Gradient-Domain Metropolis Light Transport

Jaakko Lehtinen[1,2] Tero Karras[1] Samuli Laine[1] Miika Aittala[2,1] Frédo Durand[3] Timo Aila[1]

[1]NVIDIA Research [2]Aalto University [3]MIT CSAIL

Horiz. differences $I^{\mathrm{d}x}$ Vertical differences $I^{\mathrm{d}y}$ Coarse image I^g Sample density Result

Figure 1: *We compute image gradients I^{dx}, I^{dy} and a coarse image I^g using a novel Metropolis algorithm that distributes samples according to path space gradients, resulting in a distribution that mostly follows image edges. The final image is reconstructed using a Poisson solver.*

Abstract

We introduce a novel Metropolis rendering algorithm that directly computes image gradients, and reconstructs the final image from the gradients by solving a Poisson equation. The reconstruction is aided by a low-fidelity approximation of the image computed during gradient sampling. As an extension of path-space Metropolis light transport, our algorithm is well suited for difficult transport scenarios. We demonstrate that our method outperforms the state-of-the-art in several well-known test scenes. Additionally, we analyze the spectral properties of gradient-domain sampling, and compare it to the traditional image-domain sampling.

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

Keywords: Metropolis, global illumination, light transport

Links: DL PDF WEB

1 Introduction

The evaluation of path-space light transport integrals requires costly numerical techniques such as stochastic sampling with high sample counts to avoid noise, especially in the face of complex reflection phenomena involving multiple bounces, glossy reflection, and challenging visibility. An algorithm such as Metropolis light transport [Veach and Guibas 1997] seeks to improve efficiency by focusing samples in regions of path space with high contributions to the final image. A Markovian process mutates paths so that the final density is proportional to the throughput of paths, which makes it easy for the algorithm to locally explore regions that are hard to find because they have a small measure but have high contribution. Unfortunately, the algorithm also needs to use many samples in smooth parts of path space where radiance is high but does not vary much. In this paper, we build on the intuition that information in images is concentrated around edges and other variations, and present a new Metropolis light transport approach that seeks to concentrate samples on paths that contribute highly to the *gradient* of the image.

Our approach is to directly compute estimates for the horizontal and vertical finite-difference gradients of the image, in addition to a coarse estimate of the image itself. We then use a Poisson solver to produce a final image that best matches these estimates. This process is illustrated in Figure 1. In order to compute the gradients, we extend the notion of path space so that each of our samples corresponds to a *pair of paths* reaching neighboring pixels. To guarantee that both the gradients and the coarse image are adequately sampled, we drive a Metropolis sampler according to how much a sample contributes to each. This results in a high density of samples around image edges, as shown in Figure 1. Our algorithm is unbiased when the final image is reconstructed using a linear L^2 Poisson solver, but we also show that the performance and robustness can be further improved by using a non-linear L^1 reconstruction.

We must pay mathematical care to the measure in path space to ensure the correctness of our integrator. Starting with the equation for the gradient as a difference between the two path-space integrals for neighboring pixels, we turn it into a single integral over one pixel by defining a *shift function* that maps paths going through a pixel i to paths going through one of its neighbor j, in a manner similar to Veach and Guibas' lens mutations [1997]. This allows us to express the gradient as a single integral over the main pixel i alone, and reveals that the Jacobian of the shift function must be taken into account. We observe that carefully crafting the shift function can be important for robust treatment of difficult near-specular cases.

Finally, we analyze the balance between gradient and throughput sampling and its effect on the frequency spectrum of the error. Intuitively, taking the gradient squeezes the regions of high contribution towards discontinuities, making it harder for the sampler to find the relevant portions of path space. Our analysis shows that in our test cases this added difficulty is offset by the fact that the gradients contain less energy than the actual image.

Our approach is complementary to recent improvements in Metropolis light transport. In particular, we greatly benefit from Jakob and Marschner's manifold exploration [2012] because our method successfully focuses high-density sampling to a small part of path space, making it all the more important to be able to stay on the narrow manifold of high-contribution paths.

ACM Reference Format
Lehtinen, J., Karras, T., Laine, S., Aittala, M., Durand, F., Aila, T. 2013. Gradient-Domain Metropolis Light Transport. ACM Trans. Graph. 32, 4, Article 95 (July 2013), 11 pages. DOI = 10.1145/2461912.2461943 http://doi.acm.org/10.1145/2461912.2461943.

DOI: http://doi.acm.org/10.1145/2461912.2461943

2 Related Work

Physically-based Light Transport Physically based light transport algorithms render images by integrating over all possible paths that light can take from the source to the sensor. Veach and Guibas [1997] formulate the rendering equation as

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

Unbiased Monte Carlo rendering algorithms sample the space of light paths to compute these integrals numerically. The result I_j is the intensity of the jth pixel, Ω is the space of all light paths of finite length (*path space*), h_j is the pixel filter of the jth pixel, and $f^*(\bar{x})$ is the *spectral image contribution function* representing the amount of light reaching the sensor through a given path $\bar{x}$ in a given wavelength. A path $\bar{x}$ of length k consists of a sequence of vertices $\mathbf{x}_0, \ldots, \mathbf{x}_k$, and $\mathrm{d}\mu(\bar{x})$ is the *area product measure* $\prod_{i=0}^{k} \mathrm{d}A(\mathbf{x}_i)$.

Adaptive Sampling and Reconstruction Several Monte Carlo light transport techniques aim to adaptively sample light transport "where it matters", followed by a subsequent image reconstruction step. Several techniques sparsely sample radiance and its gradients, and employ higher-order interpolation to save on the number of samples that need to be shaded, e.g. [Ward and Heckbert 1992; Dayal et al. 2005; Ramamoorthi et al. 2007]. We focus on the finite differences of path throughput between pixels, and do not analytically reason about the underlying factors that give rise to the gradients. Rousselle et al. [2011] present state-of-the-art results in screen-space adaptive sampling and reconstruction. Bolin and Meyer [1995] directly estimate the DCT coefficients of the image based on point samples, and stop when a block is estimated to be represented faithfully. In a similar vein, Overbeck et al. [2009] use a wavelet decomposition to estimate whether the variance in the samples is due to visible image features or sharp path space features that will result in a smooth image when integrated, in which case they smooth the result. Several recent algorithms rely on frequency analysis for highly efficient sampling of individual effects [Soler et al. 2009; Egan et al. 2009]. All of the above algorithms are biased.

In their Multidimensional Adaptive Sampling algorithm (MDAS), Hachisuka et al. [2008] sample the light transport integrand in regions of contrast determined from local neighborhoods of samples, and produce a piecewise constant approximation to the integrand through a k-D tree. The reliance on the tree makes the algorithm suffer from the curse of dimensionality, excluding several use cases such as multiple bounces of indirect light. Further, the piecewise constant approximation introduces bias. Similar to their technique, our algorithm distributes samples along path space boundaries.

Gradient-Domain Image Processing Image gradients are a powerful and popular tool for image editing [Pérez et al. 2003; Georgiev 2005], panorama stitching, style transfer, and other applications too numerous to list here. We employ similar machinery to determine the primal image from computed gradients by solving a Poisson equation. Like Bhat et al. [2010], we also employ a coarse primal image to aid the reconstruction of low frequencies.

Natural Images It is well known that gradients of natural images tend to be sparse [Ruderman 1994]. As a result, the gradients provide a succinct representation for the image. The power spectrum of natural images is also known to be approximately inversely proportional to the square of frequency, which means that the gradients can be expected to contain much less energy than the original image. Inspired by this, we compute the gradients directly in hopes of making the most of our sample budget. Interestingly, Tumblin et al. [2005] envision a camera that directly captures gradient images.

2.1 Metropolis Sampling

Markov Chain Monte Carlo (MCMC) methods are used for drawing samples from high-dimensional functions that are difficult or impossible to sample directly. They start with an initial state $\mathbf{x}_0$ and, at each iteration t, apply a random change to the current state $\mathbf{x}_t$ to obtain the next state $\mathbf{x}_{t+1}$. If the *transition function* $K(\mathbf{s_1} \to \mathbf{s_2})$ denotes the probability density of going from state $\mathbf{s_1}$ to state $\mathbf{s_2}$ and μ is a throughput measure, the distribution of $\mathbf{x}_{t+1}$ is given by:

$$p_{t+1}(\mathbf{x}) = \int_\Omega K(\mathbf{y} \to \mathbf{x}) p_t(\mathbf{y}) d\mu(\mathbf{y}). \tag{2}$$

With mild conditions on K, the distributions p_t will converge to a stationary equilibrium distribution p^*.

The Metropolis-Hastings algorithm [Metropolis et al. 1953; Hastings 1970] constructs a Markov chain with f as the equilibrium distribution by crafting a special transition function K given a *tentative transition function* T (see [Veach and Guibas 1997] for requirements concerning T). Given the current state $\mathbf{x}_t$, MH generates a new candidate sample $\mathbf{x}'$ with probability $T(\mathbf{x}_t \to \mathbf{x}')$ and defines the next state $\mathbf{x}_{t+1}$ as $\mathbf{x}'$ with probability $a(\mathbf{x}_t \to \mathbf{x}')$, and as $\mathbf{x}_t$ otherwise, where

$$a(\mathbf{x} \to \mathbf{y}) = \min\left\{1, \frac{f(\mathbf{y})T(\mathbf{y} \to \mathbf{x})}{f(\mathbf{x})T(\mathbf{x} \to \mathbf{y})}\right\}. \tag{3}$$

The generated samples will be distributed according to f only in the limit for t tending to infinity, unless the initial distribution p_0 coincides with f. This problem is known as start-up bias, and in context of light transport it is easiest avoided by starting from an approximate equilibrium distribution computed using another sampling algorithm [Veach and Guibas 1997].

Integration Using Metropolis Sampling The ability of the MH algorithm to produce samples distributed with $f(\mathbf{x})$ makes it easy to compute integrals:

$$\int g(\mathbf{x}) \, \mathrm{d}\mathbf{x} \approx \frac{1}{N} \sum_{i=1}^{N} \frac{g(\mathbf{x}_i)}{p^*(\mathbf{x}_i)} = \frac{C}{N} \sum_{i=1}^{N} \frac{g(\mathbf{x}_i)}{f(\mathbf{x}_i)}. \tag{4}$$

Here $g(\mathbf{x})$ is an arbitrary function, $\mathbf{x}_i \sim p^*(\mathbf{x}) = f(\mathbf{x})/C$, and C is the integral of f estimated using other means, usually standard Monte-Carlo integration. Note that while $f(\mathbf{x})$ has to be a scalar function, $g(\mathbf{x})$ can be vector-valued. We call $f(\mathbf{x})$ the *sampling target function* and $g(\mathbf{x})$ the *integrand*.

Metropolis Light Transport The Metropolis light transport algorithm [Veach and Guibas 1997] directly applies Equation 4 to Equation 1 in the space Ω of light paths:

$$I_j \approx \frac{C}{N} \sum_i \frac{h_j(\bar{x}_i) f^*(\bar{x}_i)}{f(\bar{x}_i)}, \tag{5}$$

where the sampling target function $f(\bar{x})$ is the scalar luminosity of $f^*(\bar{x})$. The same path samples $\bar{x}_i$ are used for estimating the integrals for all pixels. To bootstrap the algorithm, Veach and Guibas select an initial path with bidirectional path tracing. While in usual MLT f^* and f are related in this simple manner—in particular, for a monochromatic rendering f^* equals f—the above makes it clear there is more freedom: we can drive the sampler using one function but integrate another. This is well known [MacKay 2003],

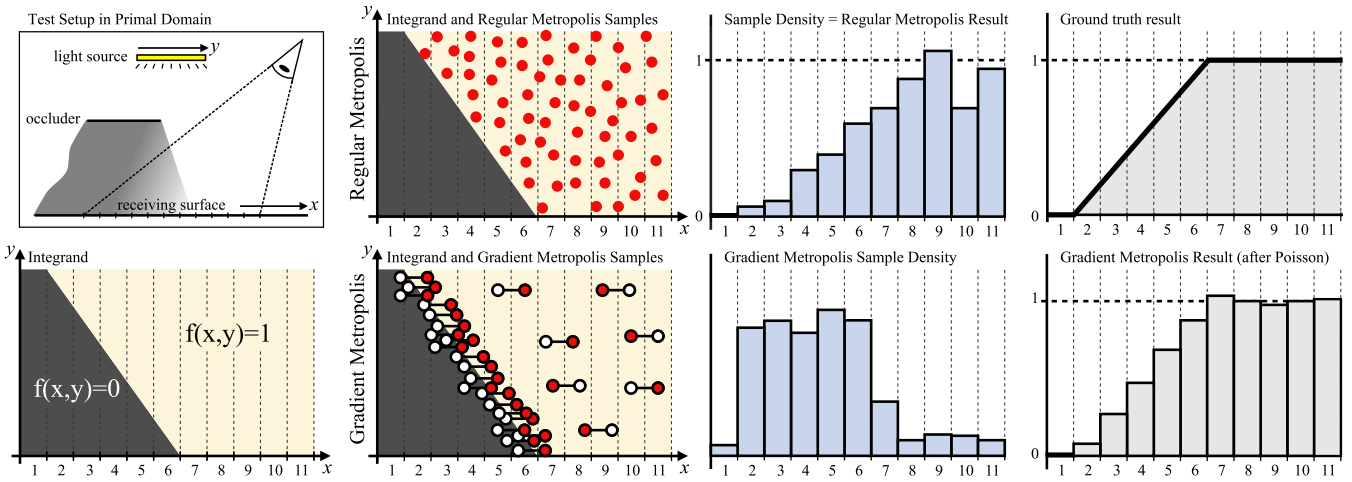


Figure 2: *A motivating example to illustrate the principle of our algorithm with simplified soft shadow computation in one-dimensional setting. In the integrand plots, the horizontal axis denotes pixels and vertical the position on a light source. The top row shows how a traditional MLT algorithm distributes the samples and computes the result, while the bottom row shows the same for our algorithm. See Section 3 for a detailed explanation.*

but in graphics it seems to only have applied by Hoberock and Hart [2010], whose MLT algorithm alters the screen distribution of paths using importance functions.

Veach and Guibas propose several mutation schemes that act on the path itself, resulting in a set of paths eventually distributed proportional to $f(\bar{x})$. The path space formulation allows for very flexible mutation strategies, but unfortunately, Ω has a complex structure and it can be challenging to compute probabilities for the tentative transitions. In order to simplify the space of integration, Kelemen et al. [2002] introduced *primary sample space* mutations. Their formulation allows symmetric mutations, removing the need to compute transition probabilities; however, some power is lost compared to path space mutations. Jakob and Marschner [2012] introduce a new mutation strategy, manifold exploration, that substantially improves the treatment of specular and highly glossy paths.

Energy-redistribution path tracing (ERPT) [Cline et al. 2005] is a variant of MLT that uses a very large number of short Markov chains. This allows them to use smaller perturbations than would be feasible in traditional MLT implementations, and potentially improves local exploration properties.

3 Overview

Our goal is to concentrate rendering effort in *regions of change* in the image. Our main idea is to compute image gradients directly, in addition to a coarse estimate of the image itself, and subsequently reconstruct the final image from these by solving a screened Poisson equation [Bhat et al. 2010]. We seek finite differences between pixels, *not* analytic (infinitesimal) derivatives of the image function. They would not benefit us, because we explicitly seek to capture the finite gradients caused by visibility and other discontinuities.

Figure 2 illustrates our approach in a simple one-dimensional scenario, where we integrate direct illumination from a partially blocked area light source. The integrand is a two-dimensional function that determines incident radiance for each screen position (x coordinate) and light source position (y coordinate). For each x, we integrate over y to find the total illumination, and these results are further integrated with box filters associated for each pixel (denoted by dashed vertical lines). Traditional Metropolis sampling (top row, Figure 2) distributes samples according to the magnitude of the integrand. Using a box filter, the final result is obtained by simply counting the samples that land in each pixel. The result is noisy even in the region where the true solution (top right) is flat.

While we want to concentrate our sampling on the gradients, we also want the coarse image to be of sufficient quality. We thus use the Metropolis-Hastings sampling to distribute our samples according to a target function that takes both the change in the integrand and its value into account. As illustrated in the bottom row of Figure 2, each of our samples consists of a pair of paths that measures the integrand in two locations one pixel apart. The final result (bottom right) captures the changed region faithfully, while remaining less noisy on the smooth part of the domain.

Working in the path integral formulation, we apply the same principle to the high-dimensional domain of light paths. We rewrite image gradients through path-space integrals by introducing a function that deterministically shifts a segment of a light path by one pixel (Section 4), and also note that driving the Metropolis-Hastings sampler by a combination of the gradient and primal path throughput is beneficial. Crucially, we must also account for the difference of path space measures by computing the Jacobian of the shift function (Section 5). Finally, we reconstruct the image by solving a screened Poisson equation (Section 6).

4 Gradient-Domain Light Transport

This section derives a path-space integral formulation for direct rendering of image gradients (Sec. 4.1) and describes how to use a Metropolis sampler for computing the gradients (Section 4.2).

4.1 Path-Space Gradients

We aim to compute image gradients, i.e., the pixelwise differences along the x and y directions in image space without computing the primal image first. We define the differences as $I_j^{\mathrm{d}x} = I_{j+1} - I_j$ and analogously for the y dimension. This section details how these can be written as a path-space integral analogous to Equation 1.

Let us first rewrite the measurement I_{j+1} in terms of a change of integration variable, i.e., in terms of paths at pixel j instead

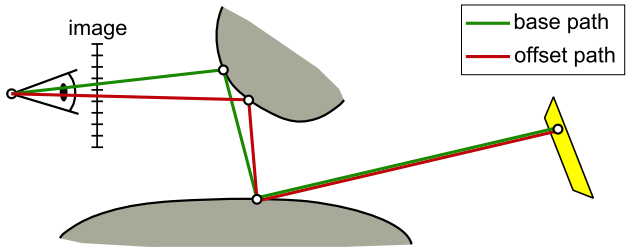


Figure 3: *Our base path corresponds to the traditional path used in bidirectional path tracing or MLT. In order to be able to compute path space gradients, we also have, for each base path, an offset path, which is—in absence of specular surfaces—constructed by perturbing the base path by one pixel, and then connecting the new primary hit to the secondary hit of the base path.*

of $j+1$. Let the screen coordinates of path $\bar{x}$ be (s_x, s_y). We define a *shift function* $T_{\delta_x,\delta_y}(\bar{x})$ that deterministically modifies the input path such that the resulting path intersects the screen at $(s_x+\delta_x, s_y+\delta_y)$. In absence of specular surfaces, the shift is similar to Veach's lens mutation, albeit deterministic (Figure 3). Now,

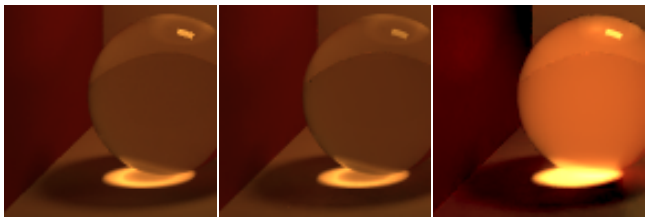
$$\begin{aligned} I_{j+1} &= \int_\Omega h_{j+1}(\bar{x}) f^*(\bar{x})\,\mathrm{d}\mu(\bar{x}) \\ &= \int_{T_{1,0}^{-1}(\Omega)} h_{j+1}(T_{1,0}(\bar{x})) f^*(T_{1,0}(\bar{x}))\,\mathrm{d}\mu(T_{1,0}(\bar{x})) \\ &= \int_\Omega h_{j+1}(T_{1,0}(\bar{x})) f^*(T_{1,0}(\bar{x}))\,\frac{\mathrm{d}\{\mu\circ T_{1,0}\}}{\mathrm{d}\mu}(\bar{x})\mathrm{d}\mu(\bar{x}) \\ &= \int_\Omega h_j(\bar{x}) f^*(T_{1,0}(\bar{x}))\,\frac{\mathrm{d}\{\mu\circ T_{1,0}\}}{\mathrm{d}\mu}(\bar{x})\,\mathrm{d}\mu(\bar{x}). \end{aligned} \tag{6}$$

The first row is essentially Equation 1. The second row is a no-operation obtained from the first by shifting the paths to one direction but integrating over path space shifted in the opposite direction. The third row is the change of variables from inversely-shifted paths to the original path space, which necessitates accounting for the *change in the density of the measure through the Jacobian determinant* $\mathrm{d}\{\mu\circ T_{1,0}\}/\mathrm{d}\mu$ of the shift function (cf. Section 5 for details); and finally, the fourth row is obtained by observing that $h_{j+1}(T_{1,0}(\bar{x})) = h_j(\bar{x})$. Finally, the difference is obtained from

$$I_j^{\mathrm{d}x} = I_{j+1} - I_j = \int_\Omega h_j(\bar{x}) \overbrace{\left[f^*(T_{1,0}(\bar{x}))\,\frac{\mathrm{d}\{\mu\circ T_{1,0}\}}{\mathrm{d}\mu}(\bar{x}) - f^*(\bar{x}) \right]}^{=f_{1,0}(\bar{x})} \mathrm{d}\mu(\bar{x}). \tag{7}$$

In the bracketed expression $f_{1,0}(\bar{x})$ the subscripts have the same meaning as in the definition of T, i.e., it is a function evaluated through shifting the horizontal screen coordinate of path $\bar{x}$ by one pixel to the right, computing the throughput of the resulting path, multiplying by the Jacobian determinant, and subtracting the throughput of path $\bar{x}$. Figure 4 shows what happens if one fails to account for the Jacobian determinant.

This completes the first stage of the derivation: given a path $\bar{x}$, we can compute its differential contribution to image gradients with respect to T. Moreover, integration over all light paths in the fashion of Equation 1 gives us the image-space gradient. Vertical differences are obtained through the same steps using $T_{0,1}$ instead of $T_{1,0}$. Note that it is simple to write the difference also to the opposite direction, i.e., using negative shifts $T_{-1,0}$ and $T_{0,-1}$ instead, and we will be utilizing this freedom below.



Ground truth | With Jacobian | Without Jacobian

Figure 4: *Ignoring the change in path space measure can lead to highly visible artifacts.*

4.2 Metropolis Sampling of Gradients

We compute the integrals in Equation 7 using Metropolis sampling. The process is largely analogous to standard path-space MLT, with some simple but important changes that necessitate extending the traditional path space.

4.2.1 Extended Path Space

Each pixel in our result contains both horizontal and vertical finite differences, i.e., the result consists of one color image for each. We compute the horizontal and vertical differences simultaneously by using a single chain that randomly alternates between the two directions. To this end, we extend the path space with two bits that determine the axis and direction along which the path is to be shifted. We denote the new path space by

$$\Omega' = \Omega \times \{(+1,0),(-1,0),(0,+1),(0,-1)\}. \tag{8}$$

Its elements, *extended paths* $\bar{z} = \{\bar{x}, \delta_x, \delta_y\}$, contain a *base path* $\bar{x}$, and two shift direction bits. The corresponding *offset path* is computed by $T_{\delta_x,\delta_y}(\bar{x})$, with T defined as in Section 4.1.

We use only mutators designed for traditional MLT for sampling base paths; we have not attempted to formulate specialized mutators particularly for sampling gradients. Specifically, this means that when proposing a mutated extended path $\bar{z}'$, its base path $\bar{x}'$ is sampled by a traditional mutator. Also the axis and direction are part of the Metropolis-Hastings sampler state, and are chosen randomly when a mutation is proposed. Building on an existing MLT implementation and its capability to compute $T(\bar{x}\to\bar{x}')$, computation of the tentative transition probability $T(\bar{z}\to\bar{z}')$ is simple because the proposals for the shift direction are uniform among the four alternatives.

4.2.2 Reversibility and Negative Offsets

We employ both negative and positive offsets because the mutators designed for traditional MLT can never propose samples where base path throughput $f^*(\bar{x})$ is zero. Such paths exist, e.g., in shadow boundaries of direct lighting, and are not reachable by the Markov chain. However, $f^*(\bar{x}) = 0$ *does not imply* $f_{\delta_x,\delta_y}(\bar{x}) = 0$; even if the base path carries no light, the offset path may have a positive throughput and thus contribute to the gradients. We employ positive and negative offsets precisely to enable sampling of all pairs of paths one pixel apart where at least one of the paths carries light.

Whenever both the base and offset paths are unblocked, they can be sampled with both positive and negative offsets, and consequently they contribute to the resulting finite differences twice—once through the forward and once through the reverse differences. We call these situations *reversible shifts*, and to ensure proper normalization of the result, the integrand must be divided by two in these cases. For extended paths where the offset path is blocked,

the integrand is left as-is to account for the fact that the path pair can only be sampled from one direction.

4.2.3 Integrand and Signs

Combining Sections 4.2.1 and 4.2.2, we can finally formulate the integrand for sampling gradients. We first define a generalized throughput function $f^*(\bar{z})$ by

$$f^*(\bar{z}) = n(\bar{z}) \begin{cases} f_{1,0}(\bar{x}), & (\delta_x, \delta_y) = (\ 1, \ 0) \\ -f_{-1,0}(\bar{x}), & (\delta_x, \delta_y) = (-1, \ 0) \\ f_{0,1}(\bar{x}), & (\delta_x, \delta_y) = (\ 0, \ 1) \\ -f_{0,-1}(\bar{x}), & (\delta_x, \delta_y) = (\ 0, -1) \end{cases}, \tag{9}$$

where $n(\bar{z}) = \frac{1}{2}$ if both the base and offset paths are unblocked, and 1 otherwise. The function $f_{\delta_x,\delta_y}(\bar{x})$ is defined in Equation 7.

When the path $\bar{z}$ sample is accumulated, it needs to be recorded to the correct gradient image (horizontal or vertical) based on which of δ_x, δ_y is non-zero. Paths with $\delta_x = -1$ or $\delta_y = -1$ estimate the finite difference $I_j - I_{j-1}$ instead of $I_{j+1} - I_j$, so we need to accumulate them into pixel $j-1$. Formally, this can be written through extended pixel filters $h_j^x(\bar{z})$ and $h_j^y(\bar{z})$ that only respond to horizontal and vertical differences:

$$h_j^x(\bar{z}) = \begin{cases} h_j(\bar{x}), & (\delta_x, \delta_y) = (\ 1, 0) \\ h_{j+1}(\bar{x}), & (\delta_x, \delta_y) = (-1, 0) \\ 0, & \delta_x = 0 \end{cases} \tag{10}$$

and similarly for the vertical differences. The filters respond in the usual way to path pairs with a positive offset, but with a negative shift of one pixel for path pairs sampled in the negative direction. The final integrands are then $h_j^x(\bar{z})f^*(\bar{z})$ and $h_j^y(\bar{z})f^*(\bar{z})$ for all j.

4.3 Coarse Estimate of the Image

Although it would be possible to reconstruct the final image based on the gradients alone—assuming that we also have an estimate of the overall image brightness—one can expect problems due to drift: small errors in the gradients can pile up over large distances, resulting in visible low frequency error in the final image. For this reason, we output a coarse estimate I^g of the primal domain image in addition to the gradients. This can be done at practically no cost during Metropolis sampling, since we already know the throughput of each base path $\bar{x}$ by virtue of having computed $f^*(\bar{z})$ for the corresponding extended path. Forming I^g is then only a matter of accumulating $\frac{1}{4}h_j(\bar{x})f^*(\bar{x})/f(\bar{z})$ for each extended path $\bar{z}$, where $f(\bar{z})$ is the target function driving the sampler. The factor $\frac{1}{4}$ is due to the fact that each base path appears 4 times in our extended path space, so we need to divide its contribution accordingly.

4.4 Target Function

We drive the Metropolis samples by a target function that is a combination of both generalized throughput $f^*(\bar{z})$ and base path throughput $f^*(\bar{x})$. Both of these functions are multispectral, and additionally $f^*(\bar{z})$ can be negative. As Metropolis-Hastings requires a non-negative scalar target function $f(\bar{z})$, we combine the luminance of their absolute values as follows:

$$f(\bar{z}) = ||f^*(\bar{z})|| + \alpha \left(\frac{1}{4} ||f^*(\bar{x})|| \right), \tag{11}$$

where α is a free parameter that controls the relative importance of the two terms. The choice of α affects the spectral characteristics of the approximation error in the final image—we defer the further analysis to Section 8. By using this target function our estimates are unbiased because $f(\bar{z}) \neq 0$ whenever $f^*(\bar{x}) \neq 0$ or $f^*(\bar{z}) \neq 0$.

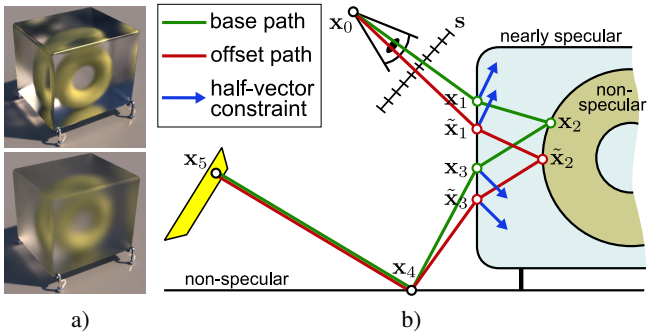


Figure 5: *a) We treat near-specular BSDFs as specular (top), and rough glossy BSDFs as non-specular (bottom). b) Offset path generation when (near-)specular surface interactions are present both before and after the first non-specular vertex. In this example, $S_e = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2\}$, $S_m = \{\tilde{\mathbf{x}}_3\}$, and unchanged suffix path is $\{\mathbf{x}_4, \mathbf{x}_5\}$.*

Using the absolute value of the path-space gradient does not directly correspond to distributing the samples according to the final image gradient, which is an integral over all pairs of paths that contribute to the pixel. This is easy to see: the image gradient can end up being zero when multiple positive and negative contributions from different paths cancel each other out. While the sampler cannot know this in advance, it does, however, concentrate on the higher-dimensional boundaries within path space similar to MDAS [Hachisuka et al. 2008].

5 Shift Function and Its Jacobian

In this section we describe our shift function in detail and derive an explicit form for the determinant of its Jacobian. Our result is valid for perspective cameras, both pinhole and thin-lens. In accordance to standard nomenclature, a specular surface interaction is one that scatters light into a discrete set of directions, corresponding to a Dirac BSDF. A non-specular interaction may be either perfectly diffuse or glossy. Like Jakob and Marschner [2012], we classify glossy interactions as either specular or non-specular for the purposes of the shift function, based on the sharpness of the corresponding BSDFs (Figure 5a).

5.1 Shift Functions

To simplify the notation, in this section we number the path vertices in a non-standard fashion starting from the camera. We denote the vertices of the base path $\bar{x}$ by $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k\}$, where $\mathbf{x}_0$ is the camera vertex, $\mathbf{x}_1$ is the primary hit, and $\mathbf{x}_k$ is the light source vertex. We denote the vertices of the offset path $T(\bar{x})$ by $\tilde{\mathbf{x}}_i$. Further, we denote the screen coordinates of path $\bar{x}$ by $\mathbf{s} = (s_x, s_y)$, and note that the primary hit is a function of $\mathbf{s}$, i.e., $\mathbf{x}_1 = \mathbf{x}_1(\mathbf{s})$, when we consider $\mathbf{x}_0$ (and the possible aperture parameter) to be fixed.

The purpose of the shift functions $T(\bar{x})$ is to deterministically alter the path $\bar{x}$ such that its screen coordinates $\mathbf{s}$ change by one pixel in the desired direction, with the camera vertex $\mathbf{x}_0$ staying fixed. While this is trivial in cases of non-specular transport—we merely change the direction of the primary ray, trace a ray to determine the new primary hit, and connect the next, unchanged path vertex to it—the specular case is significantly more involved.

Clearly, if the primary hit $\mathbf{x}_1$ lies on a specular surface, moving $\mathbf{x}_1$

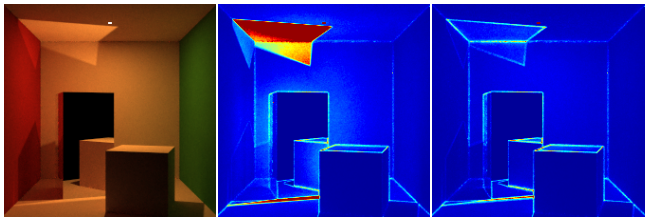


Image Lens (#samples) Manifold (#samples)

Figure 6: *When offset paths are perturbed only until the first non-specular vertex (similar to the lens mutator), the throughputs of offset paths with non-empty second specular chains S_m become zero, causing apparent gradients and leading to wasted samples on areas that do not actually need them. Perturbing a longer portion of the path, similarly to the manifold mutator, alleviates this issue. Here, the Cornell box was modified by turning the taller box into a mirror and shrinking the light source to make the features more obvious.*

but keeping the following vertices constant would cause the path to carry no light. Therefore we propagate the change deterministically along the specular chain until a non-specular vertex $\mathbf{x}_b$ is found, similarly to Veach and Guibas' lens mutator [1997]. Although this is a valid shift function, it is not always efficient (Figure 6) because if $\mathbf{x}_b$ is followed by another specular vertex, the specular chain of vertices between $\tilde{\mathbf{x}}_b$ and the *next* non-specular vertex $\mathbf{x}_c$ carries no light. Thus we further perturb the chain deterministically such that the specular chain ending at $\mathbf{x}_c$ correctly starts at $\tilde{\mathbf{x}}_b$, relying on Jakob and Marschner's [2012] manifold perturbation. If it happens that the shift moves the path off the specular manifold, the throughput of the offset path is defined to be zero.

In a general case the offset path $T(\bar{x})$ consists of four parts (Fig. 5):

1. The unchanged camera vertex $\mathbf{x}_0$;
2. The specular eye chain $S_e = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_b\}$ terminating on a non-specular vertex $\tilde{\mathbf{x}}_b$;
3. The possible second specular chain $S_m = \{\tilde{\mathbf{x}}_{b+1}, \ldots, \tilde{\mathbf{x}}_{c-1}\}$ following the first non-specular vertex; and finally
4. The unchanged suffix path $\{\mathbf{x}_c, \ldots, \mathbf{x}_k\}$.

The changed portion of the path is thus $\{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_b, \ldots, \tilde{\mathbf{x}}_{c-1}\}$, where $\tilde{\mathbf{x}}_b$ is a non-specular vertex and the rest are either purely or nearly specular. Because the propagation is deterministic, the shifted positions of the vertices in both S_e and S_m are functions of the screen coordinates of the base path. A common special case is when the primary hit lands on a non-specular surface and is followed by a non-specular vertex ($S_e = \{\tilde{\mathbf{x}}_1\}, S_m = \emptyset, c = 2$).

Following Jakob and Marschner [2012], the vertex positions in (near-)specular chains can be characterized in terms of constraint functions $c_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})$ that compute the (generalized) half-vector $\mathbf{o}_i$ at vertex i and project them to the tangent plane of the vertex. When the vertex positions are in specular configuration, $c_i = \mathbf{0}$. Importantly, any path can be characterized equivalently by the vertex positions *or* the vectors $\mathbf{o}_i$ and two vertex positions. Jakob and Marschner define the manifold perturbation as a function that keeps the offsets constant when the free vertices are moved.

5.2 Jacobian

We are now ready to compute the Jacobian determinant in Equation 7. Let $\partial\tilde{\mathbf{x}}_i/\partial\mathbf{x}_j$ denote the 2×2 matrix of partial derivatives of the local, orthogonal tangent plane coordinates of vertex $\tilde{\mathbf{x}}_i$ with respect to the coordinates of $\mathbf{x}_j$. The required Jacobian is computed by assembling the blocks for each combination of i, j into a $2(c-1) \times 2(c-1)$ matrix and computing its determinant:

$$\frac{\mathrm{d}\mu(T(\bar{x}))}{\mathrm{d}\mu(\bar{x})} = \left|\frac{\partial\tilde{\mathbf{x}}_i}{\partial\mathbf{x}_j}\right|_{ij}, \quad i, j = 1, \ldots, c-1. \tag{12}$$

We drop the index ranges for clarity from here on, and let expressions like $(\partial\tilde{\mathbf{x}}_i/\partial\mathbf{x}_j)_{ij}$ denote matrices consisting of one 2×2 block of partial derivatives for each combination of the indices. Because the suffix path is unchanged, we only need to account for terms involving the vertices that change in the shift. We now write the base and offset paths in terms of the offsets and the position of the middle non-specular vertex $\mathbf{x}_b$:

$$\{\mathbf{x}_1, \ldots, \mathbf{x}_{c-1}\} \equiv \{\mathbf{o}_1, \ldots, \mathbf{x}_b, \ldots, \mathbf{o}_{c-1}\} := \mathbf{O}$$
$$\{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{c-1}\} \equiv \{\tilde{\mathbf{o}}_1, \ldots, \tilde{\mathbf{x}}_b, \ldots, \tilde{\mathbf{o}}_{c-1}\} := \tilde{\mathbf{O}}.$$

Denoting the components of $\mathbf{O}$ by $\mathbf{O}_l$ (resp. $\tilde{\mathbf{O}}$, $\tilde{\mathbf{O}}_k$), the chain rule gives

$$\left|\frac{\partial\tilde{\mathbf{x}}_i}{\partial\mathbf{x}_j}\right|_{ij} = \left|\frac{\partial\tilde{\mathbf{x}}_i}{\partial\tilde{\mathbf{O}}_k}\frac{\partial\tilde{\mathbf{O}}_k}{\partial\mathbf{O}_l}\frac{\partial\mathbf{O}_l}{\partial\mathbf{x}_j}\right|_{ij} = \left|\frac{\partial\tilde{\mathbf{x}}_i}{\partial\tilde{\mathbf{O}}_k}\right|_{ik}\left|\frac{\partial\tilde{\mathbf{O}}_k}{\partial\mathbf{O}_l}\right|_{kl}\left|\frac{\partial\mathbf{O}_l}{\partial\mathbf{x}_j}\right|_{lj}. \tag{13}$$

Let us first deal with the middle term, the matrix of partial derivatives of the constraints. Because the manifold perturbation, by definition, keeps the offsets the same, i.e., $\tilde{\mathbf{o}}_i = \mathbf{o}_i$, the matrix has a simple structure: it is all identity, with the exception of the diagonal 2×2 block that corresponds to $\partial\tilde{\mathbf{x}}_b/\partial\mathbf{x}_b$; consequently, its determinant is simply $|\partial\tilde{\mathbf{x}}_b/\partial\mathbf{x}_b|$. Writing the middle non-specular vertex as a function of the screen coordinates $\mathbf{s}$ and employing the chain rule again, we have

$$\begin{aligned}\left|\frac{\partial\tilde{\mathbf{x}}_i}{\partial\mathbf{x}_j}\right|_{ij} &= \left|\frac{\partial\tilde{\mathbf{x}}_i}{\partial\tilde{\mathbf{O}}_k}\right|_{ik}\left|\frac{\partial\tilde{\mathbf{x}}_b}{\partial\mathbf{s}}\frac{\partial\mathbf{s}}{\partial\mathbf{x}_b}\right|\left|\frac{\partial\mathbf{O}_l}{\partial\mathbf{x}_j}\right|_{lj} \\ &= \left(\left|\frac{\partial\tilde{\mathbf{x}}_b}{\partial\mathbf{s}}\right|\left|\frac{\partial\tilde{\mathbf{x}}_i}{\partial\tilde{\mathbf{O}}_k}\right|_{ik}\right)\left(\left|\frac{\partial\mathbf{x}_b}{\partial\mathbf{s}}\right|\left|\frac{\partial\mathbf{x}_j}{\partial\mathbf{O}_l}\right|_{jl}\right)^{-1}.\end{aligned} \tag{14}$$

The two terms in the parentheses are precisely analogous, and can be computed separately for the base and offset paths. The 2×2 matrix $\partial\mathbf{x}_b/\partial\mathbf{s}$ can be computed by

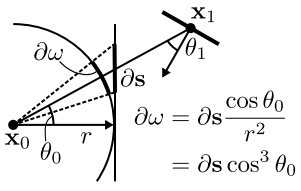
$$\begin{aligned}\frac{\partial\mathbf{x}_b}{\partial\mathbf{s}} = \frac{\partial\mathbf{x}_b}{\partial\mathbf{x}_1}\frac{\partial\mathbf{x}_1}{\partial\mathbf{s}} &= \frac{\partial\mathbf{x}_b}{\partial\omega_0^\perp}\frac{\partial\omega_0^\perp}{\partial\mathbf{x}_1}\frac{\partial\mathbf{x}_1}{\partial\mathbf{s}} \\ &= \frac{G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)}{G(\mathbf{x}_0 \leftrightarrow \ldots \leftrightarrow \mathbf{x}_b)}\frac{\partial\mathbf{x}_1}{\partial\mathbf{s}},\end{aligned} \tag{15}$$

and similarly for the terms related to the offset path. Here $\partial\omega_0^\perp/\partial\mathbf{x}_1 = G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)$ is the classical geometry term [Veach and Guibas 1997], and $G(\mathbf{x}_0 \leftrightarrow \ldots \leftrightarrow \mathbf{x}_b)$ is the generalized geometry term [Jakob and Marschner 2012].

Finally,

$$\frac{\partial\mathbf{x}_1}{\partial\mathbf{s}} = \frac{\|\mathbf{x}_1 - \mathbf{x}_0\|^2\ \cos^3\theta_0}{\cos\theta_1},$$

which is obtained by projecting from $\mathbf{x}_1$ to a sphere centered at $\mathbf{x}_0$, and projecting from the sphere to the view plane.



The above derivation is valid in the general case of sequences of nearly but not perfectly specular surfaces. Like Jakob and Marschner, we eliminate pure specular vertices from the equations

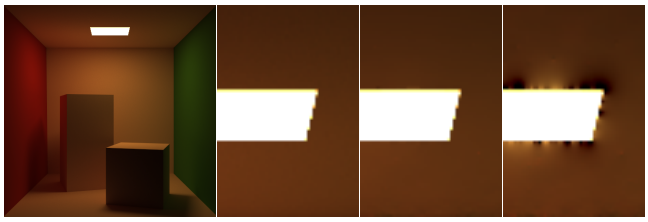


Figure 7: *L^1 reconstruction is superior to L^2 in the proximity of poorly sampled bright features, such as directly visible light sources or outlier pixels where a Markov chain got stuck for an extended time. Here, also direct lighting was computed using Metropolis sampling, and an insufficient number of samples is deliberately used to emphasize the difference.*

because this corresponds to an essentially lower-dimensional integration domain. This affects the computation of the determinants $|\partial \mathbf{x}_j / \mathbf{O}_l|$ and $|\partial \tilde{\mathbf{x}}_i / \tilde{\mathbf{O}}_k|$, and path throughput $f(\bar{x})$. The computation of the determinants $|\partial \mathbf{x}_j / \mathbf{O}_l|$ and $|\partial \tilde{\mathbf{x}}_i / \tilde{\mathbf{O}}_k|$ is somewhat involved, but fortunately the same term is employed in the manifold perturbation, and the existing implementation in Mitsuba [Jakob 2012] can be reused without modification.

In the common special case of the primary hit lying directly on a non-specular surface and followed by a non-specular vertex, $\mathbf{O} = \{\mathbf{x}_1\}$, $\tilde{\mathbf{O}} = \{\tilde{\mathbf{x}}_1\}$, and the entire Jacobian reduces to

$$\frac{\partial \tilde{\mathbf{x}}_1}{\partial \mathbf{s}} \left(\frac{\partial \mathbf{x}_1}{\partial \mathbf{s}} \right)^{-1} = \frac{\|\tilde{\mathbf{x}}_1 - \tilde{\mathbf{x}}_0\|^2}{\|\mathbf{x}_1 - \mathbf{x}_0\|^2} \frac{\cos\theta_1}{\cos\tilde{\theta}_1} \frac{\cos^3\tilde{\theta}_0}{\cos^3\theta_0}.$$

6 Image Reconstruction

Once we have the gradient images $I^{\mathrm{d}x}$ and $I^{\mathrm{d}y}$ and a rough estimate of the actual image I^g, we use a screened Poisson solver to find the image I whose gradients best match the inputs in L^2 sense [Pérez et al. 2003; Bhat et al. 2010]. This can be written as

$$\underset{I}{\operatorname{argmin}} \left(\left\| \begin{pmatrix} H^{\mathrm{d}x} I \\ H^{\mathrm{d}y} I \end{pmatrix} - \begin{pmatrix} I^{\mathrm{d}x} \\ I^{\mathrm{d}y} \end{pmatrix} \right\|_2^2 + \|\alpha (I - I^g)\|_2^2 \right). \quad (16)$$

Here $H^{\mathrm{d}x}$, $H^{\mathrm{d}y}$ are matrices that take finite differences along both rows and columns and returns appropriately vectorized gradients. Solving the Poisson equation in this way is a linear operation. α determines the relative weighting of the primal and gradient images in the reconstruction. While one could use a different α for the target function (Equation 11) and reconstruction (Equation 16), we have observed that best results are obtained by using the same weight for both. Section 8 presents a detailed analysis of the role of α and the spectral characteristics of the result.

Claim. The resulting image is unbiased. *Proof.* Let $\mathbf{S}(I^{\mathrm{d}x}, I^{\mathrm{d}y}, I^g)$ denote the screened Poisson solution operator, and let E be the expected value operator. Both are linear. Let R be the true solution image. Since our samplers are unbiased, we have, by definition, $E[I^{\mathrm{d}x}] = H^{\mathrm{d}x}(R), E[I^{\mathrm{d}y}] = H^{\mathrm{d}y}(R)$, and $E[I^g] = R$. Because the screened Poisson equation is invertible, $\mathbf{S}$ perfectly recovers R from its gradients: $\mathbf{S}(H^{\mathrm{d}x}(R), H^{\mathrm{d}y}(R), R) = R$. Since expectation commutes with fixed linear operators, we have

$$E[\mathbf{S}(I^{\mathrm{d}x}, I^{\mathrm{d}y}, I^g)] = \mathbf{S}(E[I^{\mathrm{d}x}], E[I^{\mathrm{d}y}], E[I^g]) = \mathbf{S}(H^{\mathrm{d}x}(R), H^{\mathrm{d}y}(R), R) = R, \quad (17)$$

i.e., the expected value of the solution is the true image. □

L^1 Reconstruction Our gradient estimates may contain large-magnitude errors especially with lower sampling rates and within the proximity of very bright objects or pixels where Markov chains got stuck for a longer time. The L^2 solution effectively spreads the error due to such outliers on large areas in the image. Due to its better resilience, we have found it preferable to use L^1 optimization instead [Levin et al. 2004]. Figure 7 demonstrates the difference between L^1 and L^2 reconstruction. However, since L^1 optimization is non-linear, the result is not guaranteed to be unbiased. In most practical cases the differences in image quality are small (1–2 dB in our tests), and L^1 is consistently better both numerically and perceptually. The time taken by L^2 reconstruction is negligible compared to rendering, and even the L^1 optimization takes less than 10 seconds for 720p images.

7 Results

We have implemented our algorithm as an extension to the publicly available Mitsuba renderer [Jakob 2012]. We will compare against unbiased rendering methods: path-space MLT (VMLT), primary sample space MLT (KMLT), energy-redistribution path tracing with bidirectional seeding (ERPT), and bidirectional path tracing (BDPT), all of which are available in Mitsuba. Our test platform is a 6-core 3.33GHz Intel Core i7 with 24GB of memory. All full-resolution result images are available on the project web page.

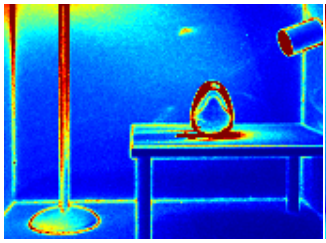
We have five test scenes. SIBENIK and DOOR (Figures 9 and 10) feature difficult light transport: light enters the visible rooms through narrow openings. TORUS (Figure 11) embeds a non-specular object inside a solid glass cube, which is a well-known difficult scenario for unbiased light transport algorithms. The other two scenes, LAMP and SPONZA (Figures 8 and 12), feature simpler light transport, and are used mainly to demonstrate various details.

The ground truth images were rendered using BDPT, except in SIBENIK, DOOR and TORUS where BDPT would have taken days or weeks to converge. In these scenes we used 12-hour VMLT renderings for ground truth. Most of the methods have a large number of parameters that could theoretically be adjusted separately for each scene. As we believe this is not an option in practical use, we rely mostly on Mitsuba's default parameters that appear to be rather carefully chosen. The set of mutators, length of Markov chains, and maximum path length were chosen on a per-scene basis. In all cases, the same set of mutators and mutator parameters were used for VMLT and our method. In particular, we use manifold exploration in all scenes. Unless stated otherwise, we use the MLT methods for indirect light only, and direct lighting is computed separately using low discrepancy sampling. Russian roulette starts from the fifth bounce in all scenes.

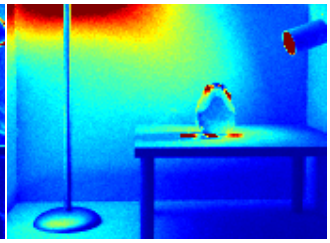
We compare the algorithms by showing non-converged images rendered with a fixed time budget. While deviations from ground truth are guaranteed to disappear over time, the images include algorithm-specific artifacts, such as high-magnitude outliers (BDPT) or brightness deviations between image regions and missing specular highlights (VMLT). Such uneven convergence is characteristic to all Metropolis algorithms that use long chains: once a narrow region of high-throughput paths is found, local exploration distributes its energy on the screen, producing a sudden change. If the region is small and consequently very difficult to find, this can happen practically at any time in the rendering process — even when the image seems otherwise converged.

Figure 8 visualizes a number of concepts from our algorithm in the LAMP scene. The second row demonstrates that our method places most of the samples around gradients in the image, whereas VMLT spends them on bright areas of the image. Gradients are shown

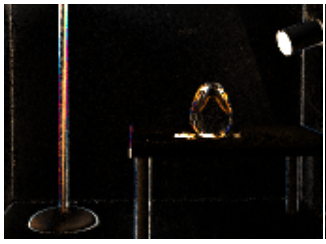
a) LAMP rendered using an unbounded number of bounces



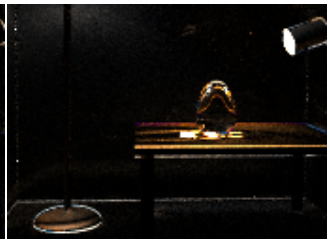
b) Our sample count



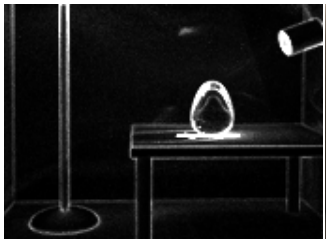
c) VMLT sample count



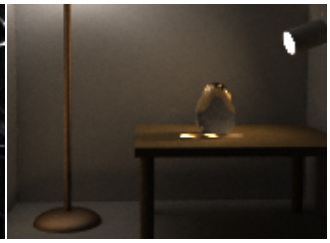
d) x gradient (absolute value)



e) y gradient (absolute value)



f) Irreversible

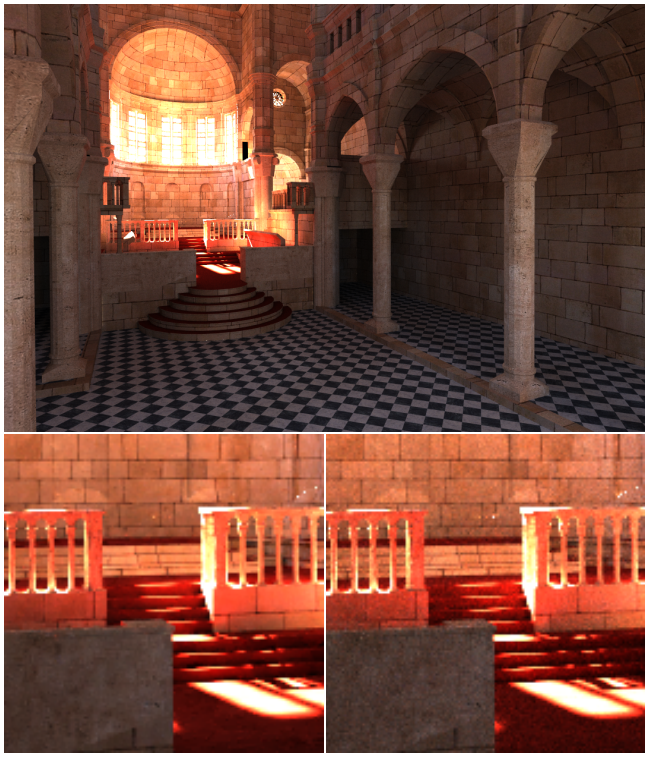


g) Closeup of indirect light only

Figure 8: *a) The* LAMP *scene with full lighting. The closeups b–g use indirect lighting only. b) and c) show where the samples are deposited by our algorithm and VMLT. d) and e) visualize horizontal and vertical gradients. f) marks the pixels where irreversible shifts were encountered. Predictably, these concentrate around geometric edges, but also around shadow boundaries and caustics. The closeups were rendered at a reduced resolution to make the pixel-scale features more visible.*

next, and "irreversible" highlights pixels where irreversible shifts occurred frequently (Section 4.2.2).

Table 1 gives scene-specific details and lists the obtained PSNRs. BDPT does reasonably well on the two simplest scenes where it can reliably find the contributing paths. KMLT and ERPT perform quite similarly to each other, giving a significant improvement over BDPT in the three more difficult scenes. ERPT is particularly good in TORUS. Of the comparison methods VMLT generally beats the rest by a wide margin, and our method consistently improves upon



Our (34.2 dB) VMLT (32.4 dB)

Figure 9: *Ground truth and result image closeups from* SIBENIK.

it, by up to 5.4 dB. Our method can afford approximately half as many samples as VMLT within a fixed time budget because our samples are more expensive due to the offset path generation and computation of the Jacobian.

SIBENIK and DOOR are difficult for BDPT, ERPT, and KMLT. VMLT uses long Markov chains (20k samples), allowing it to more thoroughly explore the high-contribution areas of path space, resulting in vastly improved convergence. Our algorithm retains this desirable property, and further improves the result by 2–4 dB.

TORUS demonstrates that our algorithm works also in presence of difficult specular transport. This is a challenging scene because the path space is fragmented into separate narrow regions of high contribution. Our method suffers from this more than VMLT because finite differencing makes an already difficult path throughput function even more so. In VMLT and our method only the bidirectional mutator [Veach and Guibas 1997] is allowed to change the path configuration (number of vertices, type of vertices) and its acceptance probability is low in this scene. This makes it hard for the sampler to reliably move between modes of specular transport, reducing the effectiveness of long Markov chains.

In LAMP and SPONZA our method shows a 4–5 dB advantage over VMLT. As for the characteristic artifacts of our method (which will disappear over time), the noise around sharp discontinuities disappears slower than the noise from smooth areas, and can therefore be visible in non-converged images.

We further investigated the effect of the number of edge pixels in the image. We studied this in SPONZA where the vast majority of the edges seen by our algorithm come from textures, by tiling the repetitive textures four times in each direction (16× more edge pixels). VMLT lost 0.1 dB and we lost 0.7 dB. As expected, we are slightly more sensitive to the density of edge pixels, but we still

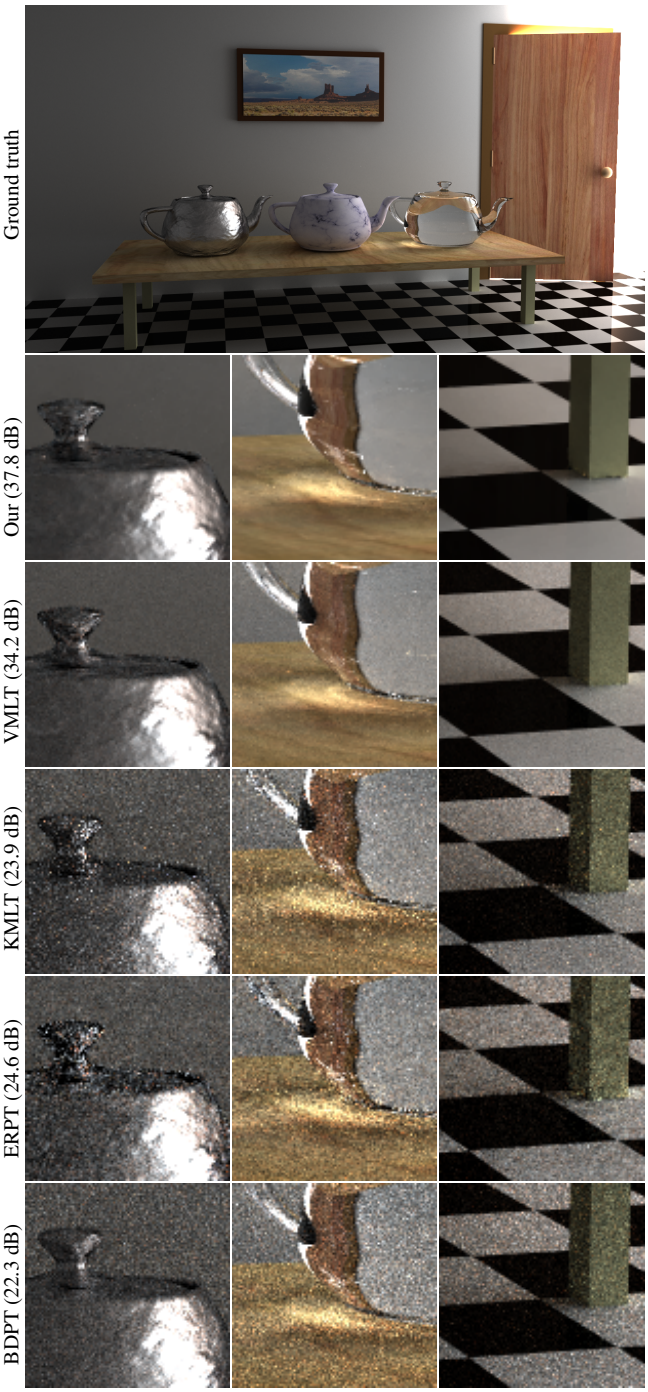
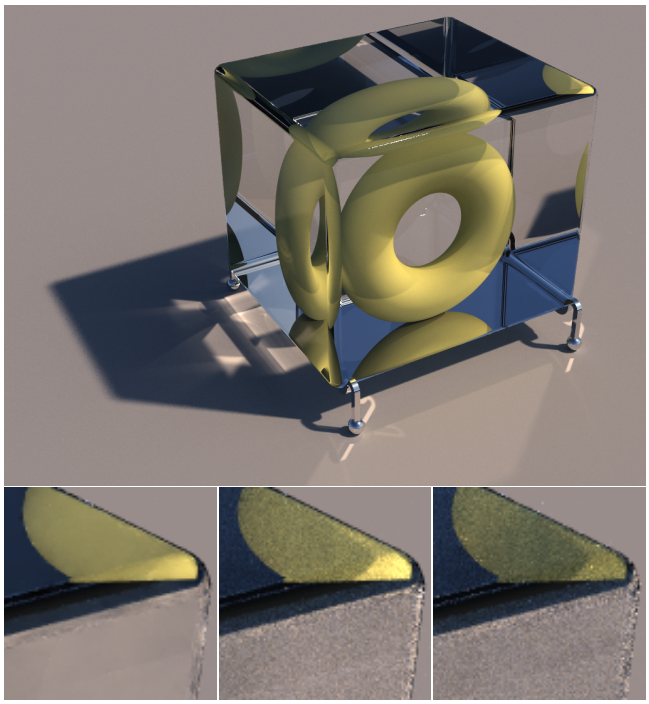


Figure 10: *Closeups of result images from* DOOR.



Our (29.8 dB) VMLT (29.8 dB) ERPT (28.9 dB)

Figure 11: *Ground truth and result image closeups from* TORUS.

Scene	Resolution	Time (min)	PSNR (dB) Our	VMLT	KMLT	ERPT	BDPT	Max path length
SIBENIK	1080×720	10	**34.2**	32.4	19.7	24.0	14.4	6
DOOR	1280×720	20	**37.8**	34.2	23.9	24.6	22.3	12
TORUS	1024×768	5	**29.8**	**29.8**	22.5	28.9	19.2	6
LAMP	1280×720	10	**38.8**	34.8	36.4	31.7	35.2	∞
SPONZA	1024×768	2.5	**39.6**	34.2	29.3	31.7	32.3	∞

Table 1: *Details and PSNR measurements from our test scenes. Measured from floating point pixel data, before gamma correction.*

maintained an almost 5 dB lead even with the dense textures.

8 Analysis and Discussion

To further understand how our method behaves with respect to the free parameter α, let us examine the characteristics of the resulting approximation error. Figure 12 shows a series of closeups of SPONZA rendered with different values of α, along with their differences from the ground truth. The value of α directly affects the frequency content of the error: higher values distribute a considerable part of the error on high frequencies, whereas lower values concentrate the error on low frequencies. The choice of α thus allows us to make a tradeoff between low and high frequency errors in order to maximize the overall PSNR.

The sampling phase of our algorithm calculates Monte Carlo approximation I^g of the ideal primal domain image R, as well as approximations $I^{\mathrm{d}x}$ and $I^{\mathrm{d}y}$ of its gradients $R^{\mathrm{d}x}$ and $R^{\mathrm{d}y}$, respectively. To study the effect of finite differencing on the signal, we take the Fourier transform of $R^{\mathrm{d}x}$:

$$\mathcal{F}\{R^{\mathrm{d}x}\} = \mathcal{F}\{R(x+1,y) - R(x,y)\} = (e^{i\omega_x} - 1)\mathcal{F}\{R\}, \quad (18)$$

and similarly for $R^{\mathrm{d}y}$. The factor $e^{i\omega_x} - 1$ is the frequency response of the finite difference operator, which we denote by $H^{\mathrm{d}x}$. The corresponding amplitude response is given by the absolute value $|H^{\mathrm{d}x}| = 2\sin\left|\frac{1}{2}\omega_x\right|$. Finite differencing thus amplifies the Nyquist frequency $\omega_x = \pi$ by a factor of two and attenuates low frequencies roughly proportional to $|\omega_x|$, killing the DC term completely. Since the energy of natural images is known to be distributed roughly according to $1/\omega^2$, we can expect R to contain most of its energy on the same frequencies that are attenuated the most by $H^{\mathrm{d}x}$. $R^{\mathrm{d}x}$ thus contains significantly less energy than R; the difference is, for instance, 18.5 dB in SPONZA.

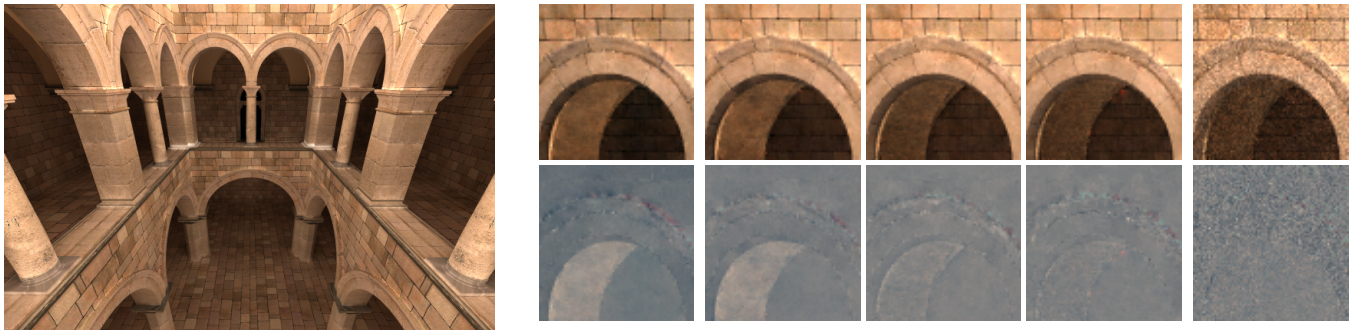


Figure 12: SPONZA *with indirect light only. Our method typically captures fine detail regardless of the value of* α*, but when the value is very small, coarse image estimation is effectively disabled and the color of uniform areas is entirely determined by the noisy edges. When* $\alpha \to \infty$*, our method becomes equal to VMLT. The closeups use only 256 samples per pixel on the average to highlight noise, and the difference images have been further amplified.*

The frequency content of the approximation error resulting from Metropolis sampling of I^g and $I^{\mathrm{d}x}$ is shown in Figure 13, top. We can see that $I^{\mathrm{d}x}$ has consistently lower error than I^g. This is mainly because the sampling error is relative in nature, so its magnitude is dependent on the magnitude of the integrand. The difference of 18.5 dB is, however, offset by the fact that gradients are generally harder to sample than luminance. We also see that the errors are concentrated toward low frequencies, especially with I^g. The explanation lies in the mutation strategies, which often propose relatively small perturbations to the path. This leads to a high amount of correlation between consecutive samples.

After sampling, we run the screened Poisson solver to determine the final image I. Qualitatively, the Poisson solver can be thought of as first integrating $I^{\mathrm{d}x}$ and $I^{\mathrm{d}y}$ by amplifying them roughly inversely proportional to the frequency, and then forming I by taking the low frequency content from I^g and the high frequency content from $I^{\mathrm{d}x}$ and $I^{\mathrm{d}y}$, with the cutoff controlled by α. This can be seen by noting that he L^2 Poisson solution described by Equation 16 is linear and translation invariant, allowing an analytic Fourier transform:

$$\mathcal{F}\{I\} = \frac{\alpha^2 \mathcal{F}\{I^g\} + \overline{H}^{\mathrm{d}x} \mathcal{F}\{I^{\mathrm{d}x}\} + \overline{H}^{\mathrm{d}y} \mathcal{F}\{I^{\mathrm{d}y}\}}{\alpha^2 + |H^{\mathrm{d}x}|^2 + |H^{\mathrm{d}y}|^2}, \qquad (19)$$

where $\overline{H}$ is used to denote the complex conjugate of H. The gain of this linear system with respect to I^g and $I^{\mathrm{d}x}$ is shown on the top row of Figure 14. We also illustrate its response to I^g by setting the gradients to zero (Figure 14, middle row), and similarly for $I^{\mathrm{d}x}$ and $I^{\mathrm{d}y}$ by setting I^g to zero (Figure 14, bottom row).

Figure 13, bottom shows the frequency content of the error in I after the Poisson solver. We only show the results for an L^1 solver, as the difference between L^1 and L^2 is consistently less than 2 dB in SPONZA. We also note that the error in I is a direct consequence of the errors in I^g, $I^{\mathrm{d}x}$, and $I^{\mathrm{d}y}$, meaning that the results in Figure 13, bottom can be predicted reliably based purely on Figure 13, top and Equation 19.

The role of α is thus two-fold. First, high α leads to a larger portion of the samples being distributed according to image luminance, which reduces the sampling error of I^g. Second, as the estimate of I^g becomes more reliable, the Poisson solver relies more on it when it comes to low frequencies of I. Although the optimal choice is dependent on the scene, we have found $\alpha = 0.20$ to work well in all of our test scenes.

Acknowledgments

We thank Wenzel Jakob for Mitsuba, and for answering our technical questions on it during the initial stages of this project. M. Aittala is supported by the HeCSE Graduate School and the MIDE program (project UI-ART) of Aalto University. Frédo Durand acknowledges NSF CGV 1116303.

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.

CLINE, D., TALBOT, J., AND EGBERT, P. 2005. Energy redistribution path tracing. *ACM Trans. Graph. 24*, 3, 1186–1195.

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.

GEORGIEV, T. 2005. Image reconstruction invariant to relighting. In *Proc. Eurographics 2005*, 61–64.

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.

HOBEROCK, J., AND HART, J. C. 2010. Arbitrary importance functions for Metropolis light transport. *Comput. Graph. Forum 29*, 6, 1993–2003.

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, 58:1–58:13.

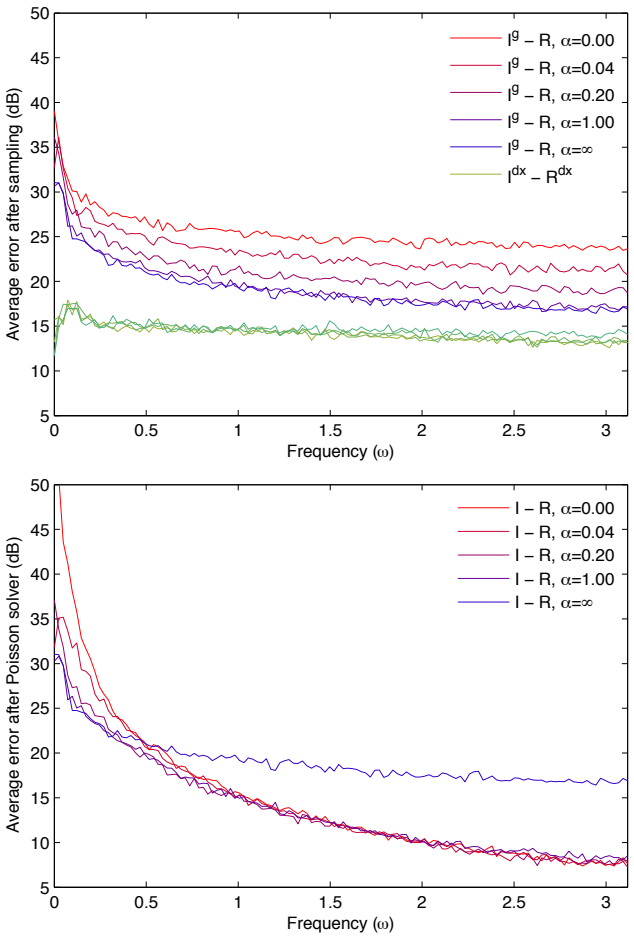


Figure 13: Top: *Ensemble power spectrum of the approximation error in coarse image I^g and horizontal gradient I^{dx} for* SPONZA, *rendered with 256 samples per pixel. The x-axis corresponds to frequency $\omega = \sqrt{\omega_x^2 + \omega_y^2}$, with the Nyquist frequency located at $\omega = \pi$. The y-axis shows spectral power in dB, averaged over all orientations. We omit I^{dy}, as it behaves exactly the same way as I^{dx}.* Bottom: *Ensemble power spectrum of the approximation error in the final image I, reconstructed using L^1 Poisson solver. $\alpha = \infty$ is equal to VMLT, and suffers from excess high frequency errors. $\alpha = 0$ corresponds to running our method without the coarse image, and suffers from excess low frequency errors.*

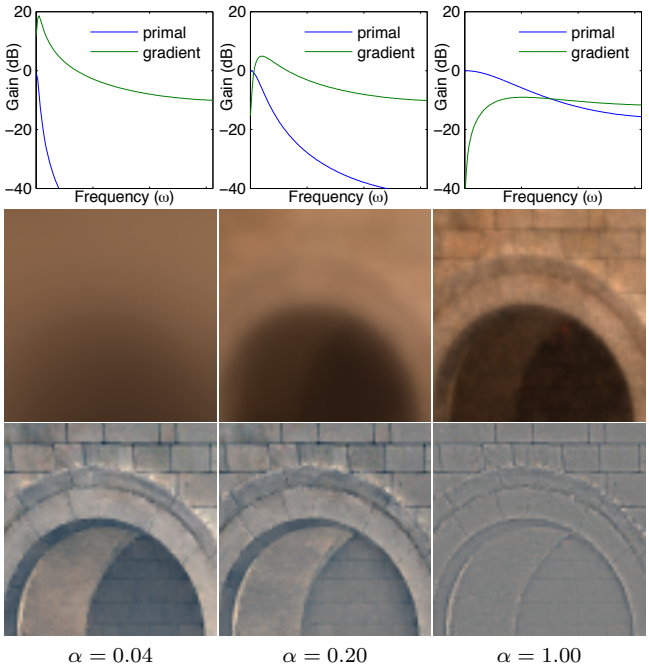


Figure 14: *Effect of α on the L^2 Poisson solver.* Top: *Gain with respect to I^g and I^{dx} in dB.* Middle: *Contribution of I^g to the final image, i.e. the result of setting the gradients to zero.* Bottom: *Contribution of the gradients, i.e. the result of setting I^g to zero.*

JAKOB, W., 2012. Mitsuba v0.4. http://mitsuba-renderer.org.

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.

LEVIN, A., ZOMET, A., PELEG, S., AND WEISS, Y. 2004. Seamless image stitching in the gradient domain. In *Proc. European Conference on Computer Vision (ECCV)*, 377–389.

MACKAY, D. J. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

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.

RUDERMAN, D. 1994. The statistics of natural images. *Network: computation in neural systems 5*, 4, 517–548.

SOLER, C., SUBR, K., DURAND, F., HOLZSCHUCH, N., AND SILLION, F. 2009. Fourier depth of field. *ACM Trans. Graph. 28*, 2, 18:1–18:12.

TUMBLIN, J., AGRAWAL, A., AND RASKAR, R. 2005. Why I want a gradient camera. In *Proc. CVPR'05*, 103–110.

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

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