

Filtering Environment Illumination for Interactive Physically-Based Rendering in Mixed Reality

Soham Uday Mehta^{1,2} Kihwan Kim² Dawid Pająk² Kari Pulli³ Jan Kautz² Ravi Ramamoorthi⁴

¹University of California, Berkeley ²NVIDIA Corporation ³Light ⁴University of California, San Diego

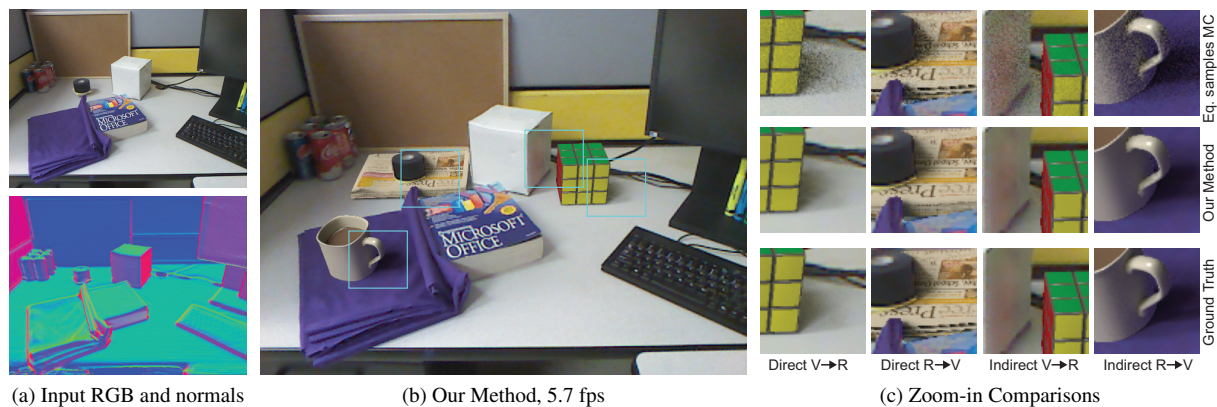


Figure 1: We start with a stream of RGBD images from a Kinect camera, and use SLAM to reconstruct the scene. (a) shows the input RGB image and reconstructed normals. In this DESK scene, we insert a virtual Rubik's cube, a newspaper, and a coffee mug, as shown in (b) where environment illumination (both direct and 1-bounce indirect) are computed using Monte Carlo (MC) path tracing followed by filtering. The full system runs at 5.7 fps. We show comparisons of our result with unfiltered MC with equal samples, which is very noisy, and reference, which takes 60× longer to render.

Abstract

Physically correct rendering of environment illumination has been a long-standing challenge in interactive graphics, since Monte-Carlo ray-tracing requires thousands of rays per pixel. We propose accurate filtering of a noisy Monte-Carlo image using Fourier analysis. Our novel analysis extends previous works by showing that the shape of illumination spectra is not always a line or wedge, as in previous approximations, but rather an ellipsoid. Our primary contribution is an axis-aligned filtering scheme that preserves the frequency content of the illumination. We also propose a novel application of our technique to mixed reality scenes, in which virtual objects are inserted into a real video stream so as to become indistinguishable from the real objects. The virtual objects must be shaded with the real lighting conditions, and the mutual illumination between real and virtual objects must also be determined. For this, we demonstrate a novel two-mode path tracing approach that allows ray-tracing a scene with image-based real geometry and mesh-based virtual geometry. Finally, we are able to de-noise a sparsely sampled image and render physically correct mixed reality scenes at over 5 fps on the GPU.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Ray-tracing—Virtual Reality

1. Introduction

Environment illumination is an important effect in physically-based as well as real-time rendering, where a surface receives illumination from light sources located at infinity, i.e. from every direction. In the past, various approx-

imate techniques for environment lighting have been proposed, such as spherical harmonics and precomputed radiance transfer. Monte Carlo (MC) ray tracing is the physically accurate technique for rendering photo-realistic imagery with environment illumination. However, the number

of rays needed to produce a visually pleasing noise-free image can be large, requiring hours to render a single image.

We observe that, on smooth surfaces, shading from environment illumination is often slowly varying, and hence one approach for fast rendering is to exploit the smoothness by appropriately filtering a sparsely-sampled Monte Carlo result (see Fig. 1). We extend the axis-aligned filtering algorithm of Mehta et al. [MWR12, MWRD13], previously limited respectively to area light direct illumination and indirect illumination only, to filter environment illumination (direct and indirect) adaptively in screen-space. The filtering scheme is fast and in screen-space, and at the same time does not overblur the shading.

In Sec. 6, we analyze the illumination, and resulting image shading in the Fourier domain. While previous work [DHS*05] has conducted such a frequency analysis, we provide new insights into the shape of spatio-angular radiance spectra. The 2D (flatland) light fields of incident illumination and visibility have different slopes. We show that convolution of the corresponding spectra in Fourier space is an oriented ellipsoid, unlike previous double-wedge models (see Fig. 3). By understanding the nature of this spectrum, we derive an axis-aligned filter and compute the spatial shading bandwidth, for both diffuse and glossy cases. Using our Fourier analysis and bandwidth prediction, we derive Gaussian image space filters (Sec. 7) for environment map direct lighting. In addition, we make two minor changes to previous axis-aligned filtering methods – (i) Temporal filtering, to use the result of the previous frame as an input to our filter, which helps reduce noise and, (ii) Anti-aliasing for primary visibility with 4 samples per pixel.

We demonstrate our technique applied to rendering mixed reality (MR). The fundamental objective of MR applications for immersive visual experiences is seamlessly overlaying virtual models into a real scene. The synthesis has two main challenges: (1) stable tracking of cameras (pose estimation) that provides a proper placement of virtual objects in a real scene, and (2) plausible rendering and post-processing of the mixed scenes. For the first challenge, there are many existing techniques that can be leveraged. In this paper, we use dense simultaneous localization and mapping (SLAM) algorithms [NLD11, IKH*11] that provide the estimated 6DOF camera pose, as well as scene geometry in the form of per-pixel positions and normals. For the second challenge, many approaches use either rasterization with a dynamic but noisy real-world mesh obtained directly from a depth camera [KTMW12], or Monte Carlo ray-trace a fixed pre-defined real-world mesh [KK13b]. We provide a two-mode path-tracing method that uses a denoised real-world vertex map obtained from the SLAM stage. As described in Sec. 5, a fast GPU ray-tracer is used for the virtual geometry, while real geometry is intersected with rays traced in screen space. Finally, we use our filtering scheme on a 16 samples/pixel noisy MC result to remove noise. This gives us the quality of a purely path-traced result at interactive rates of 5 frames/second. Note, we assume all real surfaces are diffuse, while virtual objects can be diffuse or glossy. The environment lighting is obtained by photographing a mirrored ball.

2. Previous Work

Environment Illumination: The most popular approach for real-time rendering with environment lighting is to use spherical harmonics [RH01] with pre-computed ambient occlusion, or more generally, precomputed radiance transfer [SKS02]. This approach is limited to low-frequency lighting, and requires pre-computation that does not support dynamic MR.

Rendering in MR: A pioneering work in light transport for MR was presented by Fournier et al. [FGR93], based on radiosity. It was later extended using image based lighting derived from a light probe [Deb98]. Gibson and Murta [GM00] present a hardware-rendering approach using pre-computed basis radiance-maps and shadow maps. Cossairt et al. [CNR08] synthesize inter-reflections without knowing scene geometry, using a controlled set-up to capture and re-project a 4D radiance light field. Recently, some work has adapted more modern techniques for MR, such as differential instant radiosity [KTMW12] using imperfect shadow mapping, and delta voxel cone-tracing [Fra14]. While they show fast impressive results, they are not physically accurate.

Our approach is based purely on ray-tracing. Closest to our approach are differential progressive path tracing [KK13b] and differential irradiance caching [KK13a], which compute direct and indirect illumination using path-tracing. Both methods use pre-determined real-world mesh geometry. Differential progressive path-tracing uses only one sample/pixel/frame on the combined scene geometry. Since they do not filter their result, only a very noisy image is achievable in real-time. The user must wait without camera motion for the image to become noise-free. We overcome this limitation through the use of fast yet accurate filtering, and fast accumulation of 16 samples per pixel per frame. Further, we use screen-space ray-tracing for real objects. Methods like Kán and Kaufmann [KK12] demonstrate reflections, refractions and caustics through the use of differential photon mapping, which improves the realism. We can handle reflections and refractions, but we did not implement caustics, which would require photon mapping. We use pure ray-tracing and focus on low- to mid-frequency shading.

To improve the realism of inserted virtual objects, many works focus on post-processing techniques to match color palette and noise between pixels belonging to real and virtual objects. Our goal in this paper is primarily to provide a photorealistic rendering system for MR scenes. We do not focus on tracking/reconstruction quality, lighting estimation, and post-processing techniques.

Screen-space Ray-tracing (SSRT): We introduce a hybrid ray-tracing scheme that uses screen-space rays to find intersections with real geometry represented as a vertex map. Mark et al. [MMB97] use depth information to re-render an image from a nearby viewpoint. This idea can be extended to screen-space ray-tracing, where rays traverse pixels and test depth for intersection – a technique that has been used for local specular reflections [SKS11, MMNL14]. We use SSRT to compute direct and indirect illumination

as well. Like many previous works, the real-world mesh is represented as a dynamically updated vertex map of the current frame. Hence, real objects not visible in the current frame do not affect the shading of virtual objects. One could potentially use the global volumetric signed-distance function maintained by the SLAM back-end for ray-tracing; however, this would be extremely slow.

Fourier Analysis and Axis-aligned Filtering: We are inspired by Chai et al. [CTCS00] and Durand et al. [DHS*05], who introduce the basic Fourier theory for space-angle and pixel-light light fields. The latter work, Fourier Analysis of Light Transport (FLT), models light-surface interactions atomically, and derives the Fourier equivalent for each interaction. Their Fourier spectra are parallelograms, while we show that the spectra can actually have an ellipsoidal shape; our bandwidths are more accurate. More recently, Belcour et al. [BSS*13] model the shading spectrum as a Gaussian covariance matrix that is the product of matrices for each atomic interaction, that is expensive and slow to compute. We use an end-to-end geometric approach that directly gives the object space bandwidth and is fast to compute. Bagher et al. [BSS*12] use bandwidth prediction to shade different materials under environment lighting via hierarchical shading, but do not consider occlusion.

Egan et al. [EDR11] show that the Fourier spectrum for ambient occlusion in a position-angle space is a double wedge, and demonstrate a sheared filter that tightly fits the spectrum. They demonstrate good results for low-frequency environment lighting with 32 samples/pixel, although they do not take the interaction of BRDF and visibility into account, and their filter is offline. The recent axis-aligned filtering approach of Mehta et al. [MWR12, MWRD13] derives a filter aligned with the axes, that reduces to a Gaussian filter with appropriate spatial bandwidth in image space. These works are respectively useful only for area light soft shadows and indirect illumination. We extend this approach to handle environment lighting, which requires a different curvature-dependent parametrization, since the light sources are at infinity. While these previous works treat the spectrum to be strictly a double-wedge, our spectra are not restricted to this model. The axis-aligned filter size is no longer dependent on only the BRDF or the lighting bandlimit, but combines the effect of both terms in a non-trivial way.

Denoising Monte-Carlo Images: Image filtering is a popular approach to remove noise in MC images, because of its simplicity and efficiency. Geometric information such as normals, textures, and depths, can play an important role for predicting noise in rendered images. The state of the art in this domain includes [KS13] (AMLD) that gives a noise estimation metric to locally identify the amount of noise in different parts of the image, with adaptive sampling and filtering using standard denoising techniques. Other approaches include use of Stein’s unbiased risk estimator (SURE, [LWC12]), ray histogram fusion [DMB*14] and adaptive local regression [MCY14]. These approaches support general rendering effects, but have a high reconstruction overheads in seconds, and are offline. Recently, [TSPP14]

(Fast-ANN) have shown approximate-nearest-neighbor collaborative filtering for general images at real-time speeds. We compare results to AMLD and Fast-ANN.

3. Differential Rendering

Our MR rendering system is based on the differential rendering method of Debevec [Deb98]. The idea is to estimate the pixel color considering only real objects, and considering both real and virtual objects, and add the *difference* to the raw camera image. Let L_R be the (per-pixel, outgoing) radiance due to real objects only, and L_{RV} be the radiance due to both real and virtual objects (including indirect illumination between them). Then, differential rendering composites the final image per the equation:

$$L_{final} = (1 - M) \cdot L_{RV} + M \cdot (L_{RV} - L_R + L_{cam}) \quad (1)$$

Here, M is the fraction of the pixel covered by real objects, and L_{cam} is the input radiance image.

Calculating the radiances L_R and L_{RV} implicitly requires knowledge of the real-object BRDF and texture. Under the assumption that all surfaces are diffuse, only the real-object RGB texture (albedo) k_R is unknown. We show that eqn. 1 can be written purely in terms of irradiances, without the need to explicitly estimate k_R .

We separate all outgoing radiances into a product of irradiance E and texture k . The first term in eqn. 1 corresponds to the contribution of a virtual object, so we replace L_{RV} with $k_V E_{RV}$ where k_V is known virtual object texture and E_{RV} is the pixel irradiance considering both real and virtual objects. The second term corresponds to the contribution of real objects, and we write $L_{RV} - L_R = k_R (E_{RV} - E_R)$. Like previous works, the estimate of k_R is $k_R = L_{cam} / E_R$. Substituting this into eqn. 1, we get:

$$\begin{aligned} L_{final} &= (1 - M) \cdot k_V E_{RV} + M \cdot (k_R (E_{RV} - E_R) + L_{cam}) \\ &= (1 - M) \cdot k_V E_{RV} + M \cdot \left(\frac{L_{cam}}{E_R} (E_{RV} - E_R) + L_{cam} \right) \end{aligned}$$

Simplifying,

$$L_{final} = (1 - M) \cdot k_V E_{RV} + M \cdot \left(\frac{E_{RV}}{E_R} L_{cam} \right). \quad (2)$$

Thus, we have eliminated the unknown k_R completely. Our task now is to estimate E_R and E_{RV} .

Glossy Virtual Objects: Above, we consider only diffuse real and virtual objects, so that the radiance can be factored into texture and irradiance. However, we can easily handle glossy virtual objects. In the glossy case, the L_{RV} in the first term of eqn. 1 would be split into a spatial texture and outgoing radiance that depends on viewing angle. We may represent this outgoing radiance with the same symbol E_{RV} , without changing the resulting analysis. We follow this convention for glossy virtual objects throughout the paper for simplicity. In Sec. 6.3, our theory treats glossy BRDFs rigorously.

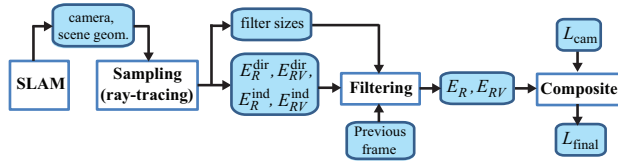


Figure 2: An abstract flowchart of our MR system. Stored input/output variables are enclosed in blue boxes, and computational steps are enclosed in white boxes. We start with dense SLAM to estimate camera pose and scene geometry from depth, followed by sparse Monte Carlo sampling to compute noisy estimates of four irradiance values (Refer to Sec. 4 for more details). We then filter the irradiances using our theory, also using the values from the previous frame for further noise reduction. The filtered values are then used to composite the final image, as explained in Sec. 3.

4. Overview

Figure 2 shows an overview of our MR system. Our system can be abstracted into four distinct stages; each stage is briefly explained below.

We take the RGBD stream from a Kinect camera and use the depth data to estimate camera pose and scene geometry. To realistically render virtual objects, the scene lighting in the form of an environment map must be known. We obtain the environment map from an image of a reflective metallic ball, using a camera looking down from above (the environment map may not be changed except for simple rotations). The entire pre-processing is done once for a single scene; each of the following stages runs per-frame at interactive speed.

1. Dense SLAM: The first step is camera tracking and scene reconstruction, to estimate per-pixel world-coordinates, normals as well as camera pose (rotation+translation), from the depth images. We use InfiniTAM [PKC*14], an open-source implementation of GPU-based voxel-hashing Kinect fusion [NZIS13]. It is fast, running at about 50 fps, and provides high quality scene reconstruction. Reconstructed normals for each scene are shown in the corresponding figures. While we can support dynamic geometry, both real and virtual, the SLAM backend is not robust to dynamic real objects. So, we only demonstrate dynamic virtual objects.

2. Sampling: We use two Monte Carlo path-tracing passes to estimate per pixel illumination without and with virtual objects. Each is the sum of direct illumination from an environment light source and 1-bounce indirect illumination, i.e., $E_R = E_R^{\text{dir}} + E_R^{\text{ind}}$, and $E_{RV} = E_{RV}^{\text{dir}} + E_{RV}^{\text{ind}}$. Thus, we compute four independent components: $E_{R/RV}^{\text{dir/ind}}$. The sampling algorithm is described in Sec. 5, with detailed pseudocode in the supplementary material.

3. Filtering: Obviously, the Monte Carlo sampled result is very noisy, and hence in the next stage, we filter each of the four irradiances using physically-based filters. The filter bandwidths are derived using Fourier analysis in Sec. 6. To reduce noise further, we also save the unfiltered irradiances from the previous frame, and use them as additional inputs to our filter. Details of this temporal filtering are discussed in Sec. 7.

4. Compositing: The last step involves compositing the final MR image, using the filtered values E_R and E_{RV} and the input RGB image, according to eqn. 2.

5. Two-mode Sampling Algorithm

In this section, we discuss our sampling algorithm (step 2 in the overview above). We aim to compute the pixel color using physically-based Monte Carlo sampling. The input is the camera position, per-pixel real object world positions and normals, and virtual object geometry as a triangle mesh. The output of the algorithm is per-pixel out-going illumination, namely E_R^{dir} , E_R^{ind} , E_{RV}^{dir} , and E_{RV}^{ind} , as explained in Sec. 3.

Previous works achieve this by tracing two kinds of rays: one that intersects only real geometry, and one that intersects both real and virtual geometry. Path-tracing using these two ray types is described in Kán and Kaufmann [KK13b], but our method is slightly different. For the virtual geometry we use meshes, since most virtual models are mesh-based, and the NVIDIA OptiX [PBD*10] ray-tracer is very suitable to intersect mesh geometry. However, we do not assume a known mesh model of the real world, and using a mesh-based ray-tracer for real geometry (per-frame vertex map) is wasteful. A screen-space ray-tracer (SSRT) computes the same result much faster, by traversing the vertex map starting from the origin pixel, and returns the world position of the intersection. We use a hierarchical traversal method adapted from [TIS08]. Since only the current frame is used, off-screen real objects will not affect the shading; the effects of this limitation are quite subtle for diffuse scenes. This can be addressed by using SSRT on a higher field-of-view vertex map rendered in the SLAM stage, at the cost of performance.

5.1. Algorithm

We propose a two-mode path-tracer that traces OptiX rays to intersect only virtual geometry, and screen-space rays to intersect only real geometry. Our sampling algorithm is explained in detail in the supplemental document.

First, we compute 4 primary samples per pixel (spp) for anti-aliasing, and we determine whether a real or a virtual object is visible at the current sample and update the mask M (see Sec. 3). Next, we compute 4 secondary samples for each of the 4 primary samples, so we compute a total of 16 spp for each of direct and indirect illumination. For direct illumination, we importance sample the environment map (see Sec. 5.2), as this gives the least amount of noise for very little overhead. For indirect illumination, we sample the cosine hemisphere for diffuse surfaces (real and virtual) and a Phong lobe for glossy surfaces (virtual only).

In the sampling step, we also save the (average) world location, normal, virtual texture k_V . We also record the minimum hit distance for direct and indirect illumination; these are required for filtering. Since texture is multiplied with irradiance after filtering, we approximate $\langle k \cdot E \rangle \approx \langle k \rangle \cdot \langle E \rangle$, where $\langle \cdot \rangle$ denotes the mean of the quantity at a pixel. This approximation works on most pixels except on silhouettes, where the error is usually small.

5.2. Importance sampling the environment map

Traditional importance sampling involves pre-computing a cumulative distribution function (CDF) of the 2D environment map and then generating samples at each pixel by finding the inverse CDF for a uniform stratified random sample. However, performing 16 such look-ups per-frame per-pixel is slow, so instead we pre-compute a large number of importance samples (4096) and store them to a buffer. Then, at each pixel we perform 16 random stratified look-ups into the buffer. This is somewhat analogous to a virtual-point-light approach, except we use a very large number of lights and sample randomly. In the supplemental material, we show that the difference between our pre-computed importance sampling strategy vs. ground truth is negligible. Complex importance sampling strategies such as [ARBJ03] are more effective at importance sampling and could also be used.

6. Fourier Analysis for Environment Lighting

So far, we have computed the quantities E_R^{dir} , etc. as per the flowchart in Fig. 2, from our sampling phase above (stage 2). However, since we only use 16 samples per pixel, the results are noisy, and accurate rendering requires filtering (stage 3). We now discuss our filtering theory, that computes the required bandwidths; Sec. 7 discusses the actual filtering given these bandwidths.

Our main contribution is the Fourier analysis of direct illumination from an environment map, considering occluders and visibility. We first perform a 2D Fourier analysis of the shading in a position-angle space, and then show that the shading is bandlimited by the BRDF in the angular dimension. The resulting axis-aligned filter provides a simple spatial bandwidth for the shading. In practice, the pre-integrated noisy per-pixel irradiance can be filtered using a Gaussian kernel of variance inversely related to the bandwidth, without altering the underlying signal.

6.1. Diffuse without visibility

As in previous works, we perform our analysis in flatland (2D). As explained in Sec. 6.4, the 2D results provide a bound on the 3D result, even though there is no simple analytic formula for 3D. We begin with the simplest case. Parameterize a diffuse receiver surface of curvature κ by x along a tangent plane defined at the origin $x = 0$. Note that κ is defined as the change in the normal angle (relative to the origin normal) per unit change in x ; it is assumed positive but the analysis extends easily for negative curvatures. Consider the set-up shown in Fig. 3(a). Let the environment illumination be $L_e(\cdot)$, with angles to the right of the normal being positive and to the left being negative. The illumination is assumed to have an angular bandwidth of B_e (i.e., 99% energy of $\|\hat{L}_e(\Omega_\theta)\|^2$ lies in $|\Omega_\theta| < B_e$). We now analyze the 1D surface irradiance given by the reflection equation in flatland:

$$E(x) = \int_{-\pi/2}^{\pi/2} L_i(x, \theta) \cos \theta d\theta = \int L_i(x, \theta) f(\theta) d\theta. \quad (3)$$

We have re-written the equation with a clamped cosine function $f(\theta) = \cos \theta$ for $\theta \in [-\pi/2, \pi/2]$ and 0 otherwise. The

Fourier transform of eqn. 3 is straightforward,

$$\hat{E}(\Omega_x) = \int \hat{L}_i(\Omega_x, \Omega_\theta) \hat{f}(\Omega_\theta) d\Omega_\theta \quad (4)$$

The incoming direction θ at x corresponds to the direction $\theta + \kappa x$ at the origin. Then,

$$L_i(x, \theta) = L_e(\theta + \kappa x). \quad (5)$$

The 2D Fourier power spectrum of L_i is a single line through the origin, with slope κ^{-1} (see [CTCS00]). This line is bandlimited in the angular dimension by \hat{f} . Let this bandwidth be B_f . Then, as shown in Fig. 3(b), the bandwidth of \hat{E} is $B_x = \kappa \cdot \min\{B_e, B_f\}$. In most interesting cases, we have $B_e > B_f$, so that

$$B_x = \kappa B_f. \quad (6)$$

This simple result (also derived in [DHS*05]) shows that higher curvature produces higher shading frequency. We now build upon this simple analysis to extend the result to include visibility and glossy BRDFs.

6.2. Diffuse with visibility

We now refine the above result by including an infinite occluder at depth z , as shown in Fig. 3(c). The occluder blocking angle θ_b for a point at a small $x > 0$ on the curved surface can be written in terms of the angle at the origin $\theta_{\text{occ}} = \theta_b(0)$:

$$\begin{aligned} \theta_b(x) &\approx \tan^{-1} \left(\frac{z \tan \theta_{\text{occ}} - x}{z + \kappa x^2} \right) - \kappa x \\ &\approx \tan^{-1} \left(\tan \theta_{\text{occ}} - \frac{x}{z} \right) - \kappa x \\ &\approx \theta_{\text{occ}} - x \frac{\cos^2 \theta_{\text{occ}}}{z} - \kappa x \end{aligned} \quad (7)$$

In the second step, we ignore the offset κx^2 of the point below the plane of parametrization, since it is quadratic in x . In the last step we use Taylor approximation to expand the arctan: $\tan^{-1}(\alpha + x) \approx \tan^{-1} \alpha + x/(1 + \alpha^2)$ for $x \ll 1$. This is similar to [RMB07]. Thus, for small curvature and small displacement x , we get $\theta_b(x) = \theta_{\text{occ}} - \lambda x$ where

$$\lambda = \kappa + \cos^2 \theta_{\text{occ}}/z \quad (8)$$

Finally, note that the visibility at x is

$$V(x, \theta) = H(\theta - \theta_b(x)), \quad (9)$$

where $H(\cdot)$ is a step function that takes value 0 when its argument is positive and 1 otherwise. Hence, the irradiance including visibility can be written as:

$$\begin{aligned} E(x) &= \int L_i(x, \theta) V(x, \theta) f(\theta) d\theta \\ &= \int L_e(\theta + \kappa x) H(\theta - \theta_{\text{occ}} + \lambda x) f(\theta) d\theta. \end{aligned} \quad (10)$$

To find the spatial bandwidth of $E(x)$, we find the Fourier transform (full derivation is provided in the supplemental

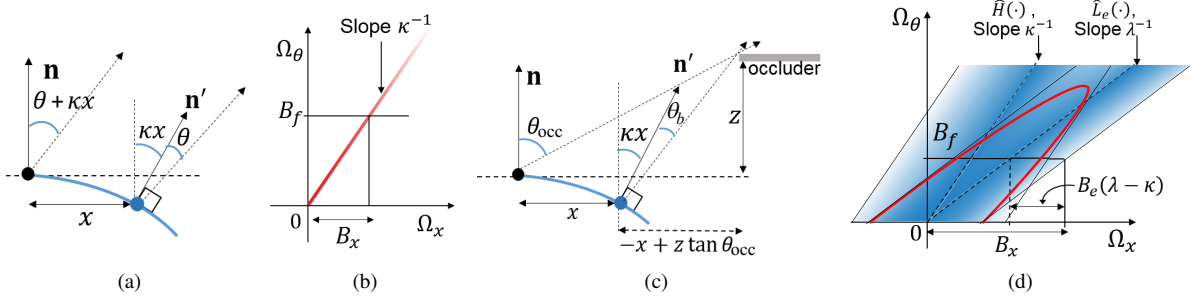


Figure 3: (a) Geometry for diffuse flatland case without considering visibility, (b) Power spectrum of L_i and axis-aligned filter. In (c) we show the flatland geometry with an occluder at depth z and (d) shows the power spectrum \hat{G} , defined in eqn. 11, which is the product of the two sheared Gaussians (shaded blue), and has an ellipsoidal shape (shaded red); B_x is our conservative estimate of its bandwidth.

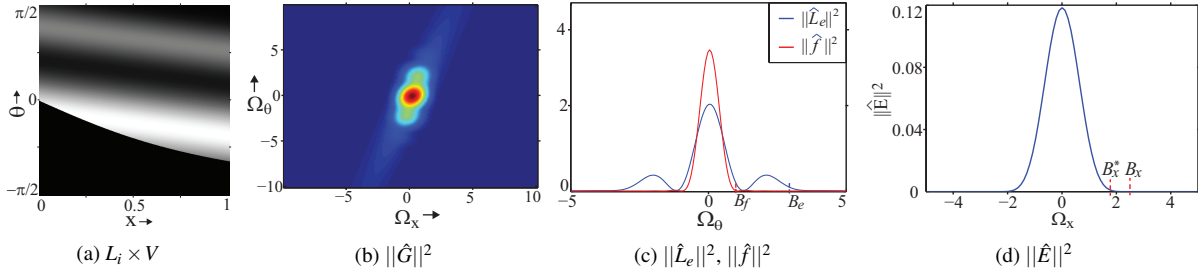


Figure 4: Verification of eqn. 13 for the simple flatland setup of Fig. 3(c) with $\kappa = 0.5$ and one occluder at $\theta_{\text{occ}} = 0$ and $z = 2$, using a Gaussian-sinusoid product for illumination. (a) shows the product $L_i \times V$ for this setup. (b) shows the power spectrum $\|\hat{G}\|^2$ of (a). In (c) we show the 1D power spectra of L_e and f , showing bandlimits $B_e = 3$ and $B_f = 1$. (d) shows the 1D power spectrum \hat{E} of the surface irradiance. Eqn. 13 gives $B_x = 2.5$, while numerically the bandwidth of \hat{E} is $B_x^* = 1.9$, thus showing that our estimate is conservative and reasonably tight. The FLT estimate in this case is $B_x = 1.0$, causing significant energy loss.

document):

$$\begin{aligned} \hat{E}(\Omega_x) &= \frac{1}{\lambda - \kappa} \int \hat{L}_e \left(-\frac{\Omega_x - \lambda \Omega_\theta}{\lambda - \kappa} \right) \hat{H} \left(\frac{\Omega_x - \kappa \Omega_\theta}{\lambda - \kappa} \right) \\ &\quad e^{j(\dots)} \hat{f}(\Omega_\theta) d\Omega_\theta \\ &= \int \hat{G}(\Omega_x, \Omega_\theta) \hat{f}(\Omega_\theta) d\Omega_\theta \end{aligned} \quad (11)$$

The phase term $e^{j(\dots)}$ due to the θ_{occ} offset in H is ignored for brevity; we are only concerned with the magnitude of the integrand. Both terms \hat{L}_e and \hat{H} are 1-D functions sheared in 2-D along lines of slopes λ^{-1} and κ^{-1} , respectively. Since the respective 1-D functions are both low-pass (i.e. 99% energy lies in a small frequency range), the product \hat{G} is shaped roughly like an ellipsoid. This is shown in Fig. 3(d). **The shape of the spectrum is no longer a simple line for a single depth occluder.**

From eqn. 11, G is bandlimited in Ω_θ by the bandwidth of f , i.e. B_f . Since \hat{L}_e has the smaller slope is λ^{-1} , the worst case spatial bandwidth of \hat{E} is that of this term. Part of the bandwidth is from the center line, specifically $B_f \lambda$. The bandwidth has an additional component due to the non-zero spread of the \hat{L}_e term in eqn. 11. Since the one-sided width of $\hat{L}_e(\Omega)$ is B_e , the width of this term is $B_e(\lambda - \kappa)$. Thus, our conservative estimate of the spatial bandwidth of

\hat{E} is

$$B_x = B_f \lambda + B_e(\lambda - \kappa) \quad (12)$$

This bandwidth is the sum of what one would get considering only visibility (first term), and the extra bandwidth due to the lighting (second term). However, it is not simply the sum of bandwidths due to illumination and visibility when considered separately, as one may have expected from the form of eqn. 10. Using the definition of λ (eqn. 8), we can re-write the filter width as:

$$B_x = \kappa B_f + (\cos^2 \theta_{\text{occ}}/z)(B_f + B_e) \quad (13)$$

We verified that our predicted bandwidth holds for many flatland cases. One such set up is shown in Fig. 4. Observe the shape of the spectrum \hat{G} . The predicted bandwidth slightly overestimates the true bandwidth.

The case with multiple occluders at different depths cannot be solved analytically; we find the most conservative bound by considering the closest occluder (smallest z). However, in this case, the spectrum is not a well-defined double-wedge as in previous works.

6.3. Glossy BRDF

We will now derive the bandwidth for shading on a curved glossy surface in flatland. Visibility is not considered, since

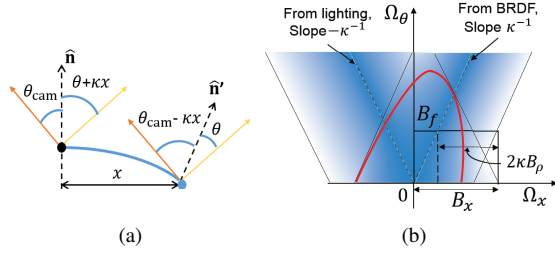


Figure 5: (a) Flatland geometry for shading with glossy BRDF. (b) The shading spectrum is the product of the two sheared Gaussians (shaded blue), and has an ellipsoidal shape (shaded red); B_x is our conservative estimate of its bandwidth.

it introduces an intractable triple product, but we intuitively add the effect of visibility at the end. As before, the surface is lit with a 1D environment illumination L_e relative to the origin, and the camera is at an angle θ_{cam} from the origin. The camera is assumed distant compared to the scale of surface being considered. The setup is shown in Fig. 5(a). The surface BRDF is assumed rotationally invariant, that is, only the difference of the incoming and outgoing angles determines the BRDF value. Numerically, $\rho(\theta_i, \theta_o) = \rho(\theta_i + \theta_o)$, where the + sign is due to angles being measured relative to the normal. Then, the radiance reflected to the camera is

$$\begin{aligned} L_0(x, \theta_{\text{cam}}) &= \int_{-\pi/2}^{\pi/2} L_i(x, \theta) \rho(\theta, \theta_o) \cos \theta d\theta \\ &= \int L_e(\theta + \kappa x) \rho(\theta_{\text{cam}} + \theta - \kappa x) f(\theta) d\theta \end{aligned} \quad (14)$$

This equation is similar to eqn. 10, except that the slopes of the two terms have opposite signs. The Fourier transform has the same form, with minor differences:

$$\hat{L}_o(\Omega_x, \theta_{\text{cam}}) = \frac{1}{2\kappa} \int \hat{L}_e \left(\frac{\Omega_x + \kappa \Omega_\theta}{2\kappa} \right) \hat{\rho} \left(\frac{\Omega_x - \kappa \Omega_\theta}{2\kappa} \right) e^{j(\dots)} \hat{f}(\Omega_\theta) d\Omega_\theta \quad (15)$$

As before, the phase term $e^{j(\dots)}$ is ignored for brevity. The terms \hat{L}_e and $\hat{\rho}$ are low-pass, and sheared along lines of slopes $-\kappa^{-1}$ and κ^{-1} , respectively. Their product is visualized in Fig. 5(b). Thus, the conservative bandwidth estimate for \hat{L}_o is

$$\begin{aligned} B_x &= \kappa B_f + 2\kappa \min \{ B_e, B_\rho \} \\ &= \kappa (B_f + 2B_\rho) \end{aligned} \quad (16)$$

Here B_ρ is the angular bandwidth of the BRDF $\rho(\cdot)$. Comparing to eqn. 13, we see that the angular bandlimit has effectively increased from B_f to $B_f + 2B_\rho$. Thus, we can modify eqn. 13 to include the effect of a visibility discontinuity and rewrite the generalized bandwidth as

$$B_x = \kappa (B_f + 2B_\rho) + (\cos^2 \theta_{\text{occ}}/z) (B_f + 2B_\rho + B_e) \quad (17)$$

We provide a numeric evaluation of this bandwidth estimate in the supplemental material, similar to Fig. 4.

6.4. Extension to 3D

The flatland results above can be extended to 3D. Directions in 3D can be parameterized in spherical coordinates (θ, ϕ) ; however, there is no simple linear form in terms of curvature in 3D which makes the analysis tedious. However, we can restrict to a fixed ϕ – along the direction of maximum curvature – and perform our analysis. The resulting bandwidth is a conservative bound for the true bandwidth considering the full hemisphere of directions, since the normal angle θ changes most rapidly along the maximum curvature direction. In practice, computing the maximum curvature per pixel is difficult, and we instead determine screen-space curvatures κ_x, κ_y , which bound the bandwidth along the image X, Y axes. The filter size is the reciprocal of the bandwidth.

In Fig. 6, using a purely virtual and untextured scene under an outdoor environment map (direct illumination only), we show that our flatland analysis works well in 3D using screen space curvatures. In (b), we show the mean of X and Y filter size. Note how the filter size depends on curvature, BRDF and occluder distance.

6.5. Indirect Illumination

The above analysis derives filter bandwidths for the direct illumination terms $E_R^{\text{dir}}, E_{RV}^{\text{dir}}$. We must also filter the sparsely-sampled noisy indirect illumination terms, $E_R^{\text{ind}}, E_{RV}^{\text{ind}}$. For indirect illumination, we use the axis-aligned filter derived in Mehta et al. [MWRD13]. For any configuration of reflectors at a minimum distance z_{min} from the receiver, with BRDF bandlimit B_h , the bandwidth formula is:

$$B_x^{\text{ind}} = B_h / z_{\text{min}} \quad (18)$$

For the diffuse case, $B_h \approx 2.8$. For a Phong BRDF with exponent m , $B_h \approx 4.27 + 0.15m$ [MWRD13].

6.6. Discussion

We discuss the novelty of and compare our contributions against previous works. We first emphasize that [MWR12] which treats area lights at finite distances and [MWRD13] which treats indirect illumination from nearby reflectors, are both not applicable to environment lighting. Bagher et al. [BSS*12] do not separate texture and illumination; they compute bandwidths numerically for glossy BRDFs and distant lighting, and their shading bandwidth is obtained by matrix multiplication. They do not consider visibility at all, which is a crucial contributor to shading bandwidth. An important difference from FLT [DHS*05] is the consideration of the source illumination bandwidth. They combine the effect of visibility and BRDF without doing a full derivation; their eqn. 21 gives the following bandwidth in object space (ignoring the scaling of $d/(\mathbf{n} \cdot \mathbf{v})$)

$$B_x^{\text{FLT}} = (2\kappa + z^{-1}) B_\rho, \quad (19)$$

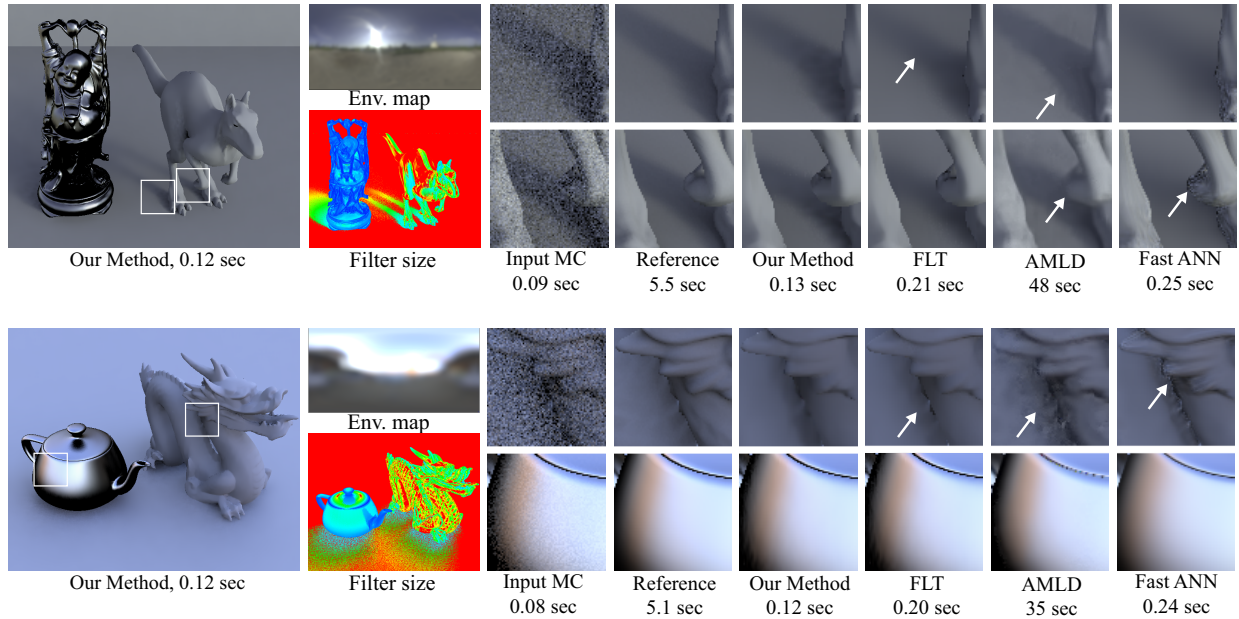


Figure 6: We demonstrate our filtering method on two purely virtual untextured scenes, with a diffuse and a phong (exponent 64) object on a diffuse plane. We show our result with an input of 16 spp (using temporal filtering, Sec. 7), that runs at about 8 fps (resolution 720×720). We also show the environment map used, and warm-to-cool filter size (inverse bandwidth). In the insets, we compare to unfiltered 16 spp MC input, reference with 1024 spp ($60 \times$ slower). We also compare to three other methods with 32 spp input: FLT [DHS*05] which blurs shadows, AMLD [KS13] which is offline and blurs geometric edges, and Fast ANN [TSP14], which is only $2 \times$ slower than ours, but produces artifacts. All methods work well on glossy objects (bottom row insets) since the noise is small.

Comparing to our bandwidth, eqn. 17, the FLT approach ignores the B_e term, which arises from our exact evaluation of the convolution of illumination and visibility, and the ellipsoidal shape of the spectrum. Thus, FLT underestimates the shading bandwidth of high frequency lighting with nearby occluders, resulting in blurring high frequency shading effects when used for filtering.

7. Practical Filtering

We now discuss how the bandwidths derived above can be used to filter noisy irradiances. Before the filtering stage, we run a CUDA kernel to compute per-pixel screen-space curvatures κ_X, κ_Y as

$$\kappa_X(x, y) = \frac{\text{angle}(\mathbf{n}(x, y), \mathbf{n}(x+1, y))}{\|\mathbf{p}(x, y) - \mathbf{p}(x+1, y)\|} \quad (20)$$

where \mathbf{n} is the normal and \mathbf{p} is the world position. The sampling pass stores the minimum occluder distances for direct (we ignore occluders which block samples with intensity below a threshold of 0.1) and indirect illumination. Then, filter bandwidths along each axis, B_X, B_Y are computed – using eqn. 17 for direct illumination and 18 for indirect illumination ($B_X = B_Y$ for indirect). B_e is taken to be the 99% energy bandwidth of the 2D Fourier spectrum of the environment map in lat-long coordinates.

A naive implementation of a 2D Gaussian filter of radius R has $\Theta(R^2)$ complexity. Analogous to previous

axis-aligned filtering papers, the filter is separable into two stages, aligned along the image X and Y axes, reducing the complexity to $\Theta(R)$. We now provide the formulae for the 2-step separated filter. Let $E(x, y)$ denote a raw noisy irradiance value at the pixel (x, y) , and \bar{E} denote the filtered value. Then,

$$E_X(x, y) = \frac{\sum_{|i| < R} w_{xy}(x+i, y) E(x+i, y)}{\sum_{|i| < R} w_{xy}(x+i, y)} \quad (21)$$

E_X denotes the intermediate value resulting from filtering only in the X -direction. The filtered result is given as:

$$\bar{E}(x, y) = \frac{\sum_{|j| < R} w_{xy}(x, y+j) E_X(x, y+j)}{\sum_{|j| < R} w_{xy}(x, y+j)}. \quad (22)$$

The filter kernel is a Gaussian:

$$w_{xy}(x+i, y) = \exp(-2\bar{B}_X^2 \|\mathbf{p}(x, y) - \mathbf{p}(x+i, y)\|^2) \quad (23)$$

Since the bandwidth estimates are also noisy, we use the average square bandwidth of the source and target pixel $\bar{B}_X^2 = 0.5(B_X^2(x, y) + B_X^2(x+i, y))$. Similarly, $w_{xy}(x, y+j)$ uses the bandwidth B_Y .

Temporal filtering: Since we only use 16 samples per pixel, the result from the sampling stage is very noisy, and the filtering can still leave some residual noise in temporal sequences, leading to distracting flickering. Hence, we do temporal filtering where the filter also extends to the previous frame. This scheme is physically accurate, assuming the

illumination does not change (at a given world location) between two consecutive frames – which is a good assumption in most situations except rapid geometry or light source motion. Let E' be the irradiance value from the previous frame, and (x', y') be the pixel in the previous frame with the closest world location to pixel (x, y) , i.e., $\mathbf{p}(x, y) \approx \mathbf{p}'(x', y')$. First, filter along X :

$$E'_X(x, y) = \frac{\sum_{|i| < R} w'_{xy}(x' + i, y') E'(x' + i, y')}{\sum_{|i| < R} w'_{xy}(x' + i, y')} \quad (24)$$

The weights are modified to

$$w'_{xy}(x' + i, y') = \exp(-2\bar{B}_X^2 \|\mathbf{p}(x, y) - \mathbf{p}'(x' + i, y')\|^2) \quad (25)$$

Note that the center of the kernel is offset to (x', y') , unlike eqn. 23. To see why this is important, imagine that there is camera motion between the two frames only along Y . Then, if the filter were to be centered at (x, y) , there may be no pixel $x + i$ where $\|\mathbf{p}(x, y) - \mathbf{p}'(x + i, y)\|$ is small resulting in few or no useful values of E' and artifacts in the final result.

We can now combine the results of eqns. 21 and 24, and filter along the Y -axis to produce the final filtered value:

$$\bar{E}(x, y) = \frac{\left(\sum_{|j| < R} w_{xy}(x, y + j) E_X(x, y + j) + \sum_{|j| < R} w'_{xy}(x', y' + j) E'_X(x', y' + j) \right)}{\sum_{|j| < R} w_{xy}(x, y + j) + w'_{xy}(x', y' + j)}. \quad (26)$$

8. Results

We show four mixed reality scenes with environment map direct and indirect illumination, all rendered at the Kinect camera's VGA (640×480) resolution. Our results include a variety of real-life scenarios augmented with diffuse as well as glossy virtual objects that blend in seamlessly. The accompanying video shows animations and screen captures demonstrating temporal stability. Our images and video are rendered on an Intel Core i7, 3.60GHz desktop with a single NVIDIA Titan GPU, using CUDA v6.5 and OptiX v3.5.

In Fig. 1, we show a simple DESK scene about (1 meter)³ in size. A diffuse Rubik's cube, coffee mug and newspaper are inserted into the real image/video, and they blend in plausibly. The insets show the 4 primary modes of interaction. Direct illumination shadows cast from virtual to real objects are shown in the left-most column 1, and from real to virtual objects are shown in column 2. Similarly indirect illumination color bleeding from virtual to real objects is captured in column 3, and virtual to real color bleeding is shown in column 4. In addition, corresponding insets from the unfiltered, and converged reference images are also shown for comparison. In Fig. 7, we show intermediate steps of our system for the DESK scene, including each of the four irradiances E_R^{dir} , E_R^{ind} , E_{RV}^{dir} , E_{RV}^{ind} , and their corresponding filter sizes, as obtained from our theory.

Figure 8, FURNITURE, shows a larger scene of about (2 meter)³ size consisting of furniture and a plastic mannequin. A diffuse sofa cushion and a wooden table, and a glossy

Scene	SLAM	Optix	SSRT	Filter	Total	FPS
DESK	22	85	44	24	175	5.7
FURNITURE	26	98	44	23	191	5.2
PLAYROOM	26	105	40	23	194	5.2
SHELF	27	110	41	25	203	4.9

Table 1: Detailed timings of our scenes (in milliseconds) rendered at 640 × 480. Our filtering overhead is small compared to the rendering time. We achieve interactive frame rates on a variety of complex scenes.

(phong exponent 64, $B_p = 10$) trashcan are inserted. The insets show the 3 key regions of real-virtual interaction. Corresponding insets from the unfiltered image, and converged reference image are also shown for comparison. All of our scenes are captured in the same large room with a number of area lights (with a different set of lights used for each scene); the environment map for this scene is shown in (a). We used $B_e = 4$.

Figure 9 shows a kid's PLAYROOM scene, of about (2 meter)³ size. We insert a diffuse doll, a toy robot and a stool (on the right) which matches the real stool (on the left). The insets show the parts of each virtual object. Corresponding insets from the unfiltered image, and converged reference image are also shown for comparison.

Figure 10, shows a multi-compartment SHELF scene about 2 meters wide, with some real or virtual objects in each compartment. We insert a diffuse gift box, a diffuse book and a glossy (phong exponent 64) metal bowl. The metal bowl reflects both the environment map and the local geometry, and is not over-blurred. Corresponding insets from the unfiltered image, and converged reference image are also shown for comparison.

Timings: We provide timings for each stage of our system in Table 1. Our overall speed is 5.7 fps for the fastest and 4.9 fps for the slowest scenes. For the SLAM step, compared to [PKC*14], we use a higher number of iterations for the ICP step for greater pose accuracy. We use a fixed voxel size of (3 cm)³; the timings differ by scene due to different voxel grid sizes. The sampling stage is split into an Optix pass and a CUDA screen-space pass, as explained in the supplemental document; each sub-stage takes roughly equal time, and the entire stage accounts for three-quarters of the computational cost. The filtering stage, using a 30 × 30 neighborhood around each pixel, runs under 25 msec (including temporal filtering), and is under 15% of the total cost. To reduce read/write overheads, we store colors with 8 bits per channel and other quantities as half-precision (16 bits) floats. The compositing stage takes negligible time, and is hence not reported.

8.1. Comparisons

Figure 6 compares insets of our result with the result of filtering with the bandwidth from FLT [DHS*05] (see Sec 6.6). Their bandwidth ignores the illumination bandlimit B_e resulting in over-blurred shadows.

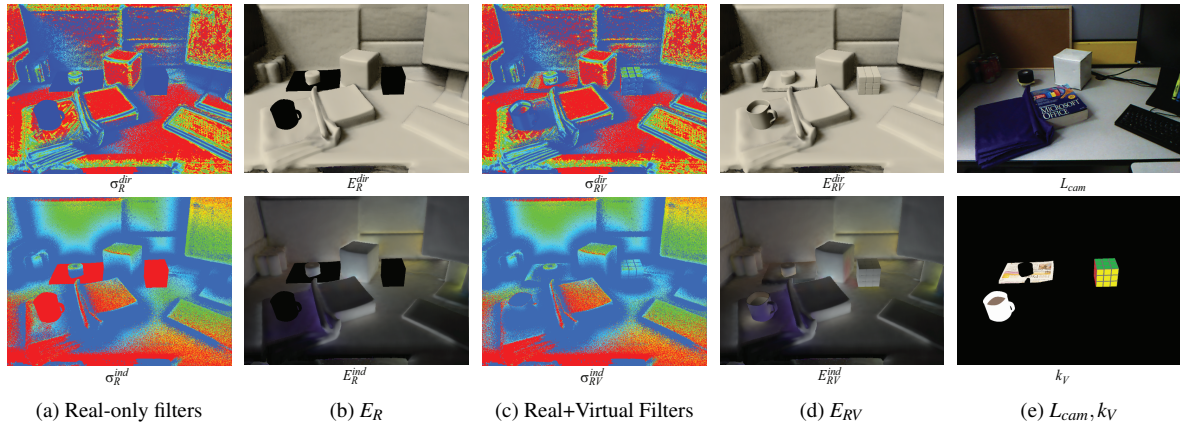


Figure 7: We show intermediate quantities in our filtering algorithm for the DESK scene of Fig. 1. Filter sizes are shown as color-coded standard deviations of the world-space Gaussians, $\sigma = 1/B_x$, using the appropriate bandwidths B_x . The filter sizes in (a) are used to filter the irradiance computed considering only real objects, i.e. $E_R^{\text{dir}}, E_R^{\text{ind}}$ using only real geometry; and the filtered results are shown in (b). Similarly, irradiances considering the full real-virtual geometry, $E_{RV}^{\text{dir}}, E_{RV}^{\text{ind}}$ are filtered using the filters in (c); the results are shown in (d). Observe how addition of virtual objects affects nearby filter sizes due to the introduced occlusions. The final result, computed using eqn. 2, requires two additional quantities – input radiance image L_{cam} , and the virtual object texture k_V – shown in (e).

We also compare to a state-of-the-art Monte Carlo adaptive sampling and multi-level denoising algorithm, AMLD [KS13], and Fast-ANN image denoising [TSPP14], each with 32 spp input. AMLD is offline, with 10 sec filtering overhead, and preserves harder shadow edges but blurs geometric edges slightly since the method is not explicitly aware of geometry. Our method also slightly blurs shadow edges due to our bandlimited light assumption, but runs at real-time (40 ms overhead). Fast-ANN is real-time with 80 ms filtering overhead ($2\times$ of ours), but produces artifacts on smooth untextured images since it relies on finding similar patches using normals.

We do not compare against real-time but approximate MR rendering methods (e.g., Knecht et al. [KTMW12]), since their goal is not to produce physically-accurate renderings. They also use different input (marker-based tracking or pre-defined real meshes) which makes it difficult to produce identical images. Delta voxel cone-tracing [Fra14] uses only point light sources. In general, these methods produce biased images (not physically accurate), or retain noise [KK13b].

8.2. Limitations and Future Work

We describe some limitations of our work that are excellent avenues for future work. Our filtering-based approach assumes that the environment illumination is band-limited, and hence cannot handle high frequency components such as small bright lights. Simply using a large B_e will result in small filter size leaving visible residual noise, while using a large filter size would result in over-blurring. As in Nowrouzezahrai et al. [NGM*11], this can be treated by separating out high frequency illumination into a small set of point lights, then using our approach for the low frequency component.

For our MR system, since we use SLAM with a coarse noisy input depth, the reconstructed geometry is often not

perfectly aligned with the RGB image, which causes artifacts. Although our theory supports dynamic real objects, we do not demonstrate it since our SLAM backend cannot handle moving geometry robustly. All these issues can be mitigated using a more robust SLAM and higher accuracy input depth. We assume that all real surfaces are diffuse, since estimating the true BRDF even with a simple model is difficult at interactive speed. This can be addressed in future work. Further, we do not handle caustics, since our method is based on ray-tracing, and the filtering theory does not treat specular to diffuse light transport.

9. Conclusion

We presented a novel Fourier analysis of environment illumination, which shows that the local spatio-angular shading spectrum is an ellipsoid, and leads to conservative object-space bandwidths. We filter a sparsely sampled noisy Monte-Carlo image to produce a noise-free and accurate result in under 30 ms, with an interesting application to mixed reality. We addressed the problem of ray-tracing both real and virtual geometry using screen-space ray-tracing.

The benefit of our work is two-fold. Our filtering scheme will prove useful in speeding up physically-based rendering of environment lighting. Further, given the rapid advances in mixed and augmented reality mobile technologies such as Google Glass or Oculus Rift, we believe that our work opens up the possibility of interactive mixed reality applications with physically-based shading.

10. Acknowledgements

This work was supported in part by NSF grants 1115242, 1451830, an NVIDIA fellowship, and the UC San Diego Center for Visual Computing.

References

- [ARBJ03] AGARWAL S., RAMAMOORTHY R., BELONGIE S., JENSEN H. W.: Structured importance sampling of environment maps. *ACM Trans. Graph.* 22, 3 (2003), 605–612. 5
- [BSS*12] BAGHER M. M., SOLER C., SUBR K., BELCOUR L., HOLZSCHUCH N.: Interactive rendering of acquired materials on dynamic geometry using bandwidth prediction. In *Proceedings of the ACM SIGGRAPH I3D'12* (2012), pp. 127–134. 3, 7
- [BSS*13] BELCOUR L., SOLER C., SUBR K., HOLZSCHUCH N., DURAND F.: 5D covariance tracing for efficient defocus and motion blur. *ACM Trans. Graph.* 32, 3 (2013), 31:1–31:18. 3
- [CNR08] COSSAIRT O., NAYAR S., RAMAMOORTHY R.: Light field transfer: Global illumination between real and synthetic objects. *ACM Trans. Graph.* 27, 3 (2008), 57:1–57:6. 2
- [CTCS00] CHAI J.-X., TONG X., CHAN S.-C., SHUM H.-Y.: Plenoptic sampling. In *SIGGRAPH '00* (2000), pp. 307–318. 3, 5
- [Deb98] DEBEVEC P.: Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH '98* (1998), pp. 189–198. 2, 3
- [DHS*05] DURAND F., HOLZSCHUCH N., SOLER C., CHAN E., SILLION F. X.: A frequency analysis of light transport. *ACM Trans. Graph.* 24, 3 (2005), 1115–1126. 2, 3, 5, 7, 8, 9
- [DMB*14] DELBRACIO M., MUSÉ P., BUADES A., CHAUVIER J., PHELPS N., MOREL J.-M.: Boosting monte carlo rendering by ray histogram fusion. *ACM Trans. Graph.* 33, 1 (2014), 8:1–8:15. 3
- [EDR11] EGAN K., DURAND F., RAMAMOORTHY R.: Practical filtering for efficient ray-traced directional occlusion. *ACM Trans. Graph.* 30, 6 (2011), 180:1–180:10. 3
- [FGR93] FOURNIER A., GUNAWAN A. S., ROMANZIN C.: Common illumination between real and computer generated scenes. In *Graphics Interface* (May 1993), pp. 254–262. 2
- [Fra14] FRANKE T.: Delta voxel cone tracing. In *ISMAR* (Sept 2014), pp. 39–44. 2, 10
- [GM00] GIBSON S., MURTA A.: Interactive rendering with real-world illumination. In *Eurographics Workshop on Rendering* (2000), pp. 365–376. 2
- [IKH*11] IZADI S., KIM D., HILLIGES O., MOLYNEAUX D., NEWCOMBE R., KOHLI P., SHOTTON J., HODGES S., FREEMAN D., DAVISON A., FITZGIBBON A.: KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Symposium on User Interface Software and Technology* (2011), pp. 559–568. 2
- [KK12] KÁN P., KAUFMANN H.: High-quality reflections, refractions, and caustics in augmented reality and their contribution to visual coherence. In *ISMAR* (2012), pp. 99–108. 2
- [KK13a] KÁN P., KAUFMANN H.: Differential irradiance caching for fast high-quality light transport between virtual and real worlds. In *ISMAR* (2013), pp. 133–141. 2
- [KK13b] KÁN P., KAUFMANN H.: Differential progressive path tracing for high-quality previsualization and relighting in augmented reality. In *ISVC 2013, Part II, LNCS 8034*, Bebis G., (Ed.). 2013, pp. 328–338. 2, 4, 10
- [KS13] KALANTARI N. K., SEN P.: Removing the noise in Monte Carlo rendering with general image denoising algorithms. *Computer Graphics Forum* 32, 2 (2013). 3, 8, 10
- [KTMW12] KNECHT M., TRAXLER C., MATTAUSCH O., WIMMER M.: Reciprocal shading for mixed reality. *Computers and Graphics* 36, 7 (2012), 846–856. 2, 10
- [LWC12] LI T.-M., WU Y.-T., CHUANG Y.-Y.: Sure-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph.* 31, 6 (Nov. 2012), 194:1–194:9. 3
- [MCY14] MOON B., CARR N., YOON S.-E.: Adaptive rendering based on weighted local regression. *ACM Trans. Graph.* 33, 5 (2014), 170:1–170:14. 3
- [MMB97] MARK W. R., MCMILLAN L., BISHOP G.: Post-rendering 3D warping. In *Symp. on Interactive 3D Graph.* (1997), pp. 7–16. 2
- [MMNL14] MARA M., MCGUIRE M., NOWROUZEZAHRAI D., LUEBKE D.: *Fast Global Illumination Approximations on Deep G-Buffers*. Tech. Rep. NVR-2014-001, NVIDIA Corp., 2014. 2
- [MWR12] MEHTA S. U., WANG B., RAMAMOORTHY R.: Axis-aligned filtering for interactive sampled soft shadows. *ACM Trans. Graph.* 31, 6 (2012), 163:1–163:10. 2, 3, 7
- [MWRD13] MEHTA S. U., WANG B., RAMAMOORTHY R., DURAND F.: Axis-aligned filtering for interactive physically-based diffuse indirect lighting. *ACM Trans. Graph.* 32, 4 (2013), 96:1–96:12. 2, 3, 7
- [NGM*11] NOWROUZEZAHRAI D., GEIGER S., MITCHELL K., SUMNER R., JAROSZ W., GROSS M.: Light factorization for mixed-frequency shadows in augmented reality. In *ISMAR* (2011), pp. 173–179. 10
- [NLD11] NEWCOMBE R. A., LOVEGROVE S. J., DAVISON A. J.: DTAM: Dense tracking and mapping in real-time. In *International Conference on Computer Vision (ICCV)* (2011), pp. 2320–2327. 2
- [NZIS13] NIESSNER M., ZOLLHÖFER M., IZADI S., STAMMINGER M.: Real-time 3D reconstruction at scale using voxel hashing. *ACM Trans. Graph.* 32, 6 (2013), 169:1–169:10. 4
- [PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., STICH M.: OptiX: A general purpose ray tracing engine. *ACM Trans. Graph.* 29, 4 (2010), 66:1–66:13. 4
- [PKC*14] PRISACARIU V. A., KÄHLER O., CHENG M. M., REN C. Y., VALENTIN J., TORR P. H. S., REID I. D., MURRAY D. W.: A Framework for the Volumetric Integration of Depth Images. *ArXiv e-prints* (2014). arXiv:1410.0925. 4, 9
- [RH01] RAMAMOORTHY R., HANRAHAN P.: An efficient representation for irradiance environment maps. In *SIGGRAPH '01* (2001), pp. 497–500. 2
- [RMB07] RAMAMOORTHY R., MAHAJAN D., BELHUMEUR P.: A first-order analysis of lighting, shading, and shadows. *ACM Trans. Graph.* 26, 1 (2007). 5
- [SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.* 21, 3 (2002), 527–536. 2
- [SKS11] SOUSA T., KASYAN N., SCHULZ N.: Secrets of CryENGINE 3 graphics technology. *SIGGRAPH Courses* (2011). 2
- [TIS08] TEVS A., IHRKE I., SEIDEL H.-P.: Maximum mipmaps for fast, accurate, and scalable dynamic height field rendering. In *Symp. on Interactive 3D Graph.* (2008), pp. 183–190. 4
- [TSPP14] TSAI Y.-T., STEINBERGER M., PAJAČ D., PULLI K.: Fast ANN for high-quality collaborative filtering. In *High-Performance Graphics* (2014). 3, 8, 10

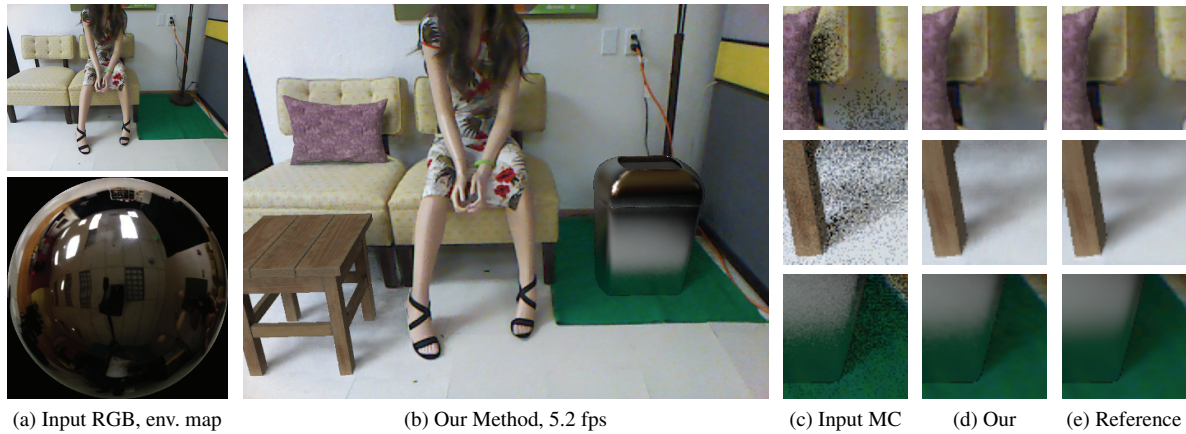


Figure 8: For this FURNITURE scene, the input image is shown in (a) (top), along with the captured environment illumination (bottom). The augmented image with a virtual cushion, wooden table and a glossy trashcan with physically correct illumination, is shown in (b). We compare our result (16 spp, 0.19 sec) with unfiltered Monte Carlo (16 spp), which is very noisy, as well as reference (1024 spp, 12 sec) which is 60× slower.



Figure 9: The PLAYROOM scene with input image and 3D reconstruction are shown in (a). We insert a virtual doll, toy robot, and a stool (the one on the right), with physically correct illumination, as shown in (b). We compare our result (16 spp, 0.19 sec) with unfiltered Monte Carlo (16 spp), which is very noisy, as well as reference (1024 spp, 12 sec) which is 60× slower.

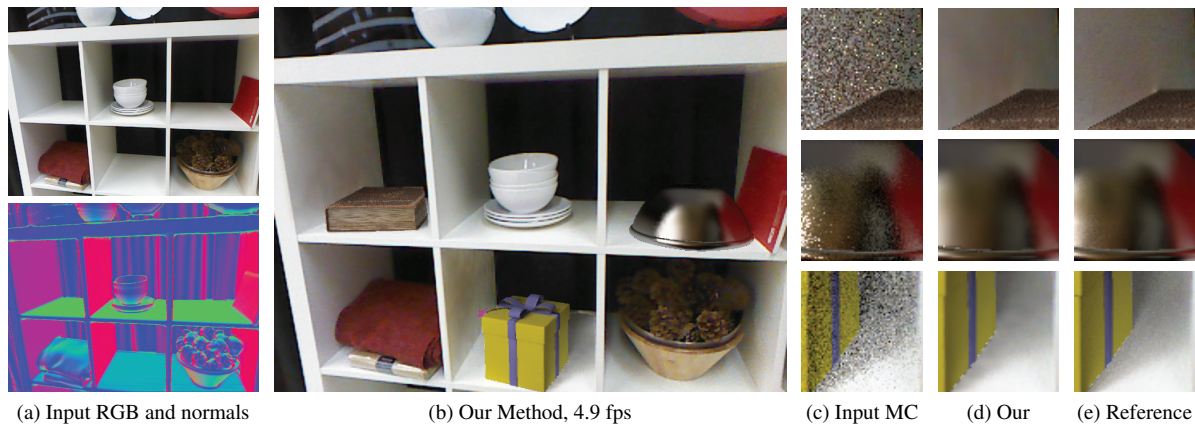


Figure 10: The SHELF scene input image and 3D reconstruction are shown in (a). We insert a diffuse gift-box and a book, and a glossy bowl, as shown in (b). We compare our result (16 spp, 0.20 sec) with unfiltered Monte Carlo (16 spp), which is very noisy, as well as reference (1024 spp, 13 sec) which is 60× slower.