approaches to obtain rational forms of $\left|\frac{\partial \boldsymbol{u}_1}{\partial \boldsymbol{u}_k}\right|$.

[3]Since I_0 is completely scene-dependent, we ignore it (i.e., simply assuming $I_0 = 1$) across all the irradiance visualizations.

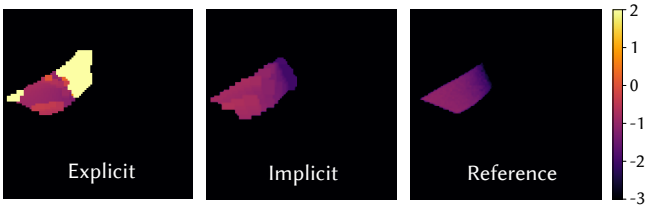

Fig. 7. The irradiance bound using explicit differentiation is extremely loose for cases involving nearly total internal reflection, even using 10^4 pieces. Fortunately, the implicit differentiation succeeds. We show $\log_{10}(E_k)$. Note that the axes are barycentric coordinates u_k and v_k on the **receiver**.

Explicit differentiation. We initiate by computing the reciprocal using the form invariance of first-order differentials:

$$\left|\frac{\partial \boldsymbol{u}_1}{\partial \boldsymbol{u}_k}\right| = \frac{1}{\left|\frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{u}_1}\right|} = \frac{1}{\begin{vmatrix} \frac{\partial u_k}{\partial u_1} & \frac{\partial u_k}{\partial v_1} \\ \frac{\partial v_k}{\partial u_1} & \frac{\partial v_k}{\partial v_1} \end{vmatrix}}. \tag{16}$$

This only requires the forward derivatives of the position expressions derived in Section 4.3. After all, the coefficients of a polynomial's partial derivative are trivially a linear combination of the original polynomial's coefficients, which is easy to compute. We acquire the final irradiance bound by substituting Eq. (15) into Eq. (3). Note that the remainder variables in the square root approximation only guarantee the correctness of the position bound, so we use a derivative-aware approximation, which compensates not only the primal values but also the derivatives to β_i; see the supplemental document for details.

Unfortunately, when the refracted angle is near $\pi/2$, β_i tends to zero. Hence, the relative approximation error tends to infinity, so the bound becomes extremely loose (Fig. 7). This motivates us to directly express $\left|\frac{\partial \boldsymbol{u}_1}{\partial \boldsymbol{u}_k}\right|$ using the derivatives of implicit functions.

Implicit differentiation. Implicit differentiation does not require representing $\boldsymbol{u}_k$ as a rational function of $\boldsymbol{u}_1$, avoiding approximations on the refraction angle at a particular vertex $\boldsymbol{x}_k$.

For refraction, we express the specular constraint on $\boldsymbol{x}_{k-1}$ as two polynomial functions F and G in variables $\boldsymbol{u}_1$ and $\boldsymbol{u}_k$. We use multivariate specular polynomials[4] as F and G. By solving

$$\begin{cases} F_{u_k} + F_{u_1} \dfrac{\partial u_1}{\partial u_k} + F_{v_1} \dfrac{\partial v_1}{\partial u_k} = 0, \\ G_{u_k} + G_{u_1} \dfrac{\partial u_1}{\partial u_k} + G_{v_1} \dfrac{\partial v_1}{\partial u_k} = 0, \end{cases} \tag{18}$$

we obtain

$$\frac{\partial u_1}{\partial u_k} = -\frac{\frac{\partial F}{\partial u_k}\frac{\partial G}{\partial v_1} - \frac{\partial F}{\partial v_1}\frac{\partial G}{\partial u_k}}{\frac{\partial F}{\partial u_1}\frac{\partial G}{\partial v_1} - \frac{\partial F}{\partial v_1}\frac{\partial G}{\partial u_1}}, \quad \frac{\partial v_1}{\partial u_k} = \frac{\frac{\partial F}{\partial u_k}\frac{\partial G}{\partial u_1} - \frac{\partial F}{\partial u_1}\frac{\partial G}{\partial u_k}}{\frac{\partial F}{\partial u_1}\frac{\partial G}{\partial v_1} - \frac{\partial F}{\partial v_1}\frac{\partial G}{\partial u_1}}. \tag{19}$$

[4]The constraints F and G are polynomials in both $\boldsymbol{u}_1$ and $\boldsymbol{u}_k$ [Fan et al. 2024]:

$$\begin{cases} F = \boldsymbol{d}_{k-1}^2 \left((\boldsymbol{d}_{k-2} \times \boldsymbol{n}_{k-1}) \cdot \boldsymbol{b}\right)^2 - \eta^2 \boldsymbol{d}_{k-2}^2 \left((\boldsymbol{d}_{k-1} \times \boldsymbol{n}_{k-1}) \cdot \boldsymbol{b}\right)^2, \\ G = (\boldsymbol{d}_{k-2} \times \boldsymbol{n}_{k-1}) \cdot \boldsymbol{d}_{k-1}. \end{cases} \tag{17}$$

Here, $\boldsymbol{b}$ represents an arbitrary vector with a non-zero norm, and η is the ratio of the index of refraction. To avoid degenerated cases, we use the intersection of bounds computed using two different $\boldsymbol{b}$ vectors $\boldsymbol{b}_1 = (1,0,0)^\top$ and $\boldsymbol{b}_2 = (0,1,0)^\top$, respectively.

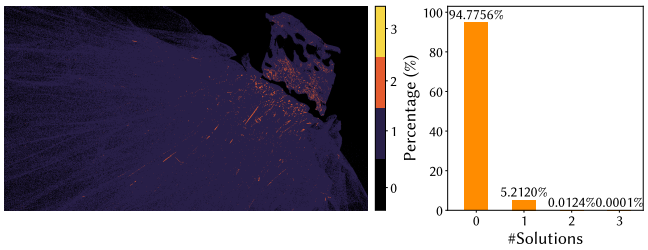

Fig. 8. **The number of solutions per tuple.** In the Plane scene, nearly all tuples have no more than 1 solutions, so we set $m \leq 1$ for rendering experiments. **Left**: the *maximal* number of solutions per tuple for each shading point. **Right**: distributions of the number of solutions per tuple.

The same goes for v_k. The resulting Jacobian $\left|\frac{\partial \boldsymbol{u}_1}{\partial \boldsymbol{u}_k}\right| = \begin{vmatrix} \frac{\partial u_1}{\partial u_k} & \frac{\partial u_1}{\partial v_k} \\ \frac{\partial v_1}{\partial u_k} & \frac{\partial v_1}{\partial v_k} \end{vmatrix}$ includes both $\boldsymbol{u}_1$ and $\boldsymbol{u}_k$. Directly substituting $\boldsymbol{u}_k$ using an expression of $\boldsymbol{u}_1$ results in extremely high degrees. We instead treat $\boldsymbol{u}_1$ and $\boldsymbol{u}_k$ as independent variables within the position bound, but we still substitute $\boldsymbol{u}_2, ..., \boldsymbol{u}_{k-1}$ with rational functions in $\boldsymbol{u}_1$. This step extends a 2D manifold to its 4D superset, so the bound may become looser but still valid. Finally, we reach the irradiance bound by substituting Eq. (15) into Eq. (3). We present the degree of final expressions and complexity analysis in the supplemental document.

4.5 Total irradiance contribution of a triangle tuple

The above-discussed irradiance $E_k(\boldsymbol{u}_1)$ pertains to an admissible specular path. Since we aim at sampling triangle tuples, we need to know the total irradiance contribution $E(\mathcal{T}, \boldsymbol{u}_k, \mathcal{T}_k)$ of each tuple, a summation over all admissible paths' contribution within $\mathcal{T}$:

$$E(\mathcal{T}, \boldsymbol{u}_k, \mathcal{T}_k) = \sum_{\boldsymbol{u}_k = \boldsymbol{u}_k(\boldsymbol{u}_1)} E_k(\boldsymbol{u}_1). \tag{20}$$

$E(\mathcal{T}, \boldsymbol{u}_k, \mathcal{T}_k)$ is no greater than m times the path's irradiance bound. Here, m denotes the number of solutions within $\mathcal{T}$. For $\boldsymbol{u}_k$ not covered by the position bound, m must be zero. For $\boldsymbol{u}_k$ covered by the position bound, following the common assumption that at most one solution exists when triangles are small, we set $m = 1$ in our experiments[5]. We show an example in Fig. 8.

When considering subdivisions on the domain of $\boldsymbol{u}_1$, we should use the maximum irradiance bound among the pieces whose position bounds cover the shading point instead. Suppose the union of disjoint rectangles $\boldsymbol{U}_1^1, \ldots, \boldsymbol{U}_1^n$ covers $\mathcal{U}^2$. Each $\boldsymbol{U}_1^i$ corresponds to a position bound $\boldsymbol{U}_k^i$ and an irradiance bound $\boldsymbol{E}_k^i$. We can give an upper bound of the total irradiance contribution:

$$E(\mathcal{T}, \boldsymbol{u}_k, \mathcal{T}_k) \leq \tilde{E}(\mathcal{T}, \boldsymbol{u}_k, \mathcal{T}_k) = m \max_{\boldsymbol{u}_k \in \boldsymbol{U}_k^i} \overline{E}_k^i. \tag{21}$$

Note that the symbol $\overline{E}$ refers to per-path irradiance bound, while $\tilde{E}$ is per-tuple. See proofs and detailed discussions in the supplemental document. Lastly, we showcase two 2D examples to demonstrate the overall process of the precomputation pass in Fig. 9.

[5]Even if m exceeds our assumption, it will not introduce bias. However, the variance may increase beyond expected levels. m is provably finite for triangle meshes [Wang et al. 2020]. A theoretically strict upper bound of m can be derived from the degree of specular polynomials, but is too loose in practice.

5 BOUND-DRIVEN SAMPLING OF TRIANGLE TUPLES

The previously computed bound provides substantial information about caustics, which we now leverage to sample triangle tuples effectively. Our discussion in this section is under a specific shading point $\boldsymbol{x}_k$, which determines $\mathcal{T}_k$ and $\boldsymbol{u}_k$, so we omit these symbols for simplicity. For instance, $E(\mathcal{T})$ is the shorthand of $E(\mathcal{T}, \boldsymbol{u}_k, \mathcal{T}_k)$, which represents the irradiance contributed by $\mathcal{T}$ at $\boldsymbol{x}_k$.

5.1 Problem definition

There is a finite set U of different triangle tuples $\mathcal{T}$ whose position bounds cover the given shading point $\boldsymbol{x}_k$. Our goal is to estimate the sum of irradiance $E(\mathcal{T})$ over all triangle tuples $\mathcal{T} \in \mathrm{U}$, i.e.,

$$E = \sum_{\mathcal{T} \in \mathrm{U}} E(\mathcal{T}). \tag{22}$$

For simplicity, we discuss the diffuse receiver, where the sum E decides outgoing radiance $L = \rho E$, with ρ being the albedo. For glossy shading points, a product with the BSDF is required.

We aim to sample a **candidate triangle tuple set** $\mathrm{S} \subseteq \mathrm{U}$. After that, we solve for admissible paths within each $\mathcal{T} \in \mathrm{S}$ and sum their contributions (dividing the corresponding sampling probability). Here, the key design choice is determining the number of tuples to sample and the probability with which each tuple is chosen.

5.2 Sampling a single triangle tuple

A straightforward way is to importance sample [Veach 1998] triangle tuples proportional to their irradiance contributions. Theoretically, this is the optimal probability for a one-sample estimator. However, the true irradiance of each triangle tuple is intractable; we can only use the bound instead:

$$P_1(\mathcal{T}) = \frac{\tilde{E}(\mathcal{T})}{\sum_{\mathcal{T}' \in \mathrm{U}} \tilde{E}(\mathcal{T}')}. \tag{23}$$

Nevertheless, $\tilde{E}(\mathcal{T})$ is sometimes substantially higher than $E(\mathcal{T})$ due to a loose bound (either position or irradiance). This could lead to an unmanageable increase in variance.

Practically, there is an intrinsic trade-off among validity, tightness, and computational cost of bounds. This motivates us to resort to a new category of estimators that possibly sample multiple tuples to guarantee the variance is controllable.

5.3 Sampling multiple triangle tuples

Instead of considering which tuple is sampled, we allow multiple tuples to be selected and evaluate the decision to choose each tuple. This effectively unifies importance sampling and selective activation (i.e., deciding where to enable specular path sampling) [Fan et al. 2023; Loubet et al. 2020] in the same framework.

Formally, we introduce $P_{\mathcal{T}}$, which represents the discrete probability that $\mathcal{T}$ is chosen, i.e., $P_{\mathcal{T}} = \mathbb{P}[\mathcal{T} \in \mathrm{S}]$. Note that $P_{\mathcal{T}}$ is not necessarily normalized, i.e., $\sum_{\mathcal{T} \in \mathrm{U}} P_{\mathcal{T}} \neq 1$. The estimator

$$\langle E \rangle = \sum_{\mathcal{T} \in \mathrm{S}} \frac{E(\mathcal{T})}{P_{\mathcal{T}}} \tag{24}$$

is unbiased as long as $P_{\mathcal{T}} > 0$ for all $\mathcal{T}$ that satisfies $E(\mathcal{T}) > 0$. We summarize the sampling process in Algorithm 1.

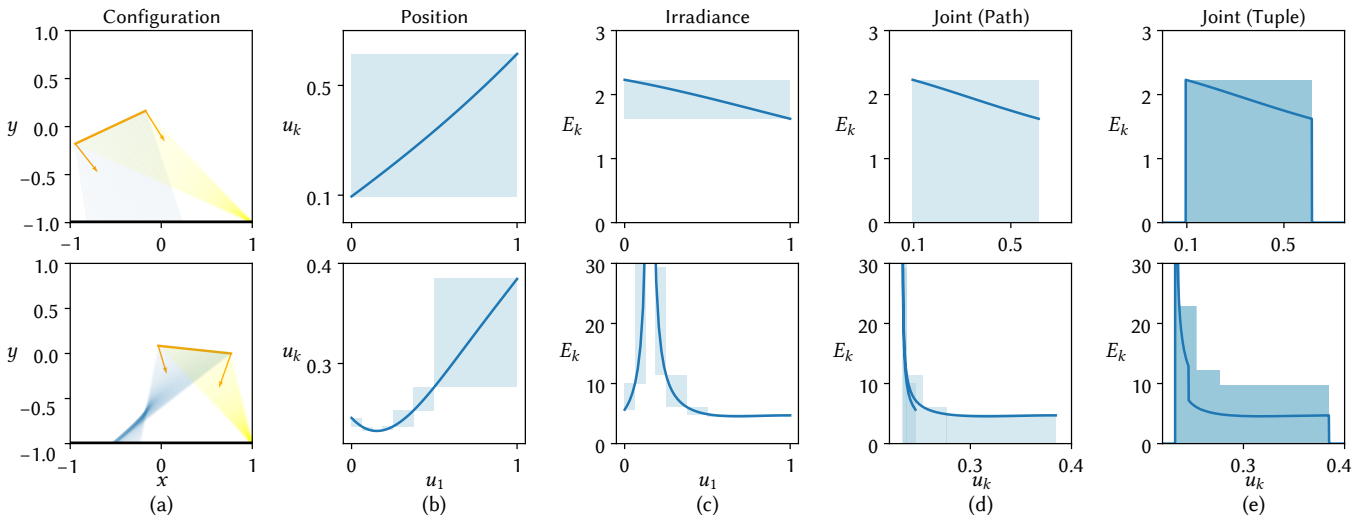

Fig. 9. **2D examples. (a)** The incident light (yellow) hits a specular triangle (orange, with interpolated normals). Reflected rays (blue) hit the receiver (black, bottom). **(b)** We show reference as solid curves and our bound as a shaded region. We perform subdivision on the domain of u_1 to acquire a piecewise constant bound. **(c)** At the singular point where $\frac{\mathrm{d}u_k}{\mathrm{d}u_1} = 0$, the irradiance tends to infinity, so we cannot obtain a finite bound. **(d)** A joint use of position and irradiance bound for bounding the **per-path** irradiance contribution at any position. Note the positional overlap and the irradiance singularity in the bottom example. **(e)** Eq. (20) and Eq. (21) provide the reference and bound of irradiance **per tuple**, respectively. Here, we set $m = 1$ (top) and $m = 2$ (bottom). Note that u_1 is hidden by the transform from (b) and (c) to (d), thus it is **never** stored. The final storage for each $\mathcal{T}$ only involves u_k and E_k.

Algorithm 1 Estimator that samples multiple triangle tuples

Input: Probabilities $P_{\mathcal{T}}$ for each triangle tuple $\mathcal{T} \in \mathrm{U}$
Output: Selected tuple set S, estimated irradiance $\langle E \rangle$

$\mathrm{S} \leftarrow \emptyset$, $\langle E \rangle \leftarrow 0$
for $\mathcal{T} \in \mathrm{U}$ **do**
  $r \leftarrow \mathtt{rng}()$ ▷ Generate a random number in $[0, 1]$
  if $r < P_{\mathcal{T}}$ **then**
    $\mathrm{S} \leftarrow \mathrm{S} \cup \{\mathcal{T}\}$ ▷ Select a triangle tuple $\mathcal{T}$
    $\langle E \rangle \leftarrow \langle E \rangle + \frac{E(\mathcal{T})}{P_{\mathcal{T}}}$ ▷ Add the contribution of all solutions in $\mathcal{T}$
  end if
end for
return S, $\langle E \rangle$

The time complexity of Algorithm 1 is $O(|\mathrm{U}|)$. Such a brute-force implementation already works since the root-finding process is complex. Yet, when U is substantially larger than S, we can reduce the complexity to $O(|\mathrm{S}| \log(|\mathrm{U}|))$. We pack tuples into groups (denoted as B) whose sum of probabilities is no greater than one as a classical bin-packing problem. In each group, we simply importance sample one tuple. We use the first-fit greedy algorithm with $O(|\mathrm{U}|)$ time preprocessing and a guaranteed approximation ratio of 2.

As an important property, variance measures the quality of sampling. A significant benefit of our multi-sample estimator is that we can represent the upper bound of the (population) variance σ^2 (a.k.a. $\mathbb{V}[\langle E \rangle]$) only in terms of $\tilde{E}(\mathcal{T})$, $P_{\mathcal{T}}$, and a constant μ:

$$\sigma^2 = \mu_2 - \mu^2 \le \mu_2, \quad \mu_2 \le \overline{\mu_2} = \sum_{\mathcal{T} \in \mathrm{U}} \frac{\tilde{E}^2(\mathcal{T})}{P_{\mathcal{T}}}. \tag{25}$$

Here, μ denotes the mean value, and μ_2 is the second-order moment.

The irradiance bound $\tilde{E}(\mathcal{T})$ is precomputed and fixed now, so $\overline{\mu_2}$ is controlled only by $P_{\mathcal{T}}$. A good design of $P_{\mathcal{T}}$ should have a relatively low variance. Thus, we determine $P_{\mathcal{T}}$ by optimizing $\overline{\mu_2}$.

5.4 Optimized sampling probabilities

We aim to minimize $\overline{\mu_2}$ given the expected number of candidates $W = \mathbb{E}[|\mathrm{S}|]$. Note that we first treat W as a given parameter. Since $\mathrm{S} \subseteq \mathrm{U}$, we assume $W \le |\mathrm{U}|$. This is a nonlinear optimization with linear equality and inequality constraints:

$$\begin{aligned} \min_{P} \quad & \sum_{\mathcal{T} \in \mathrm{U}} \frac{\tilde{E}^2(\mathcal{T})}{P_{\mathcal{T}}}, \\ \text{s.t.} \quad & \sum_{\mathcal{T} \in \mathrm{U}} P_{\mathcal{T}} = W, \\ & 0 \le P_{\mathcal{T}} \le 1, \quad \forall \mathcal{T} \in \mathrm{U}. \end{aligned} \tag{26}$$

According to the Karush-Kuhn-Tucker (KKT) conditions, we solve the above optimization using the Lagrangian with Lagrange multipliers λ and $\lambda_{\mathcal{T}}$:

$$\mathcal{L} = \sum_{\mathcal{T} \in \mathrm{U}} \frac{\tilde{E}^2(\mathcal{T})}{P_{\mathcal{T}}} - \lambda \left(\sum_{\mathcal{T} \in \mathrm{U}} P_{\mathcal{T}} - W \right) - \sum_{\mathcal{T} \in \mathrm{U}} \lambda_{\mathcal{T}} (P_{\mathcal{T}} - 1) \tag{27}$$

and obtain that for each $\mathcal{T}$, either $P_{\mathcal{T}} = 1$ or $P_{\mathcal{T}} = \gamma \tilde{E}(\mathcal{T})$ is satisfied. Therefore, we reach the final probability

$$P_{\mathcal{T}} = \min\left(\gamma \tilde{E}(\mathcal{T}), 1\right) \tag{28}$$

with γ being a constant parameter. Note that the above equation naturally satisfies $P_{\mathcal{T}} \ge 0$, so we can ignore this constraint. In contrast, the condition $P_{\mathcal{T}} \le 1$ must be considered explicitly as a key

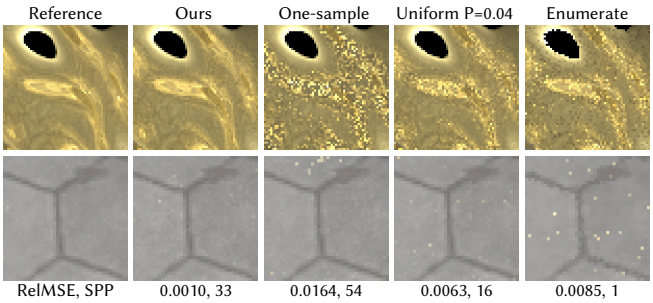

Fig. 10. **Validations on different sampling schemes.** While it is possible to use the position bound only to render a pixel-perfect image (Enumerate), it increases the runtime cost for each spp, leading to high variance in equal time comparison and obvious aliasing artifacts. Uniformly assigning P introduces visible noise. Conventional importance sampling (One-sample) easily causes extremely high variance when the bound is loose. Our multi-sample estimator leveraging the irradiance bound performs the best.

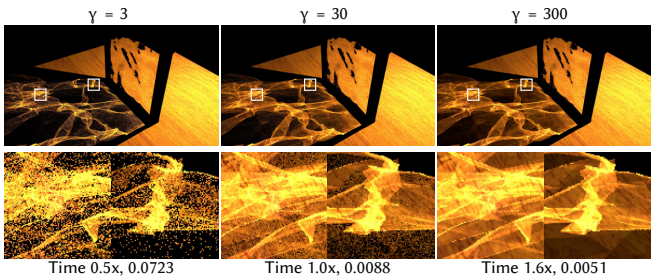

Fig. 11. **The impact of the sampling parameter γ.** All methods use 4 spp for path tracing. Higher gamma reduces variance by increasing sampling probability. Notably, the probability easily reaches 1 (thus no variance) for high-energy tuples, while noise still persists for low-energy ones.

difference from continuous cases [Rath et al. 2020]. As a summary, we compare different estimators we discussed in Fig. 10.

Additionally, the impact of the parameter γ is intuitive, as shown in Fig. 11. While it is possible to let users specify γ, obtaining it automatically from the expected number of samples W or variance is also feasible since there is a one-to-one mapping between these parameters. See the supplemental document for details.

6 EVALUATION

We implement our method in Mitsuba 0.6 [Jakob 2010]. The precomputation partially utilizes the Numba JIT compiler. All timings are conducted on an Intel Core i9-13900KF processor.

6.1 Implementation

The supplemental document provides detailed algorithms, complexity analysis, and pseudo-code snippets. We briefly outline some important design choices here.

Tuple construction. In precomputation, we construct triangle tuples by extending triangles at the end of a given prefix. For $\mathcal{T}_1, ..., \mathcal{T}_i$, we compute the bound of $\boldsymbol{x}_i$ and $\boldsymbol{d}_i$. Then, we traverse the bounding volume hierarchy (BVH) to find all possible $\mathcal{T}_{i+1}$ according to whether the bound of $(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) \times \boldsymbol{d}_i$ covers zero.

Bound storage. We rasterize bounds into a grid parameterized by the texture coordinates. Each cell stores a list containing several pairs of irradiance bound $\tilde{E}(\mathcal{T}, \boldsymbol{u}_k, \mathcal{T}_k)$ and triangle indices of $\mathcal{T}$. We splat the irradiance bound to each cell intersecting with the position bound. We choose uniform 512×512 grids for simplicity. During rendering, for each shading point, we use its coordinate as $\boldsymbol{u}_k$ to query the grid on texture space, returning a list of tuples.

Domain subdivision. Our bound is valid but loose when the triangles are large, leading to slow rendering. To compute the piecewise constant bound, we recursively subdivide the box domain of $\boldsymbol{u}_1$ into four boxes at the midpoint. The subdivision stops when

- the area of $\boldsymbol{U}_k$ is less than $\sigma = 10^{-4}$,
- the ratio $\overline{E_k}/\underline{E_k}$ is smaller than a threshold[6] α, or
- the subdivision depth reaches a limit that varies among scenes.

Note that we do not subdivide $\mathcal{T}_2, \ldots \mathcal{T}_k$, and the domain subdivision process is completely optional. Again, for different pieces after subdivisions, their domains (i.e., the range of $\boldsymbol{u}_1$) are completely disjoint, but position bounds on $\boldsymbol{u}_k$ may overlap with each other.

Domain initialization. For each triangle tuple, the effective domain of $\boldsymbol{u}_1$ is usually smaller than $\mathcal{U}^2$. Therefore, for each $\mathcal{T}$, we first compute the bound of $\boldsymbol{u}_1$ where the corresponding $\boldsymbol{u}_k \in \mathcal{U}^2$, implemented using a recursive subdivision of at most 100 pieces.

Degree reduction. We found the numerical stability and complexity become infeasible when the degree exceeds a certain threshold, e.g., 40. Thus, we convert these high-degree polynomials into low-degree ones and add a new remainder variable to maintain bounding validity. Specifically, we fit low-degree approximants using linear regression based on singular-value decomposition. Each reduction also eliminates all existing remainder variables.

Area light sources. Our method easily generalizes to area light sources by incorporating two additional variables in the expression for $\boldsymbol{x}_0$. In Fig. 12, we present a 2D example for illustration.

[6] We set the approximation ratio α to 2 for single scattering and 10 for multiple ones.

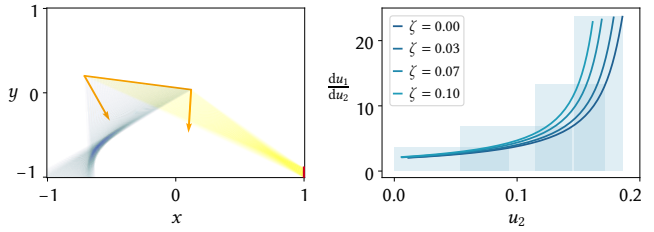

Fig. 12. **Handling area light sources.** We introduce an extra variable ζ to represent the position along the line light source. The rational functions have two variables u_1 and ζ. We succeeded in bounding the irradiance distribution of all points on the area light source. The curves correspond to different positions (denoted as ζ) on the light source.

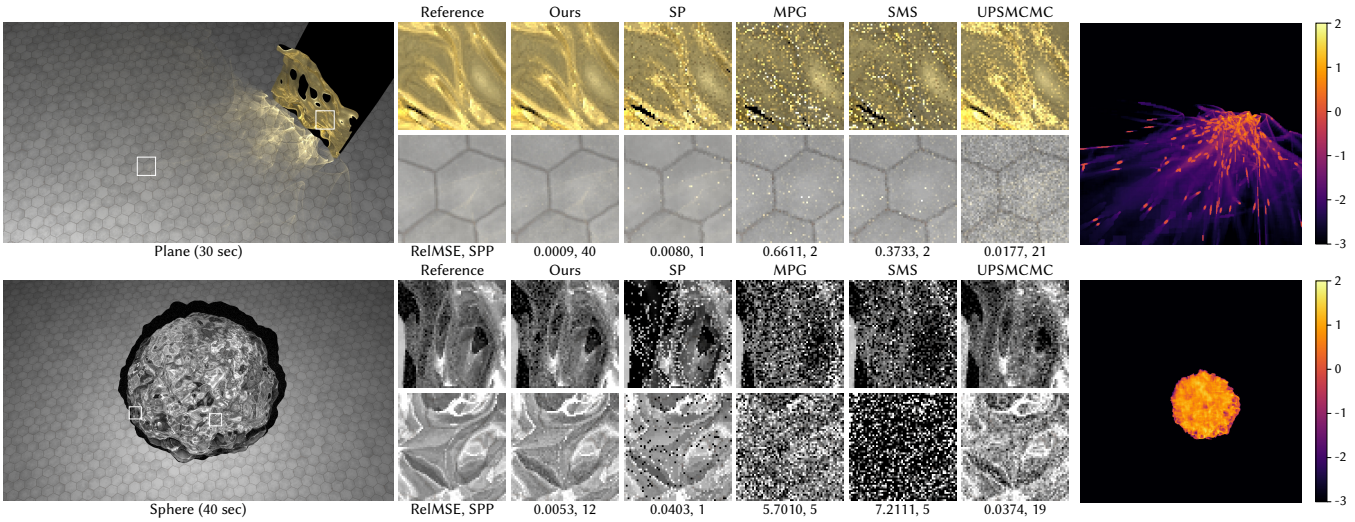

Fig. 13. **Equal-time comparisons on single scattering.** Precomputation takes 2.1 sec and 2.4 sec, respectively. We visualize irradiance bounds (in the base 10 logarithmic space) summed over tuples. We compare with Specular Polynomials (SP) [Fan et al. 2024], Manifold Path Guiding (MPG) [Fan et al. 2023], Specular Manifold Sampling (SMS) [Zeltner et al. 2020], and Metropolised Bidirectional Estimators (UPSMCMC) [Šik et al. 2016].

Root-finding. Rendering caustics requires finding admissible specular paths within the sampled tuple. We employ specular polynomials [Fan et al. 2024] for single scattering. However, existing deterministic methods fail to find all solutions in multiple scattering while maintaining a low computational cost. Therefore, we choose Newton's method, assessing two different schemes of initialization and weighting to evaluate various aspects:

- **Deterministic** (Det) initialization [Wang et al. 2020] may leak solutions but does not introduce additional variance, which helps evaluate the amount of variance introduced by our proposed triangle sampling.
- **Stochastic** initialization with an unbiased[7] weighting (Stoc) [Zeltner et al. 2020] helps validate the overall unbiasedness. However, it could introduce outliers.

6.2 Equal-time comparisons

In Figs. 13 and 14, we compare our method to several approaches:

- Deterministic search. We compare with specular polynomials [Fan et al. 2024] for one bounce and Path Cuts [Wang et al. 2020] for multiple bounces. Note that specular polynomial also uses Path Cut's interval tests to select triangles. For multiple scattering, we also note that the interval tests are extremely slow. To ensure a fair comparison with roughly equal time, we develop a variant (marked with an asterisk, e.g., Path Cuts*) that uniformly samples 1% nodes.
- Manifold sampling methods, including the unbiased variant of Specular Manifold Sampling (SMS) [Zeltner et al. 2020] and Manifold Path Guiding (MPG) [Fan et al. 2023].
- Photon-based (biased) methods, including stochastic progressive photon mapping (SPPM) [Hachisuka and Jensen 2009] and metropolised bidirectional estimator (UPSMCMC) [Šik et al. 2016]. We tune the initial photon lookup radius to lower the bias and only compare noise.
- Regular MC methods, including (bidirectional) path tracing [Kajiya 1986; Veach and Guibas 1995], path guiding [Müller et al. 2017], and Metropolis light transport [Jakob and Marschner 2012; Veach and Guibas 1997].

We also evaluate the temporal stability in equal-time and equal-sample settings in the supplemental video.

Comparisons with deterministic search. The main drawback of deterministic search is its high computational cost, which leads to extremely low sample rates. Consequently, although it produces zero-variance estimations of incident radiance, the rendering result still suffers from aliases and noise. This issue is particularly pronounced for specular-diffuse-specular (SDS) effects, which require path tracing to sample diffuse shading points (e.g., in Fig. 13, the caustics viewed through the reflection of the gold plane). Besides, there are many fireflies on the floor, which come from the sampling of non-specular paths (with potential connections to the deterministic specular ones). Fortunately, our method significantly speeds up rendering by stochastically reducing the search domain using bounding information. As a result, we can utilize substantially higher sample rates within equal time, effectively decreasing the overall noise and aliases present in the final image.

For multiple scattering, deterministic search becomes too slow due to combinatorial explosions. As a result, we can only uniformly sample a portion (Path Cuts*) for roughly equal-time comparisons. However, uniform sampling does not consider energy, thus introducing significant noise, as shown in Fig. 14. Our bound-driven

[7]We briefly note that it is possible to combine a **stochastic** initialization with a **biased** weighting scheme [Zeltner et al. 2020], which also suffers from energy loss but in a smoother pattern. These biased estimations remain below or equal to the ground truth, ensuring the overall second moment is still controllable.

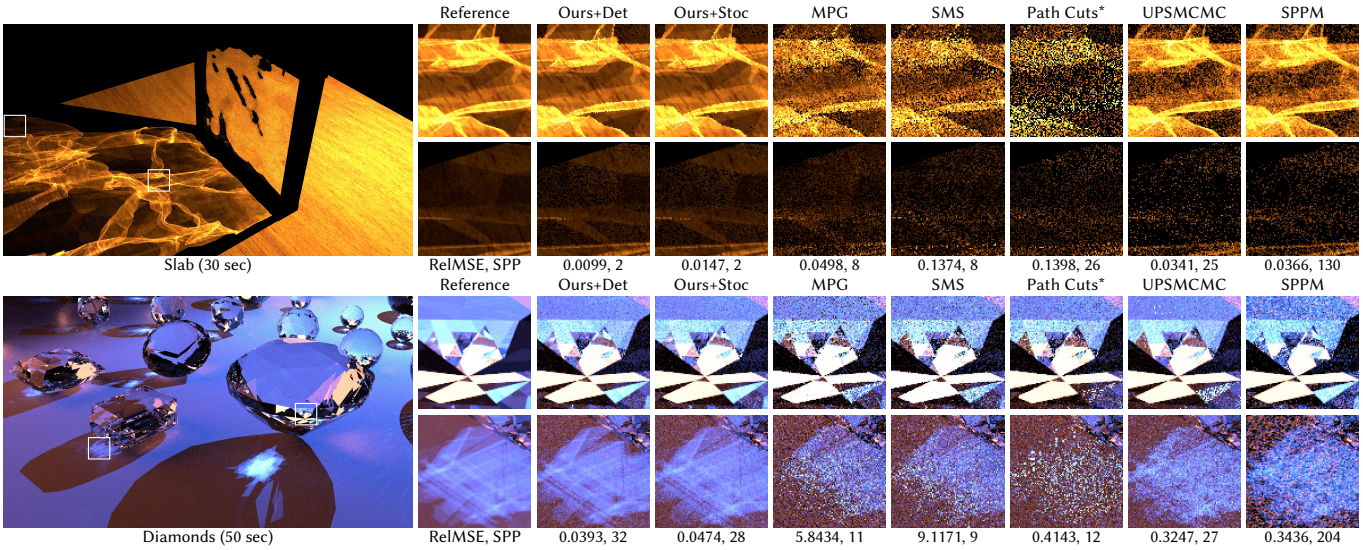

Fig. 14. **Equal-time comparisons on double scattering.** Precomputation time is included, which takes 23 sec and 25 sec, respectively. We combine our triangle sampling with deterministic (Det, biased) and stochastic (Stoc, unbiased) initialization for Newton's iteration-based root-finding. Since the original Path Cuts is extremely slow, we use a modified version (Path Cuts*) that samples 1% paths, which already takes 60× more time than the other methods.

sampling reduces the number of tuples that need to perform root-finding. Consequently, rendering with one sample per pixel takes just one to several seconds while maintaining a low variance.

Comparisons with manifold sampling. As state-of-the-art methods for sampling specular paths, both SMS and MPG rely on point sampling to search for admissible chains. In particular, SMS tends to exhibit noticeable noise, largely due to its uniform sampling of seed chains. While MPG mitigates this issue through importance sampling, it requires a fairly long time to learn accurate distributions. With a limited budget, MPG still produces noisy outputs. Moreover, these methods possess an unbounded probability of finding a solution, which, depending on the initialization of distributions, could be quite small and result in extremely high variance.

Although we also introduce stochastic sampling, we can keep the variance controllable thanks to the bound of caustics. Consequently, we guarantee that important solutions can be immediately found with a large enough probability, thus achieving low variance.

Note the speed difference between our triangle-based approach and manifold sampling when handling complex geometry. Specifically, even if we sample multiple solutions per shading point while manifold sampling generates at most one, our sample rates are usually higher. The reasons for this phenomenon are two-fold:

- For manifold sampling methods, the use of **reciprocal probability estimation** contributes a significant amount of variance and overhead, resulting in fireflies. In contrast, our triangle sampling does not require probability estimations.
- Manifold walks require tracing a full specular chain in each iteration of Newton's solver, which includes several **intersection tests** (i.e., querying the BVH). Since our triangle sampling has bounded the domain to a local region, our solver does not require ray tracing except for the visibility check after a solution is found.

Comparison with photon-based methods. Photons are often distributed non-uniformly across the receiver. Low-energy regions, such as the bottom crop of the Slab scene, may receive an insufficient number of photons. However, accurate density estimations require sufficient photon samples to reconstruct the true distribution reliably. As a consequence, rendering results may exhibit either noise or blurriness, depending on the choice of the kernel radius. Our approach operates on functions within finite regions and directly solves for admissible paths, avoiding the issues related to point sampling and density estimations. As a result, we achieve low-variance rendering that preserves the sharp details of caustics.

Comparison with regular MC methods. Traditional MC methods face significant challenges when dealing with SDS paths (Fig. 15) because of the high-frequency radiance distribution. Even with effective guiding or Metropolis sampling, they still rely on the base sampler to find initial paths for subsequent learning and mutations. As an intrinsic limitation, these issues also persist with more advanced guiding and Metropolis sampling.

6.3 Validations

In addition to the above rendering results, we provide a direct visualization of the correctness and tightness of our bound.

Bounding correctness and tightness. In Fig. 17, we present the ratio $\tilde{E}/E$ between the irradiance bound and the true value, shown in a logarithmic scale with base 10. This ratio is expected to be no smaller than 1 as long as the bound is valid, with smaller values representing a tighter bound. The absence of red regions in the image indicates that our bound is consistently valid. The predominance

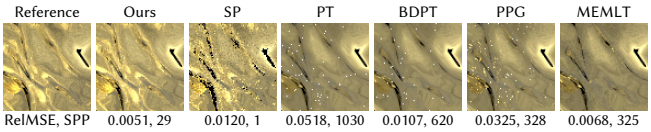

Fig. 15. **Equal-time (30 sec) comparisons on handling area light sources**. We compare our method with path tracing (PT), bidirectional path tracing (BDPT), Practical Path Guiding (PPG), and Manifold Exploration Metropolis light transport (MEMLT) in a Plane scene lit by an area light. Precomputation takes 11 sec.

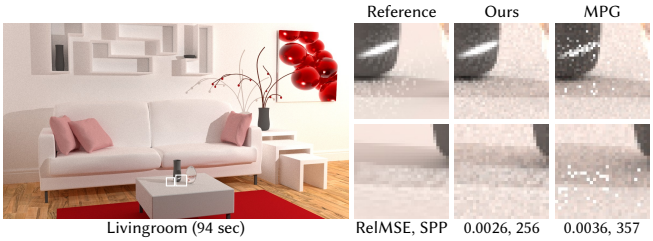

Fig. 16. Rendering scenes with non-planar caustics receivers. Our method is still accurate, though relatively slow. Precomputation takes 20 sec.

of light blue areas suggests that the bound is generally tight. Note that the bound may become loose for various reasons, including a loose position/irradiance bound and insufficient resolution.

The number of solutions. We further validate the number of solutions for each triangle tuple, as illustrated in Fig. 8. Across our tested scenes, nearly all triangle tuples exhibit at most one solution. This indicates that our assumption of $m = 1$ is reasonable.

6.4 Ablation studies

We study the impact of some important components in our pipeline.

With vs. without remainder variables. In Fig. 18, we examine the impact of the remaining variables. The absence of these variables significantly accelerates precomputation; however, this comes at the cost of bounding validity, leading to (red) regions where the irradiance exceeds bounds. By incorporating the remaining variables, we ensure validity, albeit with a slower precomputation.

For our experiments, we enable the remaining variables unless otherwise noted for the sake of strict correctness. Nonetheless, we acknowledge that in certain scenarios, maintaining bounding validity may not be critical. In such cases where slight leaking or increased variance is acceptable, one might consider omitting the remaining variables to enhance performance.

Position-only vs. our complete method. It is possible to compute only the position bound and then either enumerate or uniformly sample all tuples that cover the shading point. We evaluate these variants in Fig. 10. As seen, enumeration yields a significantly smaller number of samples, leading to higher overall noise. Uniform sampling, which does not account for energy, introduces visible variance. Our method, by sampling a small subset based on irradiance bound, accelerates rendering while maintaining low variance, ultimately resulting in high-quality rendering within equal time.

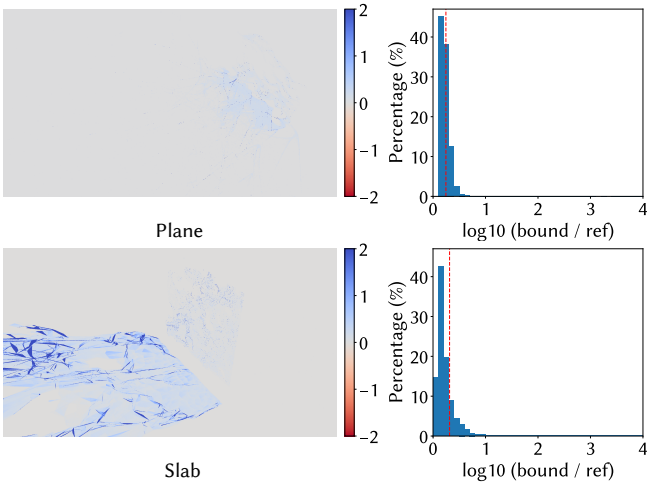

Fig. 17. **Visualization of the ratio between the bound and the true irradiance** for each solution in the Plane and Slab scenes. The image illustrates the overall situation, where the ratio is averaged per pixel. The accompanying histogram represents the ratios for each solution. These ratios are displayed on a logarithmic scale with base 10. All ratios are greater than zero, and the red dashed lines indicate the average.

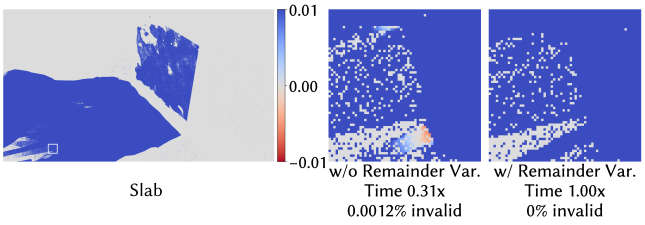

Fig. 18. **Without remainder variables**, there exists some solutions whose bound $\tilde{E}$ is slightly lower than the true irradiance E (red). By using remainder variables, all solutions are properly bounded (blue). The ratio $\tilde{E}/E$ is displayed on a logarithmic scale with base 10.

However, it is important to note that the effectiveness of irradiance-based sampling is scene-dependent. In scenarios where the number of tuples that cover a grid cell is low (e.g., the Slab scene in Fig. 14 and the Pool scene in Fig. 20), tuple sampling may contribute minimally. In such cases, bypassing the irradiance computation can shorten the precomputation time. Just allocating these budgets to the rendering pass would result in higher overall quality.

Interval arithmetic vs. Bernstein polynomials. Our pipeline can support various methods for computing the bounds of rational functions. In Fig. 4, we compare our approach with interval arithmetic, which has been widely adopted in previous works [Walter et al. 2009; Wang et al. 2020]. Despite its general applicability, we observed that interval arithmetic exhibits slow convergence and often generates excessively loose bounds, particularly for irradiance. In contrast, our use of Bernstein polynomials takes advantage of the properties of rational functions, yielding tighter bounds in both scenarios of equal piece count and equal time allocation.

Table 2. **Rendering statistics**. We show the percentage of precomputation time used for position bound (including tuple constructions), irradiance bound, and recording bound into grid cells. For rendering, we show the percentage of time used by sampling S from U and the average size of S, B, and U, respectively. We only render the specified chain type for fair comparisons. Additionally, we report the number of triangle tuples, the number of pieces (that require computing position bounds and irradiance bounds, respectively) averaged per tuple, and the size of bound storage.

Figure	Type		Scene	#Tri.	Subdiv.	Precomputation Time			#Tuples	Avg. #Pieces		Mem.	Render Time	Size of sets		
	Chain	Normal		(K)	Max Level	Pos.	Irr.	Rec.	(K)	Pos.	Irr.	(MB)	Sampling	|S|	|B|	|U|
Fig. 1	R	Interp	Dragon (1)	354	20	23.4%	58.9%	16.5%	354.26	43.44	32.12	1948.7	3.2%	11.8	13.0	1454.4
Fig. 1	R	Interp	Dragon (23)	354	20	21.5%	74.8%	2.6%	354.26	193.19	170.49	1297.4	2.0%	1.3	2.0	1053.7
Fig. 13	R	Interp	Plane	131	12	21.6%	55.2%	21.8%	131.07	10.89	7.75	220.9	3.1%	3.4	4.1	204.7
Fig. 13	T	Interp	Sphere	82	1	4.1%	92.6%	0.7%	81.77	1.76	1.17	37.0	1.4%	1.5	1.7	51.6
Fig. 14	TT	Interp	Slab	10	1	20.4%	62.4%	10.1%	126.87	1.58	0.37	28.6	1.3%	18.7	19.5	20.5
Fig. 14	TT	Flat	Diamonds	10	3	4.9%	76.3%	11.7%	192.57	3.63	0.77	29.8	6.5%	2.5	3.3	15.1
Fig. 16	TT	Flat	Livingroom	3	2	2.2%	91.8%	2.1%	48.79	3.36	0.65	4.6	0.4%	0.7	0.8	0.8
Fig. 20	T	Interp	Pool	20	1	6.5%	84.2%	0.9%	20.00	1.12	1.12	13.0	2.6%	3.3	3.8	7.6

6.5 Performance analysis

In Table 2, we report the statistics of our rendering experiments.

Precomputation time. In the precomputation pass, the majority of time is spent calculating the irradiance bounds due to the high degrees. The recording process also incurs some overhead because we utilize a simple uniform grid, which becomes inefficient when the position bounds cover numerous cells.

Sampling. The size of our sampled set S is often smaller than that of U, which validates the effectiveness of sampling. As we pack tuples into bins whose accumulation of probabilities never exceeds 1, the number of bins |B| is slightly above |S|. Sampling a tuple from a bin only requires a simple bisection, while solving for an admissible path is inherently more time-consuming. As a result, the time added by our sampling process is relatively minimal.

The number of triangle tuples and pieces. For double scattering, outgoing rays from each $\mathcal{T}_1$ intersect with only 10 to 20 different $\mathcal{T}_2$ on average. This indicates that our bounds effectively mitigate the combinatorial explosion associated with triangle-based methods.

The average number of pieces after subdivision remains far below the quartic of the maximum subdivision level, thanks to the stopping criteria for domain subdivisions.

Impact of mesh tessellations. We further investigate the relationship between performance and tessellations in Table 3. Generally, our method effectively keeps rendering time and error at a stable level, albeit with a slight growth as the mesh tessellation increases. Precomputation time and memory usage also grow sublinearly.

We highlight the double refraction case, where a naïve combination of triangle tuples would result in a quadratic increase relative to the number of triangles. Thanks to our bound-driven tuple constructions, the growth in the number of tuples is linear to the number of triangles. Note that the bound-driven tuple constructions also enable handling of non-planar caustics receivers as shown in Fig. 16, with a sublinear (nearly square root) growth of precomputation time with respect to the number of triangles of the receiver. Additionally, the average precomputation time and storage requirements per tuple decrease as the position bounds become smaller and the irradiance variation within each tuple is reduced.

Table 3. **The impact of mesh tessellation (uniform) on single reflection (top) and double refractions (bottom). Top:** rendering variants of the Plane scene. We observe a sublinear growth of precomputation time, rendering time, and memory with respect to the increase in the number of triangles. We use the fixed maximal subdivision level at 10. **Bottom:** We show statistics on rendering the double refraction of a ball (with interpolated normal). Here, level refers to the maximal domain subdivision depth, which we decrease as the mesh gets finely tessellated.

#Triangles/K	Precomputation/s	Rendering/s	Memory/MB	RelMSE
7	0.4	6.0	56.2	0.00327
28	0.4	6.0	89.7	0.00359
114	0.7	6.8	138.5	0.00391
458	1.5	8.0	205.8	0.00417

#Triangles	Level	#Tuples/K	Pre./s	Render./s	Mem./MB	RelMSE
80	4	15.0	4.1	3.2	22.1	0.00007
320	3	52.4	7.1	2.2	23.8	0.00006
1280	2	198.3	17.5	5.1	72.1	0.00006
5120	1	770.6	47.1	7.3	110.4	0.00006
320	0	15.0	8.8	199.9	687.1	0.00002
1280	0	52.4	10.9	91.0	641.1	0.00002
5120	0	198.3	18.2	31.6	400.9	0.00003

In the absence of domain subdivisions (Level = 0), precomputation time, rendering time, and storage costs all increase significantly compared to scenarios that utilize subdivisions. This underscores the necessity of introducing domain subdivisions.

Impact of hyper-parameters. We assess the impact of precomputation parameters in Fig. 19. Generally, these parameters govern the trade-off between precomputation time and rendering time required to achieve a consistent noise level. For instance, a smaller spatial threshold σ, finer subdivision depth, and higher grid resolutions facilitate faster[8] rendering convergence, albeit at the expense of increased precomputation time. For each scene, our selected parameters strike an appropriate balance in this regard. Yet, we leave the automatic selection strategies for future work. Meanwhile, it is important to note that these precomputation parameters have minimal influence on rendering quality, which is predominantly determined by the sampling parameter γ, as illustrated in Fig. 11.

[8]The rendering time has a limit proportional to γ times the sum of true irradiance.

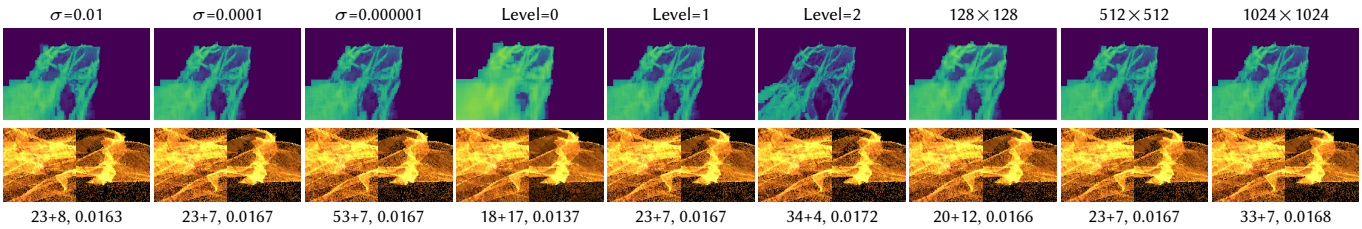

Fig. 19. **The influence of precomputation parameters**, including the spatial threshold σ, the maximal level of subdivisions, and grid resolutions. We visualize the irradiance bound summed over tuples. Precomputation time p, rendering time q, and RelMSE values r are reported in the form $p + q, r$.

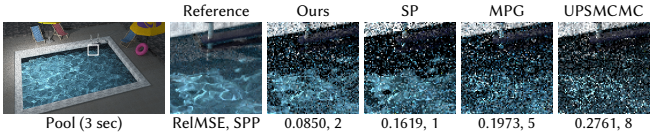

Fig. 20. The advantage of our method is not significant in cases already well handled by deterministic search. We show an example of a pool scene with shallow water, where the number of tuples related to each shading point is small, and irradiance does not have a significant difference. Precomputation time is included, which takes 1 sec.

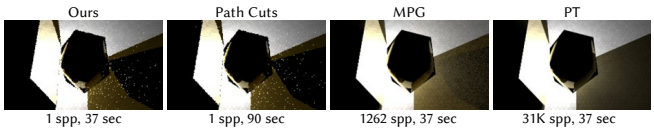

Fig. 21. **Triple reflections** between a metallic plane and a sphere. We precompute only the position bounds, which requires 8 sec. Due to the looseness of our bounds, the resulting speedup is not substantial.

7 LIMITATIONS

Convergence of bounds. Due to the approximations employed and the necessity of rational functions, our current framework does not yet include a theoretical analysis regarding the guaranteed convergence rate and the tightness of the bound, so our precomputation could be costly. Practically, in some cases, such as when triangles intersect with each other, the bound is extremely loose, necessitating substantial subdivisions. Establishing a tight and efficient bound for these challenging scenarios deserves future research.

Generalization with remainder variables. Our method is currently designed for triangle meshes with a computational cost sublinear to the number of triangles. However, theoretically, this approach is not constrained due to the strong capability of remainder variables in expressing uncertainty and approximation errors. Just like how we handle area light sources, future work could explore extensions to near-specular vertices, non-planar triangles, and triangle aggregations, which could no longer depend on the number of triangles in the scene but rather on the actual geometric complexity.

Long chains. Our method is tailored for high-quality rendering of short specular chains (i.e., one or two bounces), where it demonstrates the most substantial improvements over existing techniques. As the length increases, we may encounter additional challenges. Within a triangle tuple, the degree of rational functions becomes higher. Although we can convert them into low-degree ones, it results in looser bounds and an increased computational burden (Fig. 21). More critically, the number of triangle tuples will grow, even with our bound-driven tuple constructions. Therefore, we believe it is impractical to consider all possible triangle tuples during precomputation. Instead, it would be more appropriate to focus on those with high contributions from the beginning.

Simplifying assumptions. Visibility and the Fresnel term are ignored during precomputation. We also assume a single, small emitter and purely specular scattering. Thus, the precomputation time would scale linearly with the number of emitters. Also, we only consider specular chains connecting to light sources. Future work could relax these assumptions to make the method more practical.

Bound representations. We parameterize the positional bound using the texture coordinates of receivers and employ a uniform grid for simplicity. This approach is advantageous for planar receivers; however, it faces performance degradation as the complexity of the receivers increases (see Fig. 16). Future work could store volumetric bounds with vector irradiance [Arvo 1994], thereby eliminating dependence on the receiver configuration. Implementing a spatial hierarchy could further reduce computational and memory costs.

8 CONCLUSION

The challenge of unbounded convergence is crucial to robustly render complex light transport. By bounding both the position and irradiance of caustics, we succeed in controlling the estimator's variance, resulting in efficient and robust rendering. With analytic and functional modeling on both the light transport behaviors and geometric information, we finally reach a bound of caustics using the properties of rational functions. Unlike methods based on point sampling and online learning, our bound is intrinsically reliable and conservative. We finally leverage our bound to achieve a variance reduction of over an order of magnitude in equal time.

We believe our method represents a step forward in controlling the complex behaviors of stochastic sampling, indicating great potential for efficient and reliable rendering. The established bounds may have further applications beyond triangle sampling, such as manifold sampling and general path guiding. Additionally, we hope our method will inspire future research focused on developing enhanced bounds for caustics and beyond.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable suggestions. We also thank Pengpei Hong, Pengcheng Shi, and Huzhiyuan Long for proofreading and valuable discussions.

This work was supported by the National Natural Science Foundation of China (No. 61972194 and No. 62032011) and the Natural Science Foundation of Jiangsu Province (No. BK20211147).

The model in the scene Diamond has been created by the cgtrader user alexmit. The scene Sphere, Slab, Pool, and Plane are modified from the test scenes from Zeltner et al. [2020]. Other scenes are acquired from Bitterli [2016] with modifications.

REFERENCES

James Arvo. 1994. The irradiance Jacobian for partially occluded polyhedral sources. In Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94). Association for Computing Machinery, New York, NY, USA, 343–350.

S. Bernstein. 1912. Démoistration du théoréme de Weierstrass fondée sur le calcul des probabilités. Communications de la Société mathématique de Kharkow. 2-ée série 13, 1 (1912), 1–2.

Benedikt Bitterli. 2016. Rendering resources. https://benedikt-bitterli.me/resources/.

Min Chen and James Arvo. 2000. Theory and application of specular path perturbation. ACM Trans. Graph. 19, 4 (oct 2000), 246–278.

Marc Droske, Johannes Hanika, Jiří Vorba, Andrea Weidlich, and Manuele Sabbadin. 2023. Path tracing in Production: The Path of Water. In ACM SIGGRAPH 2023 Courses (Los Angeles, California) (SIGGRAPH '23). Association for Computing Machinery, New York, NY, USA, Article 12, 66 pages.

Zhimin Fan, Jie Guo, Yiming Wang, Tianyu Xiao, Hao Zhang, Chenxi Zhou, Zhenyu Chen, Pengpei Hong, Yanwen Guo, and Ling-Qi Yan. 2024. Specular Polynomials. ACM Trans. Graph. 43, 4, Article 126 (July 2024), 13 pages.

Zhimin Fan, Pengpei Hong, Jie Guo, Changqing Zou, Yanwen Guo, and Ling-Qi Yan. 2023. Manifold Path Guiding for Importance Sampling Specular Chains. ACM Trans. Graph. 42, 6, Article 257 (Dec 2023), 14 pages.

Rida T. Farouki. 2012. The Bernstein polynomial basis: A centennial retrospective. Comput. Aided Geom. Des. 29 (2012), 379–419.

J. Garloff. 1986. Convergent bounds for the range of multivariate polynomials. In Interval Mathematics 1985, Karl Nickel (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 37–56.

Jürgen Garloff, Antek Schabert, and Andrew P. Smith. 2012. Bounds on the Range of Multivariate Rational Functions. PAMM 12 (2012).

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light Transport Simulation with Vertex Connection and Merging. ACM Transactions on Graphics 31, 6 (Nov. 2012), 1–10.

Toshiya Hachisuka and Henrik Wann Jensen. 2009. Stochastic Progressive Photon Mapping. ACM Trans. Graph. 28, 4, Article 141 (2009), 8 pages.

Johannes Hanika, Marc Droske, and Luca Fascione. 2015. Manifold Next Event Estimation. Computer Graphics Forum 34, 4 (July 2015), 87–97.

Paul S. Heckbert and Pat Hanrahan. 1984. Beam tracing polygonal objects. SIGGRAPH Comput. Graph. 18, 3 (Jan. 1984), 119–127.

Kei Iwasaki, Yoshinori Dobashi, and Tomoyuki Nishita. 2003. A Fast Rendering Method for Refractive and Reflective Caustics Due to Water Surfaces. Computer Graphics Forum 22, 3 (2003), 601–609.

Wenzel Jakob. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

Wenzel Jakob and Steve Marschner. 2012. Manifold Exploration: A Markov Chain Monte Carlo Technique for Rendering Scenes with Difficult Specular Transport. ACM Transactions on Graphics 31, 4 (Aug. 2012), 1–13.

Henrik Wann Jensen. 1995. Importance Driven Path Tracing Using the Photon Map. In Rendering Techniques '95, Patrick M. Hanrahan and Werner Purgathofer (Eds.). Springer Vienna, Vienna, 326–335.

Henrik Wann Jensen and Niels Jørgen Christensen. 1995. Photon maps in bidirectional Monte Carlo ray tracing of complex objects. Comput. Graph. 19 (1995), 215–224.

James T. Kajiya. 1986. The Rendering Equation. Proceedings of the 13th annual conference on Computer graphics and interactive techniques - SIGGRAPH '86 (1986), 143–150.

Anton S. Kaplanyan and Carsten Dachsbacher. 2013. Path Space Regularization for Holistic and Robust Light Transport. Computer Graphics Forum 32, 2pt1 (2013), 63–72.

Anton S. Kaplanyan, Johannes Hanika, and Carsten Dachsbacher. 2014. The Natural-Constraint Representation of the Path Space for Efficient Light Transport Simulation. ACM Trans. Graph. 33, 4, Article 102 (jul 2014), 13 pages.

Guillaume Loubet, Tizian Zeltner, Nicolas Holzschuch, and Wenzel Jakob. 2020. Slope-Space Integrals for Specular next Event Estimation. ACM Trans. Graph. 39, 6, Article 239 (nov 2020), 13 pages.

Don Mitchell and Pat Hanrahan. 1992. Illumination from Curved Reflectors. Proceedings of the 19th annual conference on Computer graphics and interactive techniques - SIGGRAPH '92 (1992), 283–291.

Tomas Möller and Ben Trumbore. 1997. Fast, Minimum Storage Ray-Triangle Intersection. Journal of Graphics Tools 2, 1 (Jan. 1997), 21–28.

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. Computer Graphics Forum 36, 4 (July 2017), 91–100.

Anthony Narkawicz, Jürgen Garloff, Andrew P. Smith, and César A. Muñoz. 2012. Bounding the Range of a Rational Function over a box. Reliable Computing 17 (2012), 34–39.

Alexander Rath, Pascal Grittmann, Sebastian Herholz, Petr Vévoda, Philipp Slusallek, and Jaroslav Křivánek. 2020. Variance-Aware Path Guiding. ACM Transactions on Graphics 39, 4 (Aug. 2020).

Mikio Shinya, T. Takahashi, and Seiichiro Naito. 1987. Principles and applications of pencil tracing. In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87). Association for Computing Machinery, New York, NY, USA, 45–54.

Martin Šik, Hisanari Otsu, Toshiya Hachisuka, and Jaroslav Křivánek. 2016. Robust Light Transport Simulation via Metropolised Bidirectional Estimators. ACM Transactions on Graphics 35, 6 (Nov. 2016), 1–12.

Eric Veach. 1998. Robust Monte Carlo Methods for Light Transport Simulation. Ph. D. Dissertation. Stanford University, Stanford, CA, USA. Advisor(s) Guibas, Leonidas J. AAI9837162.

Eric Veach and Leonidas Guibas. 1995. Bidirectional Estimators for Light Transport. In Photorealistic Rendering Techniques, Georgios Sakas, Stefan Müller, and Peter Shirley (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 145–167.

Eric Veach and Leonidas J. Guibas. 1997. Metropolis Light Transport. Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97 (1997), 65–76.

Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. 2019. Path Guiding in Production. , Article 18 (2019), 77 pages.

Bruce Walter, Shuang Zhao, Nicolas Holzschuch, and Kavita Bala. 2009. Single Scattering in Refractive Media with Triangle Mesh Boundaries. ACM SIGGRAPH 2009 papers (July 2009), 1–8.

Beibei Wang, Miloš Hašan, and Ling-Qi Yan. 2020. Path Cuts: Efficient Rendering of Pure Specular Light Transport. ACM Transactions on Graphics 39, 6 (Dec. 2020), 1–12.

Mark Watt. 1990. Light-water interaction using backward beam tracing. In Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques (Dallas, TX, USA) (SIGGRAPH '90). Association for Computing Machinery, New York, NY, USA, 377–385.

Tizian Zeltner, Iliyan Georgiev, and Wenzel Jakob. 2020. Specular Manifold Sampling for Rendering High-Frequency Caustics and Glints. ACM Transactions on Graphics 39, 4 (Aug. 2020).