Energy Redistribution Path Tracing

David Cline Justin Talbot Parris Egbert *

Brigham Young University

Abstract

We present Energy Redistribution (ER) sampling as an unbiased method to solve correlated integral problems. ER sampling is a hybrid algorithm that uses Metropolis sampling-like mutation strategies in a standard Monte Carlo integration setting, rather than resorting to an intermediate probability distribution step. In the context of global illumination, we present Energy Redistribution Path Tracing (ERPT). Beginning with an inital set of light samples taken from a path tracer, ERPT uses path mutations to redistribute the energy of the samples over the image plane to reduce variance. The result is a global illumination algorithm that is conceptually simpler than Metropolis Light Transport (MLT) while retaining its most powerful feature, path mutation. We compare images generated with the new technique to standard path tracing and MLT.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism.

Keywords: Monte Carlo integration, correlated integrals, energy redistribution, global illumination, path tracing, Metropolis light transport.

1 Introduction

Today, a number of rendering programs exist that can produce photorealistic output. Methods that create such "synthetic photographs" by measuring light transport are collectively known as *global illumination* algorithms. The distinguishing characteristic of a global illumination algorithm, as opposed to an ad-hoc lighting algorithm, is the goal of accounting for all light scattering events that lead to the creation of an image. In a very real sense, the process of global illumination is a physical simulation in which light transport paths are followed through a virtual scene and recorded on a virtual film plane.

The most versatile global illumination algorithms currently available are based on ray tracing and numerical integration. [Kajiya 1986] was the first to publish a global illumination algorithm of this type. Drawing on heat transfer literature and Monte Carlo integration theory, Kajiya described the now classic path tracing algorithm, which samples the light reaching the image plane by tracing potential light paths backwards from the eye point.

Despite the generality of path tracing, it can be quite inefficient even in common lighting situations. The reason for this inefficiency is variance in the Monte Carlo light estimate, which shows up as noise in a rendered image. Practically speaking, the Monte Carlo sampler does not have enough global context to quickly find all of the important light transport paths. Importance sampling techniques, such as [Veach and Guibas 1995] and [Lawrence et al. 2004] can provide some of this context, but usually only in a local way.

Noting the difficulty of finding all of the significant light transport paths starting at the eye point, [Lafortune and Willems 1993] and [Veach and Guibas 1994] independently developed bidirectional path tracing, which generates paths starting at the light sources as well as the eye point. Some parts of path space are better sampled this way, so variance is reduced.

Other techniques cache and interpolate portions of the light transport that are similar between pixels, reducing variance at the expense of biasing the solution. Irradiance caching [Ward et al. 1988], density estimation [Shirley et al. 1995] and photon mapping [Jensen 1996] all take this approach.

An innovative global illumination algorithm that has received a lot of attention in recent years is Metropolis Light Transport (MLT) [Veach and Guibas 1997]. MLT replaces the Monte Carlo integrator used in path tracing with a Metropolis sampler. The main advantage of the Metropolis algorithm over Monte Carlo integration is the ability to preserve the sampling context. This is done by using path mutation to explore path space in a localized way. Thus, when high contribution paths are found, nearby paths will likely be explored as well.

Since the original 1997 paper, researchers have sought to extend the MLT algorithm in a number of ways. [Pauly et al. 2000] added mutation strategies to MLT that handle participating media such as smoke and fog. [Kelemen et al. 2002] simplified the MLT algorithm, and increased the mutation acceptance rate, by defining mutation over an abstract space of random numbers rather than the geometric space of ray paths. Other work has focused on the statistical properties of MLT. [Szirmay-Kalos et al. 1999] characterized the start-up bias problem of the algorithm, and [Ashikhmin et al. 2001] analyzed its variance.

Combining path tracing and MLT. In this paper we propose a new global illumination algorithm that combines Monte Carlo path tracing with Metropolis Light Transport mutation strategies. The algorithm works by redistributing the energy of initial path traced samples over the image plane, so we call it Energy Redistribution path tracing, or ERPT.

Our motivation for combining path tracing and MLT comes from the observation that Monte Carlo integration tends to be easy to stratify and control, whereas Metropolis has better convergence properties in many hard sampling situations. However, even though Metropolis sampling may exhibit better convergence properties at low sampling densities, it will still have a worse *order* of convergence than stratified Monte Carlo sampling if the dimensionality of the integral is low enough. This effect becomes readily apparent when comparing MLT to path tracing for direct lighting.

The ERPT algorithm begins with a set of Monte Carlo samples taken from a path tracer. It then uses a filter step based on path mutation to spread the energy of the MC samples over the image plane in an unbiased way. Unlike the Metropolis algorithm, which uses a single, very long Markov chain of sample locations, our algorithm generates shorter sample chains starting at each path traced sample. We can use short chains because the initial Monte Carlo step eliminates startup bias.

*cline@rivit.cs.byu.edu, jft2@email.byu.edu, egbert@cs.byu.edu

Paper organization. The remainder of this paper is organized as follows: section 2 reviews a number of ideas leading up to ER sampling, including a brief review of Monte Carlo integration, correlated integrals and energy flow. Section 3 describes the ER sampling algorithm in detail. Section 4 presents Energy Redistribution Path Tracing, followed by comparisons between ERPT, standard path tracing and MLT in section 5. Finally, section 6 concludes and suggests ways to improve the algorithm.

2 Sampling Issues

This section gives an overview of sampling ideas leading up to ER sampling. We give a brief overview of Monte Carlo integration, and present the concepts of correlated integrals, energy flow, and general and detailed balance. Finally, Metropolis sampling is reviewed, and we show how it relates to energy flow and detailed balance.

2.1 Monte Carlo Integration

Consider the problem of integrating the function f over some domain Ω:

$$\int_\Omega f(\bar{x})\, d\mu(\bar{x}).$$

We place a bar over the x to indicate that it may be a vector rather than just a scalar quantity. Monte Carlo integration solves this integral by creating a random variable $\mathbf{X}_f$ with expected value equal to the integral:

$$E[\mathbf{X}_f] = \int_\Omega f(\bar{x})\, d\mu(\bar{x}).$$

$\mathbf{X}_f$ is constructed starting with a sampling procedure $\mathbf{S}_p$ which generates samples from Ω according to some probability distribution, p. To complete $\mathbf{X}_f$, a sample location $\bar{x}$ is chosen using $\mathbf{S}_p$, and $\mathbf{X}_f(\bar{x})$ is evaluated

$$\mathbf{X}_f(\bar{x}) = \frac{f(\bar{x})}{p(\bar{x})}. \tag{1}$$

This expression forms an unbiased estimate of the integral, which may have a high variance. The usual way to reduce the variance is to average a number of samples taken from $\mathbf{X}_f$. We will refer to the quantity $\mathbf{X}_f(\bar{x})$ as the "initial energy" deposited at point $\bar{x}$.

Equation 1 can also be rearranged to obtain values of f in terms of $\mathbf{X}_f$ and p:

$$\mathbf{X}_f(\bar{x})\; p(\bar{x}) = f(\bar{x}). \tag{2}$$

We will refer to $\mathbf{X}_f(\bar{x})p(\bar{x})$ as the "expected energy" at $\bar{x}$.

2.2 Correlated Integrals

A good number of integration problems involve the estimation of not just one, but a large number of integrals. Path tracing is a particularly pertinent example. Each pixel in a path traced image is an integral that is evaluated using Monte Carlo integration. A standard path tracer evaluates the pixel integrals independently, but it is well known that the integrals have highly correlated integrands (see figure 1). The most successful correlated integral solutions tend to exploit the correlation between integrals to reduce variance. In fact, the correlation between pixel integrands is the implied basis for many of the global illumination algorithms currently in use, including irradiance caching [Ward et al. 1988], photon mapping [Jensen 1996] and Metropolis Light Transport [Veach and Guibas 1997]. Irradiance caching and photon mapping take adavantage of inter-pixel correlation by caching incident light values, which are later used to approximate parts of the pixel integrals that are difficult to evaluate independently. MLT leverages the correlation between pixel integrals in a different way, using mutation strategies

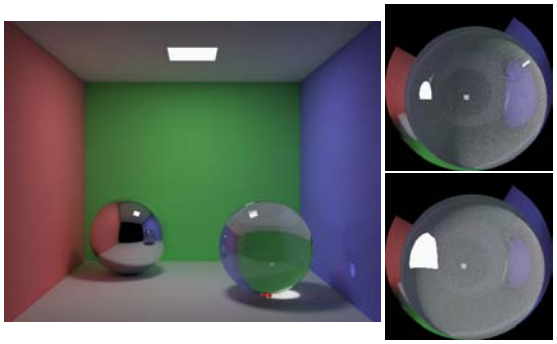


Figure 1: Correlated integrals in path tracing. A path tracer must integrate the light incident on surface points seen from the camera. Nearby pixels often have very similar integrands, as can be seen above. The right images show the incident light at the two pixels marked by the red dots in the left image.

to share integrand information between pixels. Our work takes a similar approach, utilizing path mutations to spread the energy of initial Monte Carlo pixel estimates over the image plane.

2.3 Energy Flow

One way to coordinate sampling efforts between correlated integrals is to use a process of *energy flow*. (By *energy*, we simply mean the value of a real-valued function. For a color-valued function, such as an image, energy refers to the luminance.) Energy flow allows a sampling procedure to perform a directed search between similar points in the domains of correlated integrals. To see why this can be useful, consider two correlated integrals, I_1 and I_2, with domains Ω_1 and Ω_2. Suppose that in the process of sampling, a high contribution point $\bar{x}$ is found in Ω_1 (i.e. $\mathbf{X}_f(\bar{x})$ is large). Since I_1 and I_2 are correlated, it is likely that a high contribution point $\bar{y}$ will exist in a location similar to $\bar{x}$ in Ω_2. Energy flow establishes a connection between points $\bar{x}$ and $\bar{y}$ and transfers some of the energy at $\bar{x}$ to $\bar{y}$. Figure 2 shows this graphically. Often, energy flow can be more efficient than standard Monte Carlo sampling because the cost of finding high contribution points is amortized over multiple integrals.

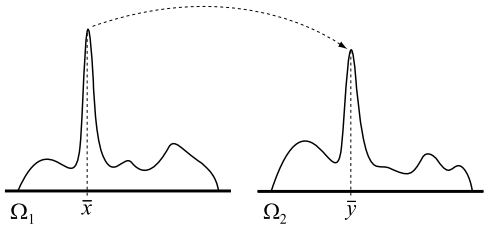


Figure 2: Energy flow connects points in correlated integrals, and transfers function energy between them. When done properly, energy flow provides a mechanism for directed searching within the domains of correlated integrals without biasing the integral estimates.

The expected energy flow. In practice, energy flow is created by perturbing or "mutating" a source point, $\bar{x}$, to produce a destination point, $\bar{y}$. (Imagine laying a pipe from $\bar{x}$ to $\bar{y}$ along which energy can flow.) Some of the energy at $\bar{x}$ is then tranferred to $\bar{y}$. Let $T(\bar{x} \to \bar{y})$ be the *transition probability* from $\bar{x}$ to $\bar{y}$, that is, the probability that $\bar{y}$ is chosen as the destination point given that $\bar{x}$ is the source point. In this situation, the expected flow from $\bar{x}$ to $\bar{y}$ is

$$E[\phi(\bar{x} \to \bar{y})] = E[\mathbf{X}_f(\bar{x})\; p(\bar{x})\; T(\bar{x} \to \bar{y})\; q(\bar{x} \to \bar{y})], \tag{3}$$

where $\phi(\bar{x} \to \bar{y})$ denotes the energy flow from $\bar{x}$ to $\bar{y}$, $E[\ \cdot\ ]$ is the expected value, $X_f(\bar{x})p(\bar{x})$ is the expected energy located at $\bar{x}$ from an initial Monte Carlo estimate (equation 2), and $q(\bar{x} \to \bar{y})$ is the percentage of energy at $\bar{x}$ that flows to $\bar{y}$ once a connection has been established.

General and detailed balance. Astonishingly, energy flow can occur without biasing the integral estimates, as long as certain conditions on the flow amount are met. In particular, the integral estimates will remain unbiased as long as the expected flow of energy out of any point $\bar{x}$ equals the expected flow back in. We will refer to this property as *general balance*. More formally, we say that general balance holds if

$$E\left[\int \phi(\bar{x} \to \bar{y})\, d\mu(\bar{y})\right] = E\left[\int \phi(\bar{y} \to \bar{x})\, d\mu(\bar{y})\right] \quad \forall\, \bar{x}. \qquad (4)$$

An even stronger constraint that guarantees unbiased-ness is called *detailed balance*. Detailed balance requires that the expected flow between any two points be equal. In other words,

$$E[\phi(\bar{x} \to \bar{y})] = E[\phi(\bar{y} \to \bar{x})] \quad \forall\, \bar{x}, \bar{y}. \qquad (5)$$

Figure 3 shows the two kinds of balance graphically.

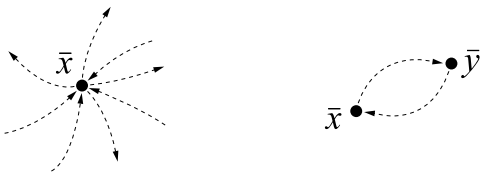


Figure 3: General and detailed balance of energy flow. General balance (left) requires the total expected flow out of a point to equal the expected flow back in. Detailed balance (right) requires that the expected energy flow between any two points be equal.

2.4 Review of Metropolis Sampling

Here we provide a brief overview of Metropolis sampling, also called the Metropolis algorithm or the Metropolis-Hastings algorithm. We refer the reader to [Pharr 2003] for a more thorough introduction to Metropolis sampling and its application to rendering.

Suppose that it is desired to evaluate a set of correlated integrals, $I_1 \ldots I_n$. Monte Carlo integration would solve this problem by taking a separate set of samples from each of the integral domains. Sampling the integrals separately can be inefficient, however, since the high contribution points in each integral domain must be found independently. The Metropolis algorithm [Metropolis et al. 1953] takes a different approach that allows sampling efforts to be coordinated among the different integrals. The main idea is to create a probability distribution (pdf) that is proportional to the correlated integrals and then draw samples from this distribution. Metropolis sampling does this by using detailed balance to migrate a single sample through the domains of the correlated integrals, $\Omega_1 \ldots \Omega_n$. As the sample moves, a histogram is kept of its location, and the number of samples deposited in the domain of each integral ends up being proportional to the value of the integral (see figure 4).

Metropolis sampling and detailed balance. Instead of using detailed balance to define the amount of flow in the system, Metropolis sampling uses it to define the *acceptance probability*, the probability that flow will occur given a proposed mutation. When flow does occur, a single unit of energy is transferred. Thus,

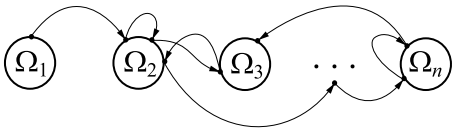


Figure 4: The Metropolis algorithm evaluates a set of correlated integrals by moving a sample through the domains of the integrals, tracing out a distribution proportional to the integral values. Sampling efforts are coordinated because the moving sample can jump between similar locations in the domains of different integrals.

the expected number of flow events between any two points, $\bar{x}$ and $\bar{y}$, must be equal for detailed balance to hold, and the ratio of the acceptance probabilities between points $\bar{x}$ and $\bar{y}$ is given by

$$\frac{a(\bar{x} \to \bar{y})}{a(\bar{y} \to \bar{x})} = \frac{f(\bar{y})T(\bar{y} \to \bar{x})}{f(\bar{x})T(\bar{x} \to \bar{y})}.$$

In practice, it is usually best to maximize the acceptance probabilities, so the actual acceptance probability used is

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

Limitations of the Metropolis algorithm. Although Metropolis sampling has proven useful in a variety of sampling contexts, it has several limitations that make it difficult to use in a global illumination setting. For example, the Metropolis algorithm is based on the idea of drawing samples from a probability distribution, even though it doesn't explicitly calculate a pdf. This framework tends to be less flexible than working directly with function energies. Metropolis also exhibits the so called "startup bias" problem, which in practice means that it can only be used if a large number of samples will be taken. Furthermore, Metropolis sampling does not stratify well, and its convergence characteristics are hard to analyze.

Metropolis versus ER sampling. As will be seen in the next sections, Energy Redistribution sampling attempts to do away with some of the limitations of Metropolis sampling while maintaining its most powerful features. ER sampling works directly with function energies. It eliminates the startup bias problem because it begins with an unbiased Monte Carlo estimator. Consequently, ER sampling does not require a complete set of mutation strategies to work (ergodicity is ensured by the initial MC samples). Furthermore, since it is an extension of Monte Carlo integration, ER sampling can leverage stratified sampling and other Monte Carlo variance reduction techniques.

3 Energy Redistribution Sampling

In the last section we saw that Metropolis sampling is closely related to the ideas of energy flow and detailed balance. In this section, we describe Energy Redistribution sampling, a new algorithm that exploits these same principles, but directly in a Monte Carlo integration setting.

Energy Redistribution sampling evaluates a set of correlated integrals in a two step process. In the first step, Monte Carlo samples are taken from the integral domains. The second step uses a process of energy flow to redistribute the energy of the MC samples over the domains of the correlated integrals in an unbiased way. Figure 5 summarizes this process. The heart of ER sampling lies in choosing a flow filter to redistribute the energy of the MC samples. This section describes several *balanced energy flow filters* (rules that define energy flow in such a way that general balance holds) We start with the *detailed balance flow rule* and then modify it to produce the *equal deposition flow rule*, which forms the basis of our ER sampling algorithm.

```
EnergyRedistributionSampling ()
    for each integral domain, Ω_i
        for j = 1 to m
            create an MC sample, x̄, in Ω_i according to S_p
            evaluate X_f(x̄) = f(x̄)/p(x̄)
            if X_f(x̄) > 0
                redistribute the energy of X_f(x̄) using
                    a balanced energy flow filter.
```

Figure 5: The Energy Redistribution sampling algorithm.

3.1 The Detailed Balance Flow Rule

To work, ER sampling must define a set of mutation strategies and determine the amount of energy transferred during flow events (q from equation 3). The main objective is to define a flow rule that reduces the variance of a set of Monte Carlo estimates while satisfying general balance. As an initial attempt, we follow the lead of Metropolis, and use *detailed* balance directly to define q. Borrowing the standard acceptance probability used by the Metropolis algorithm, we derive

$$q(\bar{x} \to \bar{y}) = min\left(1,\ \frac{\mathbf{X}_f(\bar{y})p(\bar{y})T(\bar{y} \to \bar{x})}{\mathbf{X}_f(\bar{x})p(\bar{x})T(\bar{x} \to \bar{y})}\right). \tag{6}$$

We call this flow rule the *detailed balance* flow rule for obvious reasons. In practice, we mutate the original Monte Carlo samples multiple times and transfer some energy to each of the mutated samples. For example, if the MC sample $\bar{x}$ is mutated n times, it will produce n mutated samples, $\bar{y}_1 \ldots \bar{y}_n$, and the amount of energy transferred to each of the $\bar{y}_i$ will be $X_f(\bar{x})q(\bar{x} \to \bar{y}_i)/n$.

After flow occurs, any energy that has not flowed out of $\bar{x}$ stays there, and contributes to the integral estimate at that point. Consequently, the detailed balance flow rule is unbiased, but has the serious drawback that a large portion of the energy may not flow anywhere. Thus, if a bright spot exists in one of the initial integral estimates, it will likely continue to exist after flow has occurred, and the variance will remain high.

3.2 The Equal Deposition Flow Rule

The main problem with the detailed balance flow rule is that some of the energy of the Monte Carlo estimates does not flow anywhere. A modification that partially solves this problem is to apply the detailed balance rule recursively on the original sample and all the mutated samples that are created. Each time the rule is applied, some of the energy at $\bar{x}$ gets "whittled off" so that after a few flow events very little of the original energy remains at that point. Unfortunately, iterating in this manner results in an exponential growth in the number of samples. In essence, the recursion creates a tree of splitting Markov chains that multiplies the number of samples at each iteration. Another problem with this approach is that the amount of energy in the mutated samples will vary wildly, leading to increased variance.

Equal deposition. A better solution is to create linear Markov chains that emanate from each MC sample point rather than splitting chains. Sample chains can be prevented from splitting by probabilistically keeping all of the energy at the current location or transferring all of it to the mutated location. These chains of samples create a set of unbiased estimates of the correlated integrals, one for each flow iteration. (Note that in order for the estimates to remain unbiased, the sample chains must all be the same length.)

```
EqualDepositionFlow (x̄, e, m, e_d)
    numChains = ⌊random(0,1) + e/(m × e_d)⌋
    for i = 1 to numChains
        ȳ = x̄
        for j = 1 to m
            z̄ = mutate(ȳ)
            if q(ȳ → z̄) ≥ random(0,1)
                ȳ = z̄
            deposit e_d energy at ȳ
```

Figure 6: The equal deposition flow rule. $\bar{x}$ is the location of a Monte Carlo sample, e is the initial energy at $\bar{x}$, $\mathbf{X}_f(\bar{x})$, m is the sample chain length, and e_d is the deposition energy.

To reconstruct the integrals, the ER sampler desposits an equal fraction of the original energy imparted to the sample chains after each iteration. Hence, we call this rule the *equal deposition* flow rule.

To see that the equal deposition flow rule is unbiased, notice that the sample chains, taken as a whole, form n unbiased estimates of the correlated integrals. Although each sample is processed separately, the end result is nothing more than the average of the n unbiased estimates, and is therefore unbiased as well.

Equal deposition and Metropolis sampling. In the special case where all of the sample chains start with the same amount of energy, the equal deposition flow rule becomes a form of Metropolis sampling that does not exhibit startup bias. To put it another way, since the amount of energy deposited by each sample chain on each flow iteration is equal, the sampler is implicitly taking draws from a distribution proportional to f, which is exactly what Metropolis does. This process is very similar to how [Veach and Guibas 1997] eliminate startup bias, except that in addition, our algorithm uses the Monte Carlos samples to provide initial coverage of the entire sample space.[1] Pseudo-code for this form of the algorithm is given in figure 6.

Starting the sample chains with the same energy is not the only option, however. Any number of sample chains (even zero) can be started from a given MC sample as long as the expected energy imparted to the chains equals the sample's initial energy. Another point is that there are situations in which the sample chains do not all have to have the same length. This happens when the set of available mutations splits the domains of the integrals into disjoint sets. As long as the chains in these disjoint sets are all the same length, the integral estimates will remain unbiased.

The deposition energy. An essential part of the equal deposition flow rule is the *deposition energy*, or how much energy will be deposited after flow events. To determine the deposition energy, we estimate the expected energy of MC samples within the sampling domain and divide by the desired number of mutations per integrand as follows:

$$e_d = e_{ave}/k, \tag{7}$$

where e_d is the deposition energy, e_{ave} is the average energy of a number of samples taken with the Monte Carlo sampler, and k is the desired number of mutations per correlated integrand. Note that a poor estimate of e_{ave} will not change the accuracy of the algorithm, only its run time. Contrast this with Metropolis sampling, in which a value similar to e_{ave} acts as a global scale factor for the integral estimates.

[1] These results also imply that if Veach's resampling algorithm to eliminate startup bias is used, and multiple sample chains are created, we cannot use the energy of the original MC samples to determine the sample chain length without reintroducing bias.

4 Energy Redistribution Path Tracing

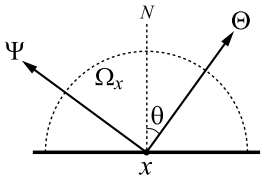
This section gives the details of our Energy Redistribution path tracing algorithm, which we will refer to as ER path tracing or ERPT. Conceptually, the algorithm is nothing more than Energy Redistribution sampling with a path tracer as the Monte Carlo sampler. Figure 7 gives pseudocode for ERPT. Notice that the algorithm is quite similar to a path tracer. The only difference is that the step that deposits the energy onto the image plane has been replaced by a balanced energy flow filter. The remainder of this section describes several essential details of ER path tracing. We give a brief overview of the rendering equation in the context of path tracing, and then discuss the idea of Monte Carlo path density. Next we give a set of rules to determine the relative Monte Carlo sampling density between two ray paths, which is needed to calculate the value of q. We also discuss the specific mutation strategies used by our algorithm.

```
ERPathTracing(m_c)
    determine the deposition energy, e_d    // equation 7
    for each pixel in the image
        for j = 1 to n
            create a path, x̄, in the current pixel
            X_f(x̄) = f(x̄)/p(x̄)   // evaluate the path
            if  X_f(x̄) > 0
                EqualDepositionFlow(x̄, X_f(x̄), m_c, e_d)
```

Figure 7: The ER path tracing algorithm. The code above specifies the equal deposition flow rule, but any balanced energy flow filter could be used. The value m_c is the user-specified sample chain length. In practice, we found values between about 100 and 1000 to work well.

4.1 Ray Paths and Monte Carlo Path Density

A path tracer creates an image by sampling the incoming light over the area of each pixel on the image plane. This incoming light is described by the *rendering equation* [Kajiya 1986], one form of which is given below:



$$L(x \rightarrow \Psi) = L_e(x \rightarrow \Psi) + \int_{\Omega_x} L(x \leftarrow -\Theta) f_r(\Psi \leftrightarrow \Theta) |\cos\theta| d\omega_\Theta. \quad (8)$$

In brief, the rendering equation describes the light coming from a surface point x in a particular direction, Ψ, $L(x \rightarrow \Psi)$. The term $L_e(x \rightarrow \Psi)$ is the light emitted directly from from x in direction Ψ, Ω_x is the hemisphere above point x, and $f_r(\Psi \leftrightarrow \Theta)$ is the BRDF function at x. We refer the reader to [Dutré et al. 2003] for a complete discussion of the various forms of the rendering equation.

A path tracer samples the rendering equation by means of ray paths that connect the eye point to a light source through a number of scattering events (reflections or refractions). To build a path, a path tracer sends out a ray from the eye point into the scene. The path tracer then extends the ray through a number of scattering events to produce an *eye subpath*, using a probabilistic sampling function to choose the outgoing direction at intersection points. We will call this function p_d. The path tracer may connect the eye subpath to a light source in one of two ways. First, p_d may happen to choose a direction that hits a light source. We will refer to this kind of path as an *implicit path*. Second, the path tracer may connect the eye subpath directly to a point on a light source. We will refer to paths created in this way as *explicit paths*.

Since the ray paths created by a path tracer are Monte Carlo samples of the rendering equation, the path tracer evaluates them in such a way that the expected value of the paths that contribute to a given pixel is equal to the pixel brightness. To see how this is done, consider the path in figure 8 below that connects the eye point to a light source:

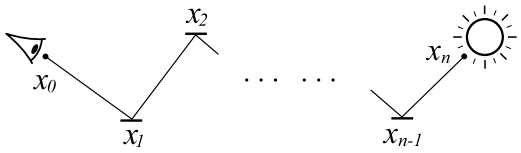


Figure 8: A ray path.

To form an unbiased estimate of the light reaching the eye along direction $x_1 \rightarrow x_0$, the MC sampler in a path tracer multiplies the pertinent terms of the rendering equation together (i.e. f_r, $cos\theta$ and L_e), and divides by the probability that the path was generated by the sampler. For an implicit path, the estimate is given by

$$\frac{L_e(x_n \rightarrow \Psi_n)}{p_{length}(n)} \times \prod_{i=1}^{n-1} \frac{f_r(\Psi_i \leftrightarrow \Theta_i) |\cos\theta_i|}{p_d(\Psi_i \rightarrow \Theta_i)}, \quad (9)$$

where Ψ_i and Θ_i are the incoming and outgoing directions at x_i, θ_i is the angle between Θ_i and the surface normal at x_i, $p_{length}(n)$ is the probability that the MC sampler chose to create a path of length n, and p_d is defined with respect to solid angle.

Explicit paths are evaluated similarly, except that the term $p_d(\Psi_{n-1} \rightarrow \Theta_{n-1})$ is replaced by a term that converts area sampling on the surface of a light source to sampling over the solid angle. Suppose that the ray path in figure 8 was made by connecting an eye subpath to light k. The path would then be evaluted

$$\frac{L_e(x_n \rightarrow \Psi_n)}{p_{length}(n)} \times \prod_{i=1}^{n-2} \frac{f_r(\Psi_i \leftrightarrow \Theta_i) |\cos\theta_i|}{p_d(\Psi_i \rightarrow \Theta_i)} \times \frac{f_r(\Psi_{n-1} \leftrightarrow \Theta_{n-1}) \ |\cos\theta_{n-1}\cos\phi_n|}{p_{light}(k)\ p_{area}(x_n)\ \pi\ d^2}. \quad (10)$$

As can be seen, the first two terms are nearly identical to the implicit case. The third term converts sampling over the surface of light source k to sampling over the solid angle from point x_{n-1}. The new terms in the expression are as follows: $\cos\phi_n$ describes the angle between the normal at point x_n on the light source and the incoming direction Ψ_n; d is the distance between x_{n-1} and x_n; $p_{light}(k)$ is the probability that light k was chosen by the MC sampler, and $p_{area}(x_n)$ is the probability that point x_n was chosen on the light source with respect to surface area.

Monte Carlo path density. The product of all of the terms related to probability in a ray path (p_{length}, p_d, p_{light} and p_{area}) can be thought of as the *path density* in path space with respect to the given MC sampler. In section 4.2 we will use this fact to compute the relative sampling density in different parts of path space.

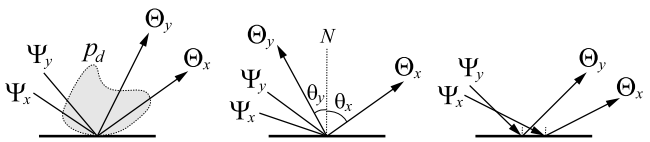
4.2 Mutation and Changes in Path Density

ER path tracing relies on two fundamental sampling steps: an initial Monte Carlo step, and an Energy Redistribution step that uses path mutation. To allow energy flow between a path $\bar{x}$ and a mutated path $\bar{y}$ during the energy redistribution step, the ER sampler must compute the ratio of the path density at $\bar{y}$ to the path density at $\bar{x}$ with respect to the path tracer's sampling routines ($p(\bar{y})/p(\bar{x})$).

[Cline and Egbert 2005] give a set of rules that describe these path density changes for ideal diffuse and specular surfaces sampled in a particular way. Here we give an extended set of rules that are valid for arbitrary BRDFs for the mutation types that we use.

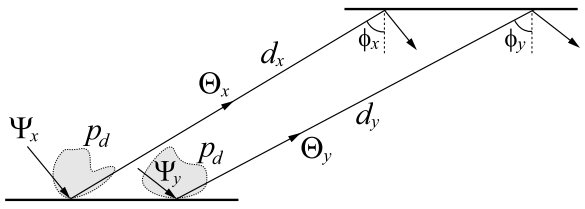
Rule 1: Changes to pixel coordinates. Explicit changes to the pixel coordinates of a path do not change the relative path density, unless the MC sampler samples different image coordinates with different densities. Note that this rule is only applied if the pixel coordinates of the path are explicitly manipulated. Rule 5 handles incidental changes to the pixel coordinates of a path.

Rule 2: Changes to directions. When the MC sampler chooses an outgoing direction at a surface, it does so according to the probability distribution p_d, which was described in section 4.1. Perturbing an outgoing direction at a surface changes the path density in a manner proportional to the relative density of samples taken by the MC sampler in the original outgoing direction and the mutated direction. In the case of a diffuse surface sampled with a cosine-weighted distribution about the normal, the density change is proportional to the ratio of the cosines of the two angles. In the case of an ideal specular surface, perturbing the outgoing direction to lie in the specular direction changes the path density proportional to p_s, the probability that the MC sampler would choose to send a ray in the specular direction. (The value of p_s might change if, for example, the MC sampler uses a Fresnel term to decide whether to send a reflection or refraction ray.) Equation 11 summarizes all three cases.



$$\frac{p_d(\Psi_y \to \Theta_y)}{p_d(\Psi_x \to \Theta_x)} \qquad \frac{|\cos\theta_y|}{|\cos\theta_x|} \qquad \frac{p_s(\Psi_y \to \Theta_y)}{p_s(\Psi_x \to \Theta_x)} \tag{11}$$

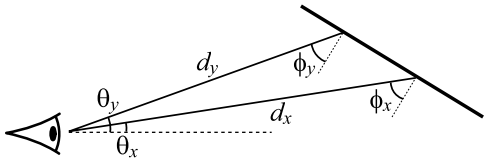
Rule 3: Connecting points in the middle of a path. Connecting two non-specular vertices in a path is a way of sampling surface areas instead of directions, and thus we must convert between area sampling and directional sampling. If we are sampling directions according to a cosine-weighted distribution about the surface normal, the density change is proportional to the familiar geometry term $|\cos\theta\cos\phi/d^2|$. On the other hand, if some other method is used to sample directions, the density change replaces the $\cos\theta$ term with with the probability of sampling different directions, as shown in equation 12.



$$\frac{p_d(\Psi_y \to \Theta_y)|\cos\phi_y|}{d_y^2} \times \frac{d_x^2}{p_d(\Psi_x \to \Theta_x)|\cos\phi_x|} \tag{12}$$

Rule 4: Connecting points to light sources. For explicit paths, connecting to an established or perturbed point on a light source does not change the path density if the light source is sampled uniformly with respect to area. Otherwise the density change is proportional to the ratio of probabilities of choosing the mutated and original points on the light. Implicit paths, on the other hand, incur the same change in path density as connecting points in the middle of a path.

Rule 5: Connecting a point to the viewer. Assuming a pinhole camera model, if a connection is made in which one of the vertices is the eye point, the density change is proportional to the modified geometry term $|\cos\phi/(d^2\cos^3\theta)|$. The actual change in density for this case is given in equation 13.



$$\frac{|\cos\phi_y|}{d_y^2|\cos^3\theta_y|} \times \frac{d_x^2|\cos^3\theta_x|}{|\cos\phi_x|} \tag{13}$$

Applying the density change rules. The path density change rules just described must be applied beginning at the eye point regardless of the manner in which the mutated path was generated. This is a consequence of the fact that we are trying to capture path density changes **with respect to the MC sampler used by the path tracer.**[2] To compute the total density change between two paths of the same length, we apply all of the pertinent rules, and multiply their results together. Appendix A gives several examples of how to apply the rules.

4.3 Mutation Strategies

Our implementation uses two mutation types, which correspond to lens and caustic perturbations as described in [Veach and Guibas 1997]. We are able to get away with such a small set of mutations because the path tracing step provides complete coverage of path space, and roughly distributes path energy over the image plane. Besides our descriptions here, [Veach and Guibas 1997] and [Cline and Egbert 2005] provide descriptions of lens and caustic perturbations as well as other mutation types.

Lens perturbations. A lens perturbation is a mutation that creates a new path $\bar{y}$ from an existing path $\bar{x}$ beginning at the eye point. To start the mutation, the pixel coordinates of $\bar{x}$ are perturbed by a random amount on the image plane.[3] A ray is cast from the eye point through this new pixel coordinate, and the new eye subpath is propagated through the same number and types of specular bounces as the original path, arriving at a non-specular vertex. If the next vertex in the original path is non-specular, $\bar{y}$ is completed by connecting the eye subpath directly to the next vertex in the original path. If the next vertex is specular, however, the outgoing direction from the diffuse vertex is perturbed, and the eye subpath is extended through another specular chain looking for two non-diffuse vertices in a row. This process repeats until either two non-diffuse vertices or the light source are found. Appendix A gives an example lens perturbation, and shows how to compute the relative path density between $\bar{x}$ and $\bar{y}$.

[2] It is not necessary to use the same MC sampler as the path tracer. Any valid sampler will work, as long as it is used to compute all parts of q.

[3] Our implementation mutates uniformly within a small square (9×9 pixels) centered on the current location.

Caustic perturbations. Caustic perturbations are created in much the same way as lens perturbations, except that they start at the light source, or second diffuse vertex in the path (from the eye point). For example, consider the path *LSSDE*. The caustic mutation starts by perturbing the direction $L \rightarrow S$ by a random angle.[4] The new light subpath is propagated through two specular bounces, and arrives at a non-specular vertex, producing the light subpath *LSSD*... . This subpath is then connected directly to the eye point. Note that this type of mutation can only be used on paths in which the eye point is connected to a non-specular vertex. Appendix A gives an example of a caustic perturbation and the path density change that it incurs.

Mutation probabilities. In practice, our strategy is to choose caustic perturbations exclusively for paths of the type *L*...*SDE*, and lens perturbations in all other cases. Another reasonable strategy would be to choose randomly between the two mutation types when they are both valid.

4.4 Noise Filtering

Since ER path tracing is a stochastic process, it can naturally introduce noise into a rendered image. Here we describe two filters that can significantly reduce the noise of images produced by ERPT. Although they are theoretically biased, we have found the filters to be effective at eliminating noise while producing few visible artifacts.

Proposed mutations noise filtering. One of the main causes of noise in ERPT is an imbalance in the number of potential flow events into different pixels. We can compensate for this imbalance by keeping a tally of the number of proposed mutations to each image pixel. This "proposed mutations" image is then blurred to produce an approximate expected number of proposed mutations into each pixel. Our current implementation uses a box filter to compute this "expected proposed mutations" image. The rendered image is then de-noised by scaling the pixel values by the ratio of the expected number of proposed mutations at the pixel to the actual number. Figure 9 shows the "proposed mutations" and "expected proposed mutations" images for a simple scene along with the effect of applying the filter.

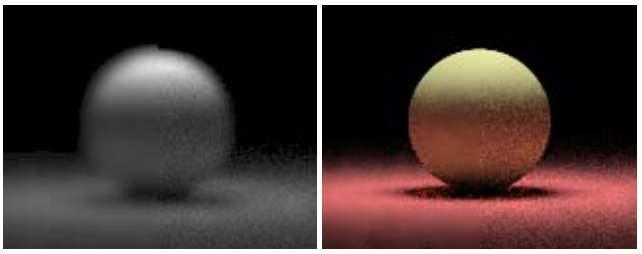
We have been quite pleased with the results of this filter, but there may be room for improvement. For example, a median filter might be a better choice than a box filter to smooth the "proposed mutations" image. Also, since the expected proposed mutations image is essentially a convolution of the luminance image, it may be better to blur the unfiltered luminance to produce the "expected proposed mutations" image.

Consecutive sample filtering. A second noise filter that works well in practice is to refuse to accumulate more than a small number of consecutive samples on a given pixel, say 10 or 20. During energy redistribution, the ERPT sampler counts how many times in a row a sample chain deposits energy onto the same pixel. Once the maximum number has been reached, the sampler continues mutating the sample chain, but it throws away any energy that would normally be deposited until the sample chain migrates off of the offending pixel, at which time energy deposition begins again. This filter tends to clean up speckles that occur when the sampler gets stuck on a given pixel.

5 Results

This section demonstrates different aspects of the ERPT algorithm, and compares ERPT to standard path tracing and MLT. To make the comparison as fair as possible, the functions that mutate paths are shared between MLT and ERPT.

[4]When perturbing the angle, we follow the formulation of [Veach and Guibas 1997], and mutate in an exponential distribution between 0.0001 and 0.1 radians. [Cline and Egbert 2005] describes this procedure in detail.



(a) (b)

Figure 9: Proposed mutation noise filtering. (a) The right half of the image shows the number of proposed mutations into each pixel, the "proposed mutations" image. The left half of the image was produced by convolving the proposed mutations image with a 7×7 smoothing kernel. This is the "expected proposed mutations" image. (b) The left half of the image has been smoothed with our proposed mutations noise filter, and the right half has not. In spite of the large kernel applied to determine the expected number of proposed mutations, edges in the filtered image remain sharp. Very large kernel sizes can produce ringing artifacts, however.

Comparison of flow rules. We experimented with the different flow filters described in section 3, plugging them in as the flow filter for our ER path tracer. One of these experiments is summarized in figure 10. In virtually every case, the equal deposition flow rule was superior to the others, so we use it for all of the other examples in the paper.

Figure 10: The effect of different flow rules. The images show the indirect lighting from the scene in figure 9, using different flow filters for the Energy Redistribution step of ERPT. (Left) The detailed balance flow rule works poorly because energy cannot migrate away from the MC sample site. (Middle) An iterated version of detailed balance works better, but the splitting sample chains produced by the rule are too short to spread evenly over the image plane. (Right) Equal deposition flow, while not perfect at this sample density, spreads the energy of the MC samples more evenly than the other two rules.

Sample chain length. We have found that the sample chains emanating from the Monte Carlo samples need to be fairly long to produce good results. Values between about one hundred and one thousand seem to work well. As a general rule, very short chains produce an uneven appearance, and very long chains start to lose the stratifying properies of the initial MC samples as more and more of the initial samples do not have any sample chains assigned to them. Figure 11 shows the effect of varying the sample chain length.

Approximate stratification of path space. In most lighting situations, MLT and ERPT produce similar results. However, ERPT tends to stratify a little better over the image plane. For example, consider the images in figure 12 showing indirect lighting of a yellow dragon placed on a red ground plane. The left image gives the desired result, computed by path tracing using a large number of samples. The middle image was produced by MLT using two mutations per pixel, and the right image was produced with ER path tracing using one Monte Carlo sample and one mutation per pixel.

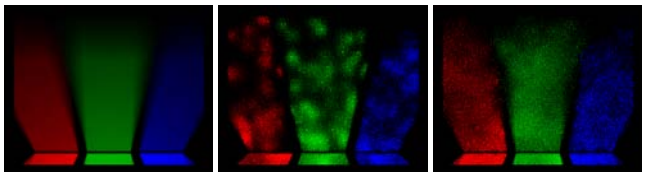


Figure 11: Sample chain length. In the scene above, light reflects off of three colored mirrors onto a diffuse wall. The desired image (left) is reproduced with ERPT using sample chains of length 10 (middle), and 100 (right).

Note that the ERPT image more faithfully reproduces the contours of the dragon than MLT, and the lighting in the MLT image has more bright artifacts. This is because ERPT creates an initial distribution of energy that covers path space evenly. By contrast, MLT must rely on large mutations to fully account for all contributing paths.

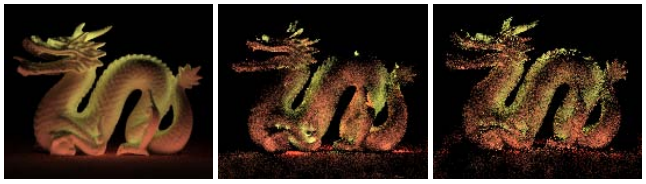


Figure 12: Comparison of stratification over the image plane between MLT (middle) using 2 samples per pixel and ERPT (right) using one MC sample and one mutation per pixel.

Indirect lighting example. Figure 13 shows renderings of a gallery scene with strong indirect lighting, taking approximately equal time, for path tracing, MLT and ERPT. The only direct illumination visible in the scene comes from the overhead spot light shining on the dragon. Image (a) was computed with standard path tracing. Since path tracing cannot coordinate sampling efforts between pixels, it wastes a lot of time sampling unimportant regions of the path space, resulting in a very noisy image. Image (b) was produced with Metropolis Light Transport. MLT is able to coordinate sampling efforts between pixels, producing a better image; however, some noise is still visible. The bottom row shows ERPT without (c) and with (d) the noise filters described in section 4.4. ERPT without the noise filters achieves a slightly better result than MLT in this scene. Adding the flow filters further reduces the noise, producing a much smoother result.

Difficult caustic lighting. Figure 14 compares ERPT to MLT and path tracing for a difficult lighting situation in which a large portion of the lighting comes from implicit caustic paths seen through a glass surface. Even the "direct" lighting on the torus is made up of these paths. (b) Path tracing produces a very noisy image. (c) MLT does much better, but still results in a splotchy appearance. A more full set of mutation strategies might remedy this problem; however, ERPT using the exact same set of mutations (d) has a much smoother appearance. This is because the deposition energy of any given sample chain is limited. Image (e) shows the effect of applying the noise filters from section 4.4. The top image (a) shows a higher quality ERPT rendering of the same scene.

6 Conclusion and Ideas for Future Study

This paper presented Energy Redistribution path tracing as an algorithm to solve the general global illumination problem. The algorithm has at its core a novel sampling technique called Energy Redistribution sampling, which can efficiently solve correlated integral problems. We compared images generated by ER path tracing to images created with standard path tracing and Metropolis Light Transport, and demonstrated several situations in which the new algorithm outperforms standard path tracing and MLT.

In addition to the algorithmic contributions of the paper, we feel that some of the most important contributions of this work are pedagogical. The concepts of correlated integrals and energy flow offer profound insights into the inner workings of Metopolis as well as ER sampling, and a new perspective from which to view numerical intergration problems in general.

We have only scratched the surface in comparing ER to Metropolis sampling. Because it works directly with function energy, we have found the ER sampling framework to be more malleable than Metropolis sampling. For example, ER sampling can be easily adapted to handle negative-valued functions as well as all positive ones. Are there domains besides global illumination that would benefit from ER sampling rather than Metropolis sampling? In what situations does ER sampling work better?

There are a number of questions that remain in regards to ER sampling flow rules. For example, are there ways to define better flow rules based on the less restrictive general balance condition rather than detailed balance? What is the optimal tradeoff between Monte Carlo samples and ER samples? Can flow rules and mutation strategies be defined which stratify better?

Noise filtering also seems to be a good avenue for further exploration. We were quite pleased with the results of the two simple filters presented in the paper, and are currently looking at the use of other simulation statistics besides "proposed mutations" to reduce the noise. Finally, we note that the noise filters that were defined in this paper are biased. Ideally, we would like to create effective noise filters that are unbiased.

Acknowledgements. We would like to thank the Siggraph reviewers for their many helpful comments and suggestions, the Stanford 3D Scanning Repository for the dragon model, and everyone in the "Siggraph support group" who read early drafts of the paper.

References

ASHIKHMIN, M., PREMOŽE, S., SHIRLEY, P., AND SMITS, B. 2001. A Variance Analysis of the Metropolis Light Transport Algorithm. *Computers and Graphics 25*, 2, 287–294.

CLINE, D., AND EGBERT, P. 2005. A Practical Introduction to Metropolis Light Transport. *Technical Report: Department of Computer Science, Brigham Young University*, 1–19.

DUTRÉ, P., BEKAERT, P., AND BALA, K. 2003. *Advanced Global Illumination*. A. K. Peters.

JENSEN, H. W. 1996. Global Illumination Using Photon Maps. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, Springer-Verlag/Wien, New York, NY, 21–30.

KAJIYA, J. T. 1986. The Rendering Equation. In *Computer Graphics (Proceedings of ACM SIGGRAPH 86)*, ACM Press, 20, 4, 143–150.

KELEMEN, C., SZIRMAY-KALOS, L., ANTAL, G., AND CSONKA, F. 2002. A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm. *Computer Graphics Forum 21*, 3, 1–10.

LAFORTUNE, E. P., AND WILLEMS, Y. D. 1993. Bi-directional Path Tracing. In *Proceedings of the Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, H. P. Santo, Ed., 145–153.

LAWRENCE, J., RUSINKIEWICZ, S., AND RAMAMOORTHI, R. 2004. Efficient BRDF Importance Sampling Using a Factored Representation. *ACM Transactions on Graphics 23*, 3, 494–503.

METROPOLIS, N., ROSENBLUTH, A., ROSENBLUTH, M., TELLER, A., AND TELLER, E. 1953. Equations of State Calculations by Fast Computing Machines. *Chemical Physics 21*, 1087–1091.

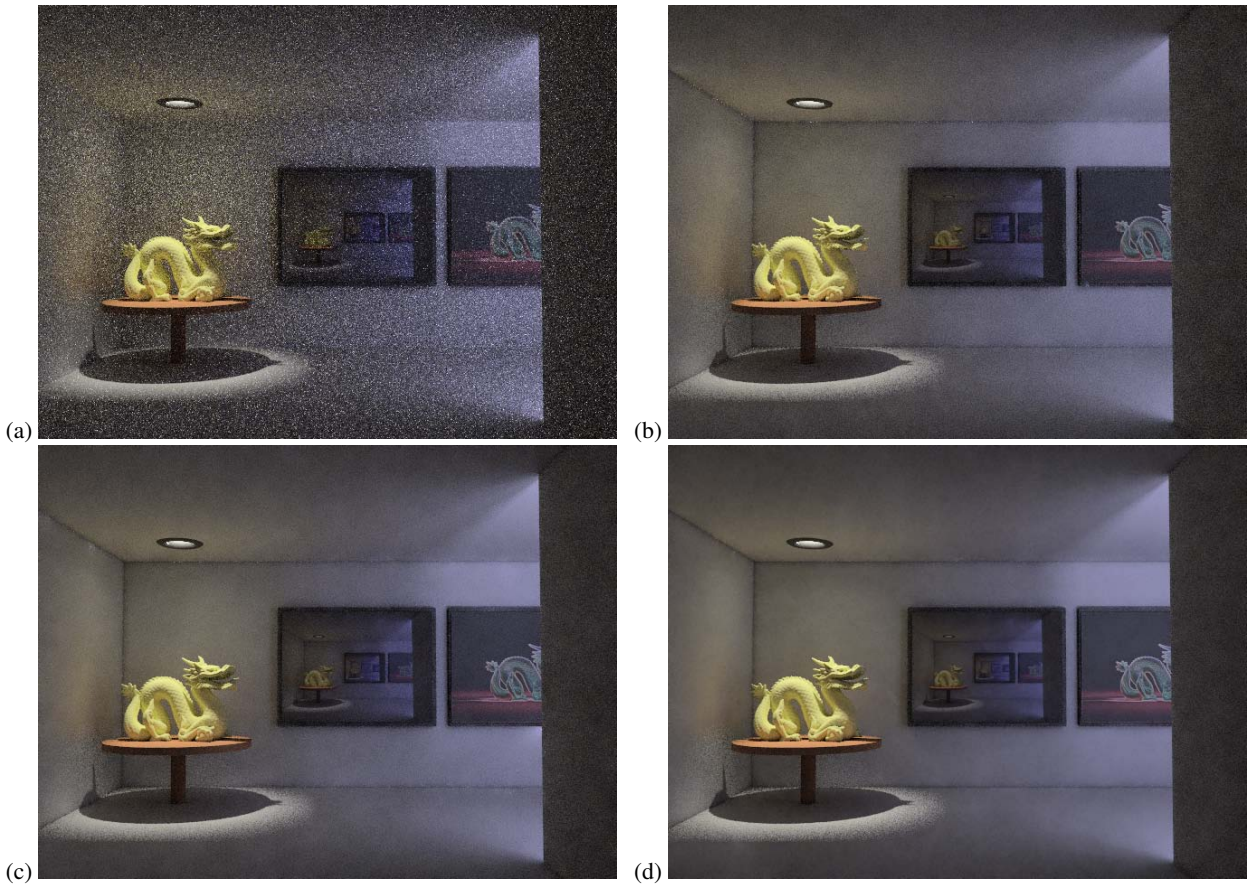


Figure 13: Comparison of a scene with strong indirect lighting. All render times are approximately 20 minutes. (a) Standard path tracing with 84 paths per pixel has a hard time finding the indirect lighting paths, resulting in a very noisy image. (b) MLT with 200 mutations per pixel produces a much better result, but some noise is still visible. (c) ERPT with 36 MC samples and 200 mutations per pixel achieves a slight improvement over MLT. (d) Adding the noise filters described in section 4.4 to ERPT removes most of the remaining noise.

PAULY, M., KOLLIG, T., AND KELLER, A. 2000. Metropolis Light Transport for Participating Media. In *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, Springer Wien, New York, NY, B. Peroche and H. Rushmeier, Eds., 11–22.

PHARR, M. 2003. Chapter 9: Metropolis Sampling. In *Monte Carlo Ray Tracing, SIGGRAPH 2003 Course 44, course notes*.

SHIRLEY, P., WADE, B., HUBBARD, P. M., ZARESKI, D., WALTER, B., AND GREENBERG, D. P. 1995. Global Illumination via Density Estimation. In *Rendering Techniques 1995 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, Springer-Verlag, New York, NY, P. M. Hanrahan and W. Purgathofer, Eds., 219–230.

SZIRMAY-KALOS, L., DORNBACH, P., AND PURGATHOFER, W. 1999. On the Start-up Bias Problem of Metropolis Sampling. *Technical Report: Department of Control Engineering and Information Technology, Technical University of Budapest*, 1–8.

VEACH, E., AND GUIBAS, L. J. 1994. Bidirectional Estimators for Light Transport. In *Rendering Techniques 1994 (Proceedings of the Fifth Eurographis Workshop on rendering)*, 147–162.

VEACH, E., AND GUIBAS, L. J. 1995. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Proceedings of ACM SIGGRAPH 1995*, ACM Press, 419–428.

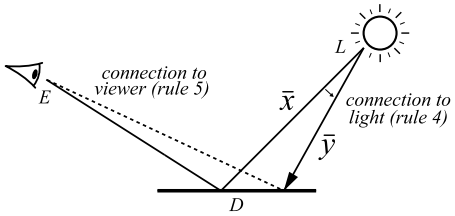
VEACH, E., AND GUIBAS, L. J. 1997. Metropolis Light Transport. In *Proceedings of ACM SIGGRAPH 1997*, ACM Press, 65–76.

WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. 1988. A Ray Tracing Solution for Diffuse Interreflection. In *Computer Graphics (Proceedings of ACM SIGGRAPH 88)*, ACM, 22, 4, 85–92.

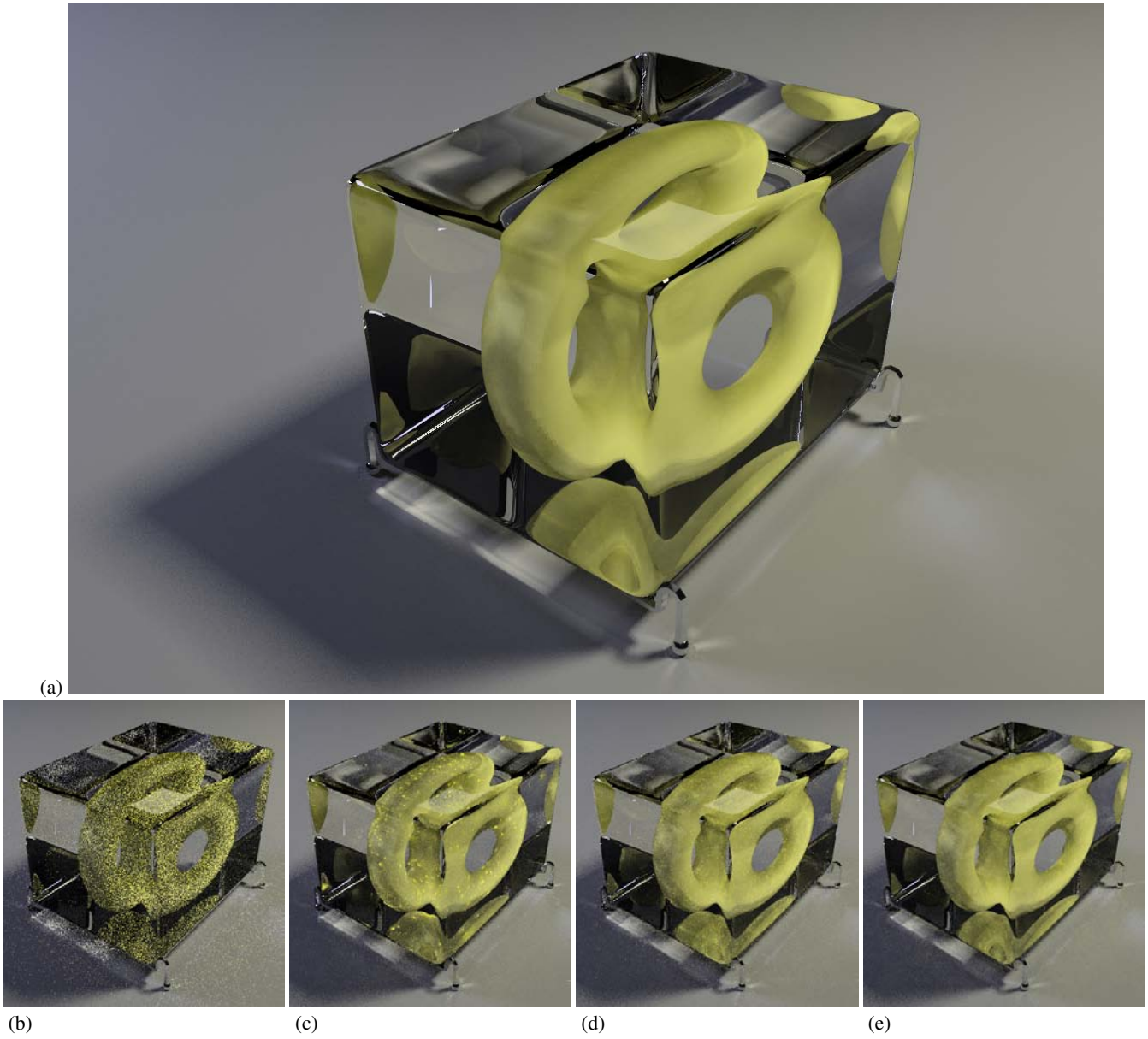
A Computing Path Density Change

Since computing the change in path density is essential to ERPT, we give several examples of how to do it here.

As a first example, let us assume that we are mutating a path of the form *LDE* using a caustic mutation, as shown below:



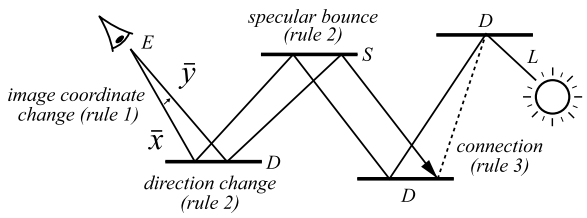
The direction starting at the light source is perturbed, and a ray is cast from the light source in this direction, creating the light subpath *LD*.... This subpath is connected to the eye point, once again creating a path of the form *LDE*. Note that even though the mutation was generated starting at the light source, the path density change rules must be applied starting at the eye point. From the eye point, we apply the following rules: A connection was made from the eye point, so we apply rule 5. Now note that even though the mutation was generated by perturbing a direction from the light source, from the point of view of the path tracer, we must make a direct connection to the light source from the *D* vertex. Thus, we apply rule 4.



(a) (b) (c) (d) (e)

Figure 14: Difficult caustic lighting. In this scene, a large portion of the light transport comes from implicit "caustic" paths. (b) Path tracing with 100 paths per pixel produces a very noisy image. (c) MLT, using 100 mutations per pixel, gets stuck on some of the caustic paths, producing a splotchy appearance. (d) ERPT using 36 MC samples and 50 mutations per pixel. Although some bright spots are visible, they are much less pronounced than in the MLT case. This is so despite the fact that both algorithms use the same mutation strategies. (e) Adding the noise filters to ERPT removes most of the small speckles in the image. (a) A high quality ERPT rendering of the scene using 192 MC samples and 800 mutations per pixel, again using the noise filters. The bottom row images were rendered at a 640×480 resolution in about fifteen minutes, and the top image was rendered at a resolution of 1200×800 in about seven and a half hours on a 3.2 Ghz Pentium 4.

As another example, consider a lens subpath mutation on a path of the form *LDDSDE*:



In this case, the pixel coordinates of the ray from the eye are changed, and we cast a ray in the new direction. The ray hits a diffuse surface, which is followed by a specular surface. In this case, the outgoing direction from this *D* vertex is perturbed, and extended through a specular bounce to produce the eye subpath of the form $\ldots DSDE$. This eye subpath is then connected directly to the next vertex in the path, to once again produce a path of the form *LDDSDE*. Now we apply the density change rules. First, we changed the pixel coordinates, so we apply rule 1. Next, we perturbed the outgoing direction, so we apply rule 2. Then, we extended the path through a specular bounce, so we apply rule 2 again. Finally, we connected two diffuse vertices in the middle of the path, so we apply rule 3.