Improved Sampling for Gradient-Domain Metropolis Light Transport

Marco Manzi
University of Bern

Fabrice Rousselle
University of Bern
Disney Research Zurich*

Markus Kettunen
Aalto University

Jaakko Lehtinen
Aalto University
NVIDIA Research

Matthias Zwicker
University of Bern

Coarse image | L_2 (OURS) | Gradients (OURS) | L_2 (GDMLT) | Gradients (GDMLT)

Figure 1: *Our generalized framework for gradient-domain Metropolis rendering includes three techniques to avoid sampling artifacts and reduce variance in sampled gradients. We (*OURS*) achieve visually and numerically improved results compared to previous work (*GDMLT*).*

Abstract

We present a generalized framework for gradient-domain Metropolis rendering, and introduce three techniques to reduce sampling artifacts and variance. The first one is a heuristic weighting strategy that combines several sampling techniques to avoid outliers. The second one is an improved mapping to generate offset paths required for computing gradients. Here we leverage the properties of manifold walks in path space to cancel out singularities. Finally, the third technique introduces generalized screen space gradient kernels. This approach aligns the gradient kernels with image structures such as texture edges and geometric discontinuities to obtain sparser gradients than with the conventional gradient kernel. We implement our framework on top of an existing Metropolis sampler, and we demonstrate significant improvements in visual and numerical quality of our results compared to previous work.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image generation—Display Algorithms;

Keywords: global illumination, metropolis light transport

Links: DL PDF WEB

1 Introduction

Monte Carlo sampling is firmly established as the most practical realistic image synthesis approach because of its flexibility and generality, but variance, which appears as visually distracting noise in the results, is a persistent challenge. In this paper, we build on the gradient-domain Metropolis light transport (GDMLT) algorithm of Lehtinen et al. [2013]. They realized that image space gradients between neighboring pixels, that is, pixel differences, can be sampled with little noise by sampling *pairs of paths* through the corresponding pixels such that they are *close to each other in path space*. Such paths tend to make similar contributions to the image, and hence they contribute small values to the gradient. By combining the estimated gradients with a noisy image using an L_2 Poisson reconstruction step, they obtained unbiased rendering results with significantly lower noise and lower error compared to sampling the image pixels only. Their approach, however, suffers from frequent sampling artifacts and singularities in the gradients, partially canceling their potential benefit. As a consequence, they proposed a reconstruction step using an L_1 error metric to increase robustness towards outliers, at the cost of introducing bias. Our goal is to avoid these issues by developing more robust gradient sampling schemes.

Here, we introduce a generalized framework for GDMLT that follows the same basic procedure as the original algorithm: first, we compute a coarse image and finite-difference gradients in image space using Metropolis sampling, and then reconstruct a higher quality image using a Poisson solver. Our contributions include three novel techniques that reduce sampling artifacts and variance in the sampled gradients, leading to significantly improved performance, both in L_2 (unbiased, see Figure 1) and L_1 reconstruction.

Our first technique provides a way to combine multiple gradient sampling strategies and weight them, akin to multiple importance sampling. In practice, we use a heuristic binary weighting function that weights down gradient samples around singularities and falls back to standard finite differencing there, effectively handling problematic paths more robustly. The two remaining techniques exploit the considerable freedom to determine neighboring paths to obtain a gradient integrand with significantly lower variance. In the second technique, given two pixels defining an image space gradient, we introduce a strategy to determine better *path space neighbors* that lead to more similar path contributions than in Lehtinen et al.'s [2013] original approach. In particular, our method automatically cancels out certain singularities that previously led to visual artifacts. In the third technique, we generalize the notion of image space gradients to include differences between arbitrary pairs of pixels. We show that by selecting pairs of similar pixels, we obtain gradients with smaller magnitudes and therefore less noise. In addition, we show that our technique more effectively preserves image structures during Poisson reconstruction than previous approaches.

*The research for this project was done while Fabrice Rousselle was a student at the University of Bern.

We implement our framework on top of an existing Metropolis sampler using manifold exploration [Jakob and Marschner 2012], and we demonstrate significant improvements in visual and numerical quality of our results compared to previous work. In summary, we make the following contributions:

- We introduce a generalized framework based on gradient-domain Metropolis light transport. It provides more flexibility for sampling gradients, leading to fewer artifacts.
- We describe how to evaluate gradients by computing multiple weighted integrals. We use a heuristic strategy to determine the weighting functions to avoid gradient singularities.
- We introduce a new method to determine path space neighbors when computing gradients between given pixel pairs. We leverage path space manifold walks to cancel out singularities.
- We generalize the concept of gradients to allow for more flexible image space gradient kernels.

2 Related Work

Physically-based Light Transport Physically based light transport algorithms render images by integrating over all possible light paths from a source to an image sensor. In its general form, the rendering equation [Veach and Guibas 1997]

$$I_j = \int_\Omega h_j(x) f^*(x)\,\mathrm{d}\mu(x) = \int_\Omega f_j(x)\,\mathrm{d}\mu(x) \qquad (1)$$

describes the radiance value I_j for each pixel j. The integral is over the space Ω of all light paths of finite length (*path space*), h_j is the pixel filter of the jth pixel, and $f^*(x)$ is the *spectral image contribution function* representing the amount of light reaching the sensor through a given path x in a given wavelength. We will also use the path contribution function $f_j(x) = h_j(x)f^*(x)$, which is the contribution of a path to a specific pixel j. A path x of length k consists of a sequence of vertices $\mathbf{x}_0, \ldots, \mathbf{x}_k$, and $\mathrm{d}\mu(x)$ is the *area product measure* $\prod_{i=0}^{k} \mathrm{d}A(\mathbf{x}_i)$.

Unbiased Monte Carlo rendering algorithms evaluate the pixel integrals with probabilistic methods. Basic Monte Carlo integration with importance sampling exploits that the integral I_j equals the expected value of $f_j(\mathbf{X})/p(\mathbf{X})$, with $\mathbf{X}$ a random variable distributed according to $p(\mathbf{X})$. Path tracers repeatedly draw random paths, evaluate the path contribution, and accumulate the weighted sample f/p, employing multiple importance sampling (MIS) when bidirectional samplers are used [Veach and Guibas 1995].

Adaptive Sampling and Reconstruction A large body of light transport algorithms adaptively sample the image (or the transport integrand), followed by an image reconstruction step. They attempt to direct computation so as to maximize attained image quality per unit of effort expended. Several techniques sparsely sample radiance and its (semi-analytic) gradients [Ward and Heckbert 1992; Dayal et al. 2005; Ramamoorthi et al. 2007], whereas we focus on the finite differences between pixels. Adaptive sampling (and reconstruction) techniques distribute more samples in image locations estimated to need them, and employ various sophisticated filters for reconstructing the final image [Rousselle et al. 2011; Bolin and Meyer 1995; Hachisuka et al. 2008; Overbeck et al. 2009; Egan et al. 2009; Egan et al. 2011]. They produce excellent results, but with no guarantee of unbiasedness.

Smart Filtering Filtering radiance estimates using auxiliary information gleaned from properties of the primary hits, such as normals, world space positions, and materials, is a powerful approach for reducing noise in Monte Carlo renderings [Ward et al. 1988; McCool 1999; Kontkanen et al. 2004; Sen and Darabi 2012; Rousselle et al. 2013]. These techniques make the natural but largely heuristic argument that the illumination solution correlates strongly with local scene features; thus, noisy estimates from "similar" regions can be blended to reduce variance. We build on the same observation, but as a crucial difference to earlier work, we present an unbiased algorithm that merely takes suggestions from such similarities.

Gradient-Domain Image Processing Finite difference gradients form the basis for an immense range of powerful image editing algorithms [Pérez et al. 2003]. We employ similar machinery to determine the image from computed gradients. We generalize standard finite differences, however, to include arbitrary pairs of pixels. This leads to generalized, data dependent Laplacians, similar to the Laplacians used in image segmentation [Shi and Malik 2000] and matting [Levin et al. 2008; Chen et al. 2013]. Recent work by Krishnan et al. [2013] shows how to solve the resulting Poisson problems efficiently. Like previous work, we also use a coarsely sampled primal image to aid reconstruction [Bhat et al. 2010; Lehtinen et al. 2013].

Metropolis Sampling Markov Chain Monte Carlo (MCMC) techniques draw random samples distributed according to functions that are difficult or impossible to sample from directly. In particular, given a target equilibrium distribution $f(x)$ and a *tentative transition function* $\tau(x \to y)$, the Metropolis-Hastings algorithm [Metropolis et al. 1953; Hastings 1970] constructs a Markov chain of samples distributed according to f. Starting with an initial state $\mathbf{x}_0$, it applies, at each step, a carefully chosen random change to the current state $\mathbf{x}_t$ to obtain the next state $\mathbf{x}_{t+1}$. In the limit, the samples will be distributed proportional to the desired target.

Metropolis Light Transport Assuming a converged chain, the samples produced by the Metropolis process can be used for integrating arbitrary (potentially vector-valued) functions. Metropolis Light Transport [Veach and Guibas 1997], short MLT, directly applies the above machinery to Equation 1 by generating a Markov chain of paths distributed according to the scalar luminosity $f(x)$ of their image contribution $f^*(x)$, and evaluating

$$I_j \approx \frac{C}{N} \sum_i \frac{h_j(x_i) f^*(x_i)}{f(x_i)}. \qquad (2)$$

The paths are distributed according to their luminance contribution to the image, and C is the integral of f^* estimated using other means, usually standard Monte-Carlo integration. Veach and Guibas propose several mutation schemes that act on the path itself. To alleviate the difficulty of implementation, Kelemen et al. [2002] introduced *primary sample space* mutations that remove the need to compute transition probabilities due to symmetry; however, some power is lost compared to path space mutations. Jakob and Marschner [2012] introduced a new mutation strategy, manifold exploration, that substantially improves the treatment of specular and highly glossy paths. Our algorithm builds on this approach.

Gradient-domain Metropolis Light Transport [Lehtinen et al. 2013] directly evaluates the horizontal and vertical finite differences $I_{j+1} - I_j$ between neighboring pixels without computing the actual values first. It does so by directly integrating in an extended path space that contains nearby pairs of paths, one through each pixel in question. Feeding the difference estimates and a low-fidelity version of the actual image to a screened Poisson solver [Bhat et al. 2010] then produces the final result. We defer further discussion to Section 3.1, as our novel derivation subsumes theirs.

3 Gradient Domain Rendering Framework

Here we introduce our gradient domain rendering framework, and three techniques to reduce variance in the gradient estimation. We first review the basic idea of computing gradient integrals in Section 3.1. In Section 3.2, we formulate a symmetric expression for these integrals, which is necessary to compute gradients in practice. In Section 3.3, we extend this approach to *multiple weighted* gradient integrals, which allows us to avoid sampling artifacts similarly to multiple importance sampling. In Section 3.4 we introduce an improved piecewise mapping function that leads to higher quality gradients than in previous work. Finally, in Section 3.5, we introduce generalized image space gradient kernels, which further improve the gradient quality. We simplify exposition using scalar radiance, and extension to the usual tristimulus (or spectral) rendering is easy.

3.1 Background

We start from Equation 1, which determines pixel intensities. The core idea in gradient domain rendering is to directly sample gradients, defined as differences between pairs of pixels, in addition to the pixel values themselves. This is beneficial because it is possible to sample the gradients with less variance than pixel intensities. Having sampled gradients and pixel values, we reconstruct the final image by solving a (screened) Poisson equation, where we use the pixel values as an additional constraint. This leads to results with less noise compared to the pixel values themselves.

Let us define a gradient $\Delta_{i,j}$ as the difference between two pixels i and j, where the pixel values I_i and I_j are given by their path space integrals. Hence,

$$\Delta_{i,j} = I_i - I_j = \int_\Omega f_i(x)\mathrm{d}\mu(x) - \int_\Omega f_j(x)\mathrm{d}\mu(x).$$

Instead of evaluating these two integrals separately, in gradient domain rendering we evaluate a gradient by sampling a single integral,

$$\Delta_{i,j} = \int_\Omega \left(f_i(x) - f_j(T_{ij}(x)) \left| \frac{\mathrm{d}T_{ij}}{\mathrm{d}x} \right| \right) \mathrm{d}\mu(x). \qquad (3)$$

Here T_{ij} is a *shift mapping* that deterministically maps a *base path* x to an offset path $\tilde{x} = T_{ij}(x)$. Below we drop the subscript from T_{ij} to reduce clutter. The indices will be clear from the context. The factor $|\mathrm{d}T/\mathrm{d}x|$ denotes the determinant of the Jacobian of $T(x)$ accounting for the change of integration variables for f_j.

A core idea is that we can design T such that $f_i(x) - f_j(\tilde{x})|\mathrm{d}\tilde{x}/\mathrm{d}x|$ generally has less variance than $f_i(x)$. Note that $T(x)$ usually only modifies a few vertices on path x while leaving the rest unchanged. Lehtinen et al. [2013] provide details on how to construct a suitable shift mapping, and we will build on and improve their method.

3.2 Symmetric Gradient Computation

For efficiency reasons, it is useful to sample the integrals for the pixel values I_i and the gradients $\Delta_{i,j}$ using the same probability density. This allows us to reuse a sample of the path contribution $f_i(x)$ for both I_i and $\Delta_{i,j}$. A probability density designed to sample the pixel integral correctly, however, may not sample the gradient correctly. In Metropolis sampling, for example, paths x with zero image contribution $f^*(x) = 0$ are never sampled. Yet, the corresponding offset paths may have a non-zero throughput, that is, $f(T(x)) > 0$, meaning that the sampler may not correctly sample Equation 3. Lehtinen et al. [2013] circumvented the issue by specifically checking for reversibility of the shift mapping, and weighting samples accordingly when local bijectivity was violated.

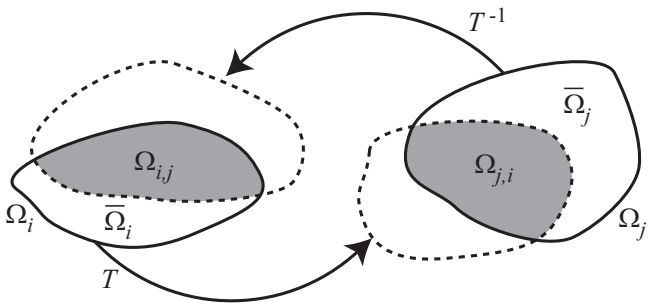


Figure 2: *Notation for the symmetric gradient computation: Ω_i is the region of path space contributing to pixel i. The region $\bar{\Omega}_i$ contributes to pixel i, but cannot be sampled from Ω_j via the inverse mapping. In contrast, Ω_{ij} can be sampled from Ω_j.*

We address this issue more generally by formulating an expression for the gradient Δ_{ij} that is symmetric in i and j. Specifically, we will integrate not only over differences $f_i(x) - f_j(T(x))|\mathrm{d}T/\mathrm{d}x|$ using the forward mapping T, but also $f_i(T^{-1}(x))|\mathrm{d}T^{-1}/\mathrm{d}x| - f_j(x)$ using the inverse mapping T^{-1}.

Let Ω_i be the region of path space that contributes to pixel i, that is $\Omega_i = \{x | h_i(x)f(x) > 0\}$, and similarly Ω_j, illustrated in Figure 2. We will apply the forward mapping only to paths in Ω_i, and the backward mapping to paths in Ω_j. In addition, we define $\bar{\Omega}_j = \Omega_j \setminus T(\Omega_i)$, that is, the paths that contribute to pixel j but that we do not sample using the forward mapping (because the corresponding base path has zero image contribution, $f^*(x) = 0$, or the corresponding base path has $h_i(x) = 0$ and does not contribute to pixel i). Similarly $\bar{\Omega}_i = \Omega_i \setminus T^{-1}(\Omega_j)$ are the paths that contribute to pixel i but that we do not sample using the backward mapping. Finally, $\Omega_{ji} = \Omega_j \setminus \bar{\Omega}_j$ are the paths that contribute to pixel j and that we do sample using the forward mapping, similarly Ω_{ij}, and $T(\Omega_{ij}) = \Omega_{ji}$. This means that we sample the differences between base-offset path pairs in Ω_{ij} and Ω_{ji} twice (using the forward and backward mapping), whereas for paths in $\bar{\Omega}_i$ and $\bar{\Omega}_j$ we sample the differences only once.

Hence, when sampling Ω_i using the forward mapping we compute its contribution Δ^i_{ij} to the gradient Δ_{ij} as

$$\Delta^i_{ij} = \int_{\bar{\Omega}_i} f_i(x)\mathrm{d}\mu(x) + \underbrace{\frac{1}{2}\int_{\Omega_{ij}} f_i(x) - f_j(T(x)) \left| \frac{\mathrm{d}T(x)}{\mathrm{d}x} \right| \mathrm{d}\mu(x)}_{\int_{\Omega_{ij}} f_i(x)\mathrm{d}\mu(x) - \int_{\Omega_{ji}} f_j(x)\mathrm{d}\mu(x)}, \qquad (4)$$

where we exploited $x \in \bar{\Omega}_i \Rightarrow f_j(T(x)) = 0$, therefore we do not need to evaluate this in $\bar{\Omega}_i$, and the factor $1/2$ compensates for duplicate sampling. In practice, we evaluate Δ^i_{ij} using one set of path samples x. We distinguish whether $x \in \bar{\Omega}_i$ or $x \in \Omega_{ij}$ and add a sample of the corresponding term to Δ^i_{ij}. We proceed analogously when sampling Ω_j and compute

$$\Delta^j_{ij} = -\int_{\bar{\Omega}_j} f_j(x)\mathrm{d}\mu(x) + \underbrace{\frac{1}{2}\int_{\Omega_{ji}} f_i(T^{-1}(x)) \left| \frac{T(x)^{-1}}{\mathrm{d}x} \right| - f_j(x)\mathrm{d}\mu(x)}_{\int_{\Omega_{ij}} f_i(x)\mathrm{d}\mu(x) - \int_{\Omega_{ji}} f_j(x)\mathrm{d}\mu(x)}. \qquad (5)$$

The desired gradient is then simply the sum of these two auxiliary values,

$$\begin{aligned}\Delta_{i,j} &= \int_{\bar{\Omega}_i \cup \Omega_{ij}} f_i(x)\mathrm{d}\mu(x) - \int_{\bar{\Omega}_j \cup \Omega_{ji}} f_j(x)\mathrm{d}\mu(x) \\ &= \Delta^i_{ij} + \Delta^j_{ij}.\end{aligned}$$

This strategy also works with partial mappings that may fail and not produce an output path at all for certain inputs, that is, $T(x)$ may be undefined for some x. This may happen for example due to numerical problems. For each x in Ω_i, if $T(x)$ fails, or $T^{-1}(T(x))$ fails, we can simply treat x as not belonging to Ω_{ij}. In addition, if we use an identity mapping for T (i.e., the offset and base paths coincide) the above symmetrized computation reduces to usual finite differencing between pixel intensities, since the Jacobian becomes unity and the identity mapping leads to $\bar{\Omega}_i = \Omega_i$, $\bar{\Omega}_j = \Omega_j$ and $\Omega_{ij} = \Omega_{ji} = \emptyset$.

3.3 Multiple Weighted Gradient Integrals

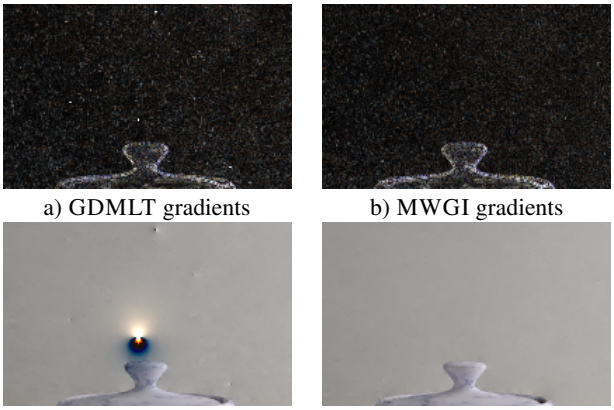
In the standard area product form, the path contribution function $f_i(x)$ consists of the product of the pixel filter, values of the BSDFs at the scattering event vertices, geometry terms, and light emission at the vertex on the source [Veach 1997]. Due to the geometry terms, $f_i(x)$ contains singularities, that is, it takes on infinite values for paths x where several vertices coincide. Besides paths with singularities, however, also paths close to them (paths with very short segments) are ill-behaved because their geometry terms contain divisions by small numbers. This leads to "exploding" path contributions. These issues can be observed in methods using virtual point lights where weak singularities occur in proximity of the cache entries [Kollig and Keller 2006; Walter et al. 2012]. In basic path tracing algorithms, however, this is not a problem — except numerically — because the geometry terms appear in both $f_i(x)$ and the sampling density $p(x)$, and thus cancel out in the sample weight.

Computation of $\Delta_{i,j}$ is not as forgiving. Because we sample both $f_i(x)$ and $f_i(x) - |T(x)/\mathrm{d}x|f_j(T(x))$ with the same density over x (see also Section 3.2), cancellation between $|T(x)/\mathrm{d}x|f_j(T(x))$ and $p(x)$ does not occur like it does with $f_i(x)$ and $p(x)$. Hence, a shift mapping $T(x)$ that moves offset paths closer to (or away from) singularities may lead to large finite differences between the two paths. Note that this is not incorrect as such: it merely means that the integrand that defines $\Delta_{i,j}$ has high variance, and is hence difficult to sample properly.

Our goal in this section is to design a method that detects cases close to singularities, and automatically falls back to a better-behaving sampling strategy when necessary (another approach is to design better shift mappings, cf. Section 3.4). In practice, we switch between two sampling strategies in a binary fashion. We show the validity of this approach by introducing a more general formalism that extends the integral in Equation 3 to multiple weighted integrals using a partition of unity defined by weight functions $w_k(x)$, with $\sum_k w_k(x) = 1, \forall x$. In addition, we apply a different mapping T_k to each weighted integral and obtain

$$\Delta_{i,j} = \sum_k \int_\Omega w_k(x)f_i(x) - w_k(T_k(x))f_j(T_k(x)) \left|\frac{dT_k}{\mathrm{d}x}\right| \mathrm{d}\mu(x).$$

This formulation is similar to multiple importance sampling, except that we determine the weights using different heuristics. The key idea here is that by adjusting the weights $w_k(x)$ locally in path space according to the properties of the mappings $T_k(x)$, we can avoid sampling artifacts by automatically weighting down the sampling scheme close to a singularity. In practice we evaluate the symmetric formulation of the gradients with each mapping, but we omit the explicit formulation, which would be tedious (we simply need to include the weight functions in Equations 4 and 5). Also, we sample all integrals simultaneously with a single set of samples. Given a path sample x, we apply all mappings to it, and then compute a weighted sum of the gradients from all mappings.



a) GDMLT gradients b) MWGI gradients

c) GDMLT reconstruction d) MWGI reconstruction

Figure 3: *Avoiding singularities with multiple weighted gradient integrals (*MWGI*): (a) basic gradients from gradient-domain MLT; (b) using our multiple weighted gradient integrals scheme; (c) and (d) corresponding L2 reconstructions. Note how artifacts in (c) correspond to outliers visible as bright peaks in the gradients.*

We propose a simple approach with two mappings: T_1 is the mapping described by Lehtinen et al. [2013], with important extensions designed to minimize the magnitude of gradients (Section 3.4), and T_0 is the identity mapping. As noted above, the identity mapping is equivalent to computing pixel differences at the end of the Metropolis sampling process. Our goal is to fall back on it when using the shift T_1 is numerically unstable due to singularities. While this approach is guaranteed not to add singularities to the gradient that are not present in the base path, it generally produces gradients with more variance (this is the rationale in using the shift mapping in the first place).

For this purpose we define binary weights

$$w_0(x) = \begin{cases} 1 & \text{if } \max\left(\frac{\|f_i(x)\|}{\|f_j(T_1(x))\|}, \frac{\|(f_j(T_1(x))\|}{\|f_i(x)\|}\right) > t \\ 0 & \text{otherwise,} \end{cases} \quad (6)$$

where t is a user specified threshold, and $w_1(x) = 1 - w_0(x)$. This strategy falls back to using the identity mapping T_0 to compute the gradient if the offset path contribution $f_j(T_1(x))$ relative to the current path contribution $f_i(x)$ is above a threshold t. In Figure 3, we show that this simple strategy effectively reduces artifacts, although it comes with the disadvantage of requiring a user parameter. We leave the development of more sophisticated weighting strategies for future work.

3.4 Improved Mapping to Reduce Variance

Recall that in some geometric configurations, the offset path contribution function contains singularities due to divisions by zero in the geometry terms. In this section, we describe a novel shift mapping that reduces the occurrence of these singularities and variance caused by them compared to previous work [Lehtinen et al. 2013]. The result is a better-behaved integrand for $\Delta_{i,j}$ that is amenable to sampling with less noise.

We start by introducing the necessary notation to describe our approach. Let us denote a path parameterized by its vertices as $x = \{\mathbf{x}_0, \dots \mathbf{x}_n\}$, where $\mathbf{x}_0$ is the eye vertex, $\mathbf{x}_1$ is the primary hit vertex, and $\mathbf{x}_n$ is a vertex on a light source. The screen position of $\mathbf{x}_1$ is $\mathbf{s}_1$. The offset path produced by the mapping T is $\tilde{x} = T(x) = (\tilde{\mathbf{x}}_0, \dots \tilde{\mathbf{x}}_n)$, and $\tilde{\mathbf{s}}_1$ is the screen position of $\tilde{\mathbf{x}}_1$. In addition, let us assume each vertex is classified deterministically either as diffuse or specular based on its BSDF properties. Let $a < b < c$ be the indices of the first three diffuse vertices along a path, where the eye vertex is classified as diffuse by definition, so $a = 0$. Finally, $G(\mathbf{x}_i \leftrightarrow \mathbf{x}_j)$ is the (generalized) geometry term between vertex i and j [Jakob and Marschner 2012].

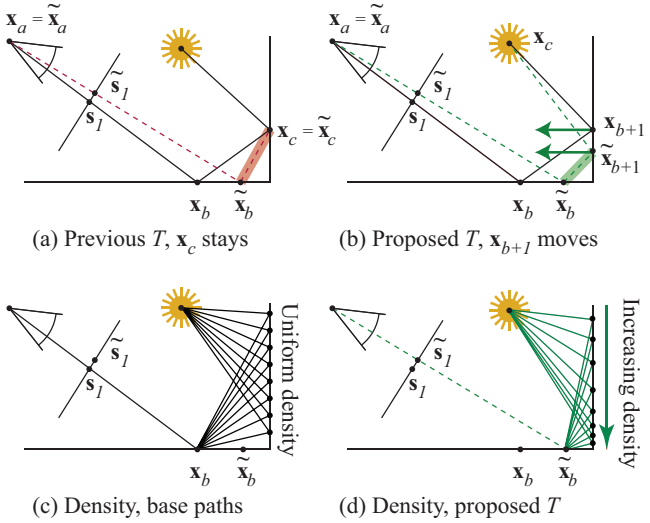
We build on the mapping proposed previously [Lehtinen et al. 2013] to obtain a gradient between pixels i and j. We review the previous approach first before introducing our improvement. The mapping consists of a concatenation of two steps: the first step updates vertices $\mathbf{x}_a, \dots, \mathbf{x}_b$, and the second step updates the rest, that is $\mathbf{x}_i$ with $i > b$. The Jacobian determinant of this concatenation is simply the product of the Jacobian determinants of both steps. In the first step, we calculate $\tilde{\mathbf{x}}_1$ such that $\tilde{\mathbf{s}}_1 - \mathbf{s}_1$ corresponds to the screen space offset between the centers of pixels i and j. Then if $b > 1$, the vertices $\mathbf{x}_i$ with $1 < i \leq b$ are updated by ray tracing to maintain a specular chain between $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}_b$. The second step makes a case distinction based on whether $\mathbf{x}_{b+1}$ is diffuse or specular:

- **Diffuse:** The mapping T leaves all other vertices $\mathbf{x}_i$ with $i > b$ unchanged, that is, the second step is identity and its Jacobian determinant is one. Below we focus on analyzing and improving this case. Also note that in this case $c = b+1$.
- **Specular:** We update the path segment $\mathbf{x}_b, \dots, \mathbf{x}_c$ such as to maintain the specular chain between $\mathbf{x}_b$ and $\mathbf{x}_c$. We achieve this using a manifold walk [Jakob and Marschner 2012].

Analysis We now analyze the offset path contribution $|\mathrm{d}\tilde{x}/\mathrm{d}x| f_j(\tilde{x})$ for the diffuse case. We discuss a common cause for singularities and variance, and then propose an approach to reduce them. Remember that the path contribution function is a product of BSDFs at the scattering vertices, (generalized) geometry terms between vertices, and light emission and importance functions. We observe that after the first step of the mapping, $\tilde{\mathbf{x}}_b$ and $\tilde{\mathbf{x}}_c$ may get arbitrarily close, which may lead to a singularity in the geometry term $G(\tilde{\mathbf{x}}_b \leftrightarrow \tilde{\mathbf{x}}_c)$. We illustrate this schematically in Figure 4(a). Conversely, $\mathbf{x}_b$ and $\mathbf{x}_c$ may be very close in the base path, and move away from each other in the offset path, leading to a much smaller geometry term. Both cases cause variance in the gradients. In addition, the Jacobian of the first step of the mapping does not involve $\tilde{\mathbf{x}}_c$, and the Jacobian of the second step is identity, hence it is impossible that the Jacobian determinants would somehow cancel the problematic geometry term.

We illustrate the effect of singularities on the gradients in Figure 5, where we visualize the gradients next to the geometry terms $G(\tilde{\mathbf{x}}_b \leftrightarrow \tilde{\mathbf{x}}_{b+1})$. In concave regions, very large gradients occur, and outliers in the geometry terms and gradients correlate closely.

Canceling Geometry Terms Our key observation and improvement is that the problems in concave regions can be addressed by modifying the second step of the mapping. Instead of doing nothing in the second step, we treat vertex $b+1$ as specular, update the index c accordingly, and then run a manifold walk on the chain $\{\mathbf{x}_b, \dots, \mathbf{x}_c\}$ while shifting $\mathbf{x}_b$ to $\tilde{\mathbf{x}}_b$, similarly to the specular case (Figure 4(b)). Interestingly, one can show that the problematic geometry term $G(\tilde{\mathbf{x}}_b \leftrightarrow \tilde{\mathbf{x}}_{b+1})$ cancels out with the Jacobian induced by the manifold walk. Intuitively, this is because the manifold walk changes the path densities as shown in Figure 4(c) and (d).



(a) Previous T, $\mathbf{x}_c$ stays (b) Proposed T, $\mathbf{x}_{b+1}$ moves

(c) Density, base paths (d) Density, proposed T

Figure 4: *(a) With Lehtinen et al.'s mapping T, singularities occur if $\tilde{\mathbf{x}}_b$ and $\tilde{\mathbf{x}}_c$ get arbitrarily close (red bar), which happens often in concave regions. (b) Declaring $\mathbf{x}_{b+1}$ specular and mapping it to $\tilde{\mathbf{x}}_{b+1}$ using a manifold walk on the segment $\mathbf{x}_b, \mathbf{x}_{b+1}, \mathbf{x}_c$ to preserve the half vector of $\mathbf{x}_{b+1}$ (green arrows) avoids this problem. Figures (c) and (d) explain this: (c) shows a family of base paths with uniform vertex density on the vertical surface. Because the proposed mapping T preserves the half vectors, it transforms these paths such that the vertex sampling density on the vertical wall increases as the geometry term $G(\tilde{\mathbf{x}}_b \leftrightarrow \tilde{\mathbf{x}}_{b+1})$ approaches a potential singularity in the corner. Mathematically, the change of vertex densities is reflected in the Jacobian of the proposed mapping T, which cancels out the problematic geometry terms.*

For a mathematical explanation of this effect we need some more notation. First, a path segment $x^{bc} := \{\mathbf{x}_b, \dots, \mathbf{x}_c\}$ may be reparameterized in the *projected half vector domain* [Kaplanyan et al. 2014] as $h^{bc} = M(x^{bc}) = \{\mathbf{x}_b, \mathbf{h}^{\perp}_{b+1}, \dots, \mathbf{h}^{\perp}_{c-1}, \mathbf{x}_c\}$, where the $\mathbf{h}^{\perp}_i$ are the half vectors at the vertices projected onto the tangent planes. Now we can express our approach to map the path segment x^{bc} as

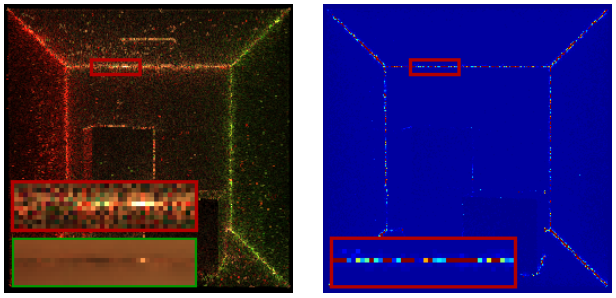
$$\tilde{x}^{bc} = M^{-1}(S(M(x^{bc}))),$$

that is, a reparameterization M into the half vector space, followed by a shift mapping S, and finally mapping back to area parameterization. The shift mapping $\tilde{h}^{bc} = S(h^{bc})$ simply moves vertex $\mathbf{x}_b$ to $\tilde{\mathbf{x}}_b$, where $\tilde{\mathbf{x}}_b$ is obtained in the first step described above. The key is that S operates in the half vector parameterization, and it does only shift the starting vertex $\mathbf{x}_b$, but it keeps the half vectors constant. Hence our procedure can be implemented using a manifold walk, which is designed to preserve half vectors while moving a single vertex position in a path.

Let $f(x^{bc})$ be the factors of the image contribution function of $f(x)$ that include only the vertices i with $b \leq i \leq c$ (we use the image contribution function f, since the pixel filter is not involved). We can now write the corresponding contribution $f(\tilde{x}^{bc})$ after our mapping,

$$f(M^{-1}(\underbrace{S(M(x^{bc}))}_{\tilde{h}^{bc}})) \left|\frac{\mathrm{d}M(x^{bc})}{\mathrm{d}x^{bc}}\right| \left|\frac{\mathrm{d}S(h^{bc})}{\mathrm{d}h^{bc}}\right| \left|\frac{\mathrm{d}M^{-1}(\tilde{h}^{bc})}{\mathrm{d}\tilde{h}^{bc}}\right|$$

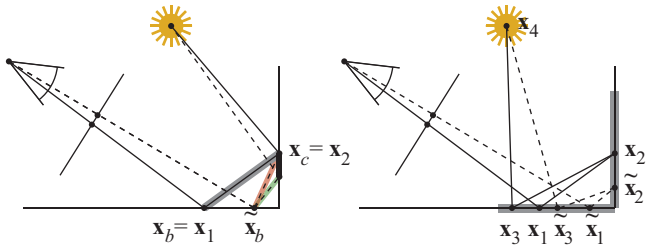
$$= g(\tilde{h}^{bc}) \left|\frac{\mathrm{d}M(x^{bc})}{\mathrm{d}x^{bc}}\right| \left|\frac{\mathrm{d}S(h^{bc})}{\mathrm{d}h^{bc}}\right|,$$



a) No manifold walk b) Geometry terms

c) One step IncMW d) IncMW

Figure 5: *(a) Gradients without our technique; (b) max-ratio of geometry terms $G(\tilde{\mathbf{x}}_b \leftrightarrow \tilde{\mathbf{x}}_{b+1})$ over $G(\mathbf{x}_b \leftrightarrow \mathbf{x}_{b+1})$ (or vice-vera; blue is 1, red is high); (c) gradients where only vertex $b+1$ may be classified as specular, that is, one step of our incremental approach (one step* IncMW*); (d) gradients with the incremental approach (*IncMW*). We visualize gradient magnitudes and scale them for better visibility. The green close-ups show the result of the L1 reconstruction in the close-up region. All images used structure-adaptive gradients (Section 3.5).*



(a) Distance criterion (b) Ambient occlusion criterion

Figure 6: *(a) Assume we determine whether $\mathbf{x}_2$ should be declared specular using a threshold on the distance to $\mathbf{x}_1$. The threshold distance is indicated by the thick gray line. For a path on the decision boundary, the two possible offset paths (dotted lines) on either side of the boundary (red bar: identity mapping, green bar: manifold walk) leave a gap in path space, indicated by the black bar below $\mathbf{x}_2$. (b) The gray bars indicate regions where vertices are declared specular because of high ambient occlusion. The incremental manifold walk first generates $\tilde{\mathbf{x}}_2$ by applying a manifold walk on $\mathbf{x}_1, \mathbf{h}_2, \mathbf{x}_3$ while shifting $\mathbf{x}_1$ to $\tilde{\mathbf{x}}_1$. Then it generates $\tilde{\mathbf{x}}_3$ by applying a manifold walk on $\mathbf{x}_2, \mathbf{h}_3, \mathbf{x}_4$ while shifting $\mathbf{x}_2$ to $\tilde{\mathbf{x}}_2$.*

where the Jacobians account for the change in integration variables. In addition, $g(\tilde{h}^{bc})$ can be interpreted as the image contribution function in the half vector parameterization. Kaplanyan et al. [2014] showed that in $g(\tilde{h}^{bc})$ all geometry terms are canceled by the Jacobian of M^{-1}, except for the geometry term $G(\tilde{\mathbf{x}}_{c-1} \leftrightarrow \tilde{\mathbf{x}}_c)$, where $\tilde{x}^{bc} = M^{-1}(\tilde{h}^{bc})$. In addition, since S is a shift by a constant its Jacobian is identity. Also the Jacobian of M is given by the base path but independent of the offset path. Intuitively, the cancellation of the geometry terms by the Jacobian corresponds to the change of densities in Figure 4(d). Hence our approach avoids the potential singularity produced by $G(\tilde{\mathbf{x}}_b \leftrightarrow \tilde{\mathbf{x}}_{b+1})$, and the benefits of this approach can be seen in Figure 5(c).

Incremental Approach An intuitive idea to avoid the remaining singularities from $G(\tilde{\mathbf{x}}_{c-1} \leftrightarrow \tilde{\mathbf{x}}_c)$ would be to declare further vertices as specular and increase the index c, until $\mathbf{x}_c$ is sufficiently far from $\mathbf{x}_{c-1}$. Unfortunately, this leads to "gaps" in the output of the mapping from x^{bc} to $\tilde{x}^{bc}$, as illustrated in Figure 6(a), causing bias in the resulting images.

We avoid this problem using a different heuristic, shown in Figure 6(b), where the key is that we decide for each vertex $\mathbf{x}_i$, one-by-one, in ascending order, whether it should be declared specular, in such a way that the decision does not depend on $\mathbf{x}_i$ itself. We observe that singularities tend to occur in paths where subsequent vertices are close to each other. Intuitively, such path configurations are common in regions of high ambient occlusion. If $\mathbf{x}_{i-1}$ lies in a region with high ambient occlusion, chances are high that $\mathbf{x}_i$ is close-by. In this case, we are likely close to a singularity. In practice, we test the ambient occlusion factor at $\mathbf{x}_{i-1}$, and if it is above a threshold, then $\mathbf{x}_i$ is considered specular. If $\mathbf{x}_i$ is declared specular, we map the vertex position $\mathbf{x}_i$ to $\tilde{\mathbf{x}}_i$ using a manifold walk on $\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}$, where the shift of $\mathbf{x}_{i-1}$ is given by the difference to $\tilde{\mathbf{x}}_{i-1}$, which was obtained in the previous step. Since this approach amounts to a concatenation of mappings via manifold walks, the overall Jacobian determinant is simply the product of the Jacobian determinants of each step. In addition, since each step is a valid bijective mapping, this also applies to the concatenation of the mappings. We illustrate the benefits of this approach in Figure 5(d).

3.5 Structure-Adaptive Gradient Kernels

A basic intuition why gradient domain rendering reduces noise is that high-dimensional path contribution functions of neighboring pixels are often very similar, and one can exploit this similarity using suitable mapping functions in path space. Such mappings compute gradients as differences between similar base and offset paths, which are in general much smaller than the variance of the path contributions of either pixel. Hence the sampled gradients have less variance than the sampled path contributions. Our key observation is that we may compute gradients between arbitrary pairs of pixels, and select pairs such that their *path contribution functions are as similar as possible*. We find pixel pairs that are usually more similar than pairs among the 4-connected neighborhood of usual finite difference gradients. Therefore, the differences between base and offset paths become even smaller, and we further reduce variance.

Motivated by this, we define *structure-adaptive gradients* at each pixel as the pairwise differences to the n most similar pixels in a small window, as opposed to standard gradients consisting of pixel differences between horizontally and vertically adjacent pixels. We describe an affinity function used to determine similarity below. Given the similarity relations between pixels, we then define a generalized Poisson reconstruction problem.

More precisely, assume that the gradient Δ_{ij} occurs in the structure-adaptive kernel. Then we add a constraint $\tilde{I}_i - \tilde{I}_j = \Delta_{ij}$ to our equation system, where $\tilde{I}_i$ and $\tilde{I}_j$ are the unknown pixel values we would like to reconstruct. Like previous work, we also use the noisy "primal" image obtained as a by-product of gradient sampling as an additional constraint, and obtain an overconstrained equation system even if the structure-adaptive gradient matrix is

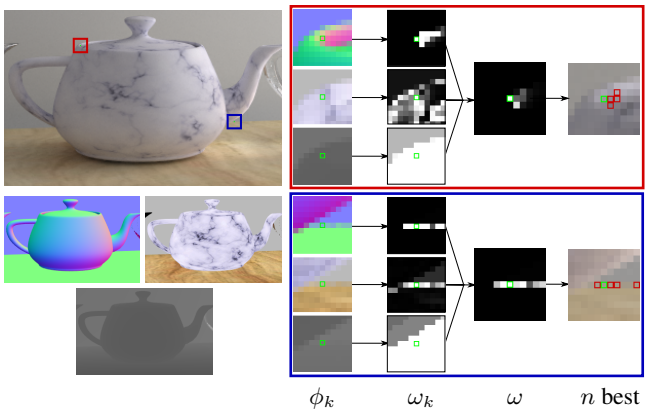


Figure 7: *Structure adaptive gradients are defined using the most similar neighbors. First, we compute bilateral weights ω_k for each feature ϕ_k in a box shaped neighborhood (brighter pixels mean bigger weights). For every pixel in the neighborhood we denote the weight of the most restrictive feature as ω. Finally, we select the n neighbors with the n biggest ω weights (here $n = 4$).*

arbitrarily rank deficient. This corresponds to the reconstruction used by Lehtinen et al. [2013], except with differences computed between adaptively-determined neighbors.

Pixel affinity We estimate similarities of path contribution functions by leveraging auxiliary per-pixel features (Figure 7). Features such as normals, depth, texture, and so on, serve this purpose well, since path contribution functions across feature discontinuities tend to be quite different. Indeed, using features for cross-bilateral filtering is highly successful for denoising Monte Carlo renderings [Ward et al. 1988; Dammertz et al. 2010]. Hence, we estimate the similarity between pixels i and j as $\omega(i,j) = \min_k[\omega_k]$ with

$$\omega_k = \exp\left(-\frac{(\max(0, |\phi_k(i) - \phi_k(j)|^2 - \tau_k)}{\sigma_k^2}\right) \qquad (7)$$

where $\phi_k(i)$ is the kth feature at pixel i, τ_k a user-defined threshold and σ_k a bandwidth parameter for feature k. In other words, we define $\omega(i,j)$ similarly as a cross-bilateral filter weight using the feature that is least similar between i and j. We compute $\omega(i,j)$ for all pixels around i in a box shaped neighborhood. We define our n structure-adaptive gradients at each pixel as the differences to the n pixels with the largest weights in the neighborhood.

Since pixel differences are signed quantities, we should select one of the two possible orderings for each pixel pair defining a gradient. We choose to avoid this issue as follows to simplify implementation: Let ij be the ordered pair of pixels i and j. In addition, let P be the set of all pairs generated by our data adaptive procedure above, where i is always the center pixel and j a neighbor. Note that $ij \in P$ does not imply $ji \in P$. Now we distinguish two cases:

- **Non-Symmetric Neighbors:** If only $ij \in P$ but $ji \notin P$, we include the constraint $\tilde{I}_i - \tilde{I}_j = \Delta_{ij}$ in the equation system. We sum Δ^i_{ij} and Δ^j_{ij} to Δ_{ij} as described in Section 3.2.
- **Symmetric Neighbors:** If both $ij \in P$ and $ji \in P$, however, we add two separate constraints $\tilde{I}_i - \tilde{I}_j = 2\Delta^i_{ij}$, and $\tilde{I}_j - \tilde{I}_i = 2\Delta^j_{ji}$. This works because the second constraint is equivalent to $\tilde{I}_i - \tilde{I}_j = 2\Delta^j_{ij}$ (note $\Delta^j_{ij} = -\Delta^j_{ji}$), hence the average of these constraints represents the desired gradient.

4 Bilateral-Domain Metropolis Light Transport

In this section we present an MLT algorithm that implements the ideas presented in the previous chapter. We call the algorithm *Bilateral-Domain Metropolis Light Transport*. As per common practice, we apply the algorithm only to indirect illumination, and handle direct illumination separately in a prior pass using a simple recursive ray tracer. Our approach is currently limited to box functions as pixel filters.

Structure-Adaptive Gradients To compute the neighbors for structure-adaptive gradients, we gather pixel features in the direct illumination pass. We store features such as normals, depths and textures in auxiliary images, assuming that the features contain only insignificant residual noise. We achieve this by using sufficiently many samples in the direct illumination pass, or by denoising the features images [Rousselle et al. 2013].

Ambient Occlusion for Incremental Manifold Walk We guide our incremental manifold walk (Section 3.4) using screen space ambient occlusion coefficients, which we compute together with the feature images in the direct illumination pass. We define a threshold on the length of secondary ray segments, and for each pixel compute its ambient occlusion coefficient as the percentage of secondary rays whose lengths are below this threshold. Since our gradients are defined in screen space, we also define the threshold length in terms of its projection to screen space. As a consequence, the world space threshold increases for primary hit points further away from the camera. Following the iterative scheme from Section 3.4, we decide whether a vertex $\mathbf{x}_i$ should be perturbed using a manifold walk based on the ambient occlusion value at vertex $\mathbf{x}_{i-1}$. Since we use a screen space ambient occlusion map, we project the position of $\mathbf{x}_{i-1}$ to screen space and read the corresponding value. We perform a depth test to check whether $\mathbf{x}_{i-1}$ is occluded from the camera, and only use the ambient occlusion value if it is not occluded. While this approach is limited to path vertices that are visible to the camera, it is still effective at removing the most prominent artifacts due to near singular geometry terms in concave regions visible in the image. In addition, we suppress occasional remaining singularities with our weighting scheme from Section 3.3.

Gradient Sampling Here we describe in more detail how we sample the symmetric gradient integrals from Section 3.2 using a Metropolis sampler. For this we define an extended path space Ω'_i for each pixel,

$$\Omega'_i = \Omega_i \times N_i,$$

where N_i is a set of neighbor indices, such that for all neighbors $k \in N_i$, either k is one of the closest neighbors of i, that is, $ik \in P$, or vice versa i is one of the closest neighbors of k, that is, $ki \in P$. Hence, $|N_i|$ is, in general, different for each pixel i. The Metropolis algorithm samples the union Ω' of all per-pixel extended path spaces, $\Omega' = \bigcup \Omega'_i$. Denote a path in Ω' by z, consisting of a usual path x and a neighbor index j. Since we use box filters, x is uniquely assigned to a pixel i, and together with the neighbor index this specifies a gradient contribution Δ^i_{ij}.

In the Metropolis sampler we use conventional mutators to propose new paths x. Assuming x belongs to pixel i, we then complete the extended path z by proposing one neighbor $j \in N_i$ randomly. Finally we evaluate the image contribution $f^*(x)$ and the gradient contribution Δ^i_{ij}. Since this means we are counting the image contribution $|N_i|$ times more often than each gradient, we multiply the gradients by $|N_i|$. This procedure also assures that we sample any

Algorithm 1: SAMPLE CONSTRAINT

Input: Extended path z, consisting of base path x in pixel i and a neighbor index j.

begin
  /* Sum over mappings T_k and weight functions w_k. */
  for $k \in 0, 1$ **do**
    /* Sample gradient contribution Δ^i_{ij}, Equation 4. */
    /* Sample called s. */
    if $x \in \Omega_{ij}$ **then**
      $\hat{x} = T_k(x)$
      $s = 1/2(w_k(x)f_i(x) - w_k(\hat{x})f_j(\hat{x})|d\hat{x}/dx|$
    else
      $s = w_k(x)f_i(x)$
    /* Multiply with $|N_i|$, since we sample only one gradient for each base path. */
    $s = |N_i|s$
    /* Check neighbor symmetry, Section 3.5. */
    if $ij \notin P$ **then**
      /* Non-symmetric, constraint $\tilde{I}_j - \tilde{I}_i = \Delta_{ji}$ */
      $b(r(j,i)) = b(r(j,i)) - s$
    else if $ji \notin P$ **then**
      /* Non-symmetric, constraint $\tilde{I}_i - \tilde{I}_j = \Delta_{ij}$ */
      $b(r(i,j)) = b(r(i,j)) + s$
    else
      /* Symmetric, constraint $\tilde{I}_i - \tilde{I}_j = 2\Delta^i_{ij}$ */
      $b(r(i,j)) = 2s$

gradient Δ_{ij} symmetrically, because both $j \in N_i$ and $i \in N_j$. Finally, we use a target function $f(z)$ similar to Lehtinen et al. [2013],

$$f(z) = |N_i|\|\Delta^i_{ij}(x)\| + \alpha\|f^*(x)\|,$$

where the extended path z implies the indices i and j. Again, we include the factor $|N_i|$ because gradients are sampled $|N_i|$-times less often than the image contribution, hence their weight should be emphasized by the same factor in the target function.

Assembling the Gradient Constraints Algorithm 1 summarizes how we construct the gradient constraints from the sampled gradient contributions. We represent the gradient constraints as $H\tilde{I} = b$, where $\tilde{I}$ is the unknown image we will reconstruct, H has rows that are zero except for two entries with values -1 and 1 that encode the pixel pair involved in a gradient. The vector b stores the corresponding gradient values. The function $r(i,j)$ returns the row index where constraint $\tilde{I}_i - \tilde{I}_j$ is stored in the equation system. The pseudo-code takes as input an extended path z obtained from the Metropolis sampler, and it stores the sampled gradient contribution Δ^i_{ij} in the gradient constraints as described in Section 3.5. It computes a weighted sum over all mappings as described in Section 3.3, although in practice we implement this more efficiently by exploiting that one of our two mappings is the identity.

Poisson Reconstruction We reconstruct an image that best fits the gradient constraints and the coarse image that we sampled during rendering by solving a Poisson problem,

$$\min_{\tilde{I}} ||H\tilde{I} - b||^2 + ||\alpha(\tilde{I} - I_g)||^2_2, \tag{8}$$

where $\tilde{I}$ is the reconstructed image and I_g is the image obtained by sampling the path contributions. This approach leads to an unbiased estimate $\tilde{I}$ of the true image if we use the L_2-norm as above. For a proof we refer to Lehtinen et al.'s work [2013]. We can also obtain visually improved results by minimizing the L_1-norm with an iteratively reweighted least-squares solver, although this introduces bias.

5 Results

In this section we report on results and comparisons obtained with our approach, which we implemented on top of the Mitsuba renderer [Jakob and Marschner 2012].

Parameter Settings We set the threshold t for the binary weight ω_0 in Equation 6 (Section 3.3) to 20 for all scenes, which conservatively suppresses outliers. This is important since a more aggressive, lower threshold will fall back to the identity mapping too often and reintroduce noise in the gradients. We compute screen space ambient occlusion coefficients (Section 3.4) using 0.75% of the image size as the threshold for the length of secondary path segments projected to image space. The threshold on the ambient occlusion coefficient that triggers the incremental manifold walk is 0.95, so that weakly concavely curved surfaces do not always lead to an extended manifold walk.

For the structure-adaptive gradients (Section 3.5) we use $\sigma_k = 0.01$ for all features, since the selection is very insensitive towards this parameter. The parameter τ_k is critical to make sure our adaptive gradients are not overly sensitive to the features. In particular, we need to avoid having them degenerate to 1D gradients along 1D contours in the features. Empirically we found the values $\tau_{normal} = 0.02$, $\tau_{texture} = 0.001$ and $\tau_{depth} = 0.01$ to work well for features normalized to range $[0, 1]$ in all tested scenes. In addition we use a bilateral window of 5×5 pixels to compute affinities $\omega(i,j)$ and $n = 4$ as the number of structure-adaptive gradients in all our experiments.

Benefits of Each Technique Figure 8 shows how each of our suggested strategies improves quality of the L_2 reconstruction in certain regions of the DOOR scene. We compare the rMSE (relative mean squared error) and the per-pixel gradient energy of the close-up regions. The per-pixel gradient energy is defined as the sum of squared values of all gradient constraints on each pixel, that is, $\|H^T b\|^2_2$. Using structure-adaptive gradients (ADAP, Section 3.5) instead of ordinary gradients strongly reduces ringing artifacts along edges as can be seen in the second row of the red and green close-ups. Since our adaptive gradients are based on features from the direct illumination pass only, they do not necessarily minimize gradient energy in regions where indirect illumination effects like caustics or indirect shadows occur. Therefore, the quality does not improve in such regions compared to using ordinary gradients, as can be seen in the second row of the blue close-up. Adding our multiple weighted gradient integrals (MWGI, Section 3.3) to avoid singularities helps removing bullet hole artifacts. This improves the quality of regions where structure-adaptive gradients alone are useless as can be seen in the third row of the blue close-up. Since avoiding singularities means using the noisier but less singularity-prone identity mapping T_0, regions where a lot of singularities occur still tend to become noisier. This mostly happens around concave geometry edges, as can be seen in row three of the green close-up. Adding the ambient occlusion guided incremental manifold walk (INCMW, Section 3.4) greatly reduces the need to fall back to T_0 in those regions, which effectively reduces noise there (bottom row of the green close-up).

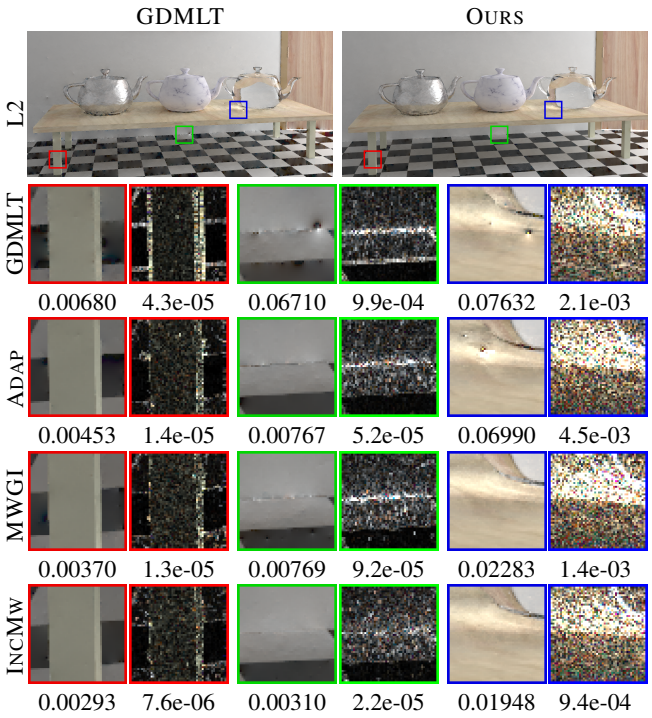


Figure 8: *We incrementally add our techniques on top of* GDMLT *in top to bottom order: unmodified* GDMLT*, structure adaptive gradients (*ADAP*, Section 3.5), multiple weighted gradient integrals (*MWGI*, Section 3.3) and incremental manifold walk (*INCMW*, Section 3.4). For every crop we show the L_2 reconstruction with the relative MSE and the energy of the gradients.*

Comparison to Previous Work In Figure 13 we demonstrate the effectiveness of our approach by comparing it to gradient-domain Metropolis (GDMLT) [Lehtinen et al. 2013] and manifold exploration MLT (MEMLT) [Jakob and Marschner 2012]. We computed reference images of the DOOR, SIBENIK and BOX scenes with MEMLT with 32000 mutations per sample, and for SPONZA and BIDIR we used bidirectional path tracing with 32000 samples per pixel. For MEMLT we used the default parameters of Mitsuba, and for GDMLT we used the parameters suggested by Lehtinen et al. [2013]. We adjusted the set of path mutators, the maximum path length, and the length of the Markov chains for every scene separately. For each scene we used the same parameters for all compared methods. We applied GDMLT and our method on the indirect illumination only. We computed direct illumination separately and then simply added it to the result of the reconstruction. Our approach has less than 5% computational overhead compared to GDMLT.

We compare the relative mean square error (rMSE) and observe an improvement of our method over GDMLT of 20%-60% using L_2 reconstruction, and of 5%-40% using L_1. We also measured how many mutations per pixel were needed for MEMLT to achieve similar rMSE as the L_1 reconstruction of our method. Results suggest that the required number of mutation per pixel is highly scene dependent. While for SIBENIK only two and a half times more mutations achieve similar rMSE, for BIDIR we required 20 times more mutations.

In general we observed that the L_2 reconstruction of GDMLT often performs poorly because the sampling tends to get stuck due to gradient outliers. Huge gradient values may appear even in smooth

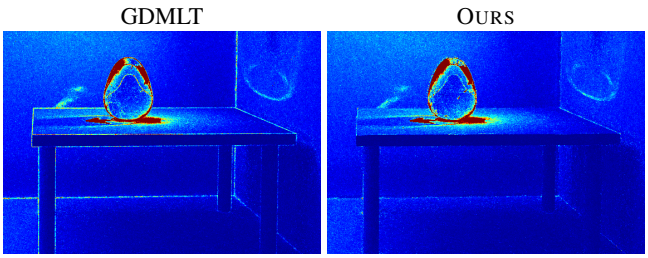


Figure 9: *A comparison of the sampling densities of our method and GDMLT. Note how our method distributes less samples along edges that are avoided by our structure-adaptive gradients.*

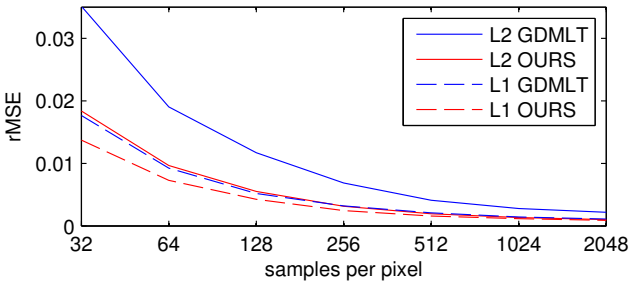


Figure 10: *The rMSE for* DOOR *using our method compared to* GDMLT *over increasing numbers of mutations per pixel.*

image regions due to geometrical singularities in the offset paths, which are caused by specular objects in the scene. The Poisson reconstruction then attempts to reconcile these gradient outliers with the coarse sampled image I^g. This leads to visually and numerically prominent bullet hole artifacts. The L_1 reconstruction of GDMLT suffers less from these problems, since entries in the linear equation system leading to big errors are weighted down. Yet some artifacts remain, especially when they occur near edges (see L_1 reconstruction of GDMLT in green close-up of DOOR scene). In all scenes our algorithm suffers less from singularity artifacts than GDMLT, as can be seen when comparing L_2 reconstructions of our approach and GDMLT.

In Figure 9 we compare the sampling densities of our approach to GDMLT in the BIDIR scene using approximately 256 mutations per pixel. One interesting observation is that our method concentrates samples exclusively in those regions where gradients occur that are not detectable in the features we used to generate the structure-adaptive gradients. GDMLT on the other hand distributes samples along all gradients of the image. This can be seen at the edges belonging to the table. This is a desirable property of our approach since it means that most samples are concentrated around regions that require more effort to be rendered correctly (e.g., caustics, indirect shadows and so on).

Figure 10 plots the rMSE of GDMLT and our method for L_2 and L_1 reconstructions against increasing numbers of mutations per pixel using the DOOR scene. The plotted error values are the averages over 25 runs. L_2 OURS is consistently better than L_2 GDMLT. The difference between L_1 GDMLT and L_1 OURS is smaller but still significant. Notably L_2 OURS has similar rMSE values as L_1 GDMLT, meaning our unbiased L_2 reconstruction is of similar quality as the biased L_1 reconstruction of GDMLT.

Limitations Our binary weighting scheme cannot avoid artifacts due to singularities in the base path. Considering Equation 6, we

observe that a singularity in $T_1(x)$ but not in x will likely lead to a big max-ratio and therefore we fall back to sampling strategy T_0. This means the singularity in $T_1(x)$ will not affect the final result. If x instead of $T_1(x)$ is close to a singularity, however, our approach is ineffective. Hence, like in other MLT methods our Markov chain may sometimes get stuck. Artifacts arising from this are isolated bright pixels, but they do not cause bullet holes as do outliers in the gradients.

The global parameters τ_k and t of our method can lead to locally suboptimal results in complex scenes: τ_k leads to a global trade-off between sensitivity towards a feature and avoidance of degenerate one-dimensional constraints, whereas t leads to a trade-off between bullet holes and more noise due to the fall-back to strategy T_0.

Like all MLT methods our method suffers from potentially incomplete coverage of path space, as can be seen in the missing highlight on the glass-egg in the BIDIR scene (Figure 13).

Analysis of Adaptive Gradients The Poisson reconstruction problem in Equation 8 is equivalent to solving

$$(H^T H + \alpha \mathrm{Id})\tilde{I} = \alpha I^g + H^T b, \tag{9}$$

where Id is the identity matrix, and the matrix $H^T H$ can be interpreted as a Laplacian matrix given by the gradient kernels. In Figure 11 we compare the eigenvectors of $H^T H$ of GDMLT and our approach. We visualize only a subset of all eigenvectors, since there are as many eigenvectors as pixels in the image. The Laplacian of the GDMLT kernel is the usual discrete Laplacian, and its eigenvectors are 2D sinusoidal functions corresponding to the discrete Fourier transform. The eigenvectors of our kernels, however, reflect and preserve the structures and edges of the feature images that we used to construct the kernels, even for eigenvectors corresponding to low eigenvalues (that is, low frequencies). Intuitively, the Poisson solver reconciles contradicting information in the coarse image and the gradients by suppressing high frequencies (see also the analysis by Lehtinen et al. [2013]). While suppressing high frequencies in the Fourier transform is prone to ringing artifacts, suppressing high frequency eigenvectors in our approach does not suffer from this problem, since our low frequencies still contain image edges.

In Figure 12 we analyze the contribution of the coarse image I^g and the gradients $H^T b$ to the solution of Equation 9 by simply setting the other part of the right hand side ($H^t b$ and I^g, respectively) to zero. A comparison of our approach to the conventional gradient kernel reveals how the image structure is built into our structure-adaptive kernels. Even if we force gradients to be zero and give little weight to errors with respect to the coarse image (low α values, top row) we still preserve sharp edges. In comparison, conventional kernels behave like low-pass filters in this setting.

6 Conclusions

In this paper we introduced a generalized framework for gradient domain Metropolis rendering, which we exploited to develop three techniques to avoid singularities and reduce noise in sampled gradients. A common insight that we explore in all techniques is that there is considerable freedom in how to determine gradients by computing offset paths in path space.

The first technique introduces the idea of applying several different shift mappings simultaneously to generate the offset paths. We show that we can weigh each mapping, akin to multiple importance sampling, to suppress the contribution of mappings that yield outliers and high variance. The second technique is an improved mapping function that avoids certain singularities that lead to artifacts in

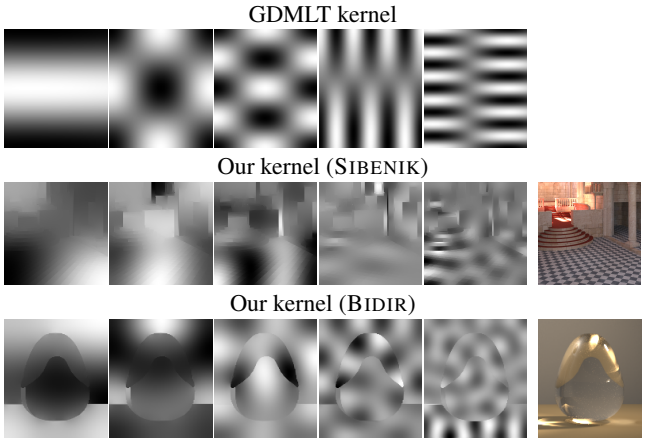


Figure 11: *Comparison of the eigenvectors corresponding to the 5th, 10th, 20th, 40th and 80th smallest eigenvalues (in that order) using the GDMLT kernels and our data-adaptive kernels. Note that the eigenvectors using our kernel adapt to the scene, while the eigenvectors using the GDMLT kernel do not. All visualized eigenvectors are normalized to* $[0, 1]$.

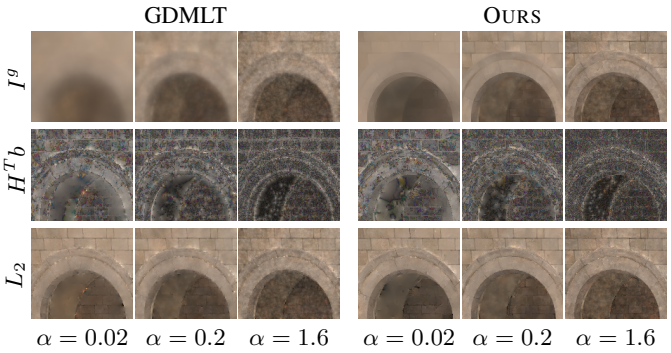


Figure 12: *Comparison of the effect of* α *on the Poisson solver on a close-up region of* SPONZA *with only 32 mutations per pixel. The top row shows the contribution of* I^g *to the solution, the middle row shows the contribution of the gradients* $H^T b$ *to the solution and the bottom row shows the* L_2 *reconstruction, which is equal to the sum of both images above. The tone-mapping of the gradient contribution has been adjusted to be more visible.*

previous methods. This technique leverages the half vector parameterization of light paths, which by construction avoids most singularities of the usual area parameterization. Finally, our third technique builds on the observation that gradient domain rendering is not restricted to conventional image gradients. We exploit this and introduce structure-adaptive kernels that encode edges and details in auxiliary feature images. The structure-adaptive kernels avoid sampling gradients between pixels with highly different path contribution functions because of geometry or other discontinuities in the scene. Avoiding such difficult gradients further reduces noise. We also show that our kernels have interesting properties with respect to the Poisson reconstruction step. An eigenanalysis of the Laplacian matrix induced by our kernels shows that image structures are preserved even in eigenvectors with low eigenvalues, which means that the Poisson reconstruction process is less prone to ringing than conventional kernels.

We obtain results that significantly reduce sampling artifacts of previous approaches. Our unbiased L_2 reconstructions generally

Figure 13: *Our proposed method using L_2 and L_1 reconstruction compared to gradient-domain MLT and manifold exploration MLT using approximatively the same number of mutations per pixel (mpp). MEMLT-Eq shows MEMLT with more samples in order to achieve approximately the same quality in terms of rMSE as L_1 Ours. The numbers below the close-ups with exception of MEMLT-Eq show the rMSE of the full images. All references where generated using MLT with 32k mpp.*

match previous biased L_1 results, and our L_1 results further improve on this. We believe our work shows that it is possible to achieve high quality gradient sampling, which may pave the way to robust, general purpose gradient rendering algorithms. In the future, we would like to further investigate robust techniques to sample gradients, for example by improving our weighting scheme.

Acknowledgements

This work was supported by the Swiss National Science Foundation under project nr. 143886. The box scene in Figure 13 was modeled by Toshiya Hachisuka [Hachisuka et al. 2008] .

References

BHAT, P., ZITNICK, L., COHEN, M., AND CURLESS, B. 2010. GradientShop: A gradient-domain optimization framework for image and video filtering. *ACM Trans. Graph. 29*, 2, 10:1–10:14.

BOLIN, M. R., AND MEYER, G. W. 1995. A frequency based ray tracer. In *Proc. ACM SIGGRAPH 95*, 409–418.

CHEN, Q., LI, D., AND TANG, C.-K. 2013. Knn matting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 35*, 9 (Sept), 2175–2188.

DAMMERTZ, H., SEWTZ, D., HANIKA, J., AND LENSCH, H. P. A. 2010. Edge-avoiding À-trous wavelet transform for fast global illumination filtering. In *Proc. High Performance Graphics 2010*, 67–75.

DAYAL, A., WOOLLEY, C., WATSON, B., AND LUEBKE, D. 2005. Adaptive frameless rendering. In *Proc. Eurographics Symposium on Rendering 2005*.

EGAN, K., TSENG, Y., HOLZSCHUCH, N., DURAND, F., AND RAMAMOORTHI, R. 2009. Frequency analysis and sheared reconstruction for rendering motion blur. *ACM Trans. Graph. 28*, 3, 93:1–93:13.

EGAN, K., HECHT, F., DURAND, F., AND RAMAMOORTHI, R. 2011. Frequency analysis and sheared filtering for shadow light fields of complex occluders. *ACM Trans. Graph. 30*, 2, 9:1–9:13.

HACHISUKA, T., JAROSZ, W., WEISTROFFER, R. P., DALE, K., HUMPHREYS, G., ZWICKER, M., AND JENSEN, H. W. 2008. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Trans. Graph. 27*, 3, 33:1–33:10.

HASTINGS, W. 1970. Monte Carlo samping methods using Markov chains and their applications. *Biometrika 57*, 1, 97–109.

JAKOB, W., AND MARSCHNER, S. 2012. Manifold exploration: A markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph. 31*, 4 (July), 58:1–58:13.

KAPLANYAN, A. S., HANIKA, J., AND DACHSBACHER, C. 2014. The natural-constraint representation of the path space for efficient light transport simulation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 33*, 4.

KELEMEN, C., SZIRMAY-KALOS, L., ANTAL, G., AND CSONKA, F. 2002. A simple and robust mutation strategy for the Metropolis light transport algorithm. *Comput. Graph. Forum 21*, 3, 531–540.

KOLLIG, T., AND KELLER, A. 2006. Illumination in the presence of weak singularities. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, H. Niederreiter and D. Talay, Eds. Springer Berlin Heidelberg, 245–257.

KONTKANEN, J., RÄSÄNEN, J., AND KELLER, A. 2004. Irradiance filtering for monte carlo ray tracing. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, Springer, 259–272.

KRISHNAN, D., FATTAL, R., AND SZELISKI, R. 2013. Efficient preconditioning of laplacian matrices for computer graphics. *ACM Trans. Graph. 32*, 4 (July), 142:1–142:15.

LEHTINEN, J., KARRAS, T., LAINE, S., AITTALA, M., DURAND, F., AND AILA, T. 2013. Gradient-domain metropolis light transport. *ACM Trans. Graph. 32*, 4 (July), 95:1–95:12.

LEVIN, A., RAV-ACHA, A., AND LISCHINSKI, D. 2008. Spectral matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 10, 1699–1712.

MCCOOL, M. D. 1999. Anisotropic diffusion for Monte Carlo noise reduction. *ACM Trans. Graph. 18*, 2, 171–194.

METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H., AND TELLER, E. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics 21*, 1087–1092.

OVERBECK, R., DONNER, C., AND RAMAMOORTHI, R. 2009. Adaptive wavelet rendering. *ACM Trans. Graph. 28*, 5, 140:1–140:12.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph. 22*, 3, 313–318.

RAMAMOORTHI, R., MAHAJAN, D., AND BELHUMEUR, P. 2007. A first-order analysis of lighting, shading, and shadows. *ACM Trans. Graph. 26*, 1, 2:1–2:21.

ROUSSELLE, F., KNAUS, C., AND ZWICKER, M. 2011. Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph. 30*, 6, 159:1–159:12.

ROUSSELLE, F., MANZI, M., AND ZWICKER, M. 2013. Robust denoising using feature and color information. *Computer Graphics Forum 32*, 7, 121–130.

SEN, P., AND DARABI, S. 2012. On filtering the noise from the random parameters in monte carlo rendering. *ACM Trans. Graph. 31*, 3 (June), 18:1–18:15.

SHI, J., AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 22*, 8 (Aug.), 888–905.

VEACH, E., AND GUIBAS, L. J. 1995. Optimally combining sampling techniques for Monte Carlo rendering. In *Proc. ACM SIGGRAPH 95*, 419–428.

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. In *Proc. ACM SIGGRAPH 97*, 65–76.

VEACH, E. 1997. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University.

WALTER, B., KHUNGURN, P., AND BALA, K. 2012. Bidirectional lightcuts. *ACM Trans. Graph. 31*, 4 (July), 59:1–59:11.

WARD, G. J., AND HECKBERT, P. 1992. Irradiance gradients. In *Proc. Eurographics Workshop on Rendering '92*.

WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. 1988. A ray tracing solution for diffuse interreflection. In *Computer Graphics (Proc. ACM SIGGRAPH '88)*, 85–92.