Glossy Probe Reprojection for Interactive Global Illumination

SIMON RODRIGUEZ, Université Côte d'Azur and Inria
THOMAS LEIMKÜHLER, Université Côte d'Azur and Inria
SIDDHANT PRAKASH, Université Côte d'Azur and Inria
CHRIS WYMAN, NVIDIA Corporation
PETER SHIRLEY, NVIDIA Corporation
GEORGE DRETTAKIS, Université Côte d'Azur and Inria

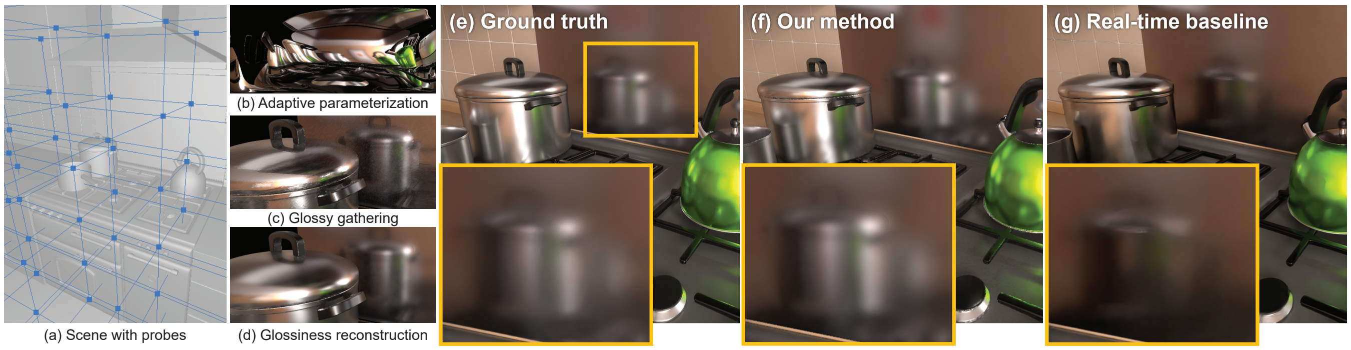

Fig. 1. *(e)* Light bounces from the light to the wall behind, to the kettle, then to the pots, then the glossy panel and finally the camera, involving several glossy surface interactions. Such complex light paths are hard to render interactively, even with modern ray-tracing GPUs. We precompute a set of directional probes in a 3D grid *(a)*, storing such light paths, and interactively render by reprojecting probe information to novel views. Rendering by reprojecting glossy probes raises three challenges: potentially high memory consumption, finding the correct samples for the glossy reflection in the probe and correctly treating reflection occlusion boundaries that have artifacts if reprojected directly, due to parallax changes between the probe and the novel view. We address these three problems by introducing: *(b)* An adaptive parameterization of probes allocating more resolution to glossy materials and complex geometry, by distorting a regular parameterization; *(c)* A gathering algorithm based on path perturbation theory to accurately reproject glossy reflections from probes; *(d)* High quality glossiness reconstruction by splitting the BRDF convolution into precomputation with reduced roughness and filtering at runtime. We achieve interactive global illumination walkthroughs *(f)* for static scenes with opaque objects, with quality close to the path-traced ground truth *(e)* and better than a real-time GPU ray-tracing baseline that queries the light map online *(g)*.

Recent rendering advances dramatically reduce the cost of global illumination. But even with hardware acceleration, complex light paths with multiple glossy interactions are still expensive; our new algorithm stores these paths in precomputed light probes and reprojects them at runtime to provide interactivity. Combined with traditional light maps for diffuse lighting our approach interactively renders all light paths in static scenes with opaque objects. Naively reprojecting probes with glossy lighting is memory-intensive, requires efficient access to the correctly reflected radiance, and exhibits problems at occlusion boundaries in glossy reflections. Our solution addresses all these issues. To minimize memory, we introduce an adaptive light probe parameterization that allocates increased resolution for shinier surfaces and regions of higher geometric complexity. To efficiently sample glossy paths, our novel gathering algorithm reprojects probe texels in a view-dependent manner using efficient reflection estimation and a fast rasterization-based search. Naive probe reprojection often sharpens glossy reflections at occlusion boundaries, due to changes in parallax. To avoid this, we split the convolution induced by the BRDF into two steps: we precompute probes using a lower material roughness and apply an adaptive bilateral filter at runtime to reproduce the original surface roughness. Combining these elements, our algorithm interactively renders complex scenes while fitting in the memory, bandwidth, and computation constraints of current hardware.

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

Additional Key Words and Phrases: Light probes, parameterization, filtering, interactive global illumination

ACM Reference Format:
Simon Rodriguez, Thomas Leimkühler, Siddhant Prakash, Chris Wyman, Peter Shirley, and George Drettakis. 2020. Glossy Probe Reprojection for Interactive Global Illumination. *ACM Trans. Graph.* 39, 6, Article 237 (December 2020), 16 pages. https://doi.org/10.1145/3414685.3417823

Authors' addresses: Simon Rodriguez, Université Côte d'Azur and Inria, simon.rodriguez@inria.fr; Thomas Leimkühler, Université Côte d'Azur and Inria, thomas.leimkuhler@inria.fr; Siddhant Prakash, Université Côte d'Azur and Inria, siddhant.prakash@inria.fr; Chris Wyman, NVIDIA Corporation, cwyman@nvidia.com; Peter Shirley, NVIDIA Corporation, pshirley@nvidia.com; George Drettakis, Université Côte d'Azur and Inria, george.drettakis@inria.fr.

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM Transactions on Graphics*, https://doi.org/10.1145/3414685.3417823.

1 INTRODUCTION

Interactive global illumination has been a major goal of computer graphics since its inception. The introduction of GPU-accelerated ray tracing [Burgess 2020; NVIDIA 2018; Wyman and Marrs 2019] brings closer the prospect of real-time, physically-based global illumination. Current hardware can create a G-buffer [Saito and Takahashi 1991] and trace specular paths very efficiently. However, more complex, typically longer and glossy light paths are still very expensive: see for example the path with multiple glossy bounces in Fig. 1(e). In this paper we introduce a new approach for precomputing and reprojecting such view-dependent glossy paths. When augmented with view-independent paths stored in a traditional light map, our solution enables full global illumination in interactive walkthroughs of static environments containing opaque surfaces (Fig. 1).

Numerous researchers have explored real-time global illumination approximations [Ritschel et al. 2012], often storing direct light or irradiance in light probes [McGuire et al. 2017b] and approximately reconstructing a subset of light paths with various heuristics [Robison and Shirley 2009]. These methods achieve impressive results, but can be less efficient on more expensive, glossy light paths. We take a different approach, precomputing *all* light paths and carefully handling storage and reprojection for each novel view.

Our approach relies on a simplifying assumption. We split light paths, treating view-dependent and view-independent light differently. Separating paths has a long history in graphics [Chen et al. 1991; Slusallek et al. 1998; Wallace et al. 1987], allowing significant acceleration of illumination computations. We focus on view-dependent paths, while for view-independent paths we use traditional light maps.

For such view-dependent paths, i.e., paths from the eye through several glossy bounces, we precompute a new kind of light probe to store them. While precomputing all light paths can enable interactive rendering of realistic lighting, *reprojecting* this data into novel views raises three main challenges. First, the dense angular sampling needed to capture view-dependent effects can impose high memory requirements, since glossiness and complex geometry imply the need for denser sample rates. Second, reprojecting glossy probe samples into a novel view can be challenging and costly. This is because complex reflector and reflected geometry/materials make it hard to find the best samples in the probes for a given novel view. Third, directly reprojecting paths can cause sharpening at glossy reflections of occlusion boundaries, as parallax changes between the probe position and the novel view can sharpen the precomputed blur.

Our work addresses these three challenges. Using Heckbert's [1990] notation, we store $L(S|D)^*DE$ in a light map and we store $L(S|D)^*S^+E$ paths in light probes (here S signifies any non-diffuse reflection). Our approach has three main contributions:

- An adaptive light probe parameterization to increase resolution depending on scene geometry and material properties, reducing the overall memory footprint.
- An algorithm using efficient reflection estimation and on-the-fly search to *gather* view-dependent texels from probes, providing high-quality interactive rendering of glossy paths.
- To avoid sharpening at glossy occlusion boundaries, we introduce a new approach that splits the convolution effect of the BRDF into two steps. First, we render the probes using materials with lower roughness in precomputation, and second, during rendering we apply efficient, adaptive-footprint bilateral filtering reproducing the original material roughness.

Our algorithm plausibly approximates ground truth illumination, with complex light paths, at interactive rates for scenes with opaque objects (Fig. 16-18, supplemental videos). We compare to modern hardware-accelerated ray tracing baselines: a lightmap with real-time glossy ray casting and real-time path tracing. We also compare with light-probe based illumination [McGuire et al. 2017b] and image-space gathering [Robison and Shirley 2009]. Overall, we show better quality compared to ground truth, when other methods run at the same framerate as ours.

2 RELATED WORK

Since our algorithm precomputes and reprojects light paths from a new kind of glossy light probe, in this section we review prior work on sample reuse, real-time reflections, and probe-based rendering. Light maps are used widely in games, and recent research efforts have explored efficient light map approximations [Luksch et al. 2013, 2019]. Our work is largely orthogonal, and we use diffuse light maps computed offline with a modified path tracer.

2.1 Reusing Samples for Rendering

Several methods have been developed to precompute and reuse samples. Shaded sample generation and rendering final images can be decoupled by caching of radiance samples [Dayal et al. 2005; Walter et al. 1999; Ward and Simmons 1999]. Samples are warped and accumulated into a novel view using depth and surface information to account for occlusions. A feedback loop guides new sample generation, which is typically provided by ray or path tracing. More recently, Nehab et al. [2007] reproject previous frame information to avoid expensive shading computations in large areas of the novel image. Other forms of sample reuse exist, e.g., for temporal supersampling and filtering [Karis 2014; Schied et al. 2017]. While our work relies on sample reuse, our preprocessing is more exhaustive than these solutions.

Object space shading can be used for remote rendering [Hladky et al. 2019; Mueller et al. 2018]. Scene triangles are packed in an atlas, while shading is updated on the server for visible triangles, and streamed to a local device using video compression. Because updates are progressive and no view extrapolation is performed on device, view-dependent effects are hard to render accurately in motion with these techniques.

Specialized methods to render reflections and specular effects often use image-based rendering (IBR) [Shum and Kang 2000]. Lischinski et al. [1998] generate axis-aligned layered depth images containing diffuse BRDF parameters as well as layered light fields for specular effects. They splat diffuse color and gather from the light fields using the stored parameters. Lehtinen et al. [2012] store glossy and diffuse secondary rays from a conventional path tracer in a 4D light field representation for a single viewpoint. The light field allows reconstruction of diffuse and glossy reflections by using neighbors in the 4D space weighted by the degree of glossiness. This provides high quality reconstruction of indirect lighting from a small number of samples for a single viewpoint. Lochmann et al. [2014] handle more complex view-dependent effects by computing

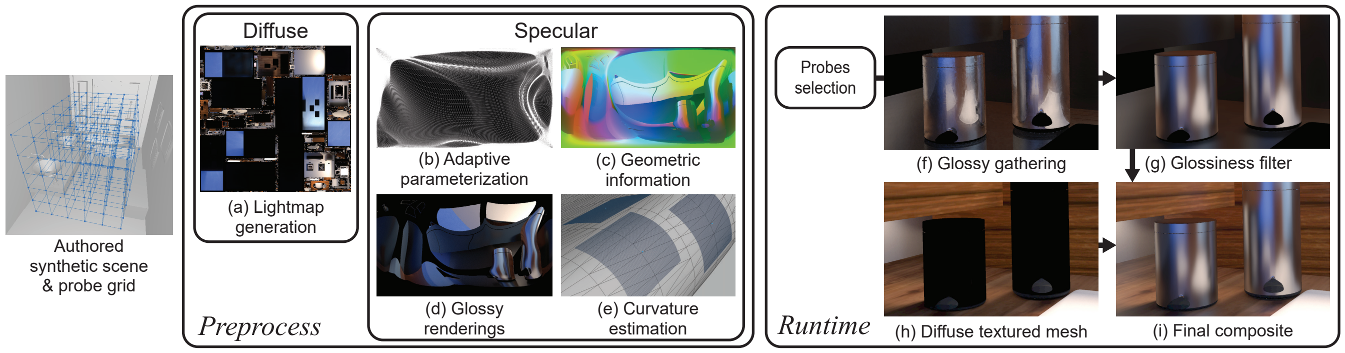

Fig. 2. For a given synthetic scene, both diffuse *(a)* and specular illumination are computed. A series of environment probes are placed on a regular grid and parametrized *(b)* to capture view-dependent effects. In preprocess, high quality renderings are generated *(d)* along with additional geometric information for each probe *(c, e)*. At render time, the closest probes are selected and glossy information is gathered at the novel viewpoint *(f)* ; glossy effects are then reconstructed *(g)*. The diffuse scene is rendered separately *(h)* and both layers are composited to generate the final image *(i)*.

warps from prior frames for diffuse, reflective and refractive surfaces. A small disparity between the previous and current view is required. In contrast, our work uses light probes for glossy paths, requiring a more involved approach to reproject these paths to the novel view. Nonetheless, IBR techniques inspired our combination of precomputed data feeding interactive rendering.

Image space gathering [Robison and Shirley 2009] stores perfect specular reflections in a buffer and applies a post-process image-space filter to create "phenomenologically compelling" approximations of blurry reflections and soft shadows. Our two-step convolution to overcome the sharpening at reflected occlusions draws inspiration from this work, but by precomputing all paths – not just perfect reflections – we simulate all lighting at the same cost.

Filters based on the BRDF footprint have also been used to tackle specular aliasing [Kaplanyan et al. 2016; Tokuyoshi and Kaplanyan 2019]. The material normal distribution function is filtered in the surface local frame, on a parallel plane or unit disk. The exact filter is derived from the pixel footprint considered and the variations of the specular path half vector around the surface point. We rely on similar assumptions to reconstruct the final glossy materials but use a larger-scale footprint and a simplified surface representation.

2.2 Real-time Reflection Rendering

Real-time rendering of reflections on graphics hardware has been extensively explored in the last twenty five years; Szirmay-Kalos et al. [2009] provide a good survey.

Ofek et al. [1998] warp the geometry of the reflected scene to virtual reflected positions. This requires highly tessellated geometry for non-planar reflections, and each reflector has to be locally approximated by planes. Similarly, Estalella et al. [2005] store virtual reflected objects in cube maps to accelerate reflection computation. This method builds on the property that the reflected light path half-vector coincides with the surface normal; we also use this property when reprojecting from the glossy light probe.

Other methods rely on the perturbation of existing specular paths. Chen et al. [2000a; 2000b] precompute reflection paths on a regular image grid and express the reflection displacement at other pixels as a second-order approximation. They rely on an analytical implicit representation of the reflectors. We build on and generalize this formulation as part of reprojecting probe texels to novel views. Manifold exploration [Jakob 2013] has been used to upsample and interpolate noisy path traced images while preserving specular effects [Zimmer et al. 2015]. While it allows for perturbation and update of complex specular paths under a small baseline, to our knowledge it has not been applied to real-time rendering problems.

One-bounce glossy interreflections under distant illumination can be computed analytically when projecting to a suitable representation, e.g., spherical Gaussians [Xu et al. 2014]. In contrast, our approach handles arbitrary lighting and materials, while imposing no restrictions on path length.

Recent ray-tracing GPUs allow fast path-tracing [Wyman and Marrs 2019], especially when coupled with denoising (e.g., [Schied et al. 2017]). Multiple-bounce glossy paths however require a high sample count to evaluate the BRDF and each path vertex, and thus result in degraded performance, as seen in our comparisons (Sec. 7.3.1). In addition, poor reconstruction of dynamic glossy and specular reflections from low-sample renderings is a known limitation of existing denoisers.

Neural rendering [Tewari et al. 2020] has been used to predict view-dependent shading effects in novel views. Implicit neural representations [Mildenhall et al. 2020] bake the entire plenoptic function of a small scene into a neural network. Closest to our approach in this space is the work of Ren et al. [2013], which allows dynamic lighting of glossy surfaces in static scenes. They evaluate a multi-layer perceptron per point light, per pixel, limiting the applicability for scenes with more complex lighting such as area lights. In contrast, the runtime performance of our approach is independent of the lighting complexity. A property we share with learning-based methods is a scene-specific pre-processing stage.

2.3 Rendering with Light Probes

Precomputed environment maps are often used in production to generate approximate real-time reflections. A reflected ray can look in these maps to estimate the incident radiance. Blending between multiple probes allows variations in the environment to be captured.

These ideas originate with the *irradiance volume* [Greger et al. 1998], which precomputes probes to light diffuse dynamic objects. More recently, Silvennoinen et al. [2017] use a set of radiance probes combined with a local reconstruction step to estimate indirect radiance on diffuse surfaces in dynamic scenes. For mirror reflections, Szirmay-Kalos et al. [2005] store an environment depth map and iteratively update the sampled location by assuming locally planar scene geometry. Yu et al. [2005] store a light field for all six cubemap faces, increasing memory use. The light fields are queried at runtime to obtain the closest sample to the reflected ray. Lagarde et al. [2012] preserve parallax effects in their probes by intersecting rays with a simplified analytic scene representation (a box or cylinder). Hakura et al. [2001; 2001] only cast rays between reflector vertices before fetching radiance from a layered environment map. Each layer is optimized to accurately represent certain reference viewpoints. The above solutions work only for specular reflections, in contrast to our approach that handles general material properties.

Light probe memory consumption can be significant. We minimize this with an adaptive parameterization [Sander et al. 2002; Sloan et al. 1998; Zayer et al. 2005], which relates to continuous magnification techniques to obtain spatially-varying resolution [Friston et al. 2019]. Unlike our solution, Friston et al. magnify according to a simple foveation pattern, independent of the scene content.

Hirvonen et al. [2019] precompute environment maps that can be lit at runtime like a G-buffer. When shading, secondary rays are cast according to the BRDF. The radiance at secondary intersections is estimated from the lit probes or the previous frame. McGuire et al. [2017b] compute radiance, surface normals and depth for a set of local probes. At runtime specular rays are marched through the probes to find intersections. Probe resolutions and reflection accuracy are limited due to ray marching cost. Wang et al. [2019] improve performance using hierarchical depth information and non-uniform probe placement. Majercik et al. [2019] rely on similar probes at a low resolution to estimate irradiance, but update them at runtime. They fall back to ray-tracing for all reflections. Handling glossy reflection paths in these method requires significantly increasing sample count, degrading performance (see comparisons in Sec. 7.3.2). In contrast, thanks to our precomputed probes containing all light paths, glossy reflections are handled naturally by our solution.

3 OVERVIEW

We introduce a novel approach to interactive rendering of global illumination in static synthetic scenes containing opaque objects, using lightmaps for diffuse and probes for glossy paths. We address three challenges of probe-based glossy rendering: first, reducing memory footprint of the probes; second, efficiently and accurately reprojecting glossy path information to the novel view and third, avoiding sharpening at glossy reflection occlusion boundaries. Our method is outlined in Fig. 2.

In preprocess (middle box in Fig. 2), we use a path-tracer to compute diffuse global illumination stored in a lightmap (a), while the glossy component of radiance – i.e., $L(S|D)^*S^+E$ paths – is precomputed (d) and stored in light probes placed in the scene, far left in figure.

For the first challenge, we maximize the amount of information stored where glossy surfaces are visible by computing a parameterization for each probe with more resolution assigned to shinier surfaces and objects with higher geometric complexity (Fig. 2b). We generate this parameterization using quasi-harmonic maps [Zayer et al. 2005]. We also precompute scene geometric curvature information which is used at runtime for gathering (Fig. 2e). A visible geometry map is also generated along with a map containing the reflected positions visible in each direction of a probe (Fig. 2c).

Rendering is performed at runtime (right-hand box in Fig. 2). We first render the diffuse component as a surface textured by the lightmap.

For the second challenge, we efficiently render accurate view-dependent paths by introducing a hierarchical gathering approach. We first perform trilinear interpolation between probes. We compute the perfect mirror reflected position visible at each point of the novel view, and reproject it into each selected probe while taking specular motion into account (Fig. 2f). We base our approach on specular path perturbation [Chen and Arvo 2000b], but generalize it to arbitrary geometry using curvature approximation. This estimate provides an initialization for a search process performed in probe space to gather the probe texels best corresponding to the – possibly glossy – reflection. We accumulate these points from the probes and blend them according to the material properties at the reflector surface. The gather process is critical to the success of our approach, since it renders our method robust to inaccuracies in the reprojection process and finds the best available data in the probe.

The third challenge occurs because naively reprojecting glossy reflections from the probes can create a sharpening effect at occlusion boundaries in the reflections. To overcome this issue, we introduce a new approach that separates the convolution effect of BRDFs into two steps (Fig. 2g). We first reduce the roughness of materials in the probe precomputation, and apply bilateral filtering in screen space during rendering. Importantly, we estimate a footprint for the screen space filter that closely reproduces the overall glossiness of the original materials.

The entire process is interactive, and reproduces all the light paths in our static scenes made of opaque objects.

4 PROBE GENERATION AND STORAGE

We store glossy paths in *light probes*, and reproject them to novel views when rendering. We first describe the per-probe data and how we compute it. We then present our adaptive parameterization that concentrates probe texels in important regions, reducing memory usage at a given image quality. We also describe additional geometric information needed as part of real-time rendering.

We place probes on a regular 3D grid in the scene [McGuire et al. 2017b]. After exploring various adaptive probe placements [Wang et al. 2019], we found regular sampling gives the highest quality at a given probe budget. This is because our reflection estimation (Sec. 5.1) works best with small distances between probes and the novel view. Regular sampling provides a conservative upper bound

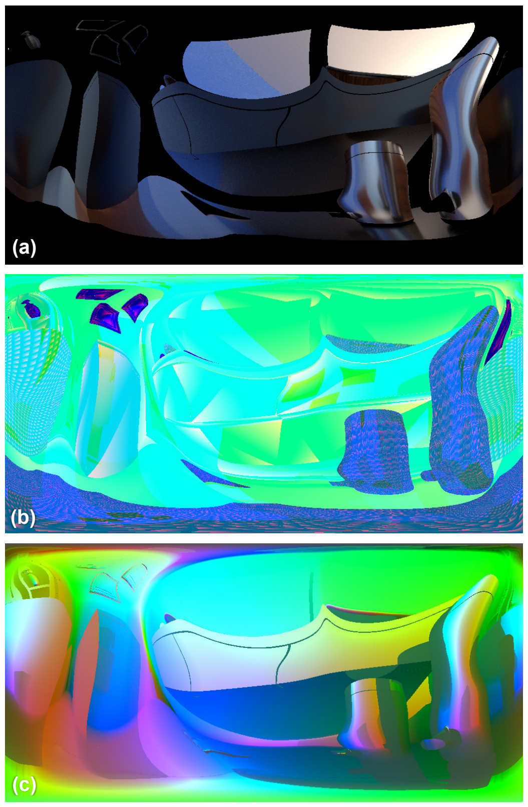

Fig. 3. Per-probe data: *(a)* glossy color, *(b)* reflector geometry information (triangle IDs and barycentric coordinates), *(c)* reflected positions (and reflected material ID in the alpha channel)

of the maximum distance, while adaptive placement naturally samples some areas sparsely.

4.1 Per-probe Data

For each probe, we render a map storing surfaces visible from the probe (Fig. 3b); this map stores triangle ID and barycentric coordinates used to reconstruct surface attributes at runtime (e.g., position, normals or curvature). We then render an environment map containing a 360° view from the probe location, storing only the glossy color at visible surfaces (Fig. 3a). We precompute this map with a modified *Mitsuba* path tracer [Jakob 2010] where we force all first-bounce rays to sample the glossy BRDF lobe and take our adaptive parameterization into account. In practice any renderer can be used, depending on desired probe quality. In a third map, we store the geometry seen with one-bounce reflection if every surface acted as a perfect mirror (Fig. 3c).

4.2 Probe Parameterization

At a high level, we aim to allocate probe resolution preferentially to regions where it is most needed. Three requirements drive our texel allocation. (*a*) Smoother surfaces require higher resolution for reprojected reflections, while we can reconstruct rough materials from fewer samples due to their lower frequencies. (*b*) Distant surfaces or those seen at grazing angles have lower effective resolution. Probe queries may occur from novel views with different perspectives requiring higher resolution. (*c*) High frequency geometric content exhibits lots of variation in reflected radiance, which needs higher resolution for reliable capture.

We start from a latitude-longitude (lat-long) parameterization (Fig. 4a) and modify it driven by the requirements above. We could start from other panoramic parameterizations, with few differences, but we chose lat-long as it allows easy lookups from ray directions and the map is a single image (i.e., not a cubemap). We proceed in two steps: first we compute an *adaptive resolution map m* (Fig. 4b) to encode relative resolution needs of various probe directions, then we use the map to compute an adaptive parameterization (Fig. 4h).

4.2.1 Adaptive Resolution Map Computation. To construct the map, we render four low resolution buffers, via ray casting, containing: (*i*) material smoothness m_{mat}, (*ii*) depth m_{depth}, (*iii*) normal m_{norm}, and (*iv*) facing angle m_{face}, the dot product of incident ray and surface normal. We render these buffers at 256×128 pixels.

The adaptive resolution map m in Fig. 4b stores:

$$m = m_{\text{mat}} \left(m_{\text{size}} + m_{\text{complexity}} \right).$$

Here, m_{mat} adapts resolution based on material smoothness (Fig. 4c), satisfying requirement (*a*), above. Diffuse texels have $m_{\text{mat}} = 0$ so no space is allocated for them, increasing resolution for shiny surfaces. m_{size} (Fig. 4d) considers distance and orientation (*b*) and is computed as:

$$m_{\text{size}} = \frac{m_{\text{depth}}^2}{m_{\text{face}}} \cos\left(\theta_{\text{long}}\right).$$

This term converts probe area to actual object size, compensating for perspective and angular foreshortening akin to a form factor. θ_{long} is the longitude angle, which compensates for size variations induced by the lat-long base parameterization.

Finally, requirement (*c*) calls for an estimate of local geometric complexity. Simple gradient-based estimates on our buffers are not meaningful, as their response increases around any discontinuity, including (curved) edges, which we do not consider "complex" for requirement (*c*). Therefore, we perform a simple frequency analysis (Fig. 4e) as follows:

$$m_{\text{complexity}} = \frac{1}{N^2} \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} w_{p,q} \|\mathbf{b}_{p,q} * m_{\text{norm}}\|_1.$$

Here $\mathbf{b}$ are basis functions of the 2D discrete cosine transform (DCT) [Ahmed et al. 1974], with $[p, q]$ integer 2D frequency vectors (Fig. 4f). See Appendix A for more on the choice of DCT. Convolving the basis functions with the normal buffer analyzes frequency content of local neighborhoods, and the 1-norm sums absolute responses of the three normal map dimensions. We weight responses by $w_{p,q} = \|[p, q]\|^k$ to ensure higher frequencies contribute heavily to our geometric complexity measure (Fig. 4g). In practice, we set $N = 16$ and $k = 5$. We normalize both m_{size} and $m_{\text{complexity}}$ by a per-probe mean to ensure equal effective contribution. Subsequent steps behave more robustly after blurring m with a small Gaussian (i.e., $\sigma = 5$).

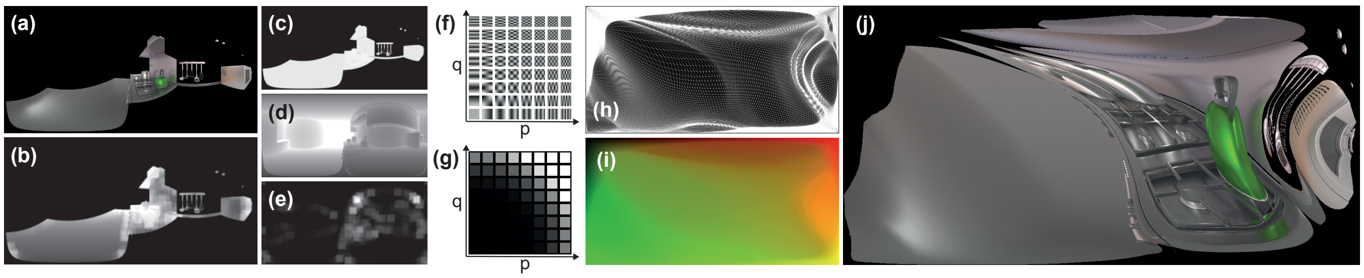

Fig. 4. Overview of our probe parameterization pipeline. Conventional content-agnostic spherical parameterizations (*a*) of a glossy rendering allocate resolution budget suboptimally. Map (*a*) is never rendered in our approach and shown here only for illustrative purposes. Instead, we design an adaptive resolution map (*b*), combining local information on surface parameters (*c*), foreshortening (*d*), and geometric complexity (*e*). The latter is estimated by convolving a G-buffer with DCT basis functions (*f*) – here shown for $N = 8$ – which are then aggregated using a weighting scheme (*g*) to detect high-frequency variation. We use quasi-harmonic maps to convert the adaptive resolution map into a corresponding adaptive parameterization (*h*). This induces an inverse flow field (*i*) – here 2D lookup coordinates are visualized using red and green channels – which we use to steer rays, obtaining a probe with spatially varying resolution (*j*).

4.2.2 Adaptive Parameterization. To turn our adaptive resolution map m into a parameterization with adapted resolution, we use tensorial quasi-harmonic maps [Zayer et al. 2005].

A quasi-harmonic map takes f_h from the plane to the plane, following the quasi-harmonic equation $\operatorname{div}(C\nabla f_h) = 0$. The 2×2 matrix C is a requested first fundamental form, which can vary spatially. In our case $C = m\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. This essentially imposes scaling proportional to m.

We use a regular quad mesh to discretize the domain, where each pixel of m corresponds to a quad. We restrict motion of boundary vertices to the domain boundary. Since the lat-long base parameterization naturally wraps, we force corresponding vertices at the vertical boundaries to move in sync, ensuring parameterization smoothness. Following Zayer et al. [2005] we solve the resulting quasi-harmonic equation iteratively using a sparse matrix solver.

This quasi-harmonic map induces a forward flow, telling us how to move our quad mesh vertices to obtain the desired parameterization (Fig. 4h). To determine where to *look up* directions in parameterized probes, we need to invert the mapping, essentially creating an inverse flow field f_h^{-1} (Fig. 4i). We do this by rasterizing the deformed mesh with the original vertex positions as colors. This happens at full probe resolution. To render our adaptively parameterized light probes, each pixel looks up its lat-long position using the inverse flow field and we trace a ray through the scene in the corresponding direction. This distorts the probe according to our magnification rules (Fig. 4j).

When rendering our glossy light probe, this inverse map specifies each texel's correct view vector to initiate path tracing. At the first intersection, we only trace paths corresponding to the glossy BRDF lobe, exploiting standard material models' separation of diffuse and specular components. Both the forward and inverse maps are also used at run time (see Sec. 5) to render the view-dependent layer.

4.3 Geometric Information

When gathering glossy light probe samples at runtime (Sec. 5), we build on Chen and Arvo's [2000a] specular path perturbation. This requires implicit representations of all reflector geometry and various derivatives. To avoid limiting scenes to implicit surfaces, we need additional geometric data to support our approximation described in Sec. 5.1.

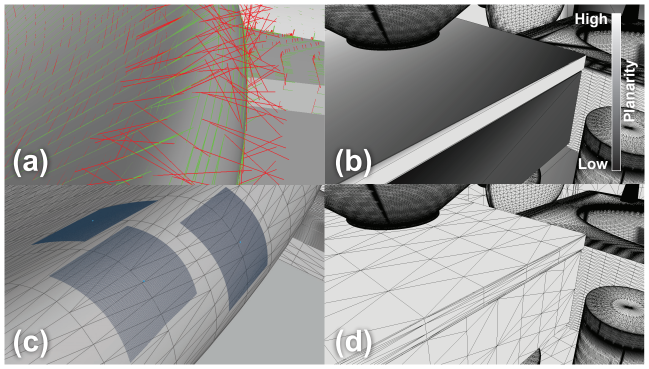

Fig. 5. *(a)* The triangle-based mesh and the estimate curvature vectors: minimal (green) and maximal (red) directions. *(b, d)* Maximal planarity estimated using the initial *(b)* and subdivided *(d)* objects: the estimation improves. *(c)* Visualization of paraboloids fitted using our estimated parameters.

For each scene vertex we first estimate both principal curvature values and directions (Fig. 5a) using standard techniques [Meyer et al. 2003]. We do this separately on each object. For best results, we tessellate large planar objects with only a few large triangles (Fig. 5b,d). To get more robust estimation, we regularize curvatures between neighboring vertices when normals deviate less than 40 degrees. We average curvatures weighting by the normal dot product and ignoring values from boundary vertices. This reduces issues at mesh boundaries (which do not have a full cycle to estimate curvature) and filters smaller, irrelevant cuvature variations.

We transfer these values back from the tessellated geometry to each scene vertex, allowing runtime interpolation of the curvature. The principal curvatures allow us to locally represent the surface by a paraboloid using the curvature directions as axes (Fig. 5c), for which we can analytically compute our required higher-order derivatives.

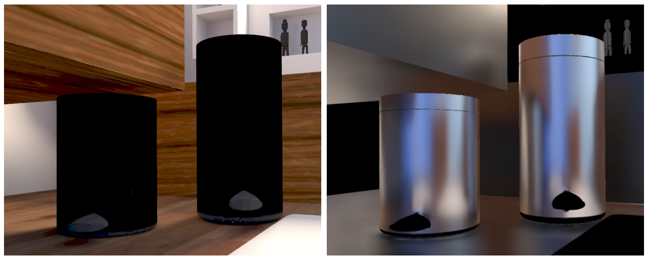

Fig. 6. Rendering of diffuse (left) and view-dependent (right) components occurs separately. Both rely on precomputed illumination.

5 RENDERING GLOBAL ILLUMINATION

To render a novel view, we handle diffuse and glossy components separately, sampling two forms of precomputed lighting (see Fig. 6). For diffuse light, we render meshes textured with a standard light map, which contains diffuse global illumination.

For glossy light paths, we reproject our light probes to the current view. To do reprojection, we first rasterize a G-buffer with position, normal, surface curvature, and material (ID and roughness). With a ray caster, we then trace perfect mirror rays at specular pixels, storing reflected hit positions and material IDs. This reflection ray is real-time on modern GPUs. We then compute per-pixel glossy lighting by gathering information from nearby probes. This gather relies on estimated specular flow, but as we gather from *glossy* probes it closely approximates a wide range of material properties. An exception are reflected occlusion boundaries, treated in Sec. 6.

For each output pixel containing a glossy surface, we use G-buffer and ray data above plus the probe data (see Fig. 3) to select relevant probe texels and merge their contributions. Let p be the current camera position. A pixel sees point x, sampled by our G-buffer, and the mirror ray from x intersects reflected point q (see Fig. 7a). To shade x using data from probe P', we need to fetch the probe sample for x' that reflects point q as seen from probe origin p' (assuming such data exists in P').

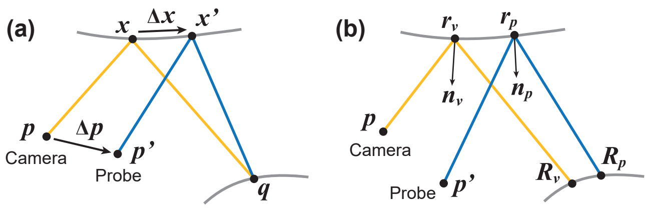

Fig. 7. The geometry of path perturbations. *(a)* From a known specular path (q, x, p) and a probe position p', a new path (q, x', p') can be determined through local path perturbation. *(b)* To assess probe samples, we compute a score based on ray information (reflector and reflected positions, reflector normal) associated with the sample (r_p, n_p and R_p) and the novel view pixel (r_v, n_v and R_v).

5.1 On-the-fly Reflection Position Estimation

First, we must determine x' as the camera moves from p to p'. The theory of specular path perturbation [Chen and Arvo 2000b] allows us to approximate displacement $\Delta x = x' - x$ given displacement $\Delta p = p' - p$ (see Fig. 7a). For any reflector represented by implicit function f, we can derive a second order approximation of the path function from Fermat's principle and the implicit function theorem. Then, there exists a Jacobian $J(p, q, x, f)$, a 3x3 matrix, and Hessian $H(p, q, x, f)$, a 3x3x3 tensor, such that

$$\Delta x = J\Delta p + [\Delta p]^T H[\Delta p],$$

where $[\Delta p]$ is a 1x1x3 tensor replicating Δp three times. We refer the reader to Chen et al. [2000a] for a detailed discussion. As we know p, p', x, and q, this gives sufficient data to lookup probe samples.

While Chen et al. [2000a] require first, second and third order derivatives of f, we do not want to limit scenes to implicit surfaces. We generalize to triangular meshes using our local curvature approximation stored during precomputation (see Sec. 4.3) and G-buffer rendering. From these curvatures we estimate a paraboloid to locally fit the surface, and use its analytical derivatives for path perturbation. We only keep points x' that share the same material as x, and reproject their probe samples.

5.2 Gathering View-dependent Color

Combining specular path perturbation with our estimated curvature only approximates the specular motion between the novel view and our probe. To be robust to inaccuracies, we explore a neighborhood in probe space before finalizing our sample selection.

We use a two-level search on a grid of decreasing step size and radius, looking for a texel with stored radiance valid at our novel location. The two-level search ensures thoroughness while maintaining efficiency. Samples corresponding to reflections on other materials are ignored.

We seek to favor probe samples containing information closely corresponding to the novel view surface. We achieve this by evaluating an energy function designed to favor such samples. We use the following notation (see Fig. 7b): reflector position r_p, normal n_p, and reflected position R_p as seen in the light probe, and corresponding values r_v, n_v, and R_v from the novel view.

Our energy function considers four criteria. First, view and probe samples (R_v and R_p) preferentially lie on the same surface; if their material IDs differ, we strongly penalize the total energy, multiplying it by $s_a = 10$. We thus use mismatched samples only if no others are available. Second, the reflected hits R_v and R_p should be close; for this we add a term $s_b = \|R_v - R_p\|$. Third, using similar surface normals n_v and n_p ensures consistent lighting; the term $s_c = 1 - (n_v \cdot n_p)$ achieves this goal. Finally, the sample should have a similar reflected ray; we use the following term:

$$s_d = 1 - \frac{(R_v - r_v) \cdot (R_p - r_p)}{\|R_v - r_v\| \|R_p - r_p\|}.$$

Combining these criteria, we use the following energy function:

$$\mathcal{E} = s_a \cdot (\min(s_b, 1) + \min(s_c, 1) + \min(s_d, 1)). \quad (1)$$

s_a and s_b have the most effect, but the other terms help fix issues in more uncommon cases, such as non convex reflectors.

Because we adaptively parameterize probes, searching probe neighborhoods with a constant-sized regular grid risks missing details in compressed regions or insufficiently exploring magnified

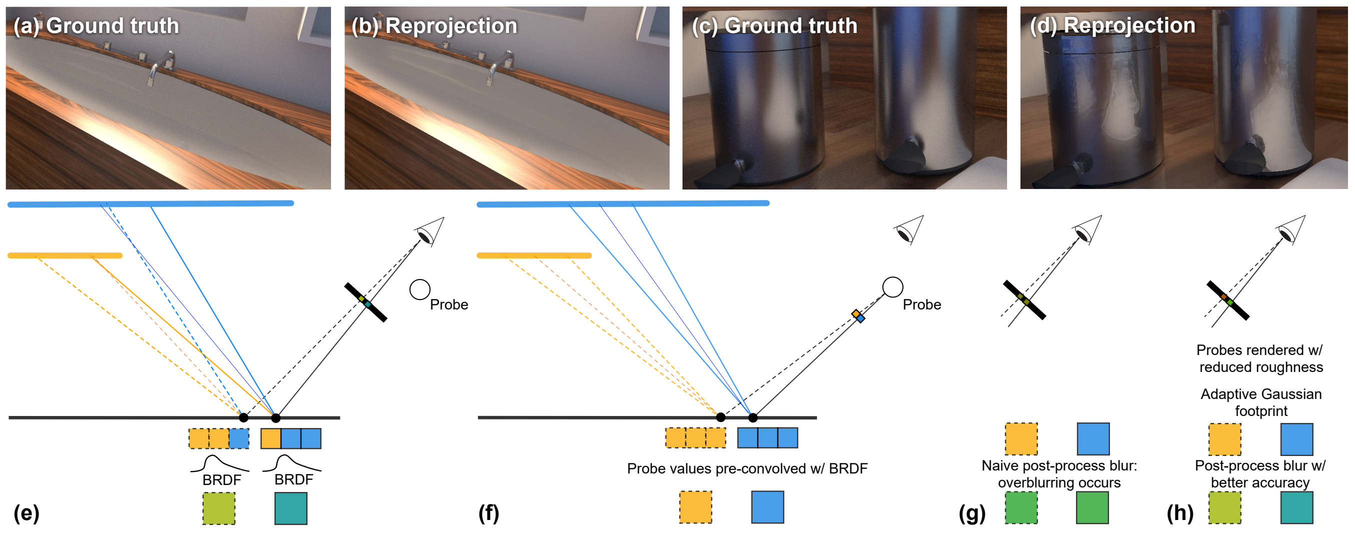
(a) Ground truth
(b) Reprojection
(c) Ground truth
(d) Reprojection
Probe
Probe
BRDF
BRDF
(e)
(f)
Probe values pre-convolved w/ BRDF
Naive post-process blur: overblurring occurs
(g)
Probes rendered w/ reduced roughness
Adaptive Gaussian footprint
Post-process blur w/ better accuracy
(h)

Fig. 8. *(a)* Path traced ground truth of unoccluded glossy reflection where *(b)* naive reprojection is accurate. *(c)* Reference reflection with occlusion boundaries; *(d)* naive reprojection sharpens these boundaries. *(e)* Ground truth: pixel colors integrate the BRDF over samples from both yellow and blue surfaces. *(f)* Naively sampling precomputed probes uses only samples from either side of the occlusion, sharpening the boundary. *(g)* Naive post-process filtering over blurs results. *(h)* By reducing roughness and estimating the footprint of the post-process filter, we improve accuracy.

areas. We thus scale the search step size by $\left|\frac{\partial f_h^{-1}}{\partial \mathbf{x}}\right|^{-1}$, where f_h^{-1} is the inverse flow field (Sec. 4.2.2). In our two-level neighbor search, our coarse level uses 7x7 samples at 4 texel spacing (before compensation). The fine level searches 3x3 samples with 2 texel spacing, centered at the minimal energy sample found by the coarse search.

To avoid popping during camera motion, we sample reflections in the eight probes nearest the novel view. Each probe selects its sample with minimal energy (Eq. 1). and the eight probe samples are combined with trilinear weights t_i into a final pixel color:

$$C = \frac{1}{Z}\sum_{i=1}^{8} t_i \cdot \exp(-\phi\mathcal{E}_i) \cdot \mathbf{c}_i, \tag{2}$$

where ϕ is a constant falloff factor we set to 8, and Z is a normalizing constant ensuring that weights sum to unity. When loading colors $\mathbf{c}_i$ from the probes, we use bilinear interpolation when this does not blend colors from different reflectors.

For pixels with no valid sample in any of the eight probes, we temporally reproject information from last frame's glossy layer. The lowest error $\mathcal{E}_i$ among the eight probes is also stored for our glossy filter pass (see Sec. 6.2).

6 TWO-STEP CONVOLUTION FOR ACCURATE WARPING OF GLOSSY PROBES

For glossy materials, we can reuse samples representing any ray in the BRDF lobe, even if they are not perfect specular reflections; this is similar to Robison and Shirley [2009]. Generally, gathered probe samples are nearly correct for the novel view, giving satisfactory results (see Fig. 8a,b)

However, reflected geometric occlusions often appear "sharpened" when gathered samples straddle these boundaries. To understand why, consider Fig. 8. Fig. 8e shows that correct pixel values integrate the BRDF lobe, combining samples from both yellow and blue surfaces. Naive probe reprojection gathers precomputed samples falling entirely on either side of the occlusion, sharpening the blurry boundary (Fig. 8d). Simply applying a post-process filter overblurs (Fig. 8g), increasing apparent surface roughness. To reduce overblurring, our solution separates the convolution effect of the BRDF into two steps. We reduce the material roughnesses when precomputing probes and then adaptively blur during lookups to approximate the desired glossiness (Fig. 8h). We first explain how to estimate the filter footprint so that we match the original material roughness in the final image (Sec. 6.1). We then explain how to perform the filter operation (Sec. 6.2) and finally our efficient filter approximation (Sec. 6.3).

6.1 Filter Footprint Estimation

To estimate our filter footprint, we use a simple configuration (see Fig. 9a): a camera looking at a plane of uniform isotropic roughness, with normal parallel to the view and reflected point colinear to the camera. We evaluate the GGX BRDF [Walter et al. 2007] for each point under a fixed field-of-view, shown in Fig. 9b. The parameters influencing the BRDF footprint are the material roughness ρ, distance to the reflector d_c, and distance to the reflected light d_r. Given this, we fit a Gaussian $\mathcal{G}_\rho(\mathbf{x}; \rho, d_c, d_r)$ with covariance matrix Σ_ρ to the image-space BRDF footprint. We can ignore the mean as the Gaussian is centered at zero. We determine Σ_ρ for specific parameters $[\rho, d_c, d_r]$ by sampling the BRDF footprint and evaluating spatial covariance numerically. Such a fit is shown in Fig. 9c along one scanline; we see this approximates the BRDF quite accurately.

We precompute and tabulate covariances in $32 \times 32 \times 32$ bins with roughness varying from 0 to 0.5 and distances from 0.01 to

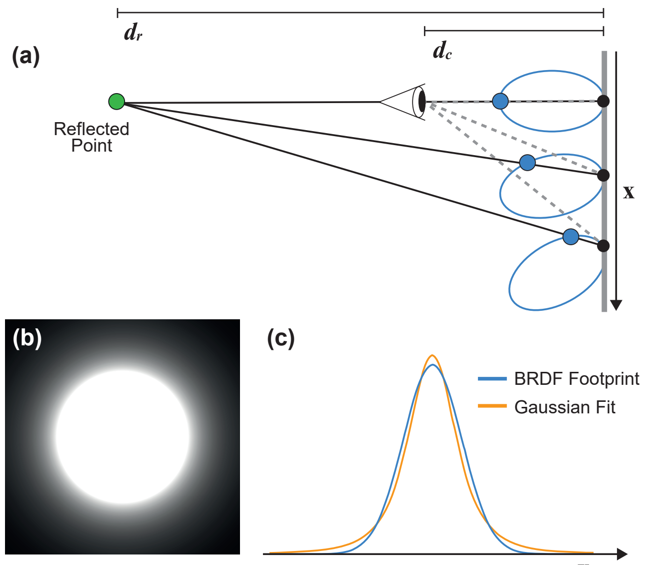

Fig. 9. Filter footprint estimation. *(a)* The setup for estimating our filter footprint. A camera observes a reflected point (green) via a planar reflector. We evaluate the BRDF lobe (blue ellipses) at various positions $\mathbf{x}$ and corresponding viewing angles (blue points) to obtain an image-space footprint. *(b)* The footprint of the GGX specular lobe, for $\rho = 0.075$, $d_c = 2m$, $d_r = 5m$. *(c)* A 1D slice of the specular lobe and our fit giving a close approximation.

10m, corresponding to typical values in our scenes. We use a power sampling scheme and decrease the covariance for small d_r (<0.5m) as we observed fitting overestimated covariance in these cases.

Slanted reflectors foreshorten the BRDF footprint along the slant direction. We dynamically incorporate this when rendering: the projected surface normal in image space gives the foreshortening direction. We then update Σ_ρ by scaling the variance in this direction by the dot product between view direction and surface normal.

Using Gaussian filtering allows splitting our glossy filter into two steps, relying on properties of Gaussians. The steps include $\mathcal{G}_\rho$, the reflector BRDFs when precomputing glossy light probes, and $\mathcal{G}_I$, the runtime image filter. Final pixel values are given by the convolution $\mathcal{G}_\rho * \mathcal{G}_I$. We can reduce material roughness during precomputation, giving a new Gaussian $\mathcal{G}_{\rho'}$ with covariance matrix $\Sigma_{\rho'}$ corresponding to the new roughness ρ'. Using the property:

$$\Sigma(\mathcal{G}_1 * \mathcal{G}_2) = \Sigma(\mathcal{G}_1) + \Sigma(\mathcal{G}_2),$$

we find the covariance matrix Σ_I of the image Gaussian $\mathcal{G}_I$ such that the operation $\mathcal{G}_{\rho'} * \mathcal{G}_I$ reproduces the effect of $\mathcal{G}_\rho$:

$$\Sigma_I = \Sigma_\rho - \Sigma_{\rho'}. \tag{3}$$

In practice, we set ρ' to $\rho/2$, and modify our preprocess probe rendering to account for this at the first glossy vertex of each path.

During interactive rendering, for each pixel, we look up the values of $\mathcal{G}_{\rho'}$ for a given initial roughness ρ and distances d_c and d_r. We compute Σ_I using Eq. 3, compensate for geometric foreshortening and apply the image filter as detailed in the next section.

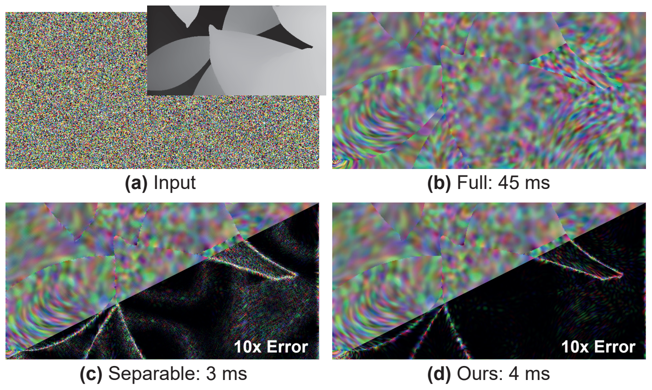

Fig. 10. Comparing filter alternatives. *(a)* Filter input is high-amplitude uniform random noise, which is adversarial for anisotropic edge-stopping filters. The guide image contains hard edges, with smoothly varying anisotropic covariance in each region. *(b)* Applying a full 2D filter, per Eq. 4, is highly inefficient for large kernels. *(c)* A naive separable implementation is fast, but suffers from strong artifacts. *(d)* Our four-pass implementation is almost as efficient as the separable version, but reduces artifacts significantly.

6.2 Gloss Filtering

Above, we estimated the image-space filter footprint. Now, we filter guided by the geometric data used to estimate colors C in Eq. 2:

$$\hat{C}(\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \sum_{\mathbf{x_i} \in \mathcal{N}(\mathbf{x})} \mathcal{G}_I(\mathbf{x}_i - \mathbf{x}) w_r(\mathbf{x}, \mathbf{x}_i) \mathcal{E}^{-1}(\mathbf{x}_i) C(\mathbf{x}_i). \tag{4}$$

Here, $\mathcal{N}(\mathbf{x})$ denotes the filter footprint at $\mathbf{x}$. $\mathcal{E}$ is the energy function in Eq. 1; the inverse acts as a confidence to limit propagation to pixels that match well [Knutsson and Westin 1993]. The range weight

$$w_r(\mathbf{x}, \mathbf{x}_i) = \mathbb{1}_{\mathbf{n}(\mathbf{x})\cdot\mathbf{n}(\mathbf{x}_i) > \alpha_\mathbf{n}} \cdot \mathbb{1}_{|d(\mathbf{x}) - d(\mathbf{x}_i)| < \alpha_d} \cdot \mathbb{1}_{m(\mathbf{x}) = m(\mathbf{x}_i)},$$

acts as a cross-bilateral term, preventing filtering across normal ($\mathbf{n}$) or depth (d) discontinuities or between different reflector material IDs (m). We deliberately use indicator functions instead of more canonical exponential cross-bilateral weights [Tomasi and Manduchi 1998], since the spatial filter footprint already accounts for local geometry using $\mathcal{G}_I$. We set indicator thresholds to $\alpha_\mathbf{n} = 0.8$ and $\alpha_d = 0.2$ in our experiments. Finally, Z is the normalizing partition function, ensuring filter weights sum to unity.

6.3 Efficient Filter Approximation

Evaluating Eq. 4 is costly for large footprints (Fig. 10b), so we employ an approximation to maintain interactivity. A two-pass separable anisotropic filter [Geusebroek et al. 2003] reduces complexity from quadratic to linear, but is only accurate for spatially invariant filter kernels. We observe that within regions defined by w_r the filter parameters vary smoothly, by construction. Naively implementing a separable edge-stopping filter introduces artifacts at edges that do not align with the filter directions [Pham and Van Vliet 2005] (Fig. 10c). To mitigate this, we split the filter into two passes, effectively using four 1D separable filters [Gastal and Oliveira 2011]. The first and third pass filter along the first principal direction of Σ_I, whereas the others filter along the second principal direction. In all

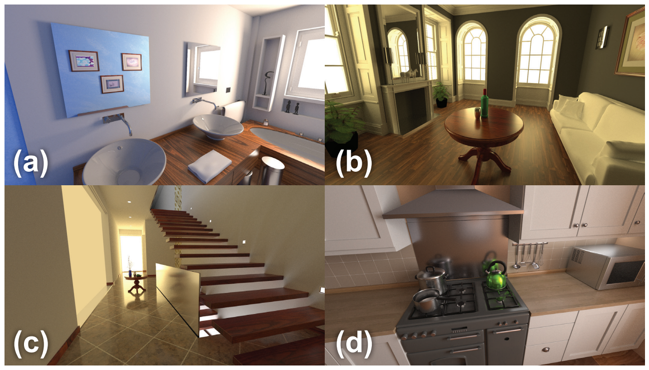

Fig. 11. Our four test scenes: *(a) Bathroom*, *(b) Livingroom*, *(c) Staircase* and *(d) Small Kitchen*.

cases, covariances must be halved to maintain the correct footprint. We observe this very closely matches the full 2D filter (Fig. 10d).

7 RESULTS, EVALUATION AND COMPARISONS

We implemented our approach in our internal framework using C++ and OpenGL. We will release the full source code, including all preprocessing and runtime components. After an initial prepass where geometric information is rendered in the novel view, the path perturbation process and gathering are run in a single fragment shader. The reconstruction filter is then applied on the gathering output in a separate set of render passes.

Specular layers and the diffuse light map were generated with a modified version of the *Mitsuba* unidirectional path tracer at 2048 samples per-pixel. Each scene contains 252 probes placed on a regular grid, each generated at a 1024x512 resolution. We render the novel view at 1920x1080.

To highlight their relative importance, we first evaluate different aspects of our algorithm on four test scenes. We then compare our results to five different methods, including some quantitative comparisons. We present results for varying levels of glossiness, from perfect mirrors to rough metal and wood. Results are best appreciated in the provided supplementary videos; we include a supplemental HTML file that allows side-by-side video comparisons of different methods.

7.1 Test Scenes

We evaluated our method on the scenes shown in Fig. 11: *Bathroom*, *Livingroom*, *Staircase* and *Small Kitchen*, all from the Bitterli [2016] model repository. The first three contain the original geometry, with some added elements to showcase glossy reflections. *Small Kitchen* contains only a portion of the repository's scene; the high object count in the original required a large diffuse light map. Solutions for handling large, complex light maps exist, but are orthogonal to our approach and we leave this as future work.

7.2 Evaluation

We evaluate four aspects of our algorithm. All figures in this subsection show only our generated glossy layer. Specifically, we explore:

(1) the effect of our probe parametrization,
(2) the effect of gathering via our approximate path perturbation,
(3) the effect of total probe count and the subset used at runtime,
(4) and the effect of our glossiness filtering.

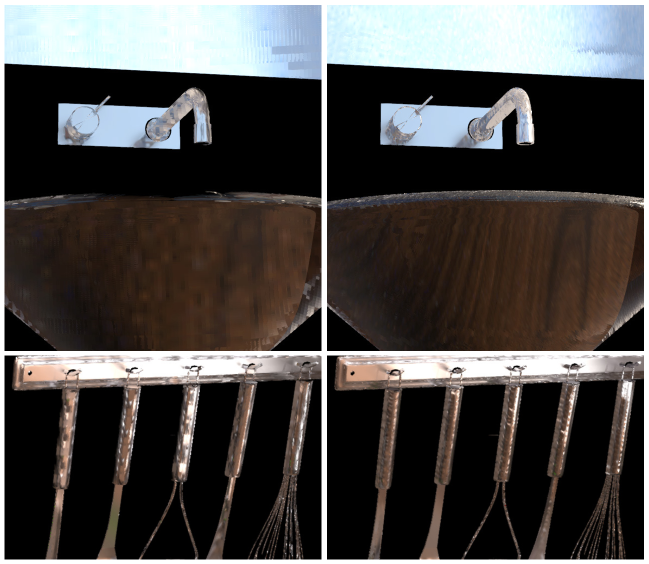

Fig. 12. Comparing (left) regular and (right) our adaptive parameterization when warping just a single probe into the novel view. Top: *Bathroom* scene, Bottom: *Small Kitchen*. For clarity, we do not apply our glossiness filter.

Fig. 12 compares results using a standard lat-long probe parameterization and our adaptive method when warping just the single closest probe into a novel view. Adding resolution for reflections viewed at a grazing angle (*Bathroom*) and on high-frequency surfaces (*Small Kitchen*) clearly reduces aliasing and subsampling artifacts. The additional resolution improves warping, as geometric information is more accurate. For clarity, renderings in Fig. 12 and 13 do not include our glossiness filtering.

For gathering using our approximate path perturbation, Fig. 13 shows the effects of our paraboloid approximation and using a single level search. We see that using a planar approximation on curved objects leads to large regions where the corrective search fails, while our paraboloid representation gives much improved results (Fig. 13 first row). Decreasing the search window can miss optimal probe samples, creating distortions on the bin (Fig. 13 second row). Using only the coarsest search level (Fig. 13 third row) we can select suboptimal probe samples, resulting in a larger number of incorrect samples.

In Fig. 14 we show the effect of using fewer probes for reprojection and reducing the number of precomputed probes. In both cases, the reduction prevents more pixels in novel views from discovering relevant probe samples, creating discontinuities and other inconsistencies on glossy surfaces.

Using probes containing full roughness materials, discontinuities are visible at reflected occlusion boundaires. In Fig. 15 we illustrate the effects of our two-step glossiness convolution, using probes with halved material roughness. This reproduces the desired material glossiness without artifacts at occlusions.

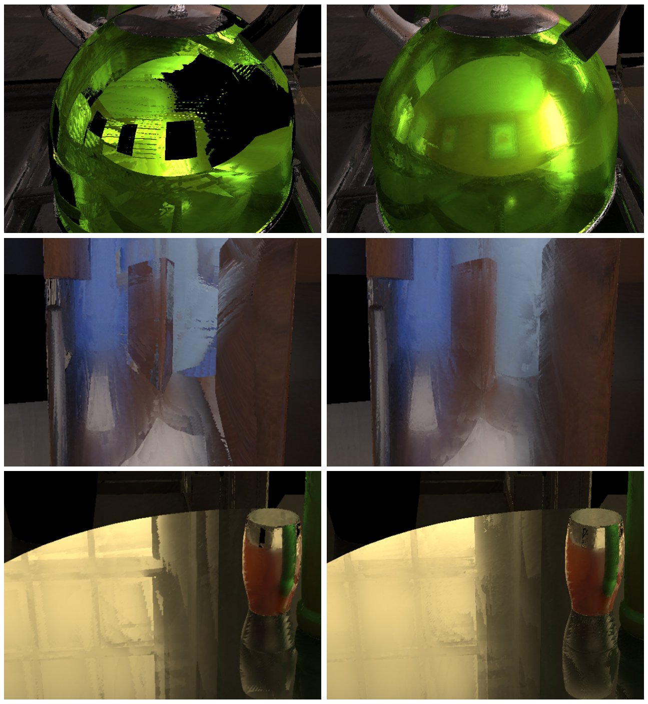

Fig. 13. Row 1: Paraboloid approximation. Left using a planar approximation and right our paraboloid. Row 2: Left using reduced (3x3;3x3) search windows, right, our solution. Row 3: Left using a one level search, right our 2-level solution. Please note that glossiness filtering is not applied here.

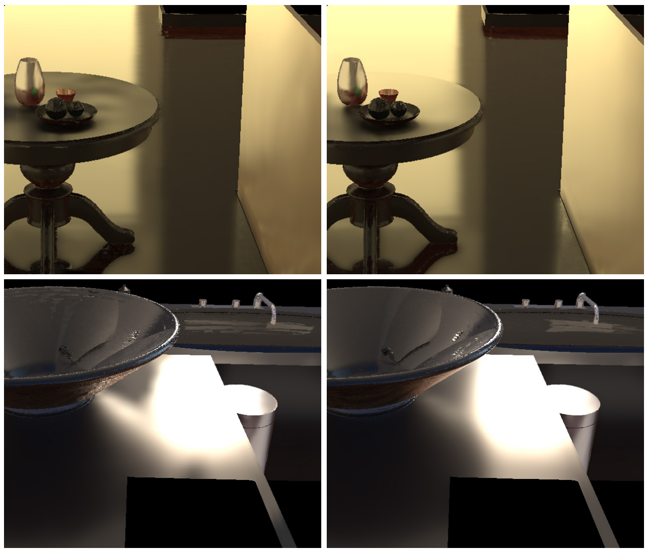

Fig. 14. Top: left, selecting 4 probes instead of 8; right our solution selecting 8 probes. Bottom: left, total of 126 probes; right our solution with 252. Fewer available probes lead to missing information on some surfaces.

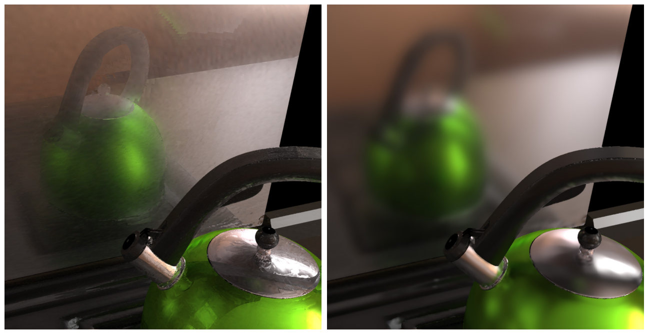

Fig. 15. Left: glossy layered rendered by gathering from probes generated using initial material roughness. The reflections are sharper. Right: our glossiness convolution applied on halved roughness probes approximate the effect of the material full roughness.

7.3 Results and Comparisons

Fig. 16 shows frames from our supplementary video's camera paths, together with the corresponding ground truth. We accurately capture complex glossy light paths at interactive frames rates, including complex secondary glossy effects (e.g., the reflection of the top of the bin—row 1, left; glossy reflections of the table—row 4, right). Nonetheless, while our approximation is quite accurate overall, some differences remain (e.g., roughness levels on the floors). Please refer to the supplemental for a visualization of the error between the path traced ground truth and our results.

We compare to three baselines and two previous methods, along with path-traced ground truth rendered with *Mitsuba*. We encourage the reader to watch the corresponding videos, provided as supplemental material. Note: due to issues converting and loading model and material formats, we only compare the *Bathroom* scene across all methods.

7.3.1 Comparisons with Baselines. The first baseline is an image-based rendering (IBR) approach akin to an unstructured lumigraph (ULR) [Buehler et al. 2001] rendering of our probes. We reproject the probes using the scene geometry and use standard ULR weights per-pixel. Evidently, this naive IBR cannot correctly capture reflections, see Fig. 18.

The second and third baselines use real-time ray tracing (RTRT) via NVIDIA's *Falcor* framework [Benty et al. 2020]. We compare to a real-time path tracer, denoised with the OptiX denoiser. We gave this path tracer the same compute budget as our prototype, resulting in 2 paths per pixel. Even though this captures the general structure of light paths, quality and stability are generally lower (e.g., in the reflections on the sink, Fig. 17, top right). Please note that materials in the *Falcor* real-time path tracer and – to a lesser extent – McGuire et al. [2017b] are slightly different from those in our system, resulting in small differences in overall appearance. We also compare to lightmap rendering augmented by BRDF-sampled rays. Again using the same budget allows for 3 bounces, using 4 samples for the glossy lobes (at the first path vertex) to obtain the best results. This method also provides good results, but misses secondary glossy effects from distant emitters that require a much higher sample count (Fig. 17, bottom left). Note, for example, the

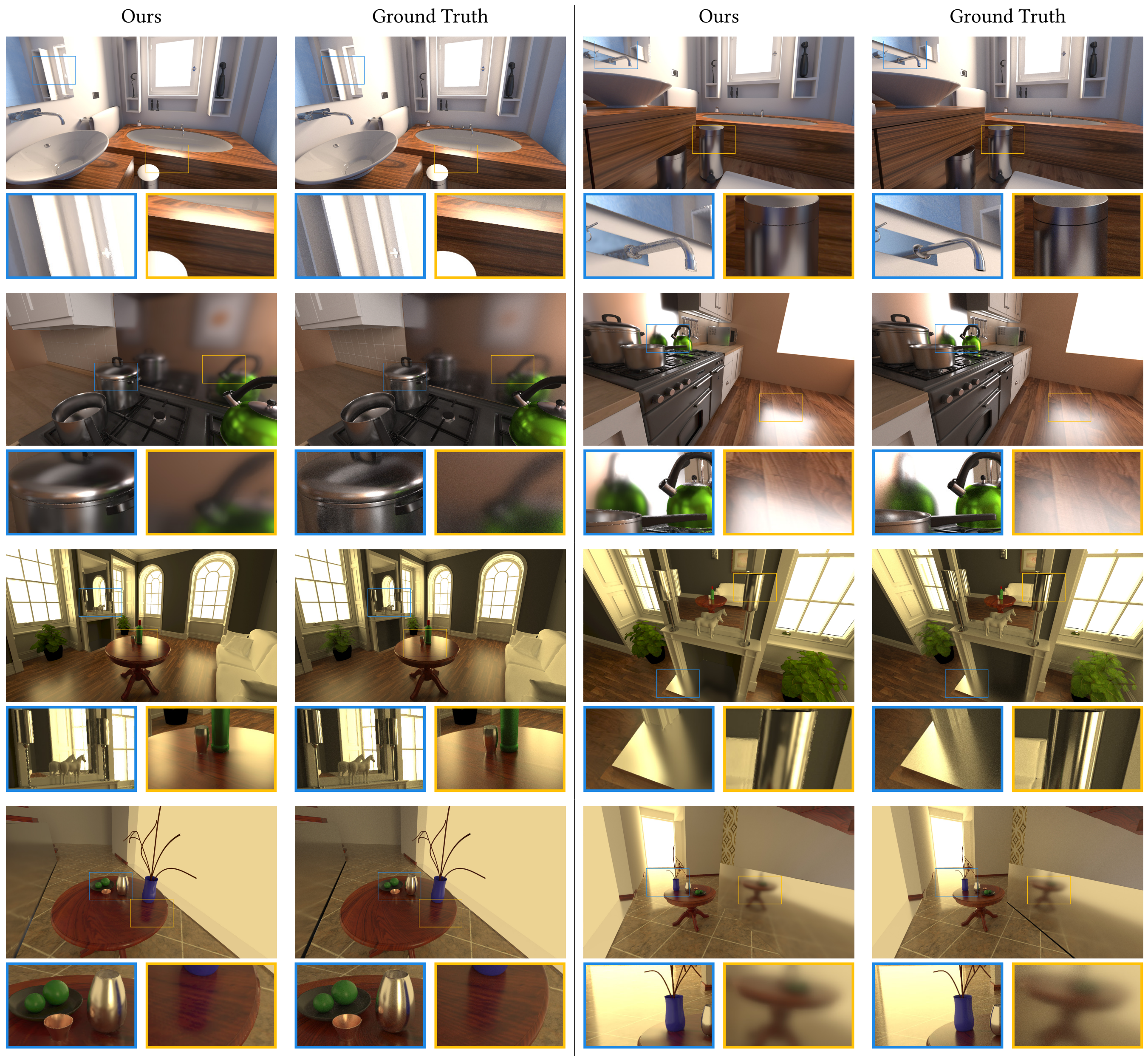

Fig. 16. Results of our method. For each scene we show two viewpoints rendered with our method and the corresponding ground truth path traced image.

missing glossy highlight on the table. In contrast, our solution has overall good image quality, even though some small inaccuracies remain in reflections. The quality is best appreciated over the entire path in the video.

7.3.2 Comparisons with Prior Art. We also compare to image-space gathering [Robison and Shirley 2009] and McGuire et al.'s [2017b] probe-based method. We reimplemented the former in our framework, using Optix [Parker et al. 2010] and fetching our lightmap to generate the perfect reflection image. For McGuire et al. [2017b], we adapted the publicly available implementation in the G3D framework [McGuire et al. 2017a]. For a fair comparison, we use *Mitsuba* to path-trace 128 regular light probes using their octahedral parameterization at 1024x1024 resolution, giving the same overall pixel count our probes used. We import these probes into G3D and use them for all processing required, e.g., irradiance. We activate their glossy reflections, which trace 8 additional rays in the probes. Both methods were given the same compute budget as ours. Again, the result is plausible, but glossy effects are missing (Fig. 17-18).

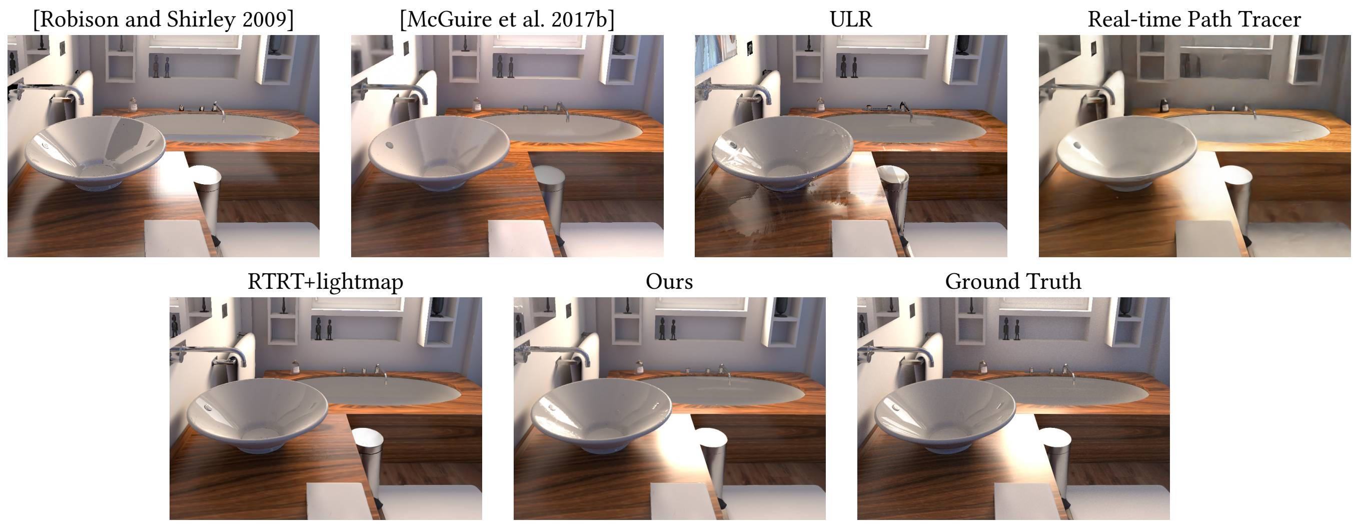

Fig. 17. Comparisons with same frame rate for each method. Previous methods: [Robison and Shirley 2009], [McGuire et al. 2017b]; baselines: unstructured lumigraph (ULR), real-time path tracing using Falcor, RTRT with light map; ours and the path-traced ground truth.

Table 1. Quantitative error metrics, using both RGB and luminance, comparing Image Space Gathering [Robison and Shirley 2009], the probe-based approach of McGuire et al. [2017b], the unstructured lumigraph (ULR), real-time path tracing and RTRT with light map baselines, and our method. Lower is better.

	RMSE		DSSIM	
Method	RGB	Lum.	RGB	Lum.
[Robison and Shirley 2009]	.086	.084	.072	.064
[McGuire et al. 2017b]	.093	.093	.087	.080
ULR	.074	.073	.100	.091
RT path tracer	.049	.042	.123	.114
RTRT+Lightmap	.050	.050	.063	.056
Ours	**.027**	**.026**	**.046**	**.039**

7.3.3 Quantitative Evaluation. We performed a quantitative evaluation using both root mean square error (RMSE) and structural dissimilarity (DSSIM) [Loza et al. 2006]. We compute the error between the generated and ground truth glossy layers. Error is averaged over 12 frames sampled regularly along the path recorded in the *Bathroom* scene. Table 1 summarizes the error for Robison and Shirley [2009], McGuire et al. [2017b], the unstructured lumigraph (ULR), real-time path tracing using Falcor and RTRT with light map baselines, and our method. The error for our method is consistently much lower than the previous work and baselines.

7.3.4 Statistics. The timings and memory consumption for our method, including preprocessing, are shown in Table 2. Rendering times are averaged over the paths shown in the supplemental video. Interactive performance was measured on a computer with an Intel Core i7-7800X processor, 64GB of RAM, and a NVIDIA Geforce RTX 2080Ti with 11GB of VRAM. We currently use the *Mitsuba* renderer to precompute probes and diffuse light maps on our cluster. A typical node has a dual Intel Xeon Silver 4110 processor and 192GB of RAM. We could instead use a real-time path tracer, greatly accelerating this step, but we opted for *Mitsuba*'s mature pipeline to handle the materials in our scene repository. Comparing with our Falcor path-tracer implementation in *Bathroom* for example, for approximately the same quality one could expect a 10x speedup in preprocessing. Our adaptive parameterization minimizes overall memory use, especially when using half-precision probe textures. Our probe texels use 24 bytes: glossy color (6 bytes), reflected positions and material ID (8 bytes), triangle ID (4 bytes), barycentric coordinates (4 bytes), parameterization and its derivatives (8 bytes, stored at half resolution). Other geometric information is stored per-vertex and interpolated at runtime.

In table 3 we display rendering time and VRAM use for the *Bathroom* scene for different probe counts and resolutions, each time doubling the number of probes or pixels. Because only the eight closest probes are gathered, performance is stable apart from a slight improvement at low resolutions thanks to texture caching. Note that the highest count test incurs a performance loss as the probes did not fit in VRAM. Memory consumption is approximately linear with the number of probes and pixels. Our method remains viable at lower memory footprints than that presented in this section, at the cost of artefacts in sharp reflectors and the shape of glossy highlights. We show example error maps with different probe counts and resolutions in supplemental.

8 LIMITATIONS AND FUTURE WORK

Our method achieves plausible results with a satisfactory accuracy in many cases (see Fig. 16-18); however it has some limitations.

The computational cost and memory of the precomputation is the main drawback of our approach, which also limits our solution to static scenes. An incremental approach to building light probes and

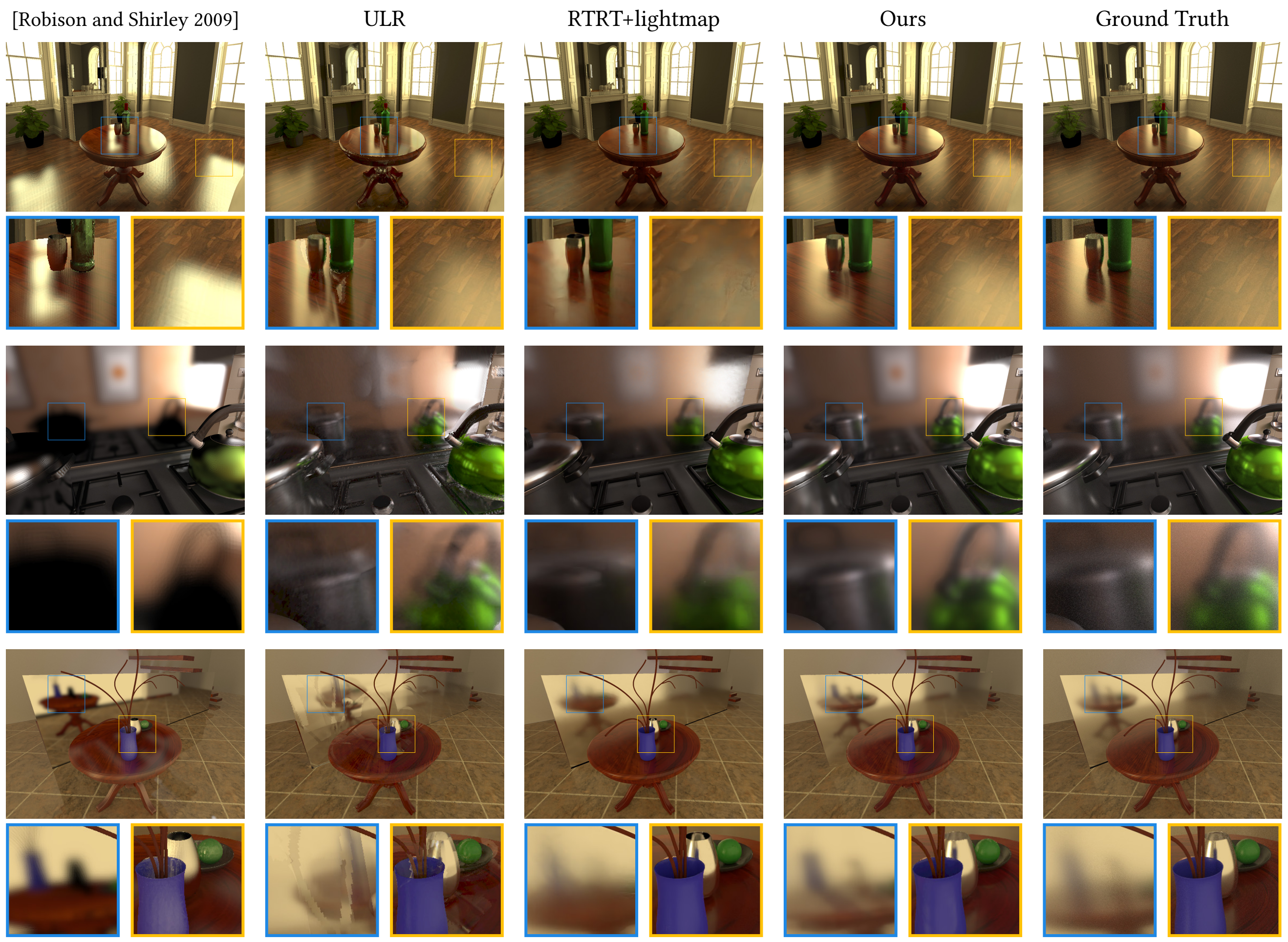

Fig. 18. Equal time comparisons for each method. Left to right: [Robison and Shirley 2009], ULR, RTRT with light map, ours and the path-traced ground truth.

Table 2. Timings and memory consumption of our method. Rendering timings average costs over frames in our videos.

	Bathroom	Kitchen	Livingroom	Staircase
Preprocess				
Parameterization	9 min	6 min	8 min	8 min
Geom. Data	3 min	3 min	3 min	2 min
Light map	10 h	9 h	15 h	12 h
Probes	16 h	13 h	22 h	23 h
Runtime				
Rasterization	0.9 ms	1.1 ms	0.9 ms	0.7 ms
Raycasting	6.3 ms	6.3 ms	6.3 ms	5.4 ms
Gathering	21 ms	27 ms	26 ms	18 ms
Filtering	10 ms	12 ms	11 ms	11 ms
Total	41 ms	47 ms	46 ms	36 ms
VRAM	5.6 GB	5.8 GB	5.7 GB	5.7 GB

Table 3. Average timings and VRAM usage of our method on *Bathroom* for different probe counts and resolutions. Rendering timings average costs over frames in our video.

	768x384px	1024x512px	1440x720px
125 probes	36ms 2.8GB	38ms 3.7GB	41ms 5.9GB
252 probes	37ms 4.0GB	41ms 5.6GB	41ms 10.1GB
504 probes	37ms 6.3GB	39ms 9.7GB	118ms 18.6GB

maps via hardware accelerated path tracing would be an interesting way to lift this limitation in future work.

Specular geometric aliasing can appear along object edges, occasionally creating bright crawling pixels. This is a common issue with sharp specular lobes; methods exist to reduce this artifact by adjusting the roughness at run-time [Kaplanyan et al. 2016; Tokuyoshi and Kaplanyan 2019]. An interesting possibility for future work would be to merge such approaches with our current gathering, by extending this formulation to large-scale filter footprints.

We are currently limited to opaque materials. Extending our approach to transparent materials probably requires storing more information in the probes plus a new gathering approach for reprojecting transmissive surfaces. Such a gathering solution is an exciting future direction. A similar argument applies to extending the method to anisotropic materials, though a simpler solution may be possible for this case.

Because our method blends reprojected specular information from nearby pixels, spatially varying BRDFs with high-frequency roughness details are not accurately supported. Our method could be extended in this direction by incorporating roughness maps in the filter footprint estimation, potentially by using the minimal roughness over the map in precomputation, and replicating the final roughness during filtering. Roughness of probe samples could also be taken into account when evaluating the gathering score, to reject candidates with a different roughness.

While the specular path perturbation framework allowed our technique to be real-time, it required a simplified representation for reflector surfaces. Manifold exploration [Jakob 2013] provides a more general way of estimating perturbed specular paths on arbitrary geometry. Exploring how it could be applied in real-time is an opportunity for future work.

Finally, our two-step convolution is approximate, as seen with the small differences in glossiness in Figs. 16-18. A deeper study of error bounds and a more accurate approximation are definitely worthy goals. Nonetheless, our current results are plausible and provide convincing and quite accurate interactive renderings.

9 CONCLUSION

We presented a novel algorithm for real-time rendering of synthetic scenes using glossy paths dynamically reprojected from probes and diffuse lighting from a light map. Our solution builds on three main contributions: an adaptive parameterization to optimize probe memory usage, an accurate gathering algorithm for reprojecting glossy paths into novel views, and a two-step solution to avoid reflection boundary sharpening that occurs when reprojecting naively.

Our solution allows interactive walkthroughs with global illumination for opaque scenes. The path we chose is based on precomputation: the advantage is that complex light paths are precomputed at high quality and that our reprojection can accurately construct novel views. However, this comes at the price of the computational overhead of precomputation and the limitation to static scenes. On the other end of the spectrum are online methods, such as a denoised real-time ray-tracer. Our results show the feasibility of using precomputed data to render complex light paths interactively, and future methods should build on the full spectrum of methods from fully online to precomputed. Our solutions for memory optimization, accurate reprojection and occlusion-aware glossiness will hopefully be useful building blocks to such solutions, moving towards the ultimate goal of real-time global illumination for complex, dynamic scenes.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their suggestions for improving the presentation. This research was funded by the ERC Advanced Grant FUNGRAPH (No. 788065, http://fungraph.inria.fr), and by software donations from Adobe. The authors are grateful to Inria Sophia Antipolis - Méditerranée "Nef" computation cluster (https://wiki.inria.fr/ClustersSophia/Clusters_Home). We also thank B. Bitterli for the scenes in his rendering repository, A. Gruson and J. Lehtinen for feedback on early versions, R. Zayer for help with the quasi-harmonic solver implementation and M. McGuire for providing comparison code.

REFERENCES

Nasir Ahmed, T Raj Natarajan, and Kamisetty R Rao. 1974. Discrete cosine transform. *IEEE Trans. Comput.* 100, 1 (1974), 90–93.

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

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

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer Graphics and Interactive Techniques.* ACM, 425–432.

John Burgess. 2020. RTX on the NVIDIA Turing GPU. *IEEE Micro* 40, 2 (2020), 36–44.

Min Chen and James Arvo. 2000a. Perturbation methods for interactive specular reflections. *IEEE Transactions on Visualization and Computer Graphics* 6, 3 (2000), 253–264.

Min Chen and James Arvo. 2000b. Theory and application of specular path perturbation. *ACM Transactions on Graphics (TOG)* 19, 4 (2000), 246–278.

Shenchang Eric Chen, Holly E Rushmeier, Gavin Miller, and Douglass Turner. 1991. A progressive multi-pass method for global illumination. In *ACM SIGGRAPH Computer Graphics*, Vol. 25. ACM, 165–174.

Abhinav Dayal, Cliff Woolley, Benjamin Watson, and David Luebke. 2005. Adaptive frameless rendering. In *ACM SIGGRAPH Courses.* ACM, 24.

Pau Estalella, Ignacio Martin, George Drettakis, Dani Tost, Olivier Devillers, and Frédéric Cazals. 2005. Accurate Interactive Specular Reflections on Curved Objects. In *Proceedings of Vision Modeling and Visualization.* Eurographics Association. http://www-sop.inria.fr/reves/Basilic/2005/EMDTDC05

Sebastian Friston, Tobias Ritschel, and Anthony Steed. 2019. Perceptual Rasterization for Head-mounted Display Image Synthesis. *ACM Transactions on Graphics (TOG)* 38, 4 (2019).

Eduardo SL Gastal and Manuel M Oliveira. 2011. Domain transform for edge-aware image and video processing. *ACM Transactions on Graphics (TOG)* (2011), 1–12.

Jan-Mark Geusebroek, Arnold WM Smeulders, and Joost Van De Weijer. 2003. Fast anisotropic gauss filtering. *IEEE Transactions on Image Processing* 12, 8 (2003), 938–943.

Gene Greger, Peter Shirley, Philip M Hubbard, and Donald P Greenberg. 1998. The irradiance volume. *IEEE Computer Graphics and Applications* 18, 2 (1998), 32–43.

Ziyad S Hakura and John M Snyder. 2001. Realistic reflections and refractions on graphics hardware with hybrid rendering and layered environment maps. In *Rendering Techniques 2001.* Springer, 289–300.

Ziyad S Hakura, John M Snyder, and Jerome E Lengyel. 2001. Parameterized environment maps. In *Proceedings of the Symposium on Interactive 3D Graphics.* ACM, 203–208.

Paul S Heckbert. 1990. Adaptive radiosity textures for bidirectional ray tracing. *ACM SIGGRAPH Computer Graphics* 24, 4 (1990), 145–154.

Antti Hirvonen, Atte Seppälä, Maksim Aizenshtein, and Niklas Smal. 2019. Accurate Real-Time Specular Reflections with Radiance Caching. In *Ray Tracing Gems.* Springer, 571–607.

Jozef Hladky, Hans-Peter Seidel, and Markus Steinberger. 2019. Tessellated Shading Streaming. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 171–182.

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

Wenzel Jakob. 2013. Light transport on path-space manifolds.

Anton S Kaplanyan, Stephan Hill, Anjul Patney, and Aaron E Lefohn. 2016. Filtering distributions of normals for shading antialiasing.. In *Proceedings of High Performance Graphics.* ACM, 151–162.

Brian Karis. 2014. High-quality temporal supersampling. *Advances in Real-Time Rendering in Games, ACM SIGGRAPH Courses* 1 (2014), 1–55.

Hans Knutsson and C-F Westin. 1993. Normalized and differential convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 515–523.

Sébastien Lagarde and Antoine Zanuttini. 2012. Local image-based lighting with parallax-corrected cubemaps. In *ACM SIGGRAPH 2012 Talks.* ACM, 36.

Jaakko Lehtinen, Timo Aila, Samuli Laine, and Frédo Durand. 2012. Reconstructing the indirect light field for global illumination. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–10.

Dani Lischinski and Ari Rappoport. 1998. Image-based rendering for non-diffuse synthetic scenes. In *Rendering Techniques' 98*. Springer, 301–314.

Gerrit Lochmann, Bernhard Reinert, Tobias Ritschel, Stefan Müller, and Hans-Peter Seidel. 2014. Real-time Reflective and Refractive Novel-view Synthesis. In *Proceedings of Vision Modeling and Visualization*. Eurographics Association, 9–16.

Artur Loza, Lyudmila Mihaylova, Nishan Canagarajah, and David Bull. 2006. Structural similarity-based object tracking in video sequences. In *9th International Conference on Information Fusion*. IEEE, 1–6.

Christian Luksch, Robert F Tobler, Ralf Habel, Michael Schwärzler, and Michael Wimmer. 2013. Fast light-map computation with virtual polygon lights. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. ACM, 87–94.

Christan Luksch, Michael Wimmer, and Michael Schwärzler. 2019. Incrementally baked global illumination. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. ACM, 4.

Zander Majercik, Jean-Philippe Guertin, Derek Nowrouzezahrai, and Morgan McGuire. 2019. Dynamic diffuse global illumination with ray-traced irradiance fields. *Journal of Computer Graphics Techniques* 8, 2 (2019).

Morgan McGuire, Michael Mara, and Zander Majercik. 2017a. The G3D Innovation Engine. https://casual-effects.com/g3d https://casual-effects.com/g3d.

Morgan McGuire, Mike Mara, Derek Nowrouzezahrai, and David Luebke. 2017b. Real-time global illumination using precomputed light field probes. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. ACM, 2.

Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*. Springer, 35–57.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934* (2020).

Joerg H Mueller, Philip Voglreiter, Mark Dokter, Thomas Neff, Mina Makar, Markus Steinberger, and Dieter Schmalstieg. 2018. Shading atlas streaming. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–16.

Diego Nehab, Pedro V Sander, Jason Lawrence, Natalya Tatarchuk, and John R Isidoro. 2007. Accelerating real-time shading with reverse reprojection caching. In *Graphics Hardware*, Vol. 41. ACM, 61–62.

NVIDIA. 2018. NVIDIA Turing GPU architecture: Graphics reinvented. https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf

Eyal Ofek and Ari Rappoport. 1998. Interactive reflections on curved objects. In *Proceedings of the 25th annual conference on Computer Graphics and Interactive Techniques*. ACM, 333–342.

Steven G Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. 2010. OptiX: a general purpose ray tracing engine. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 1–13.

Tuan Q Pham and Lucas J Van Vliet. 2005. Separable bilateral filtering for fast video preprocessing. In *IEEE International Conference on Multimedia and Expo*. IEEE, 4–7.

Peiran Ren, Jiaping Wang, Minmin Gong, Stephen Lin, Xin Tong, and Baining Guo. 2013. Global illumination with radiance regression functions. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.

Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. 2012. The state of the art in interactive global illumination. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 160–188.

Austin Robison and Peter Shirley. 2009. Image space gathering. In *Proceedings of High Performance Graphics*. ACM, 91–98.

Takafumi Saito and Tokiichiro Takahashi. 1991. NC machining with G-buffer method. In *ACM SIGGRAPH Computer Graphics*, Vol. 25. ACM, 207–216.

Pedro V Sander, Steven Gortler, John Snyder, and Hugues Hoppe. 2002. Signal-specialized parameterization. *Eurographics Workshop on Rendering* (2002).

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics*. ACM, 1–12.

Harry Shum and Sing Bing Kang. 2000. Review of image-based rendering techniques. In *Visual Communications and Image Processing*, Vol. 4067. International Society for Optics and Photonics, 2–13.

Ari Silvennoinen and Jaakko Lehtinen. 2017. Real-time global illumination by precomputed local reconstruction from sparse radiance probes. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–13.

Peter-Pike J Sloan, David M Weinstein, and J Brederson. 1998. Importance driven texture coordinate optimization. In *Computer Graphics Forum*, Vol. 17. Wiley Online Library, 97–104.

Philipp Slusallek, Marc Stamminger, Wolfgang Heidrich, J-C Popp, and H-P Seidel. 1998. Composite lighting simulations with lighting networks. *IEEE Computer Graphics and Applications* 18, 2 (1998), 22–31.

László Szirmay-Kalos, Barnabás Aszódi, István Lazányi, and Mátyás Premecz. 2005. Approximate ray-tracing on the GPU with distance impostors. In *Computer Graphics Forum*, Vol. 24. Wiley Online Library, 695–704.

László Szirmay-Kalos, Tamás Umenhoffer, Gustavo Patow, László Szécsi, and Mateu Sbert. 2009. Specular effects on the GPU: State of the art. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 1586–1617.

Ayush. Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, Rohit Pandey, Sean Fanello, Gordon Wetzstein, Jun-Yan Zhu, Christian Theobalt, Maneesh Agrawala, Eli Shechtman, Dan B Goldman, and Michael Zollhöfer. 2020. State of the Art on Neural Rendering. *Computer Graphics Forum (EG STAR)* (2020).

Yusuke Tokuyoshi and Anton S Kaplanyan. 2019. Improved geometric specular antialiasing. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. ACM, 1–8.

Carlo Tomasi and Roberto Manduchi. 1998. Bilateral filtering for gray and color images. In *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 839–846.

John R Wallace, Michael F Cohen, and Donald P Greenberg. 1987. *A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods*. Vol. 21. ACM. 311–320 pages.

Bruce Walter, George Drettakis, and Steven Parker. 1999. Interactive rendering using the render cache. In *Rendering techniques' 99*. Springer, 19–30.

Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. 2007. Microfacet Models for Refraction through Rough Surfaces. *Rendering Techniques 2007* 2007, 18th.

Yue Wang, Soufiane Khiat, Paul G Kry, and Derek Nowrouzezahrai. 2019. Fast non-uniform radiance probe placement and tracing. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. ACM, 5.

Gregory Ward and Maryann Simmons. 1999. The holodeck ray cache: an interactive rendering system for global illumination in nondiffuse environments. *ACM Transactions on Graphics (TOG)* 18, 4 (1999), 361–368.

Chris Wyman and Adam Marrs. 2019. Introduction to DirectX raytracing. In *Ray Tracing Gems*. Springer, 21–47.

Kun Xu, Yan-Pei Cao, Li-Qian Ma, Zhao Dong, Rui Wang, and Shi-Min Hu. 2014. A practical algorithm for rendering interreflections with all-frequency BRDFs. *ACM Transactions on Graphics (TOG)* 33, 1 (2014), 1–16.

Jingyi Yu, Jason Yang, and Leonard McMillan. 2005. Real-time reflection mapping with parallax. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. ACM, 133–138.

Rhaleb Zayer, Christian Rossl, and H-P Seidel. 2005. Discrete tensorial quasi-harmonic maps. In *International Conference on Shape Modeling and Applications*. IEEE, 276–285.

Henning Zimmer, Fabrice Rousselle, Wenzel Jakob, Oliver Wang, David Adler, Wojciech Jarosz, Olga Sorkine-Hornung, and Alexander Sorkine-Hornung. 2015. Path-space motion estimation and decomposition for robust animation filtering. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 131–142.

A DISCUSSION OF CHOICE OF DCT

In Sec. 4.2.1, we use the DCT to identify probe regions requiring increased resolution due to small features or high geometric complexity. While any frequency decomposition could be used, we used the DCT for simplicity and its real-valued coefficients. We could have used image depths, rather than normals, as a representation of geometry. However, due to the linearity of the convolution, larger depth differences naturally produce stronger responses. To avoid this, we use the normal buffer, which contains normalized values by construction. A canonical DCT application subdivides the image into $N \times N$ blocks and treats blocks individually, resulting in one response per block. In contrast, we convolve the image with the corresponding $N \times N$ DCT basis functions, yielding an individual response per pixel. Note that, therefore, our approach gives the same responses as the block-based method, just at an N^2 higher spatial resolution.