Path Cuts: Efficient Rendering of Pure Specular Light Transport

BEIBEI WANG, School of Computer Science and Engineering, Nanjing University of Science and Technology
MILOŠ HAŠAN, Adobe Research
LING-QI YAN, University of California, Santa Barbara

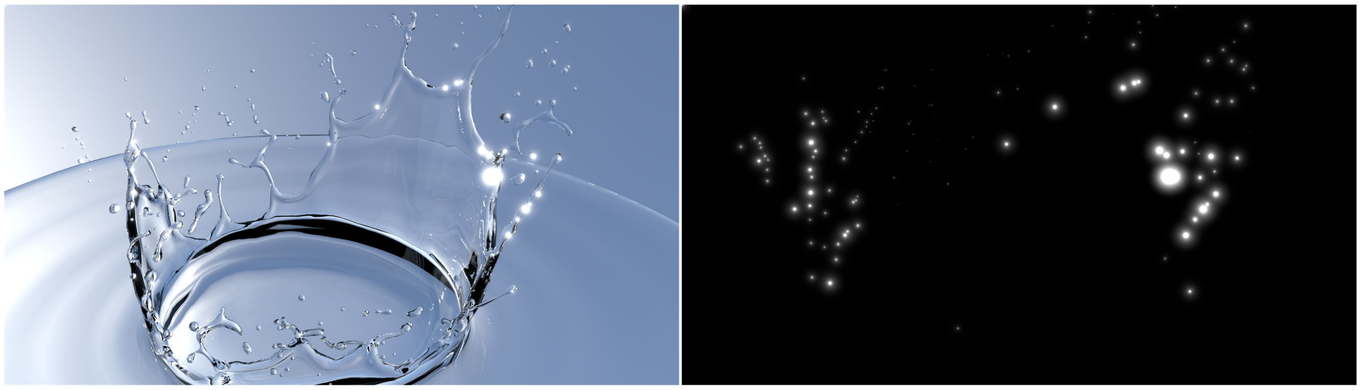
Our Result (TT + environment lighting)
Our Result (TT), 7.36s, Glint count: 159

Fig. 1. Rendering results of our method on the Splash scene: with pure specular paths and environment light (left) and pure specular paths only (right). We simulate a TT (transmittance-transmittance) and a TRT (transmittance-reflection-transmittance) light transport. The specular paths start on a point light, encounter perfectly specular reflective/refractive events, and end on the pinhole camera. We postprocess the images with a bloom filter, to better show the intensity of specular path contributions.

In scenes lit with sharp point-like light sources, light can bounce several times on specular materials before getting into our eyes, forming purely specular light paths. However, to our knowledge, rendering such multi-bounce pure specular paths has not been handled in previous work: while many light transport methods have been devised to sample various kinds of light paths, none of them are able to find multi-bounce pure specular light paths from a point light to a pinhole camera. In this paper, we present path cuts to efficiently render such light paths. We use a path space hierarchy combined with interval arithmetic bounds to prune non-contributing regions of path space, and to slice the path space into regions small enough to empirically contain at most one solution. Next, we use an automatic differentiation tool and a Newton-based solver to find an admissible specular path within a given path space region. We demonstrate results on several complex specular configurations, including RR, TT, TRT and TTTT paths.

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

Additional Key Words and Phrases: pathcuts, specular, light transport

Authors' addresses: Beibei WangSchool of Computer Science and Engineering, Nanjing University of Science and Technology; Miloš HašanAdobe Research; Ling-Qi YanUniversity of California, Santa Barbara.

0730-0301/2020/12-ART238 $15.00
https://doi.org/10.1145/3414685.3417792

1 INTRODUCTION

Geometrically complex surfaces with highly specular materials occur on many common objects around us. With sharp point-like light sources (such as the sun, or indoor LED lights), light can bounce several times on these specular materials before getting into our eyes, forming (to a close approximation) purely specular light paths. Both reflections and refractions can be present, producing interesting glinty effects. However, to our knowledge rendering such multi-bounce pure specular paths has not been handled in previous work.

In physically-based rendering, a typical approach is to find paths connecting the light source and the camera in a probabilistic fashion. A Monte Carlo estimator typically divides the contribution of a constructed path by the probability of the path being found (in an appropriate measure) [Veach 1997]. While many light transport methods have been devised to sample various kinds of light paths, none of them are able to find multi-bounce pure specular light paths from a point light to a pinhole camera. Thus, this component of the light transport will simply be missing from the resulting image in current rendering systems, research or commercial.

Of course, one could *regularize* the problem by instead using a low non-zero roughness and/or a small but finite light size, which lets one compute approximations with standard algorithms [Kaplanyan and Dachsbacher 2013]. Such regularization approaches will still have a hard time finding the solution with zero roughness. To uncover what the ground truth solution looks like, and how many path connections it contains, we need to study the pure specular setting directly. We believe that a deeper understanding of this idealized situation can lead to new visual effects and progress in algorithms for the general case.

Let an *admissible* specular path be one that at every vertex follows the rules of perfect specular reflection (refraction) of the underlying material. Finding admissible specular paths, by initializing with a nearby non-admissible path and refining through a root solver such as Newton's method, is not difficult and has been studied by a number of previous methods, though in slightly different contexts [Hanika et al. 2015a; Jakob and Marschner 2012; Mitchell and Hanrahan 1992; Walter et al. 2009]. We cannot directly use any of these approaches in the pure specular setting with many admissible paths, as they only solve the local problem of refining a path that is already close to admissible; the exception is Walter's method, which solves the global problem by using a hierarchical data structure on top of meshes with interpolated normals, but only works for a single refraction.

In this paper, we present *path cuts* to efficiently render light paths through a chain of multiple pure specular reflections and refractions, from a point light to a pinhole camera, under geometric optics. Our approach is deterministic: we enumerate the discrete set of admissible pure specular paths of a given length and type connecting the light and the camera. We globally slice the path space into small regions; within a region we solve the local problem of refining the path to become admissible. In summary, we resolve the following challenges:

- We use a path space hierarchy combined with interval arithmetic bounds to efficiently prune non-contributing regions of path space, and to slice the path space into regions small enough that local refinement becomes feasible.
- We use an automatic differentiation tool and a Newton-based solver to find an admissible specular path within a given path space region, and splat its contribution onto the image plane.
- We show that our purely specular solution can be used to initialize paths for other algorithms, such as an MCMC-based approach to render with small but non-zero roughness.

While several of these ideas were explored in different settings in previous work (some already in the work of Mitchell and Hanrahan [1992]), ours is the first approach to directly focus on multi-bounce purely specular paths on complex geometric surfaces given by triangle meshes with interpolated normals.

The rest of the paper is arranged as follows. In Sec. 2, we review related path construction methods and motivate our work. Then we present our method in three parts: path space exploration through path cuts (Sec. 3), solving for a specular path and calculating a path's contribution to the image plane (Sec. 4), and discuss the possibility of multiple solutions. After that, we show and discuss our results in Secs. 5 and 6. Finally, in Sec. 7, we summarize our method and propose future directions.

2 RELATED WORK

Solving for admissible specular paths by root finding, typically using a variation of Newton's method, has been well studied. Mitchell and Hanrahan [1992] note that admissible paths are the ones with locally extremal total lengths (minimal or maximal), according to the Fermat principle. They use the Newton method to find the vertex positions that minimize (maximize) the total length, assuming surfaces represented as implicit functions.

However, Fermat's principle no longer holds for primitives with differing geometric and shading normals, common in realistic scenes. Walter et al. [2009] propose a similar solution to handle one-bounce refractions through a mesh of primitives with interpolated shading normals. Instead of relying on Fermat's principle, the admissible paths are simply the ones whose vertices align the normal vector with the (refractive) half vector. Their method is used in the context of single scattering in media with refractive boundaries, connecting a volumetric scattering vertex to a light vertex outside of the medium, in a direct illumination (next event estimation) configuration. Furthermore, Walter et al. use a hierarchy to isolate all possible connections, though (unlike our method) it is limited to a single refractive surface event. We show a comparison to Walter's method, applying our solution to the single refraction setting.

Manifold exploration [Jakob and Marschner 2012] allows for the mutation of specular chains terminating on a non-specular vertex in a Markov chain Monte Carlo integrator (e.g. Metropolis light transport). The mutation is achieved by perturbing the location of the terminal non-specular vertex, and adjusting the whole chain to become admissible again. However, this method does not work for pure specular paths, and requires the Markov chain light transport setting to be applicable. Note that a manifold of admissible paths only exists if the terminating vertex is non-specular. In our case of pure specular paths, there is no manifold, but instead a discrete set of admissible paths (except for rare degenerate cases).

Manifold next event estimation (MNEE) [Hanika et al. 2015a] is similar to Walter et al.'s approach, but not specific to scattering media. Instead, the method can connect any shading point on a non-specular surface, through one or more refractions, to a light source sample. This is done through initializing a non-admissible connection and refining it to an admissible one. However, unlike Walter et al., no global search is used to isolate all potentially valid solutions, so the method is simpler and faster than Walter's, but it does not apply in cases where many admissible connections can be made, i.e. in settings where the global component of the problem is non-trivial.

Kaplanyan et al. [Kaplanyan et al. 2014] proposed half vector space light transport (HSLT), in which a path is represented by its start and end point constraints and a sequence of generalized half vectors. This enables efficient sampling of specular or close-to specular interaction in the context of Markov chain Monte Carlo light transport. Compared to [Jakob and Marschner 2012], HSLT does not require a specular/non-specular classification of the path vertices. Later, Hanika et al. [Hanika et al. 2015b] further improved upon this method for difficult input geometry, resulting in a more practical and faster approach. However, neither of these approaches focus on pure specular light transport.

Our approach shares the same optimization-based admissible specular path solving routine with the above methods. The difference is that we aim at exhaustively locating *pure* specular light paths from a point light to a pinhole camera. In contrast, all of the above methods are focused on finding subpaths of one or more specular vertices, terminating on a *non-specular vertex* (surface or volume). Furthermore, we are looking to deterministically enumerate all possible admissible paths up to a given length. In that sense, our work is closest to Walter et al.'s, but instead of single-refraction subpaths terminating on a scattering vertex, we support multiple reflection/refraction events terminating at a (pinhole) camera.

Recently, concurrent work by Zeltner et al. [2020] presented a specular manifold sampling technique, which is able to handle glints, reflective/refractive caustics, and specular-diffuse-specular light transport. Their method could also theoretically handle multi-bounce pure specular light transport. Ours is a deterministic approach, while theirs combines deterministic root finding with stochastic sampling in a Monte Carlo setting. The methods could also plausibly be combined, e.g. by using our path cuts to initialize their manifold sampling.

Interval arithmetic is a general method to turn a mathematical function (represented as an expression or a program), into a function that takes intervals as inputs and returns them as outputs, conservatively bounding the possible outputs of the original function. This works for both scalar and vector inputs and outputs. In graphics applications, 3D intervals (equivalent to bounding boxes) are common, with operators like dot product, norm and normalization frequently applied to such intervals. Mitchell and Hanrahan proposed the use of interval arithmetic for isolating specular paths already in 1992, in the paper discussed above. Velazquez-Armendariz et al. [2009] apply interval arithmetic to bound complex shader programs, representing reflectance functions including spatial variation and anisotropy. We use interval arithmetic to conservatively bound the possible normals and half vectors within a region of the scene; if these intervals have no intersection, we know there cannot be an admissible specular path and can reject further computation in this region of path space.

Glint rendering methods [Yan et al. 2014, 2016] turn the specular path finding problem to querying the distribution of normals of a surface patch around a shading point at the half vector between the incident and outgoing directions, which was later extended to handle wave optics effects [Yan et al. 2018]. These methods are limited to one bounce, and more importantly, can only support bump/normal mapped surfaces, otherwise the concept of a local normal distribution is not clearly defined. Jakob et al. [2014] proposed an procedural glints rendering approach, which is limited to random distributed appearance. More recently, Kuznetsov et al. [2019] proposed to use a generative adversarial network to procedurally generate specular glints from given examples with desired visual characteristics. In comparison, our method is able to solve for explicit multiple specular bounces (reflections and refractions), and works for both bump/normal mapped surfaces and actual geometric primitives, possibly with differing geometric and shading normals. We provide a comparison of our result for single reflection to Yan et al. [2014], using very low roughness in their method.

Lightcuts [Walter et al. 2006, 2005] is a many-light method designed to render direct and global illumination using virtual point lights. The lights are organized in a hierarchical data structure. During rendering, the hierarchy is traversed top-down, based on whether the contributions of all the lights within a node can be approximated well enough. In multi-dimensional lightcuts, there is an additional hierarchy on the gather (shading) points, and the two

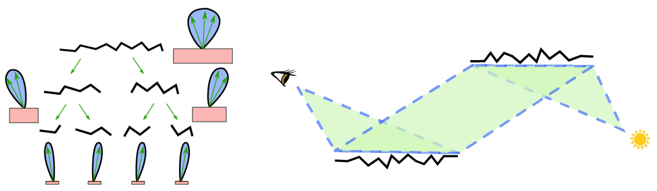
Fig. 2. Left: 2D spatial hierarchy of a surface, where each node records the range of normals (blue lobes) and positions (pink rectangles). Right: an illustration of a path cut.

hierarchies are traversed in parallel. Inspired by the idea of lightcuts, we propose path cuts to organize all the triangles, whether geometric or tessellated from bump/normal maps, into a hierarchy. We use the path cuts to efficiently prune non-contributing areas of the path space; i.e. regions where the alignment of the normal and half vectors is guaranteed to be impossible.

3 PATH SPACE TRAVERSAL USING PATH CUTS

For each number of bounces k the problem needs to be solved independently; different path types of a given length are themselves independent. In the following, we will assume a given fixed number of bounces k and a fixed path type, e.g. TT (transmit-transmit), TRT (transmit-reflect-transmit), etc.

Suppose we want to find a k-bounce specular path that connects a point light L and a pinhole camera (eye) E, thus the total number of path vertices is $k+2$, with $k+1$ segments. The path bounces at points $\mathbf{x}_i$ on scene surfaces $\mathcal{M}$, defined as general triangle meshes with interpolated normals. (This simplifies our implementation and exposition, though extensions to other surface representations would be possible.) Thus in the following, we assume our scene geometry is composed of n triangles T_i.

Our idea is to first isolate all k-tuples of triangles that could give rise to an admissible path, then find those paths within the k-tuple using a root solver. In this section, we focus on the first part, finding potentially contributing k-tuples of triangles.

3.1 Path cuts for hierarchical pruning

To find k-bounce paths of a given type, the most straightforward way is brute-force: loop over all k-tuples of primitives $(T_{j_1}, \cdots, T_{j_k})$, then consider potential paths

$$L \rightarrow (\mathbf{x}_1 \in T_{j_1}) \rightarrow (\mathbf{x}_2 \in T_{j_2}) \rightarrow \cdots \rightarrow (\mathbf{x}_k \in T_{j_k}) \rightarrow E. \quad (1)$$

This simple approach is certainly too slow: its time complexity is $O(n^k)$ for n triangles and k bounces. To improve performance, we build a tree hierarchy $\mathcal{H}$ over the set of scene surfaces $\mathcal{M}$. Each node $S_i \in \mathcal{H}$ represents a surface patch on $\mathcal{M}$ (normally a set of spatially grouped triangles), and records the 3D interval (bounding box) N_i of all surface normals and the 3D interval (bounding box) P_i of all surface positions in this patch. Note that this "root" path cut is for the entire image, as our method is independent of pixels.

When looking for a specular path of a given type with k bounces, we consider a *product hierarchy* $\mathcal{H}^k = \mathcal{H} \times \cdots \times \mathcal{H}$: a k-tuple of copies of hierarchy $\mathcal{H}$. Now suppose we choose a k-tuple of nodes

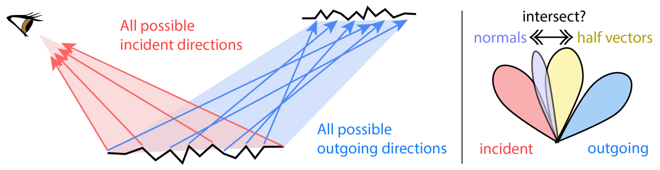
All possible incident directions
All possible outgoing directions
intersect?
normals
half vectors
incident
outgoing

Fig. 3. Left: for any node S_i, we calculate all of its incident and outgoing directions in a path cut. Right: to validate a path cut, at every node S_i along a path cut, we calculate the range of its half vectors, and compare it with the range of normals stored in this node.

$(S_{j_1}, \cdots, S_{j_k}) \in \mathcal{H}^k$; an analogy to the pair of hierarchies in multi-dimensional lightcuts [Walter et al. 2006]. Now consider a "thick path": the set of all paths from the light to the camera whose vertices are within the corresponding nodes

$$L \to (\mathbf{x}_1 \in S_{j_1}) \to (\mathbf{x}_2 \in S_{j_2}) \to \cdots \to (\mathbf{x}_k \in S_{j_k}) \to E \qquad (2)$$

We denote such a "thick path" as a *path cut*, in analogy to multi-dimensional lightcuts. This is illustrated in Fig. 2.

Our insight is that, if we can quickly determine whether a path cut potentially contains an admissible specular path, we can use this information to prune the path space to quickly converge to contributing paths. So, the first question is how to determine whether a path cut can contain an admissible specular path. The answer to this question should be *conservative*: existence of one or more admissible paths has to be always correctly detected. The reverse need not be true: if we cannot prove non-existence, we can always subdivide the path cut.

3.2 Validating a path cut

Recall that a pure specular path is admissible if at each surface bounce, the normal vector is aligned with the (reflective or refractive) half vector. For a path cut, we are able to test locally if every node is potentially able to produce such an alignment. See Figure 3.

To do that validity check, we use interval arithmetic. For a given hierarchy node S_i, we record a 3D interval P_i that bounds all the positions of surface points in S_i, and a 3D interval N_i that bounds all the normals of S_i. Some of these intervals could be trivial, i.e. contain only a single point, which is a valid interval. We also treat the camera and light positions as such trivial intervals.

For a given node S_i, all possible incident directions from S_{i-1} to S_i can then be bounded using the following interval expressions:

$$D^i_{\text{in}} = \text{normalize}(P_{i-1} - P_i), \qquad (3)$$

where both P_i and P_{i-1} are interval boxes (3D vectors with interval elements) that bound the positions, and the normalize function takes and returns an interval box, as shown in Figure 4. Similarly for the outgoing directions from S_i to S_{i+1}, we have

$$D^i_{\text{out}} = \text{normalize}(P_{i+1} - P_i). \qquad (4)$$

Bounding the reflective half vectors D of all the incident-outgoing direction pairs is then achieved by the interval expression

$$H_i = \text{normalize}(D^i_{\text{in}} + D^i_{\text{out}}), \qquad (5)$$

and similarly for refractive half vectors, where additional scaling by indices of refraction is needed.

To detect whether this interval of all half vectors H_i and the interval of all normals N_i can align, we can check whether interval intersection $H_i \cap N_i$ is non-empty. This simply entails checking whether the two intervals (bounding boxes) intersect. To determine the entire path cut's validity, we check the intersection for all bounces i. It allows us to stop the computation for the entire path cut as long as at least one $N_i \cap H_i$ is empty.

3.3 Subdividing a path cut

Once we know that a path cut has potential to contribute a pure specular path, we subdivide this path cut. The most straightforward way is to subdivide all non-leaf nodes along the path. It is also possible to randomly choose a node to subdivide. We propose to always subdivide the node with the largest bounding box.

As we repeat the subdivision process, finally we will end up with path cuts that consist of only leaf nodes, i.e. one triangle per bounce. In this case, we proceed to the next stage to actually find a pure specular light path within this region of path space.

Discussion. For the subdivision heuristics, we tried two solutions: the node with the largest position bounding box, and the node with the largest intersection interval $H_i \cap N_i$, measured using the maximum side length. We found the second solution visited more nodes in the hierarchy (about 40%). Thus, we use the first solution in our implementation.

4 SOLVING FOR A SPECULAR LIGHT PATH

Having determined the triangles T_i at each bounce of the leaf path cut, we now need to solve for a specular light path

$$L \to (\mathbf{x}_1 \in T_1) \to \cdots \to (\mathbf{x}_K \in T_K) \to E, \qquad (6)$$

and find its contribution to the image plane.

4.1 Finding an admissible path

Finding such a light path is trivial if the triangles are flat mirrors (i.e. they do not have interpolated shading normals distinct from the geometric normal). We just need to repeatedly take the virtual image of the point light across the plane containing T_i and finally connect it to the camera. However, this does not cover common situations with refractions and normal interpolation (curvature). In the general case, we need to find the vertex locations using a root solver.

We represent positions within each triangle T_i using barycentric coordinates (α_i, β_i). The third component γ_i is implied by $\alpha_i + \beta_i + \gamma_i = 1$. With this representation, given any position $\mathbf{x}_i(\alpha_i, \beta_i)$, we know its interpolated normal $\mathbf{n}_i(\alpha_i, \beta_i)$. At the same time, the incident and outgoing directions can be calculated by normalizing $\mathbf{x}_{i-1} - \mathbf{x}_i$ and $\mathbf{x}_{i+1} - \mathbf{x}_i$. This lets us compute the half vector at $\mathbf{x}_i$, denoted as $\mathbf{h}_i(\alpha_i, \beta_i)$. Note that the half vector notation here is general, which can be either reflective or refractive [Walter et al. 2007]. By convention, we assume all normals and half vectors point into the medium with the lower index of refraction (usually though not necessarily air). With the above definitions, we can write a constraint function

$$\mathbf{C}_i(\alpha_i, \beta_i) = \mathbf{n}(\alpha_i, \beta_i) - \mathbf{h}(\alpha_i, \beta_i) \qquad (7)$$

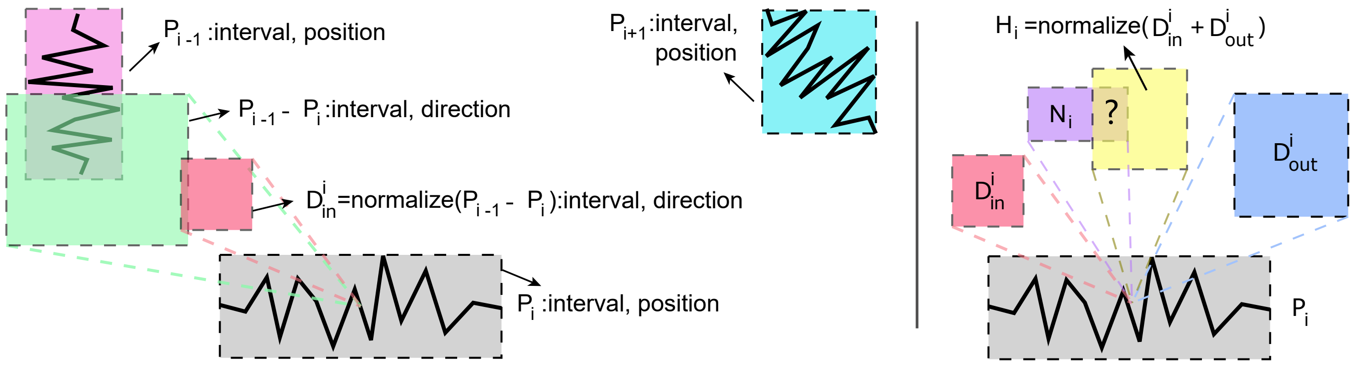
P$_{i-1}$:interval, position
P$_{i-1}$ - P$_i$:interval, direction
D$^i_{in}$=normalize(P$_{i-1}$ - P$_i$):interval, direction
P$_i$:interval, position
P$_{i+1}$:interval, position
H$_i$ =normalize(D$^i_{in}$ + D$^i_{out}$)
N$_i$
?
D$^i_{in}$
D$^i_{out}$
P$_i$

Fig. 4. Left: calculating the interval of all the possible incident directions D^i_{in} from node S_{i-1} to node S_i. Right: testing if the interval of half vectors overlap the interval of the normals.

whose roots correspond to admissible local bounces on triangle T_i, with the additional barycentric constraint $\alpha_i \geq 0, \beta_i \geq 0, \alpha_i + \beta_i \leq 0$. We finally stack all constraint functions along the path into an overall constraint

$$\mathbf{C}(\alpha_1, \beta_1, \ldots, \alpha_k, \beta_k) = (\mathbf{C}_1, \ldots, \mathbf{C}_k). \tag{8}$$

Therefore, the constraint function has $2k$ variables and $3k$ output values. (An alternative with $2k$ output values can be defined by projecting the normals and half vectors into the local geometric space, and dropping their z coordinates. We found this alternative formulation gives equivalent results and is slightly slower due to the extra vector transformations.)

As we discuss later in sec. 4.3, the constraint will have a discrete number of roots. Finding the roots of this constraint function means that the surface normals and half vectors match at all bounces, which gives us an admissible pure specular light path. This approach is similar to the target function of Walter et al. [2009] and the constraint function of Jakob et al. [2012]. We use automatic differentiation of the constraint, using the *autodiff* component of the Eigen C++ library. This gives us automatic support for all required cases.

We use Newton iteration to find the root of the constraint function $\mathbf{C}$, iteratively approximating it as locally affine, and finding the root of this affine approximation. This requires solving a $3k \times 2k$ linear system, in the least-squares sense, at each iteration. We solve this by inverting the least squares system using normal equations; this is equivalent to Walter et al.'s approach, except we have potentially more dimensions due to handling multiple bounces. We bound the maximum number of iterations to 5. If after reaching the maximum iterations, the resulting vertices are not all within their respective triangles, we assume there is no solution. A slower version of Newton's method with strong guarantees on finding all solutions can also be used, though the difference in practice is minimal, and the cost can be prohibitive, as discussed later in sec. 4.3.

When a light path has been found, we perform a visibility test for the $k+1$ segments along this path. If no segments are blocked in the middle, the path will be accepted and contribute to the image plane. The visibility test can be skipped for objects known to be convex. Next, we describe how to calculate a path's contribution to the image plane.

4.2 The contribution of an admissible path

The contribution of the path to the image consists of the product of several terms: visibility (already handled above), light intensity, the product of Fresnel terms along the path, volumetric absorption, the generalized geometry term, and the pixel importance function value. Below we discuss the last two terms; all other terms are straightforward to include.

For both of these terms, we will need to precisely define our camera model as a pinhole camera, with a sensor plane at a distance of 1 world unit from the camera position (pinhole). For the purposes of the explanation below, the camera endpoint P of the path is thus on the sensor, not on the pinhole C itself.

Pixel importance function value. Consider a ray (P, ω) leaving the sensor point P in direction $\omega = \text{normalize}(C - P)$. Denote the importance function of a single pixel (i, j) as $W_{ij}(P, \omega)$. We will need to include the corresponding W_{ij} in the path contribution.

This importance function should be normalized to integrate to 1, but in which units? These units cannot be pixels, otherwise the brightness of specular contributions would change with image resolution. Instead, we find that the normalization has to be with respect to real world units, on the image sensor at a distance of 1 unit from C.

We can easily check if the direction is outside of the view frustum, in which case the value is zero. Otherwise, if the horizontal field-of-view is θ and the aspect ratio (height/width) is r, the area of the image plane in world units is

$$A = 4r \tan(\theta/2)^2,$$

because the image width and height are $2\tan(\theta/2)$ and $2r\tan(\theta/2)$, respectively. The function $W(P, \omega)$ should be normalized across the image plane, so its value is

$$W_{ij}(p, \omega) = \frac{K_{ij}(P)mn}{A}.$$

Here we first define $K_{ij}(P)$ to be the pixel reconstruction filter normalized in pixel units, and then include the additional factor of $\frac{mn}{A}$ to make W_{ij} normalized in world units.

Note that real cameras are "photon counters", so smaller pixels record less light, thus image brightness will depend on resolution.

In rendering, however, we usually use a “radiance meter” instead of a “photon counter” model for cameras: we would like the final pixel values to be in units of radiance, not photon flux, and we generally do not prefer brightness to change with image resolution. The above solution implements the radiance meter model, and could be easily modified to the photon counter model.

Generalized geometry term. We use the terminology “generalized geometry term” (GGT), which was introduced by Jakob and Marschner [2012]; however, Mitchell and Hanrahan [1992] and Walter et al. [2009] have included the equivalent term before, referring to it as “intensity” or “distance correction factor”, respectively.

This term is essentially the Jacobian determinant of a ray propagation function (it is also related to wavefront Gaussian curvature, as noted by Mitchell and Hanrahan). To compute the Jacobian, we simply need to compute gradients of such ray propagation functions over specular paths. We could do that using automatic differentiation; however, ray differentials already give the solution to this exact problem explicitly [Igehy 1999]. Walter et al. [2009] and Holzschuch [2015] also use ray differentials to derive the value of the term for a single refraction. We show the approach works for any number of reflections and refractions.

Consider the ray propagation function $F : \mathbb{R}^2 \to \mathbb{R}^2$ that maps a neighborhood of the camera sensor point P through the admissible specular path $\bar{\mathbf{x}}$ to the imaginary plane crossing the point light position, orthogonal to the last path segment. The GGT is the value

$$G(\bar{\mathbf{x}}) = 1/|\det J_F(P)|, \tag{9}$$

where $J_F(P)$ is the Jacobian of F, a 2×2 matrix for a given value of P.

We initialize position differentials P_x and P_y of unit length on the camera sensor plane, and trace them along the specular path to the imaginary plane crossing the point light position, orthogonal to the last path segment, finding the final position differentials L_x and L_y. Finally, we can compute the GGT as

$$G(\bar{\mathbf{x}}) = 1/|L_x \times L_y|, \tag{10}$$

where $\times$ denotes the cross product.

Finally, note that the value $|L_x \times L_y|$ can become very small or even zero, causing singularities. This occasionally leads to glints that are unnaturally bright, potentially infinite. This may seem paradoxical but is actually correct in the framework of geometric optics on pure specular surfaces; the same issue was also noted by Yan et al. [2014] in the single specular reflection context. This is not observed in reality, where light transport does not follow this idealized framework; geometric optics and perfect specularity are always violated to some extent in the real world. Therefore, we optionally regularize the cross product value as

$$|L_x \times L_y| \approx \|L_x\| \cdot \|L_y\| \cdot \max(\epsilon, |\text{normalize}(L_x) \times \text{normalize}(L_y)|), \tag{11}$$

as we observed that the low values are typically due to almost parallel L_x and L_y, not due to $\|L_x\|$ or $\|L_y\|$ being close to zero. A value of $\epsilon = 0.01$ works well in general. We observe that this regularization solves the issue of occasional very bright glints, while not affecting the remaining glints.

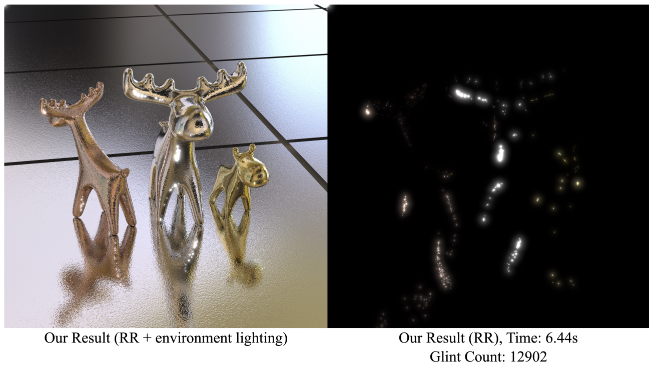
Our Result (RR + environment lighting)
Our Result (RR), Time: 6.44s
Glint Count: 12902

Fig. 5. Rendered results of RR light transport on the Deer scene.

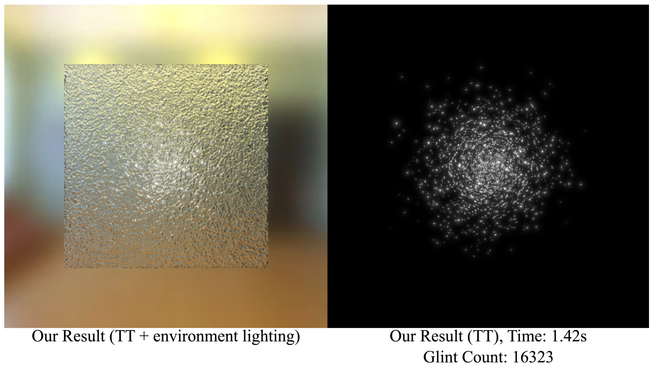
Our Result (TT + environment lighting)
Our Result (TT), Time: 1.42s
Glint Count: 16323

Fig. 6. Rendered results of TT light transport on the Double Slab scene.

4.3 Multiple solutions

The previous treatment assumed that there is at most one solution for an admissible path within a leaf path cut, and that this solution will be found by Newton’s method if it exists. While these properties appear to be generally true in our experiments, they are not theoretically guaranteed. In the following, we will discuss how to find all discrete solutions with extra effort, as well as theoretically discuss the impossibility of infinite solution regions in our current surface representation based on triangles with interpolated normals.

Finding all discrete solutions. Even if a single solution exists, standard Newton’s method still does not theoretically guarantee that the solution will be found. Furthermore, there is a possibility of multiple discrete solutions. Both of these issues can be theoretically handled with the interval version of Newton’s method, as suggested by Mitchell and Hanrahan [1992] and Walter [2009]. Specifically, both of these works used the approach introduced by Krawczyk [1969], which only requires a (pseudo-)inversion of a standard non-interval matrix. We also implemented this method.

However, even though this approach is theoretically sound, our experiments indicate that the results tend to be visually equivalent to the simpler Newton’s method (see Fig. 15). The major added computational expense is not generally worth the effort, and we do not currently use this approach in our results. Our meshes are

Table 1. Scene settings, computation time and memory costs for our test scenes. #Tri. is the count of triangles in the scene. Type means the type of light transport, such as Reflection-Reflection, Transmittance-Transmittance, or Transmittance-Reflection-Transmittance. Traversal means path cut hierarchical pruning until a path consisting of leaf nodes, Newton solving means solving for specular light paths with Newton solver, and splatting means projecting the specular light paths to the screen space and computing the pixel radiance. #Glint is the glint count.

Scene	Fig.	#Tri.(K)	Material.	Type	Time (sec.)				Memory (MB)	#Glint
					Traversal	Newton solving	Splatting	Total		
Deer	Fig. 5	9210.5	Metal	RR	4.51	1.86	0.07	6.44	1000.87	12902
Single Slab	Fig. 12	500	Metal	R	0.16	0.05	0.034	0.24	56.47	5135
Double Slab	Fig. 6	1000	Glass (1.50)	TT	1.00	0.32	0.095	1.42	112.94	16323
Double Slab	Fig. 8	1000	Glass (1.50)	TRT	5.46	6.42	0.072	11.95	112.94	9454
Double Slab	Fig. 10	2000	Glass (1.50)	TTTT	42.76	51.65	0.014	94.42	225.87	721
Splash	Fig. 1	49.8	Water (1.33)	TT	6.36	1.13	0.008	7.50	4.64	174
Splash	Fig. 1	49.8	Water(1.33)	TRT	135.32	345.35	0.009	480.68	4.64	212
Stained Glass	Fig. 9	205.5	Glass (1.50)	TT	30.28	22.73	0.018	53.03	20.84	3502

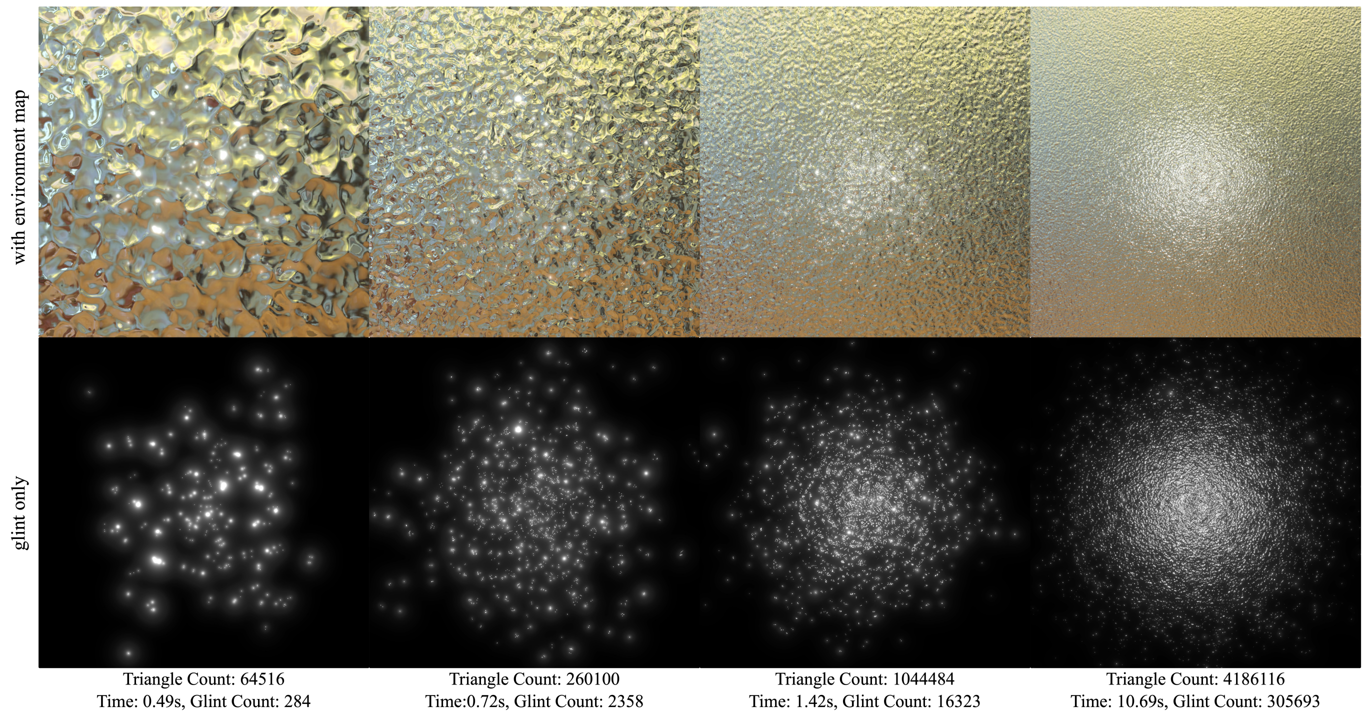
with environment map
glint only
Triangle Count: 64516
Time: 0.49s, Glint Count: 284
Triangle Count: 260100
Time:0.72s, Glint Count: 2358
Triangle Count: 1044484
Time: 1.42s, Glint Count: 16323
Triangle Count: 4186116
Time: 10.69s, Glint Count: 305693

Fig. 7. Rendered results for varying bump frequencies on the Double Slab scene.

already subdivided to small enough triangles, so multiple solutions within a single triangle do not frequently occur in practice.

Specifically, we ran experiments with interval Newton method for single reflection (R) and double transmission (TT) configurations. In the reflection case, we find that with interval Newton method, 5015 solutions are found; without intervals, there are 5007 found. In the TT case, 1052 solutions are found by interval Newton method, while 1036 are found by regular Newton. The final renderings in these experiments are visually equivalent between regular and interval Newton method when viewed side-by-side.

We note that in either case (R and TT), the regular Newton solutions are not a perfect subset of the interval Newton solutions; a small number of regular solutions are not among the interval solutions. This is because the final Newton refinement is subject to an epsilon value (0.00001 in these tests): some solutions may be found that satisfy the epsilon but are not true roots, and these special cases differ between the two methods.

In the TT case, the interval Newton solution case costs 23 hours while the regular (non-interval) solution is found in 0.6 seconds. The massive additional cost is not due to the evaluation of the additional interval conditions (which add a fairly minor expense), but due to

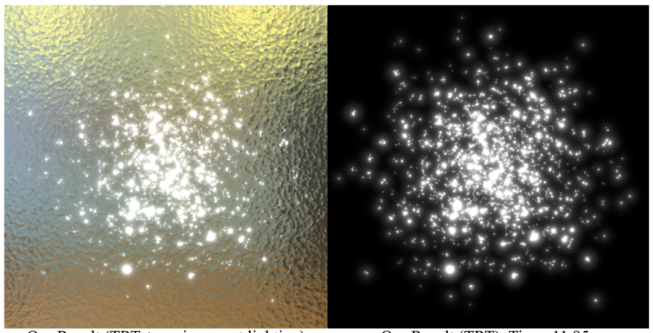
Our Result (TRT + environment lighting)
Our Result (TRT), Time: 11.95s
Glint Count: 9454

Fig. 8. Rendered results of TRT light transport on the Double Slab scene.

the fact that very fine subdivision is required until the conditions are able to prove the uniqueness of solutions.

Infinite solution regions. For general surface representations, infinite solution regions are possible for pure specular paths. Consider, for example, a cylinder that is reflective on the inside. A light and a camera are placed at the centers of each of the two caps of the cylinder. The circle at the middle of the cylinder now gives an infinite set of solutions where the normals are aligned with the half vectors.

Note, however, that our method does not currently support quadric primitives such as cylinders, only triangle meshes with linearly interpolated normals. With this primitive type, infinite regions of solutions are not possible. At a high level, the reason is that the half vector function will always contain non-linear terms, which make it impossible to match the affine function given by normal interpolation (before normalization) over any infinite continuous region. This argument is expanded in Appendix A.1.

5 RESULTS AND COMPARISON

We have implemented our algorithm inside the Mitsuba renderer [2010]. All timings in this section are measured on a 2.20GHz Intel i7 (40 cores) with 32 GB of main memory. Unless otherwise specified, all timings correspond to images with 2048×2048 pixels, except the Splash Scene with 1920×1080 pixels. In most of our results, we use a bloom filter (with a wide radial kernel) to make the intensity of the glints more perceptible; this is just an image-space post-process and not part of our core method. For most results, we provide two versions: one with specular paths from point lights only and the other with both point light and environment lighting. For environment lighting, the image component is computed using conventional path tracing and is completely orthogonal to our specular path method (the results are linearly combined).

5.1 Our scenes

In Table 1, we report all settings, computation times and memory cost for our test scenes. Below we discuss the specific scenes. We visualize the performance for different triangle counts within a fixed scene in Figure 16. Note that within a fixed scene, more subdivision inflicts only a small overhead on the performance, and the memory cost increases linearly. However, between scenes, and for different path types, the costs can differ dramatically.

Deer. Figure 5 illustrates three metallic characters on a bumpy plane under a point light and environment lighting. All objects are finely tessellated and the vertex positions are displaced with isotropic noise fields. The side objects are more rough (more noise displacement) than the object in the middle. The bumpy plane and the middle deer uses aluminum, the left deer uses copper and the right deer uses gold; these are just Fresnel term variations. We simulate the RR (reflection-reflection) pure specular light transport in this scene.

Double Slab. Figure 6 shows a slab of two refractive interfaces, whose vertices are displaced with a 512×512 height field. Both interfaces have a glass material assigned (index of refraction 1.5). A point light source is located below the bottom interface, and the camera is above the top interface. We simulate the TT light transport (transmittance-transmittance) in this scene. Note that this path type is very fast; our method took just 1.42s.

In Figure 7, we show results (TT light transport) of the Double Slab scene with different tessellation levels. As the height field resolution increases, both the glint counts and the rendering cost increase. In Figure 8, we show the results of TRT light transport. Here the point light is placed at the same side as the camera. As expected, the rendering cost is more higher when rendering three bounces.

Splash. Figure 1 illustrates a splash with a water material under four point lights and environment lighting. We simulate the TT light transport in this scene. For this scene, we also compare the results rendered with path cuts to a brute force evaluation, by looping all pairs of triangles and using Newton solver to find the roots. They produce identical results, while the brute force approach is 300x slower than our method.

Stained Glass. Figure 9 illustrates a stained glass window with a glass material under *eight* point lights and environment lighting. The glass also features colored absorption in this case. We simulate TT light transport in this scene.

5.2 Comparisons to previous methods

We compare our results to path tracing with regularization, and to previous methods handling the subproblems for a single reflection [Yan et al. 2014] and a single refraction [Walter et al. 2009].

In Figure 10, we compare our method with path tracing. We use *two double slabs*, solving for TTTT light transport (four transmissions). To render this scene with path tracing, we use a microfacet model [Walter et al. 2007] to represent it, and slightly increase the roughness of the surface (α of 0.001). Despite the regularization, we found the distribution of the glints between our method and path tracing are overall similar. Meanwhile, we measure the total energy of the two images and they have the same magnitude. On the other hand, path-tracing is extremely slow to converge in this scene. We demonstrate this in Figure 11. We choose a single pixel and compute its radiance as a function of samples per pixel. the figure shows that the path tracing does not converge to a final value even with millions of samples. We also provide the result of

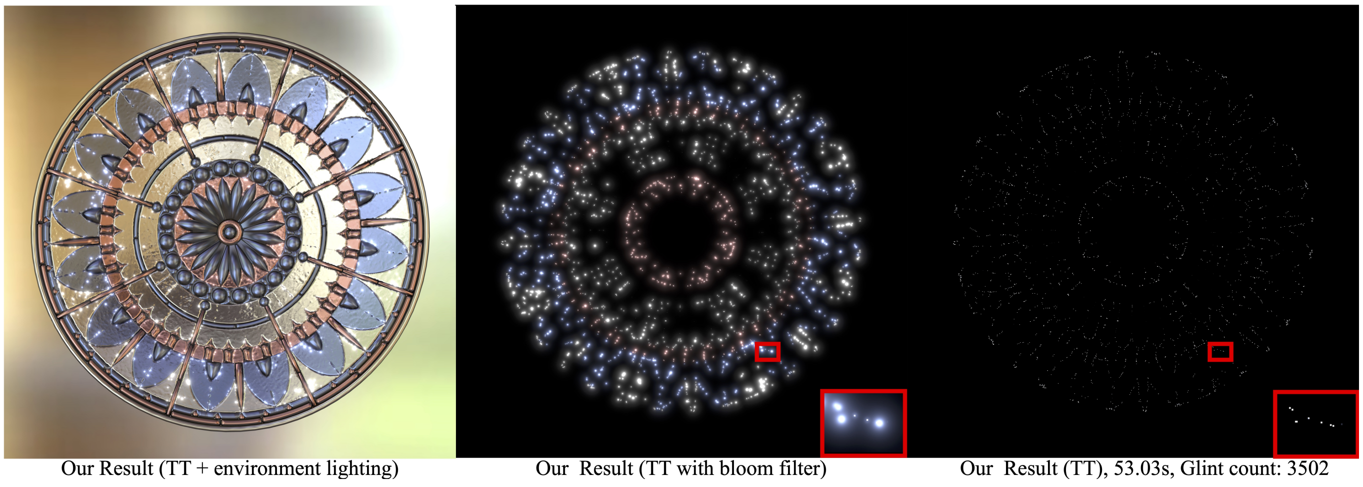
Our Result (TT + environment lighting)
Our Result (TT with bloom filter)
Our Result (TT), 53.03s, Glint count: 3502

Fig. 9. Rendered results of TT light transport on the Stained Glass scene.

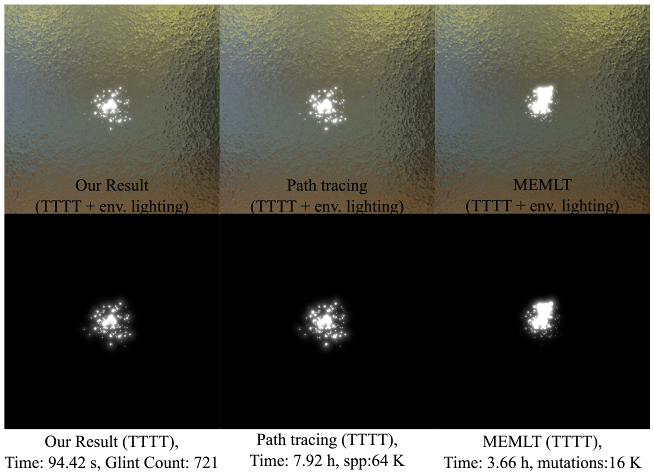

Fig. 10. Comparison between our method, path tracing and manifold exploration metropolis light transport (MEMLT) for TTTT light transport on the Two Double Slabs scene.

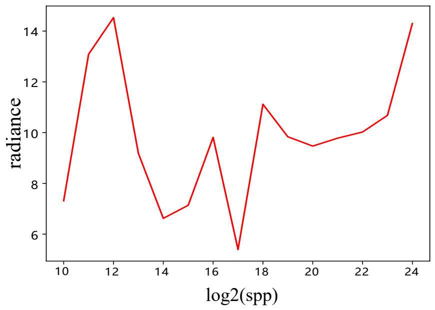

Fig. 11. Radiance of a single pixel as a function of samples per pixel in the path traced image of two double slabs (TTTT, four transmissions). Note the horizontal axis is logarithmic, showing that the path tracing does not converge even with millions of samples per pixel.

manifold exploration metropolis light transport (MEMLT) [Jakob and Marschner 2012]. This method only explores specular subpaths ending on a non-specular surface, so it has no specific estimators available to address pure specular paths. We increase the roughness of the surface (α of 0.001) and replace the point light source with a tiny spherical area light. With 16K mutations (3.66 hours), the result is still not converged.

Yan's method computes a single reflection from a normal-mapped surface, which can be easily converted to our representation. Their method assumes non-zero roughness, but works for very small values; here we compare our pure specular solution to theirs with an intrinsic roughness value of 0.00001. The result is in Figure 12. The visual agreement is close, despite very different algorithms. The brightness of some glints is slightly different due to the intrinsic roughness and different approximations made by Yan et al.

Walter's method computes single scattering in a medium with a refractive boundary. This requires integration along the refracted camera ray; for each scattering point on the ray, a connection to the point light source is found through a search for a valid refraction point on the boundary. While focused on single scattering, the search problem that is solved for a given scattering point on the ray is identical to our problem of constructing a pure specular path; we can thus replace this computation with our method. The result is shown in Figure 13. The two solutions closely match, and perform similarly, despite the completely separate code bases. Note that a later approach by Holzschuch [2015] performs better on this specific problem, but his solution is more specialized for single scattering and no longer constitutes pure specular path search.

5.3 Seeding MLT with our result

We introduce a variant of Metropolis light transport, seeded with the specular paths found by our method, where each seed path becomes an independent Markov chain for computing an image with non-zero roughness. The mutations are applied to path vertex positions

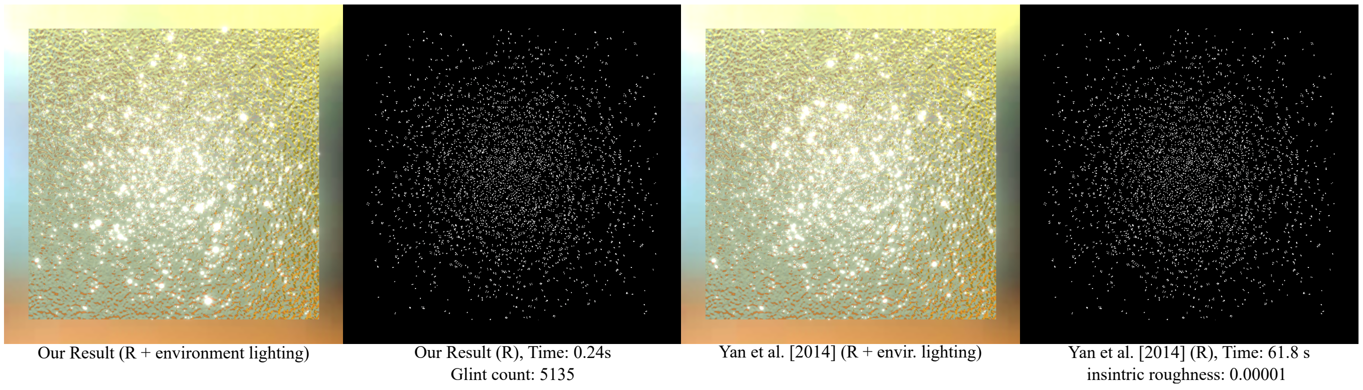
Our Result (R + environment lighting)
Our Result (R), Time: 0.24s
Glint count: 5135
Yan et al. [2014] (R + envir. lighting)
Yan et al. [2014] (R), Time: 61.8 s
insintric roughness: 0.00001

Fig. 12. Comparison between our method and Yan et al. [2014] with a slightly rough surface (intrinsic roughness 0.00001), for a single reflection. Note the close visual match, despite very different algorithms. Some difference in glint brightness is caused by the intrinsic roughness.

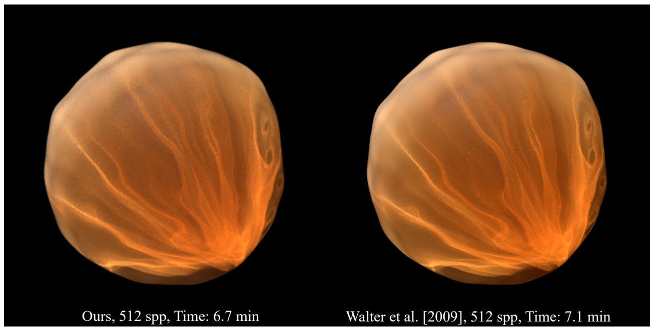
Ours, 512 spp, Time: 6.7 min
Walter et al. [2009], 512 spp, Time: 7.1 min

Fig. 13. Comparison between our method and Walter et al. [2009] on single scattering of a bumpy sphere scene with a volumetric medium within a refractive boundary. Our method is applied to find the connections from scattering points along the camera ray to the point light. The match between the images is close despite separate algorithms and code.

in texture coordinates; the mutation proposals are Gaussians in the texture space. The path contribution function uses a standard microfacet BRDF with low roughness (0.01), instead of our pure specular path contribution discussed above. Since the mutations do not significantly perturb the paths, we currently assume no visibility changes due to mutations, which speeds up the approach further. The result matches path-tracing with low roughness but is more efficient; see Figure 14.

This method is related to Bitterli et al. [2019]. They also generate some paths as seeds, but they use MLT to spread the energy and eliminate fireflies, while our method is for re-computing the energy-carrying paths in a new scene with glossy materials. Seeding MLT with our method also has several advantages over the original MLT or MEMLT, which generally suffer from two issues: getting stuck in sub-regions of the path space, and costly path mutations. Our example does well when dealing with these two issues, since our pure specular seeds cover the relevant parts of path space well. As the mutations are very local, we can also assume unchanged visibility, greatly lowering the cost.

6 DISCUSSION AND LIMITATIONS

There are several challenges not yet solved by our approach. First, the path length handled in our results is up to four specular bounces. For higher numbers of bounces, the search becomes expensive, as the number of actual solutions becomes combinatorially larger (so they cannot be pruned even with idealized pruning heuristics). Therefore, we cannot hope to deterministically enumerate all valid paths, but may still be able to find the perceptually relevant ones. This is an exciting problem for future work.

Furthermore, if one desires a hard guarantee of finding all contributing light paths of a given type, the interval Newton method needs to be used, which is slow and inconvenient. The problem of finding an efficient and yet conservative method remains open.

Finally, our interval arithmetic heuristics are conservative, and work reasonably in practice, but are not guaranteed to give the tightest possible bounds. It may be possible to further improve performance by using different bounding volumes, in addition to axis-aligned boxes.

Extending our method to handle displacement-mapped or normal-mapped geometry would be straightforward. An on-demand subdivision is first determined per geometric primitive based on local complexity of the displacement / normal map. Then every vertex is updated either by alternating its position or modifying its normal directly. The updated vertices then participate in follow-up operations as usual, including position-normal hierarchy construction, Newton iterations, and Jacobian computations. The other steps are exactly the same.

To extend the proposed MLT application to spatially-varying (textured) roughness, we would need three changes. First, during the position normal hierarchy construction, an additional roughness range is computed and stored for each node. Second, during the hierarchical traversal or path cut pruning, when validating a path cut (by intersecting the intervals of the half vector and the normal), the maximum roughness of that node is used. Third, when computing the contribution of a path, the roughness of each vertex along the path is obtained from the roughness map and then used for path contribution evaluation.

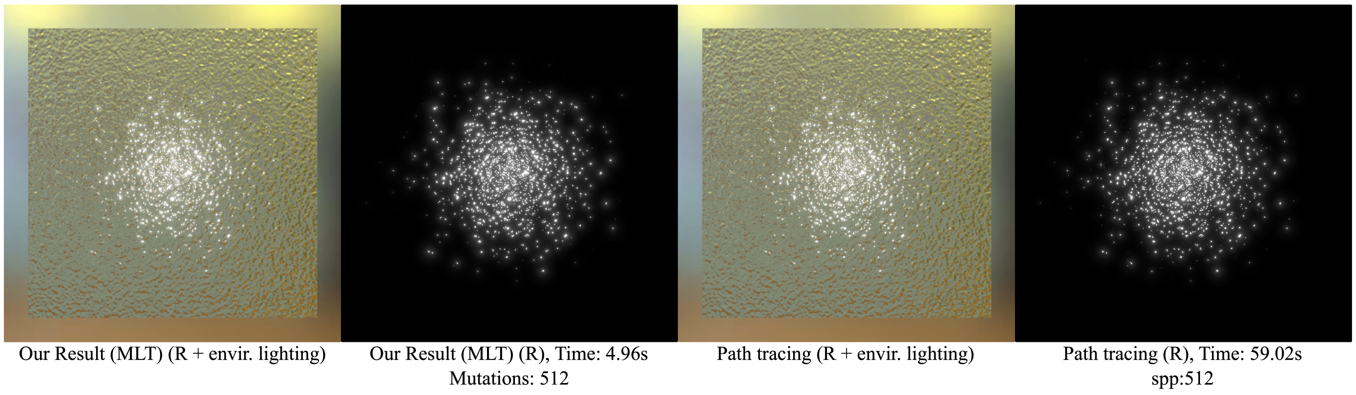
Our Result (MLT) (R + envir. lighting)
Our Result (MLT) (R), Time: 4.96s
Mutations: 512
Path tracing (R + envir. lighting)
Path tracing (R), Time: 59.02s
spp:512

Fig. 14. An image rendered using variant of Metropolis light transport, seeded with the specular paths found by our method, where each seed path becomes an independent Markov chain for computing an image with non-zero roughness. Our result matches path tracing with low roughness (note the anisotropic highlight shapes, which are not possible with zero roughness), but is more efficient.

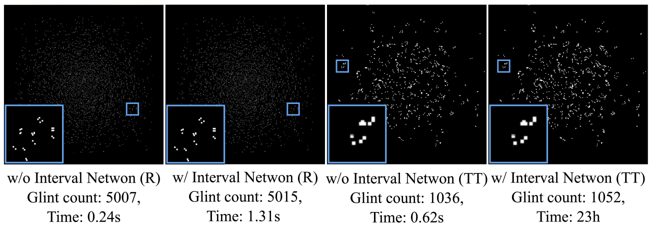

Fig. 15. Comparison between our method with and without the interval Newton method. There differences are visualy minimal, although interval Newton is much more costly.

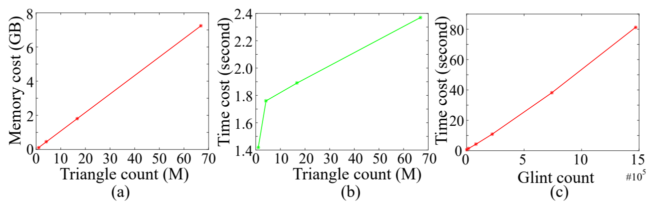

Fig. 16. (a) and (b): The impact of the triangle count on both the performance and storage costs in the Double Slab scene with different levels of subdivision. (c): The impact to the performance in the Double Slab scene of various glint counts, obtained by modifying the scales of the height field.

7 CONCLUSION AND FUTURE WORK

While many light transport methods have been devised to sample various kinds of light paths, none of them are able to find multi-bounce pure specular light paths from a point light to a pinhole camera. We presented *path cuts*, the first method able to render this component of light transport explicitly.

We use a path space hierarchy combined with interval arithmetic bounds to efficiently prune non-contributing regions of path space, and to slice the path space into small region where the search problem becomes local. Finally, we isolate admissible specular paths by a Newton solver applied to a constraint function, whose gradients are computed by automatic differentiation. We discuss in detail how such discovered paths should contribute to the image plane.

Our results show a number of geometrically complex scenes with many thousands of valid specular paths of up to four bounces, reflective and refractive. Our results closely agree with previous methods that solve related problems in the special cases of a single reflection/refraction.

We believe our approach fills a long-ignored gap in light transport algorithms, and we are interested in improving it further, by considering longer paths or more complex light transport models (e.g. based on wave optics).

ACKNOWLEDGMENTS

We thank the reviewers for the valuable comments. This work has been partially supported by the National Key R&D Program of China under grant No. 2017YFB0203000, the National Natural Science Foundation of China under grant No. 61802187 and 61872223, the Natural Science Foundation of Jiangsu under grant No. BK20170857, the Fundamental Research Funds for the Central Universities No. 30920021133 and China Postdoctoral Science Foundation under grant No.2020M671500.

REFERENCES

Benedikt Bitterli and Wojciech Jarosz. 2019. Selectively Metropolised Monte Carlo Light Transport Simulation. *ACM Trans. Graph.* 38, 6, Article 153 (Nov. 2019), 10 pages.
Johannes Hanika, Marc Droske, and Luca Fascione. 2015a. Manifold Next Event Estimation. *Comput. Graph. Forum* 34, 4 (July 2015), 87–97.
Johannes Hanika, Anton Kaplanyan, and Carsten Dachsbacher. 2015b. Improved Half Vector Space Light Transport. *Comput. Graph. Forum* 34, 4 (2015), 65–74.
Nicolas Holzschuch. 2015. Accurate computation of single scattering in participating media with refractive boundaries. *Computer Graphics Forum* 34, 6 (Sept. 2015), 48–59. https://doi.org/10.1111/cgf.12517
Homan Igehy. 1999. Tracing Ray Differentials *(SIGGRAPH '99)*. 179–186.
Wenzel Jakob. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.
Wenzel Jakob, Miloš Hašan, Ling-Qi Yan, Jason Lawrence, Ravi Ramamoorthi, and Steve Marschner. 2014. Discrete Stochastic Microfacet Models. *ACM Trans. Graph.* 33, 4, Article 115 (July 2014), 10 pages. https://doi.org/10.1145/2601097.2601186
Wenzel Jakob and Steve Marschner. 2012. Manifold Exploration: A Markov Chain Monte Carlo Technique for Rendering Scenes with Difficult Specular Transport. *ACM Trans. Graph.* 31, 4, Article 58 (2012), 58:1–58:13 pages.
Anton S. Kaplanyan and Carsten Dachsbacher. 2013. Path Space Regularization for Holistic and Robust Light Transport. *Computer Graphics Forum* (2013). https:

//doi.org/10.1111/cgf.12026

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 (July 2014), 13.

R. KRAWCZYK. 1969. Newton-algorithmen zur bestimmung von null- stellen mit fehlerschranken. *Computing* 4, 3 (1969), 187–201.

Alexandr Kuznetsov, Miloš Hašan, Zexiang Xu, Ling-Qi Yan, Bruce Walter, Nima Khademi Kalantari, Steve Marschner, and Ravi Ramamoorthi. 2019. Learning Generative Models for Rendering Specular Microgeometry. *ACM Trans. Graph.* 38, 6, Article 225 (Nov. 2019), 14 pages. https://doi.org/10.1145/3355089.3356525

Don Mitchell and Pat Hanrahan. 1992. Illumination from Curved Reflectors. *SIGGRAPH Comput. Graph.* 26, 2 (1992), 283–291.

Eric Veach. 1997. *Robust Monte Carlo Methods for Light Transport Simulation.* Ph.D. Dissertation. Stanford University.

Edgar Velázquez-Armendáriz, Shuang Zhao, Miloš Hašan, Bruce Walter, and Kavita Bala. 2009. Automatic Bounding of Programmable Shaders for Efficient Global Illumination. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 1–9. https://doi.org/10.1145/1618452.1618488

Bruce Walter, Adam Arbree, Kavita Bala, and Donald P. Greenberg. 2006. Multidimensional Lightcuts. *ACM Trans. Graph.* 25, 3 (July 2006), 1081–1088. https://doi.org/10.1145/1141911.1141997

Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. 2005. Lightcuts: A Scalable Approach to Illumination. *ACM Trans. Graph.* 24, 3 (July 2005), 1098–1107. https://doi.org/10.1145/1073204.1073318

Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. 2007. Microfacet Models for Refraction Through Rough Surfaces *(EGSR 07)*. 195–206.

Bruce Walter, Shuang Zhao, Nicolas Holzschuch, and Kavita Bala. 2009. Single Scattering in Refractive Media with Triangle Mesh Boundaries. *ACM Trans. Graph.* 28, 3, Article 92 (2009), 92:1–92:8 pages.

Ling-Qi Yan, Miloš Hašan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. 2014. Rendering Glints on High-resolution Normal-mapped Specular Surfaces. *ACM Trans. Graph.* 33, 4, Article 116 (2014), 9 pages. https://doi.org/10.1145/2601097.2601155

Ling-Qi Yan, Miloš Hašan, Steve Marschner, and Ravi Ramamoorthi. 2016. Position-normal Distributions for Efficient Rendering of Specular Microstructure. *ACM Trans. Graph.* 35, 4, Article 56 (2016), 9 pages. https://doi.org/10.1145/2897824.2925915

Ling-Qi Yan, Miloš Hašan, Bruce Walter, Steve Marschner, and Ravi Ramamoorthi. 2018. Rendering Specular Microgeometry with Wave Optics. *ACM Trans. Graph.* 37, 4, Article 75 (July 2018), 10 pages. https://doi.org/10.1145/3197517.3201351

Tizian Zeltner, Iliyan Georgiev, and Wenzel Jakob. 2020. Specular Manifold Sampling for Rendering High-Frequency Caustics and Glints. *Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020). https://doi.org/10.1145/3386569.3392408

A APPENDIX

A.1 Impossibility of infinite solution regions

We will sketch the argument for single bounce; for multiple bounces it is analogous. Assume we are looking for a single reflection or refraction at point X on a triangle. The linearly interpolated normal at point X (before normalization) can be written as an affine function $N(X)$. The half vector $H(X)$ at point X can be written as:

$$H(X) = c(X)\left(\eta_1 \frac{X - E}{\|X - E\|} + \eta_2 \frac{X - L}{\|X - L\|}\right), \tag{12}$$

where η_1 and η_2 are the indices of refraction on the camera and light side (they will be equal for reflection), and $c(X)$ is any scalar function. This expresses the fact that we are free to arbitrarily scale the half vector (normal), as its length does not matter. However, this added freedom is not enough; there is no *scalar* function $c(X)$ that would succeed in canceling the non-linear terms in the definition to make the function match the *vector* function $N(X)$ over an infinite region; equality can only be achieved for discrete points X.