Chapter 10

Bidirectional Path Tracing

In this chapter, we describe a new light transport algorithm called *bidirectional path tracing*. This algorithm is a direct combination of the ideas in the last two chapters: namely, expressing light transport as an integration problem, and then applying more than one importance sampling technique to evaluate it. The resulting algorithm handles arbitrary geometry and materials, is relatively simple to implement, and can handle indirect lighting problems far more efficiently and robustly than ordinary path tracing.

To sample each transport path, we generate one subpath starting from a light source, a second subpath starting from the eye, and join them together. By varying the number of vertices generated from each side, we obtain a family of sampling techniques for paths of all lengths. Each sampling technique has a different probability distribution over the space of paths, and takes into account a different subset of the factors of the integrand (i.e. the measurement contribution function). Samples from all of these techniques are then combined using multiple importance sampling.

This chapter is organized as follows. We start in Section 10.1 with an overview of the bidirectional path tracing algorithm. This is followed by a more detailed mathematical description in Section 10.2, where we derive explicit formulas for the sample contributions. Section 10.3 then discusses the issues that arise when implementing the algorithm, including how to generate the subpaths and evaluate their contributions efficiently, how to handle specular materials, and how to implement the important special cases where the light or eye subpath contains less than two vertices. In Section 10.4 we describe an important optimization to reduce the number of visibility tests, using a new technique called *efficiency-optimized Russian roulette*. Section 10.5 then presents some results and measurements of the algorithm, Section 10.6 compares our algorithm to other related work in this area, and Section 10.7 summarizes our conclusions.

10.1 Overview

Recall that according to the path integral framework of Chapter 8, each measurement can be written in the form

$$I_j \;=\; \int_\Omega f_j(\bar{x})\, d\mu(\bar{x})\,, \tag{10.1}$$

where $\bar{x} = \mathbf{x}_0 \ldots \mathbf{x}_k$ is a path, Ω is the set of such paths (of any length), μ is the area-product measure $d\mu(\bar{x}) \;=\; dA(\mathbf{x}_0) \,\cdots\, dA(\mathbf{x}_k)$, and f_j is the measurement contribution function

$$\begin{aligned} f_j(\bar{x}) \;&=\; L_{\mathrm{e}}(\mathbf{x}_0 \!\to\! \mathbf{x}_1)\, G(\mathbf{x}_0 \!\leftrightarrow\! \mathbf{x}_1)\, W_{\mathrm{e}}^{(j)}(\mathbf{x}_{k-1} \!\to\! \mathbf{x}_k) \\ &\quad \cdot \prod_{i=1}^{k-1} f_{\mathrm{s}}(\mathbf{x}_{i-1} \!\to\! \mathbf{x}_i \!\to\! \mathbf{x}_{i+1})\, G(\mathbf{x}_i \!\leftrightarrow\! \mathbf{x}_{i+1})\,. \end{aligned} \tag{10.2}$$

Bidirectional path tracing consists of a family of different importance sampling techniques for this integral. Each technique samples a path by connecting two independently generated pieces, one starting from the light sources, and the other from the eye. For example, in Figure 10.1 the *light subpath* $\mathbf{x}_0\mathbf{x}_1$ is constructed by choosing a random point $\mathbf{x}_0$ on a light source, followed by casting a ray in a random direction to find $\mathbf{x}_1$. The *eye subpath* $\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4$ is constructed by a similar process starting from a random point $\mathbf{x}_4$ on the camera lens. A complete transport path is formed by concatenating these two pieces. (Note that the integrand may be zero on this path, e.g. if $\mathbf{x}_1$ and $\mathbf{x}_2$ are not mutually visible.)

By varying the number of vertices in the light and eye subpaths, we obtain a family of sampling techniques. Each technique generates paths of a specific length k, by randomly generating a light subpath with s vertices, randomly generating an eye subpath with t vertices, and concatenating them (where $k \;=\; s + t \,-\, 1$). It is important to note that there is more than one sampling technique for each path length: in fact, for a given length k it is easy to see that there are $k + 2$ different sampling techniques (by letting $s = 0, \ldots, k + 1$).

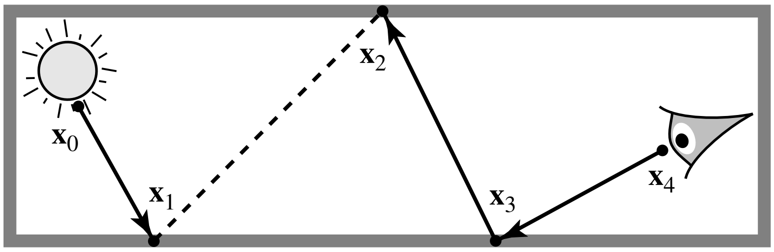
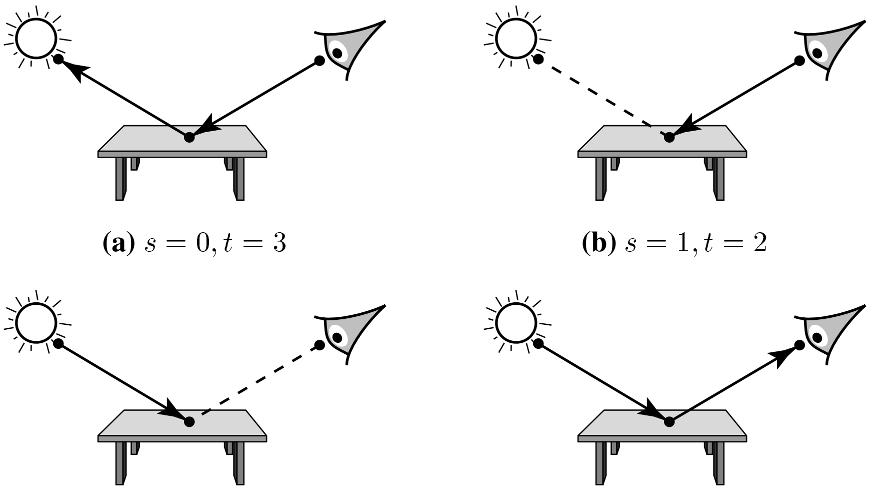


Figure 10.1: A transport path from a light source to the camera lens, created by concatenating two separately generated pieces.



(a) $s = 0, t = 3$ **(b)** $s = 1, t = 2$

(c) $s = 2, t = 1$ **(d)** $s = 3, t = 0$

Figure 10.2: The four bidirectional sampling techniques for paths of length $k = 2$. Intuitively, they can be described as **(a)** Monte Carlo path tracing with no special handling of light sources, **(b)** Monte Carlo path tracing with a direct lighting calculation, **(c)** tracing photons from the light sources and recording an image sample whenever a photon hits a visible surface, and **(d)** tracing photons and recording an image sample only when photons hit the camera lens. Note that technique (a) can only be used with an area light source, while technique (d) can only be used with a finite-aperture lens.

These techniques generate different probability distributions on the space of paths, which makes them useful for sampling different kinds of effects. For example, although technique (b) works well under most circumstances (for paths of length two), technique (a) can be superior if the table is very glossy or specular. Similarly, techniques (c) or (d) can have the lowest variance if the light source is highly directional.

Figure 10.2 illustrates the four bidirectional sampling techniques for paths of length $k = 2$.

The reason that these techniques are useful is that they correspond to different density functions $p_{s,t}$ on the space of paths. All of these density functions are good candidates for importance sampling, because they take into account different factors of the measurement contribution function f_j (as we will explain below). In practical terms, this means that each technique can efficiently sample a different set of lighting effects.

To take advantage of this, bidirectional path tracing generates samples using all of the techniques $p_{s,t}$ and combines them using multiple importance sampling. Specifically, the following estimate is computed for each measurement I_j:

$$F = \sum_{s \geq 0} \sum_{t \geq 0} w_{s,t}(\bar{x}_{s,t}) \frac{f_j(\bar{x}_{s,t})}{p_{s,t}(\bar{x}_{s,t})} . \tag{10.3}$$

Here $\bar{x}_{s,t}$ is a path generated according to the density function $p_{s,t}$, and the weighting functions $w_{s,t}$ represent the combination strategy being used (which is assumed to be one of the provably good strategies in Chapter 9, such as the balance heuristic). By combining samples from all the bidirectional techniques in this way, a wide variety of scenes and lighting effects can be handled well.

Efficiently generating the samples. So far, we have assumed that all the paths $\bar{x}_{s,t}$ are sampled independently, by generating a separate light and eye subpath for each one. However, in practice it is important to make the sampling more efficient. This is achieved by generating the samples in groups. For each group, we first generate a light subpath

$$\mathbf{y}_0 \cdots \mathbf{y}_{n_L - 1}$$

with n_L vertices, and an eye subpath

$$\mathbf{z}_{n_E - 1} \cdots \mathbf{z}_0$$

with n_E vertices (where $\mathbf{y}_0$ is a point on a light source, and $\mathbf{z}_0$ is a point on the camera lens). The length of each subpath is determined randomly, by defining a probability for terminating the subpath at each vertex (details are given in Section 10.3.3). We can then take samples from a whole group of techniques $p_{s,t}$ at once, by simply joining each prefix of the light subpath to each suffix of the eye subpath. The sample from $p_{s,t}$ is taken to be

$$\bar{x}_{s,t} \;=\; \mathbf{y}_0 \ldots \mathbf{y}_{s-1}\, \mathbf{z}_{t-1} \ldots \mathbf{z}_0 \,,$$

which is a path with $s+t$ vertices and $k = s+t-1$ edges (where $0 \le s \le n_L$, $0 \le t \le n_E$, and $k \ge 1$). The vertices $\mathbf{y}_{s-1}$ and $\mathbf{z}_{t-1}$ are called the *connecting vertices*, and the edge between them is the *connecting edge*.

The contributions of all the samples $\bar{x}_{s,t}$ are then computed and summed according to the multi-sample estimator (10.3). In order to evaluate the contribution of each path, the visibility of the connecting edge must be tested (except when $s = 0$ or $t = 0$). If the connecting edge is obstructed, or if the BSDF at either connecting vertex does not scatter any light toward the other, then the contribution for that path is zero. (The following section gives further details.)

There is an important detail that we have not mentioned yet. Notice that we have modeled the multi-sample estimator (10.3) as a sum over an infinite number of samples, one from each bidirectional technique $p_{s,t}$. We did this because of the way that multiple importance sampling was defined: it assumes that an integer number of samples $n_{s,t}$ is taken from each sampling technique, so in this case we set $n_{s,t} = 1$ for all s, t. (Note that if we placed an upper bound on the allowable values of s and t, the result would be biased.) Of course, the strategy above does not take a sample from all of the techniques $p_{s,t}$, since there are an infinite number of them. However, notice that there is always some finite *probability* of taking a sample from each technique, no matter how large s and t are. This is because for any given values of s and t, there is some probability of generating a light subpath with $n_L \ge s$ and an eye subpath with $n_E \ge t$ (since there lengths are chosen randomly).

Formally, we can show how this corresponds to the multi-sample model as follows. First we introduce the notion of an *empty path* ϵ, which is defined to have a contribution of zero. We then re-interpret the strategy above to be method for sampling all of the techniques $p_{s,t}$ simultaneously, by defining the sample from $p_{s,t}$ to be $\bar{x}_{s,t} = \epsilon$ whenever $s > n_L$ or $t > n_E$. In other words, although the estimator (10.3) is formally a combination of samples from an infinite number of techniques, in fact all but a finite number of them will be the empty path ϵ on each evaluation, so that their contributions can be ignored. Another way of interpreting this is to say that the density functions $p_{s,t}$ are allowed to integrate to less than one, since any remaining probability can be assigned to the empty path. (Notice that having an infinite number of sampling techniques does not cause any problems when computing the weights $w_{s,t}(\bar{x}_{s,t})$, since there are only $k + 2$ sampling techniques that can generate paths of any given length k.)

10.2 Mathematical formulation

In this section we derive the formulas for determining the contribution of each sample, and we show how to organize the calculations so that they can be done efficiently.

Letting $\bar{x}_{s,t}$ be the sample from technique $p_{s,t}$, we must evaluate its contribution

$$C_{s,t} \;\equiv\; w_{s,t}(\bar{x}_{s,t}) \, \frac{f_j(\bar{x}_{s,t})}{p_{s,t}(\bar{x}_{s,t})}$$

to the estimator (10.3), which can be rewritten as

$$F \;=\; \sum_{s \geq 0} \sum_{t \geq 0} C_{s,t} \,.$$

We will evaluate this contribution in several stages. First, we define the *unweighted contribution* $C^*_{s,t}$ as

$$C^*_{s,t} \;\equiv\; \frac{f_j(\bar{x}_{s,t})}{p_{s,t}(\bar{x}_{s,t})} \,.$$

We will show how to write this as a product

$$C^*_{s,t} \;=\; \alpha^L_s \, c_{s,t} \, \alpha^E_t \,,$$

where the factor α^L_s depends only on the light subpath, α^E_t depends only on the eye subpath, and $c_{s,t}$ depends only on the connecting edge $\mathbf{y}_{s-1}\mathbf{z}_{t-1}$. The weighted contribution then has the form

$$C_{s,t} \;=\; w_{s,t} \, C^*_{s,t} \,,$$

where $w_{s,t}$ depends on the probabilities with which all the other sampling techniques generate the given path $\bar{x}_{s,t}$.

We now discuss how to compute these factors in detail.

The density $\mathbf{p_{s,t}}$. We start by showing how to compute the probability density

$$p_{s,t} \;\equiv\; p_{s,t}(\bar{x}_{s,t})$$

with which the path $\bar{x}_{s,t}$ was generated. As previously discussed in Chapter 8.2, this is simply the product of the densities $P_A(\mathbf{x}_i)$ with which the individual vertices are generated (measured with respect to surface area). The vertex $\mathbf{y}_0$ is chosen directly on the surface of a light source, so that $P_A(\mathbf{y}_0)$ can be computed directly (and similarly for $\mathbf{z}_0$).

The remaining vertices $\mathbf{y}_i$ are chosen by sampling a direction and casting a ray from the current subpath endpoint $\mathbf{y}_{i-1}$. We let $P_{\sigma^\perp}(\mathbf{y}_{i-1} \to \mathbf{y}_i)$ denote the density for choosing the direction from $\mathbf{y}_{i-1}$ to $\mathbf{y}_i$, measured with respect to projected solid angle.[1] Now the density $P_A(\mathbf{y}_i)$ for choosing vertex $\mathbf{y}_i$ is simply

$$P_A(\mathbf{y}_i) \;=\; P_{\sigma^\perp}(\mathbf{y}_{i-1} \!\to\! \mathbf{y}_i)\, G(\mathbf{y}_{i-1} \!\leftrightarrow\! \mathbf{y}_i)$$

recalling that

$$G(\mathbf{x} \!\leftrightarrow\! \mathbf{x}') \;=\; V(\mathbf{x} \!\leftrightarrow\! \mathbf{x}')\, \frac{|\cos(\theta_{\mathrm{o}})\; \cos(\theta'_{\mathrm{i}})|}{\|\mathbf{x} - \mathbf{x}'\|^2}$$

(see Section 8.2.2.2 for further details).

We define symbols p_i^L and p_i^E to represent the probabilities for generating the first i vertices of the light and eye subpaths respectively. These are defined by

$$\begin{aligned}
p_0^L &= 1\,, \\
p_1^L &= P_A(\mathbf{y}_0)\,, \\
p_i^L &= P_{\sigma^\perp}(\mathbf{y}_{i-2} \!\to\! \mathbf{y}_{i-1})\, G(\mathbf{y}_{i-2} \!\leftrightarrow\! \mathbf{y}_{i-1})\, p_{i-1}^L \qquad \text{for } i \ge 2\,,
\end{aligned}$$

and similarly

$$\begin{aligned}
p_0^E &= 1\,, \\
p_1^E &= P_A(\mathbf{z}_0)\,, \\
p_i^E &= P_{\sigma^\perp}(\mathbf{z}_{i-2} \!\to\! \mathbf{z}_{i-1})\, G(\mathbf{z}_{i-2} \!\leftrightarrow\! \mathbf{z}_{i-1})\, p_{i-1}^E \qquad \text{for } i \ge 2\,.
\end{aligned}$$

[1] More precisely, it should be written as $P_{\sigma^\perp}(\mathbf{y}_{i-1} \to \mathbf{y}_i \mid \mathbf{y}_{i-2}, \mathbf{y}_{i-1})$, since the probability is conditional on the locations of the previous two vertices in the subpath.

Using these symbols, the density for generating the path $\bar{x}_{s,t} = \mathbf{y}_0 \ldots \mathbf{y}_{s-1}\, \mathbf{z}_{t-1} \ldots \mathbf{z}_0$ is simply

$$p_{s,t}(\bar{x}_{s,t}) \;=\; p^L_s \, p^E_t \,. \tag{10.4}$$

The unweighted contribution $\mathbf{C}^*_{\mathbf{s},\mathbf{t}}$. Next, we consider the unweighted contribution

$$C^*_{s,t} \;\equiv\; \frac{f_j(\bar{x}_{s,t})}{p_{s,t}(\bar{x}_{s,t})} \,. \tag{10.5}$$

To calculate this quantity efficiently, we precompute the weights α^L_i and α^E_i given below. These weights consist of all the factors of the definition (10.5) that can be computed using the first i vertices of the light and eye subpaths respectively. Specifically, we have

$$\begin{aligned}
\alpha^L_0 &= 1\,, \\
\alpha^L_1 &= \frac{L^{(0)}_{\mathrm{e}}(\mathbf{x}_0)}{P_A(\mathbf{y}_0)}\,, \\
\alpha^L_i &= \frac{f_{\mathrm{s}}(\mathbf{y}_{i-3} \to \mathbf{y}_{i-2} \to \mathbf{y}_{i-1})}{P_{\sigma^\perp}(\mathbf{y}_{i-2} \to \mathbf{y}_{i-1})}\, \alpha^L_{i-1} \qquad \text{for } i \geq 2\,,
\end{aligned} \tag{10.6}$$

and similarly

$$\begin{aligned}
\alpha^E_0 &= 1\,, \\
\alpha^E_1 &= \frac{W^{(0)}_{\mathrm{e}}(\mathbf{z}_0)}{P_A(\mathbf{z}_0)}\,, \\
\alpha^E_i &= \frac{f_{\mathrm{s}}(\mathbf{z}_{i-1} \to \mathbf{z}_{i-2} \to \mathbf{z}_{i-3})}{P_{\sigma^\perp}(\mathbf{z}_{i-2} \to \mathbf{z}_{i-1})}\, \alpha^E_{i-1} \qquad \text{for } i \geq 2\,.
\end{aligned} \tag{10.7}$$

Here we have used the conventions previously described in Section 8.3.2: the emitted radiance is split into a product

$$L_{\mathrm{e}}(\mathbf{y}_0 \to \mathbf{y}_1) \;=\; L^{(0)}_{\mathrm{e}}(\mathbf{y}_0)\, L^{(1)}_{\mathrm{e}}(\mathbf{y}_0 \to \mathbf{y}_1)\,,$$

where $L^{(0)}_{\mathrm{e}}$ and $L^{(1)}_{\mathrm{e}}$ represents the spatial and directional components of L_{e} respectively, and we define $f_{\mathrm{s}}(\mathbf{y}_{-1} \to \mathbf{y}_0 \to \mathbf{y}_1) \equiv L^{(1)}_{\mathrm{e}}(\mathbf{y}_0 \to \mathbf{y}_1)$. The quantities $W^{(0)}_{\mathrm{e}}$ and $f_{\mathrm{s}}(\mathbf{z}_1 \to \mathbf{z}_0 \to \mathbf{z}_{-1}) \equiv W^{(1)}_{\mathrm{e}}$ are defined similarly. The purpose of this convention is to reduce the number of special cases that need to be considered, by interpreting the directional component of emission as a BSDF. Also, notice that the geometry factors $G(\mathbf{x} \leftrightarrow \mathbf{x}')$ do not appear in the formulas for α_i^L and α_i^E, because these factors occur in both the numerator and denominator of (10.5) (see the definitions of p_i^L and p_i^E).

As mentioned above, the unweighted contribution can now be computed as

$$C_{s,t}^* \;=\; \alpha_s^L \, c_{s,t} \, \alpha_t^E \,, \tag{10.8}$$

where $c_{s,t}$ consists of the remaining factors of the integrand f_j that are not included in the precomputed weights. Examining the definitions of f_j, α_i^L, and α_i^E, we obtain

$$\begin{aligned} c_{0,t} &= L_\mathrm{e}(\mathbf{z}_{t-1} \to \mathbf{z}_{t-2}) \,, \\ c_{s,0} &= W_\mathrm{e}(\mathbf{y}_{s-2} \to \mathbf{y}_{s-1}) \,, \quad \text{and} \\ c_{s,t} &= f_\mathrm{s}(\mathbf{y}_{s-2} \to \mathbf{y}_{s-1} \to \mathbf{z}_{t-1}) \, G(\mathbf{y}_{s-1} \leftrightarrow \mathbf{z}_{t-1}) \, f_\mathrm{s}(\mathbf{y}_{s-1} \to \mathbf{z}_{t-1} \to \mathbf{z}_{t-2}) \\ & \qquad\qquad\qquad\qquad\qquad\qquad \text{for } s, t > 0 \,. \end{aligned}$$

Note that the factor $G(\mathbf{y}_{s-1} \leftrightarrow \mathbf{z}_{t-1})$ includes a visibility test (for the case $s, t > 0$), which is the most expensive aspect of the evaluation.

The weighting function $\mathbf{w_{s,t}}$. Finally we consider how to evaluate

$$w_{s,t} \;\equiv\; w_{s,t}(\bar{x}_{s,t}) \,,$$

whose value depends on the probability densities with which $\bar{x}$ is generated by all of the $s + t + 1$ possible sampling techniques for paths of this length. We define p_i as the density for generating $\bar{x}_{s,t}$ using a light subpath with i vertices, and an eye subpath with $s + t - i$ vertices:

$$p_i \;=\; p_{i,(s+t)-i}(\bar{x}_{s,t}) \qquad \text{for } i = 0, \ldots, s+t \,.$$

In particular, p_s is the probability with which the given path was actually generated, while $p_0 \ldots p_{s-1}$ and $p_{s+1} \ldots p_{s+t}$ represent all the other ways that this path *could* have been generated.

The evaluation of the p_i can be simplified by observing that their values only matter up to an overall scale factor. For example, if the samples are combined using the power heuristic with $\beta = 2$, we must compute

$$w_{s,t} \;=\; \frac{p_s^2}{\sum_i p_i^2} \;=\; \frac{1}{\sum_i (p_i/p_s)^2}\,.$$

The same is true for all the other combination strategies of Chapter 9. Thus we can arbitrarily set $p_s = 1$, and compute the values of the other p_i relative to p_s.

To do this, we consider the ratio p_{i+1}/p_i. It will be convenient to ignore the distinction between vertices in the light and eye subpaths, and to write the path $\bar{x}_{s,t}$ as

$$\bar{x} \;=\; \mathbf{x}_0 \ldots \mathbf{x}_k$$

where $k = s+t-1$. In this notation, the only difference between p_i and p_{i+1} lies in how the vertex $\mathbf{x}_i$ is generated: for p_i, it is generated as part of the eye subpath $\mathbf{x}_i \ldots \mathbf{x}_k$, while for p_{i+1} it is generated as part of the light subpath $\mathbf{x}_0 \ldots \mathbf{x}_i$. All other vertices of $\bar{x}$ are generated with the same probability by both techniques. Thus, the ratio of p_{i+1} to p_i is

$$\begin{aligned}
\frac{p_1}{p_0} &= \frac{P_A(\mathbf{x}_0)}{P_{\sigma^\perp}(\mathbf{x}_1 \to \mathbf{x}_0)\, G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_0)}\,, \\
\frac{p_{i+1}}{p_i} &= \frac{P_{\sigma^\perp}(\mathbf{x}_{i-1} \to \mathbf{x}_i)\, G(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)}{P_{\sigma^\perp}(\mathbf{x}_{i+1} \to \mathbf{x}_i)\, G(\mathbf{x}_{i+1} \leftrightarrow \mathbf{x}_i)} \qquad \text{for } 0 < i < k\,, \\
\frac{p_{k+1}}{p_k} &= \frac{P_{\sigma^\perp}(\mathbf{x}_{k-1} \to \mathbf{x}_k)\, G(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k)}{P_A(\mathbf{x}_k)}\,.
\end{aligned} \tag{10.9}$$

This equation can be applied repeatedly starting with p_s to find p_{s+1}, …, p_{k+1}. Similarly, the reciprocal ratio p_i/p_{i+1} can be used to compute p_{s-1}, …, p_0.

Once the p_i have been calculated, it is straightforward to compute $w_{s,t}$ according to the combination strategy being used. The final weighted sample contribution is then

$$\begin{aligned}
C_{s,t} &= w_{s,t}\, C^*_{s,t} \\
&= w_{s,t}\, \alpha^L_s\, c_{s,t}\, \alpha^E_t\,.
\end{aligned}$$

Note that the samples in each group are dependent (since they are all generated from the same light and eye subpath). However this does not significantly affect the results, since the correlation between them goes to zero as we increase the number of independent sample groups for each measurement. For example, if N independent light and eye subpaths are used, then all of the samples from each $p_{s,t}$ are independent, and each sample from $p_{s,t}$ is correlated with only one of the N samples from any other given technique $p_{s',t'}$. From this fact it is easy to show that the variance results of Chapter 9 are not affected by more than a factor of $(N-1)/N$ due to the correlation between samples in each group.

10.3 Implementation issues

This section describes several aspects of our implementation. We start by explaining how the image is sampled and filtered. Next we describe how the light and eye subpaths are generated. This includes a summary of the information that is precomputed and stored with each subpath (in order to evaluate the sample contributions more efficiently), and the methods used to determine the length of each subpath. Following this, we describe how to implement the important special cases where the light or eye subpath has at most one vertex. Finally, we consider how to handle specular surfaces correctly, and we consider several situations where the weighting functions $w_{s,t}$ cannot be computed exactly (so that approximations must be used).

10.3.1 Image sampling and filtering

So far, our discussion of bidirectional path tracing could be applied to any kind of measurements I_j. Here we discuss the special issues that arise when computing an image (as opposed to some other set of measurements).

Overall, the image sampling of bidirectional path tracing is similar to ray tracing or path tracing. The camera and lens model determine a mapping from rays in world space onto the *image plane*. This mapping is used to define an *image function* I such that $I(u,v)$ is proportional to the irradiance on the image plane at the point (u,v).[2] Each pixel value I_j is

[2]Strictly speaking, the units of $I(u,v)$ are *sensor response per unit area* $[\mathrm{S} \cdot \mathrm{m}^{-2}]$ rather than irradiance. (When $I(u,v)$ is integrated, the resulting pixel values have units of sensor response [S] rather than power.)

then defined as a weighted average

$$I_j \;=\; \iint_D h_j(u,v)\, I(u,v)\, du\, dv\,,$$

where D is the image region, and h_j is the *filter function* for pixel j (which integrates to one). In general, the filter functions are all translated copies of one another, and each one is zero except on a small subset of D.

To estimate the values of all the pixels $I_1, \ldots, I_M$, a large number of sample points are chosen across the image region. We do this by taking a fixed number of stratified samples per pixel (e.g. to take $n = 25$ samples, the nominal rectangle corresponding to each pixel would be subdivided into a 5 by 5 grid). Each sample can contribute to the value of several pixels, since the filter functions h_j generally overlap one another. Specifically, the pixel values are estimated using[3]

$$I_j \;\approx\; \frac{\sum_{i=1}^N h_j(u_i,v_i)\, I(u_i,v_i)}{\sum_{i=1}^N h_j(u_i,v_i)}\,, \tag{10.11}$$

where $N = nM$ is the total number of samples. This equation can be evaluated efficiently by storing the current value of the numerator and denominator of (10.11) at each pixel, and accumulating samples as they are taken. Note that each sample (u_i, v_i) contributes to only a few nearby pixels (because of the filter functions h_j), and that it is not necessary to store the samples themselves.

10.3.2 Estimation of the image function

The image function $I(u,v)$ is estimated using bidirectional path tracing. The initial vertex of the light subpath is chosen according to the emitted power at each surface point, while the remaining vertices are chosen by sampling from the BSDF (or some convenient approximation). Sampling the camera lens is slightly trickier: the vertex $\mathbf{z}_0$ can be chosen anywhere

[3]Note that this estimate is slightly biased. The corresponding unbiased estimate is simply

$$I_j \;=\; E\left[(|D|\,/\,N)\,\textstyle\sum_{i=1}^N h_j(u_i,v_i)\, I(u_i,v_i)\right]\,, \tag{10.10}$$

where $|D|$ is the area of the image region D. However, equation (10.11) typically gives better results (a smaller mean-squared error) because it compensates for random variations in the sum of the filter weights.

on the lens surface, but the direction $\mathbf{z}_0 \to \mathbf{z}_1$ is then uniquely determined by the given point (u, v) on the image plane (since there is only one direction at $\mathbf{z}_0$ that is mapped to the point (u, v) by the lens).[4] Note that the density $P_{\sigma^\perp}(\mathbf{z}_0 \to \mathbf{z}_1)$ is determined by the fact that (u, v) is uniformly distributed over the image region.

After generating the light and eye subpaths, we consider all possible connections between them as described above. In order to do this efficiently, we cache information about the vertices in each subpath. The vertex itself is stored in the form of a special *Event* object that has methods for sampling and evaluating the BSDF, and for evaluating the probability with which a given direction is sampled (according to a built-in sampling strategy associated with each BSDF). The vertices $\mathbf{y}_0$ and $\mathbf{z}_0$ are also stored in this form, so that the distribution of emitted radiance and importance at these vertices can be queried using the same methods.

Other per-vertex information includes the cumulative subpath weights α_i^L and α_i^E defined above, the geometric factors $G(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)$, and the probability densities $P_{\sigma^\perp}(\mathbf{x}_i \to \mathbf{x}_{i-1})$ and $P_{\sigma^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1})$ for sampling the adjacent subpath vertices on either side. The latter three fields are used in equation (10.9) to efficiently evaluate the probabilities p_i with which a given path is sampled using all the other possible techniques.

When information about the subpaths is cached, then the work required to evaluate the contributions $C_{s,t}$ is minimal (except for the visibility test, if necessary). The only quantities that need to be evaluated are those associated with the connecting edge $\mathbf{y}_{s-1}\mathbf{z}_{t-1}$ (since this edge is not part of either subpath).

10.3.3 Determining the subpath lengths

To control the lengths of the light and eye subpaths (n_L and n_E), we define a probability for the subpath to be terminated or *absorbed* after each vertex is generated. We let q_i denote the probability that the subpath is continued past vertex $\mathbf{x}_i$, while $1 - q_i$ is the probability that the subpath is terminated. This is a form of *Russian roulette* (Chapter 2).

[4]For real lens models [Kolb et al. 1995], it is difficult to determine the direction $\mathbf{z}_0 \to \mathbf{z}_1$ once the point $\mathbf{z}_0$ has already been chosen, since this requires us to find a chain of specular refractions that connects two given points on opposite side of the lens (i.e. $\mathbf{z}_0$ and the point (u, v) on the film plane). A better approach in this case is to generate $\mathbf{z}_0$ and $\mathbf{z}_0 \to \mathbf{z}_1$ together, by starting on the film plane at (u, v) and tracing a ray toward the exit pupil.

In our implementation, we set $q_i = 1$ for the first few vertices of each subpath, to avoid any extra variance on short subpaths (which typically make the largest contribution to the image). After that, q_i is determined by first sampling a candidate direction $\mathbf{x}_i \to \mathbf{x}_{i+1}$, and then letting

$$q_i \;=\; \min\{1, \frac{f_{\mathrm{s}}(\mathbf{x}_{i-1} \to \mathbf{x}_i \to \mathbf{x}_{i+1})}{P^*_{\sigma^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1})}\}\,,$$

where $P^*_{\sigma^\perp}$ is density function used for sampling the direction $\mathbf{x}_i \to \mathbf{x}_{i+1}$. Notice that if $P^*_{\sigma^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1})$ is proportional to the BSDF, then q_i is simply the albedo of the material, i.e. the fraction of energy that is scattered rather than absorbed for the given incident direction.

This procedure does not require any modification to the formulas for the sample contributions described in Section 10.2. However, it is important to realize that the final probability density for sampling each direction is now a product:

$$P_{\sigma^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1}) \;=\; q_i\, P^*_{\sigma^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1})\,.$$

The density $P_{\sigma^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1})$ can integrate to less than one, since there is a discrete probability associated with terminating the subpath at $\mathbf{x}_i$.

10.3.4 Special cases for short subpaths

Subpaths with less than two vertices require special treatment for various reasons. The most important issues are: taking advantage of direct lighting calculations when the light subpath has only one vertex, and allowing samples to contribute to any pixel of the image in the cases when the eye subpath has zero or one vertices. In addition, the cases when the light or eye subpath is empty require special handling since no visibility test is needed.

10.3.4.1 Zero light subpath vertices ($s = 0$)

These samples occur when the eye subpath randomly intersects a light source. For this to occur, the light sources must be modeled as part of the scene (so that it is possible for a ray to intersect them). We also require the ability to determine whether the current eye subpath endpoint $\mathbf{z}_{t-1}$ is on a light source, and to evaluate the emitted radiance along the ray $\mathbf{z}_{t-1} \to \mathbf{z}_{t-2}$. In order to evaluate the combination weight $w_{0,t}$, we must also compute the probability densities for generating the point $\mathbf{z}_{t-1}$ and the direction $\mathbf{z}_{t-1} \to \mathbf{z}_{t-2}$ by sampling the light sources (in order to compute the densities p_i with which the other sampling techniques generate this path).

The $s = 0$ sampling technique is very important for the rendering of certain lighting effects. These include: directly visible light sources; lights that are seen by reflection or refraction in a specular surface; caustics due to large area light sources; and caustics that are viewed indirectly through a specular surface.

A nice thing about this sampling technique is that no visibility test is required. Thus its contributions are cheap to evaluate, compared to the other $C_{s,t}$. In our implementation, we accumulate these contributions as the eye subpath is being generated.

10.3.4.2 One light subpath vertex (s = 1)

This sampling technique connects a given eye subpath $\mathbf{z}_{t-1} \ldots \mathbf{z}_0$ to a randomly chosen point on the light sources. Recall that in the basic algorithm, this point is simply the first vertex $\mathbf{y}_0$ of the light subpath (which was chosen according to emitted power). However, the variance of these samples can be greatly reduced by choosing the vertex using special direct lighting techniques. That is, we simply ignore the vertex $\mathbf{y}_0$, and connect the eye subpath to a new vertex $\mathbf{y}_0^{\mathrm{d}}$ chosen using a more sophisticated method (such as those described by Shirley et al. [1996]). This strategy is applied to each eye subpath suffix $\mathbf{z}_{t-1} \ldots \mathbf{z}_0$ separately, by choosing a different light source vertex for each one.

This optimization is very important for direct illumination (i.e. paths of length two), since it allows the same low-variance lighting techniques used in ray tracing to be applied. It is also an important optimization for longer paths; this corresponds to standard path tracing, where each vertex of the path is connected to a point on the light source. A direct lighting strategy is essentially an importance sampling technique that chooses a light source vertex $\mathbf{y}_0^{\mathrm{d}}$ according to how much it contributes to the illuminated point $\mathbf{z}_{t-1}$ (or some approximation of this distribution).

This strategy requires some changes in the way that sample contributions are evaluated:

- The unweighted contribution $C^*_{1,t}$ is computed using the density $P_A^{\mathrm{d}}(\mathbf{y}_0^{\mathrm{d}})$ with which the light vertex $\mathbf{y}_0^{\mathrm{d}}$ was chosen. This calculation is identical to standard path tracing.

- The evaluation of the combination weight $w_{1,t}$ is slightly trickier, because the direct lighting strategy does not affect the sampling of light subpaths with two or more vertices. Thus we must evaluate the density with which $\mathbf{y}_0^{\mathrm{d}}$ is sampled according to emitted power; this is used to compute the probabilities p_i for sampling the current path using the other possible techniques.

- The direct lighting strategy also affects the combinations weights for paths where $s \neq 1$. The correct probabilities p_i can be found by computing them as usual, and then multiplying the density for p_1 by $P_A^{\mathrm{d}}(\mathbf{x}_0) \,/\, P_A(\mathbf{x}_0)$. Here $P_A(\mathbf{x}_0)$ is the density for generating $\mathbf{x}_0$ according to emitted power, and $P_A^{\mathrm{d}}(\mathbf{x}_0)$ is the density for generating $\mathbf{x}_0$ using direct lighting for the point $\mathbf{x}_1$.

It is also possible to use a direct lighting strategy that takes more than one sample, e.g. a strategy that iterates over the light sources taking a few samples from each one. This is equivalent to using more than one sampling technique to generate these paths; the samples are simply combined as usual according to the rules of multiple importance sampling.

10.3.4.3 One eye subpath vertex (t = 1)

These samples are generated by connecting each light subpath prefix $\mathbf{y}_0 \ldots \mathbf{y}_{s-1}$ to the vertex $\mathbf{z}_0$ on the camera lens. These samples are important for rendering caustics (especially those from small or point light sources), some forms of direct illumination, and a variety of other lighting effects.

The main issue with this technique is that the samples it generates can lie anywhere in the image, not just at the current point (u, v). One way to handle this is to discard samples that do not contribute to the current measurement I_j. However, this is inefficient; much more information can be obtained by letting these samples contribute to any pixel of the image.

To implement this, we allocate a separate image to record the contributions of paths where 0 or 1 vertices are generated from the eye. We call this the *light image*, as opposed to the *eye image* that holds the contributions of paths where $t \geq 2$ eye subpath vertices are used.

To accumulate each sample, we first determine the point (u', v') on the image plane that corresponds to the ray $\mathbf{y}_{s-1} \rightarrow \mathbf{z}_0$. We then compute the contribution $C_{s,1}$ of this sample as usual, and record it at the location (u', v'). This is done by finding all of the pixels whose filter value $h_j(u', v')$ is non-zero, and updating the pixel values I_j^L of the light image using

$$I_j^L \;\leftarrow\; I_j^L \,+\, h_j(u', v')\, C_{s,1}\,.$$

Note that the estimate $I(u, v)$ at the current image point is not affected by this calculation. Also note that it is not necessary to store the light and eye images in memory (although this is what is done in our implementation). The eye image can be sampled and written to disk in scanline order, while the light image can be handled by repeatedly accumulating a fixed number of samples in memory, sorting them in scanline order, and merging them with an image on disk.

When the algorithm has finished, the final estimate for each pixel has the form

$$I_j \;\approx\; (|D| \,/\, N)\, I_j^L \,+\, I_j^E\,,$$

where $|D|$ is the area of the image region, N is the total number of bidirectional samples that were taken, and I_j^E is the estimate for pixel j from the eye image (sampled and filtered as described in Section 10.3.1). Note that the eye and light images are filtered differently: the eye image is normalized at each pixel by dividing by the sum of the filter weights, while the light image is not (see equations (10.11) and (10.10) respectively). Thus the final pixel values of the light image are determined by the sample density as well as the sample values; more samples per pixel correspond to a brighter image.

Note that to evaluate the contribution $C_{s,1}$ of each sample, we must evaluate the importance emitted from $\mathbf{z}_0$ toward $\mathbf{y}_{s-1}$ (or more precisely, the directional component $W_{\mathrm{e}}^{(1)}$ of the importance). The function $W_{\mathrm{e}}^{(1)}$ is defined so that

$$\int_D W_{\mathrm{e}}^{(1)}(\mathbf{z}_0, \omega)\, d\sigma^{\perp}(\omega)$$

is equal to the fraction of the image region covered by the points (u, v) that are mapped by the lens to directions $\omega \in D$. It is important to realize that this function is not uniform for most lens models in graphics, since pixels near the center of the image correspond to a set of rays whose projected solid angle is larger than for pixels near the image boundary.

10.3.4.4 Zero eye subpath vertices ($t = 0$)

These samples occur when the light subpath randomly intersects the camera lens. Because the camera lens is a relatively small target, these samples do not contribute significantly for most scenes. On the other hand, these samples are very cheap to evaluate (because no visibility test is required), and can sometimes make the computation more robust. For example, this can be an effective sampling strategy for rendering specular reflections of small or highly directional light sources.

To implement this method, the lens surface must have a physical representation in the scene (so that it can be intersected by a ray). In particular, this sampling technique cannot be used for pinhole lens models. As with the case for $t = 1$ eye subpath vertices, samples can contribute to any pixel of the image. The image point (u', v') is determined from the ray $\mathbf{y}_{s-2} \to \mathbf{y}_{s-1}$, and samples are accumulated and filtered in the light image as before.

10.3.5 Handling specular surfaces

Specular surfaces require careful treatment, because the BSDF and the density functions used for importance sampling both contain Dirac distributions. This is not a problem when computing the weights α_i^L and α_i^E, since the ratio

$$\frac{f_{\mathrm{s}}(\mathbf{x}_{i-3} \to \mathbf{x}_{i-2} \to \mathbf{x}_{i-1})}{P_{\sigma^\perp}(\mathbf{x}_{i-2} \to \mathbf{x}_{i-1})}$$

of equation (10.6) is well-defined. Although this ratio cannot be directly evaluated (since the numerator and denominator both contain a Dirac distribution), it can be returned as a "weight" when the specular component of the BSDF is sampled.

Similarly, specular surfaces do not cause any problems when computing the unweighted contribution $C_{s,t}^*$ that connects the eye and light subpaths. The specular components of the BSDF's can simply be ignored when computing the factor

$$c_{s,t} = f_{\mathrm{s}}(\mathbf{y}_{s-2} \to \mathbf{y}_{s-1} \to \mathbf{z}_{t-1})\, G(\mathbf{y}_{s-1} \leftrightarrow \mathbf{z}_{t-1})\, f_{\mathrm{s}}(\mathbf{y}_{s-1} \to \mathbf{z}_{t-1} \to \mathbf{z}_{t-2}) \,,$$

since there is a zero probability that these BSDF's will have a non-zero specular component in the direction of the given connecting edge.[5]

On the other hand, specular surfaces require careful treatment when computing the weights $w_{s,t}$ for multiple importance sampling. To compute the densities p_i for the other possible ways of sampling this path, we must evaluate expressions of the form

$$\frac{p_{i+1}}{p_i} = \frac{P_{\sigma^\perp}(\mathbf{x}_{i-1} \to \mathbf{x}_i)\, G(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i)}{P_{\sigma^\perp}(\mathbf{x}_{i+1} \to \mathbf{x}_i)\, G(\mathbf{x}_{i+1} \leftrightarrow \mathbf{x}_i)} \tag{10.12}$$

(see equation (10.9)). In this case the denominator may contain a Dirac distribution that is not matched by a corresponding factor in the numerator.

We handle this problem by introducing a *specular* flag for each vertex. If the flag is true, it means that the BSDF and sampling probabilities at this vertex are represented only up to an unspecified constant of proportionality. That is, the cached values of the BSDF $f_s(\mathbf{x}_{i-1} \to \mathbf{x}_i \to \mathbf{x}_{i+1})$ and the probability densities $P_{\sigma^\perp}(\mathbf{x}_i \to \mathbf{x}_{i-1})$ and $P_{\sigma^\perp}(\mathbf{x}_i \to \mathbf{x}_{i+1})$ are all considered to be coefficients for a single Dirac distribution δ that is shared between them.[6] When applying equation (10.12), we use only the coefficients, and simply keep track of the fact that the corresponding density also contains a Dirac distribution.

Specifically, consider a path whose connecting edge is $\mathbf{x}_{s-1}\mathbf{x}_s$. We start with the nominal probability $p_s = 1$, and compute the relative values of the other p_i by applying (10.12) repeatedly. It is easy to check that a specular vertex at $\mathbf{x}_j$ causes a Dirac distribution to appear in the denominator of p_j and p_{j+1}, so that these probabilities are effectively zero. (Notice that these densities correspond to the sampling techniques where $\mathbf{x}_j$ is a connecting vertex.) However, these are the only p_i that are affected, since for other values of i the Dirac distributions in $P_{\sigma^\perp}(\mathbf{x}_j \to \mathbf{x}_{j-1})$ and $P_{\sigma^\perp}(\mathbf{x}_j \to \mathbf{x}_{j+1})$ are canceled by each other.

The end result is particularly simple: we first compute all of the p_i exactly as we would for non-specular vertices, without regard for the fact the some of the densities are actually coefficients for Dirac distributions. Then for every vertex where $\mathbf{x}_j$ is specular, we set p_j and

[5] Even if the connecting edge happened to have a direction for which one of the BSDF's is specular (a set of measure zero), the value of the BSDF is infinite and cannot be represented as a real number. Thus we choose to ignore such paths (by assigning them a weight $\alpha_{s,t} = 0$), and instead we account for them using one of the other sampling techniques.

[6] From another point of view, we can say that BSDF and probability densities are expressed with respect to a different *measure function*, one that assigns a positive measure to the discrete direction $\mathbf{x}_{i-2} \to \mathbf{x}_{i-1}$.

p_{j+1} to zero (since these probabilities include a symbolic Dirac distribution in the denominator). Note that these techniques apply equally well to the case of perfectly anisotropic reflection, where light from a given direction is scattered into a one-dimensional set of outgoing directions. In this case, the unspecified constant of proportionality associated with the *specular* flag is a one-dimensional Dirac distribution.

10.3.6 Approximating the weighting functions

Up until now, we have assumed that the densities p_i for sampling the current path using other techniques can be computed exactly (as required to evaluate the weight $w_{s,t}$). However, there are some situation where it is difficult or impossible to do this; examples will be given below. In these situations, the solution is to replace the true densities p_i with approximations $\hat{p}_i$ when evaluating the weights. As long as these approximations are reasonably good, the optimality properties of the combination strategy being used will not be significantly affected. But even if the approximations are bad, the results will at least be unbiased, since the weighting functions sum to one for any values of the $\hat{p}_i$.[7] We now discuss the reasons that approximations are sometimes necessary.

Adaptive sampling is one reason that the exact densities can be difficult to compute.[8] For example, suppose that adaptive sampling is used on the image plane, to take more samples where the measured variance is high. In this case, it is impossible to compare the densities for sampling techniques where $t \geq 2$ eye vertices are used to those where $t \leq 1$, since the densities for $t \geq 2$ depend on the eventual distribution of samples over the image plane (which has not yet been determined). A suitable approximation in this case is to assume that the density of samples is uniform across the image.

Similarly there are some direct lighting strategies where approximations are necessary, because the strategy makes random choices that cannot be determined from the final light source vertex $\mathbf{y}_0^{\mathrm{d}}$. For example, consider the following strategy [Shirley et al. 1996]. First,

[7] Note that to avoid bias, the unweighted contribution $C^*_{s,t}$ must always be evaluated exactly; this part of the calculation is required for any unbiased Monte Carlo algorithm. The evaluation of $C^*_{s,t}$ should never be a problem, since all the random choices that were used to generate the current path are explicitly available (including random choices that are cannot be determined from the resulting path itself).

[8] Note that adaptive sampling can introduce bias, unless two-stage sampling is used [Kirk & Arvo 1991].

a candidate vertex $\mathbf{x}_i$ is generated on each light source S_i. Next we compute the contribution that each vertex $\mathbf{x}_i$ makes to the illuminated point $\mathbf{z}_{t-1}$, under the assumption that the corresponding visibility ray is not obstructed. Finally, we choose one of the candidates $\mathbf{x}_i$ at random according to its contribution, and return it as the light source vertex $\mathbf{y}_0^{\mathrm{d}}$. The problem with this strategy is that given an arbitrary point $\mathbf{x}$ on a light source, it is very difficult to evaluate the probability density $P_A^{\mathrm{d}}(\mathbf{x})$ with which $\mathbf{x}$ is sampled. This is because the sampling procedure makes random choices that are not reflected in the final result $\mathbf{y}_0^{\mathrm{d}}$: namely, the locations of the other candidate points $\mathbf{x}_i$, which are generated and then discarded. To evaluate the density exactly would require analytic integration over the all possible locations of the $\mathbf{x}_i$. A suitable approximation in this case is to use the conditional probability $A^{\perp}(\mathbf{x}_i \mid S_i)$, i.e. the density for sampling $\mathbf{x}_i$ given that the light source S_i has already been chosen.

10.4 Reducing the number of visibility tests

To make bidirectional path tracing more efficient, it is important to reduce the number of visibility tests. The basic algorithm assumes that all of the $O(n_L n_E)$ contributions are evaluated; however, typically most of these contributions are so small that a visibility test is not justified. In this section, we develop a new technique called *efficiency-optimized Russian roulette* that is an effective solution to this problem. We start with an introduction to ordinary Russian roulette and a discussion of its shortcomings. Next, we describe efficiency-optimized Russian roulette as a general technique. Finally we describe the issues that arise when applying this technique to bidirectional path tracing.

We consider the following abstract version of the visibility testing problem. Suppose that we must repeatedly evaluate an estimator of the form

$$F \;=\; C_1 + \cdots + C_n \,,$$

where the number of contributions n is a random variable. We assume that each contribution C_i can be written as the product of a *tentative contribution* t_i, and a *visibility factor* v_i (which is either 0 or 1).

The number of visibility tests can be reduced using *Russian roulette*. We define the *roulette probability* q_i to be the probability of testing the visibility factor v_i. Each contribution then has the form

$$C_i \;=\; \begin{cases} (1/q_i)\, v_i\, t_i & \text{with probability } q_i\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

It is easy to verify that $E[C_i] \;=\; E[v_i\, t_i]$, i.e. this estimator is unbiased.

The main question, of course, is how to choose the roulette probabilities q_i. Typically this is done by choosing a fixed *roulette threshold* δ, and defining

$$q_i \;=\; \min(1, t_i \,/\, \delta)\,.$$

Thus contributions larger than δ are always evaluated, while smaller contributions are randomly skipped in a way that does not cause bias.

This approach is not very satisfying, however, because the threshold δ is chosen arbitrarily. If the threshold is chosen too high, then there will be a substantial amount of extra variance (due to visibility tests that are randomly skipped), while if the threshold is too low, then many unnecessary visibility tests will be performed (leading to computation times that are longer than necessary). Russian roulette thus involves a tradeoff, where the reduction in computation time must be balanced against the corresponding increase in variance.

10.4.1 Efficiency-optimized Russian roulette

In this section, we show how to choose the roulette probabilities q_i so as to maximize the efficiency of the resulting estimator F. Recall that *efficiency* is defined as

$$\epsilon \;=\; \frac{1}{\sigma^2\, T}\,,$$

where σ^2 is the variance of the given estimator, and T is the average computation time required to evaluate it. We assume the computation time is simply proportional to the number of rays that are cast (n). Note that n includes all types of rays, not just visibility rays; e.g. for bidirectional path tracing, it includes the rays that are used to generate the light and eye subpaths.

To begin, we consider the effect that q_i has on the variance and cost of F. For the variance, we return to the definition

$$C_i \;=\; \begin{cases} (1/q_i)\, v_i\, t_i & \text{with probability } q_i\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

We can treat t_i as a fixed quantity (since we are only interested in the additional variance relative to the case $q_i = 1$), and we can also assume that $v_i = 1$ (a conservative assumption, since if $v_i = 0$ then Russian roulette does not add any variance at all). The additional variance due to Russian roulette can then be written as

$$\begin{aligned} V[C_i] &= E[C_i^2] - E[C_i]^2 \\ &= \Big[q_i\,(t_i \,/\, q_i)^2 \;+\; (1-q_i)\,0\Big] \;-\; t_i^2 \\ &= t_i^2\,(1/q_i - 1)\,. \end{aligned}$$

As for the cost, it is easy to see that the number of rays is reduced by $1 - q_i$ on average.

Next, we examine how this affects the overall efficiency of F. Here we make an important assumption: namely, that F is sampled repeatedly, so that estimates of its average variance σ_0^2 and average sample cost n_0 can be computed. Then according to the discussion above, the modified efficiency due to q_i can be estimated as

$$\epsilon \;=\; \frac{1}{\Big[\sigma_0^2 + t_i^2\,(1/q_i - 1)\Big]\,\cdot\,(n_0 - (1-q_i))}\,. \tag{10.13}$$

The optimal value of q_i is found by taking the derivative of this expression and setting it equal to zero. After some manipulation, this yields

$$q_i \;=\; t_i \,/\, \sqrt{(\sigma_0^2 - t_i^2)/(n_0 - 1)}\,.$$

Conveniently, this equation has the same form that is usually used for Russian roulette calculations, where the tentative contribution is compared against a given threshold δ. Since q_i is limited to the range $(0, 1]$, the optimal value is

$$\begin{aligned} q_i &= \min(1, t_i/\delta) \\ \text{where} \qquad \delta &= \sqrt{(\sigma_0^2 - t_i^2)/(n_0 - 1)}\,. \end{aligned} \tag{10.14}$$

However, this choice of the threshold δ has two undesirable properties. First, its value depends on the current tentative contribution t_i, so that it must be recalculated for every sample. Second, there is the possibility that an unusually large sample will have $t_i^2 > \sigma_0^2$, in which case the formula for δ does not make sense (although by returning to the original expression (10.13), it is easy to verify that the optimal choice in this case is $q_i = 1$).

To avoid these problems, we look for a fixed threshold δ^* that has the same transition point at which $q_i = 1$. It is easy to check that $q_i \geq 1$ if and only if $t_i^2 \geq \sigma_0^2/n_0$. Thus, the fixed threshold

$$\delta^* = \sqrt{\sigma_0^2/n_0}$$

leads to Russian roulette being applied to the same set of contributions as the original threshold (10.14).[9] Notice that δ^* is simply the estimated standard deviation per ray.

Summary. Efficiency-optimized Russian roulette consists of the following steps. Given an estimator F that is sampled a number of times, we keep track of its average variance σ_0^2 and average ray count n_0. Before each sample is taken we compute the threshold

$$\delta^* = \sqrt{\sigma_0^2/n_0}\,,$$

and apply this threshold to all of the individual contributions t_i that require a visibility test. The roulette probability q_i is given by

$$q_i = \min(1, t_i/\delta)\,.$$

Note that this technique does not maximize efficiency in a precise mathematical sense, since we have made several assumptions in our derivation. Rather, it should be interpreted as a heuristic that is *guided* by mathematical analysis; its purpose is to provide theoretical insight about parameter values that would otherwise be chosen in an *ad hoc* manner.

[9] The roulette probabilities will be slightly different for $q_i < 1$; it is easy to check that δ^* results in values of q_i that are slightly larger, by a factor between 1 and $\sqrt{n_0/(n_0-1)}$. Thus, visibility is tested slightly more often using the fixed threshold δ^* than the original threshold δ.

10.4.2 Implementation

The main requirement for implementing this technique is that we must be able to estimate the average variance and cost of each sample (i.e. σ_0^2 and n_0). This is complicated by the fact that the mean, variance, and sample cost can vary substantially over the image plane. It is not sufficient to simply compute the variance of all the samples taken so far, since the average variance of samples over the whole image plane does not reflect the variance at any particular pixel. For example, suppose that the left half of the current image is white, and the right half is black. The variance at most pixels might well be zero, and yet the estimated variance will be large if all the image samples are combined.

Ideally, we would like σ_0^2 and n_0 to estimate the variance and sample cost within the current pixel. This could be done by taking samples in random order over the whole image plane, and storing the location and value of each sample. We could then estimate σ_0^2 and n_0 at a given point (u, v) by computing the sample variance and average cost of the nearest N_0 samples.

In our implementation, we use a simpler approach. The image is sampled in scanline order, and we estimate σ_0^2 and n_0 using the last N_0 samples (for some fixed value of N_0). Typically we let N_0 be the number of samples per pixel; this ensures that all variance and cost estimates are made using samples from either the current pixel or the one before. (To ensure that the previous pixel is always nearby, scanlines are rendered in alternating directions. Alternatively, the pixels could be traversed according to a space-filling curve.)

The calculation of σ_0^2 and n_0 can be implemented efficiently as follows. Let n_j be the number of rays cast for the j-th sample, and let F_j be its value. We then simply maintain partial sums of n_j, F_j, and F_j^2 for the last N_0 samples, and set the Russian roulette threshold for the current sample to

$$\delta \;=\; \sqrt{\sigma_0^2 \,/\, n_0} \;=\; \sqrt{\frac{\sum F_j^2 - (1/N_0)\,\left(\sum F_j\right)^2}{\sum n_j}}\,,$$

where the sums are over the most recent N_0 samples only. (Note that the variance calculation is not numerically stable in this form, but we have not found this to be a problem.) It is most efficient to update these sums incrementally, by adding the values for the current sample j and subtracting the values for sample $j - N_0$. For this purpose, the last N_0 values of F_j and

n_j are kept in an array. We have found the overhead of these calculations to be negligible compared to ray casting.

An alternative would be to compute a *running average* of each quantity. This is done using the update formula

$$S_j \;=\; \alpha\, x_j \;+\; (1-\alpha)\, S_{j-1}\,,$$

where α is a small real number that determines how quickly the influence of each sample drops off with time. (This technique is also known as *exponential smoothing*.)

10.5 Results

We have compared bidirectional path tracing against ordinary path tracing using the test scene shown in Figure 10.3. The scene contains a floor lamp, a spotlight, a table, and a large glass egg. Observe that diffuse, glossy, and pure specular surfaces are all present, and that most of the room is illuminated indirectly.

Figure 10.3(a) was created by sampling paths up to length $k = 5$ using bidirectional path tracing, and combining the sampling techniques $p_{s,t}$ using the power heuristic with $\beta = 2$ (see Chapter 9). The image is 500 by 500 with 25 samples per pixel. Observe the caustics on the table, both directly from the spotlight and indirectly from light reflected on the ceiling. The unusual caustic pattern to the left is caused by the square shape of the spotlight's emitting surface.

For comparison, Figure 10.3(b) was computed using standard path tracing with 56 samples per pixel (the same computation time as Figure 10.3(a)). Each path was generated starting from the eye, and direct lighting calculations were used to calculate the contribution at each vertex. Russian roulette was applied to reduce the number of visibility tests. Caustics were rendered using paths that randomly intersected the light sources themselves, since these paths would otherwise not be accounted for. (Direct lighting calculations cannot be used for paths where a light source shines directly on a specular surface.)

Recall that bidirectional path tracing computes a weighted sum of the contributions made by each sampling technique $p_{s,t}$. Figure 10.4 is a visualization of how much each of these techniques contributed toward the final image in Figure 10.3(a). Each row r shows

(a) Bidirectional path tracing with 25 samples per pixel

(b) Standard path tracing with 56 samples per pixel (the same computation time as (a))

Figure 10.3: A comparison of bidirectional and standard path tracing. The test scene contains a spotlight, a floor lamp, a table, and a large glass egg. Image **(a)** was computed with bidirectional path tracing, using the power heuristic with $\beta = 2$ to combine the samples for each path length. The image is 500 by 500 with 25 samples per pixel. Image **(b)** was computed using standard path tracing in the same amount of time (using 56 samples per pixel).

the sampling techniques for a particular path length $k = r + 1$ (for example, the top row shows the sampling techniques for paths of length two). The position of each image in its row indicates how the paths were generated: the s-th image from the left corresponds to paths with s light source vertices (and similarly, the t-th image from the right of each row corresponds to paths with t eye subpath vertices). Notice that the complete set of sampling techniques $p_{s,t}$ is not shown; paths of length $k = 1$ are not shown because the light sources are not directly visible, and paths with zero eye or light subpath vertices are not shown because these images are virtually black (i.e. their weighted contributions are very small for this particular scene). Thus, the full set of images (for paths up to length 5) would have one more image on the left and right side of each row, plus an extra row of three images on the top of the pyramid. (Even though these images are not shown, their contributions are included in Figure 10.3(a).)

The main thing to notice about these images is that different sampling techniques account for different lighting effects in the final image. This implies that most paths are sampled much more efficiently by one technique than the others. For example, consider the image in the middle of the second row of Figure 10.4, corresponding to the sampling technique $p_{2,2}$ (the full-size image is shown in Figure 10.5(a)). These paths were generated by sampling two vertices starting from the eye, and two vertices starting from a light source. Overall, this image is brighter than the other images in its row, which implies that samples from this technique make a larger contribution in general. Yet observe that the glass egg is completely black, and that the inside of the spot light looks at though it were turned off. This implies that the paths responsible for these effects were sampled more efficiently (i.e. with higher probability) by the other two sampling techniques in that row.

As paths get longer and more sampling techniques are used, the effects become much more interesting. For example, consider the rightmost image of the bottom row in Figure 10.4 (enlarged in Figure 10.5(b)), which corresponds to paths with five light vertices and one eye vertex ($p_{5,1}$). Observe the caustics from the spotlight (especially the long "horns" stretching to the right), which are due to internal reflections inside the glass egg. This sampling technique also captures paths that are somehow associated with the corners of the room (where there is a $1/r^2$ singularity in the integrand), and paths along the silhouette edges of the floor lamp's glossy surfaces. Notice that it would be very difficult to take all of these factors into account if we needed to manually partition paths among the sampling techniques; multiple importance sampling is absolutely essential in order to make bidirectional path tracing work well.

It is also interesting to observe that the middle images of each row in Figure 10.4 are brighter than the rest. This implies that for the majority of paths, the best sampling strategy is to generate an equal number of vertices from both sides. This can be understood in terms of the diffusing properties of light scattering, i.e. the fact that although the emitted radiance is quite concentrated, each scattering step spreads the energy more evenly throughout the scene. The same can be said for the emitted importance function; thus by taking several steps from the light sources and the eye, we have a bigger "target" when attempting to connect the two subpaths.

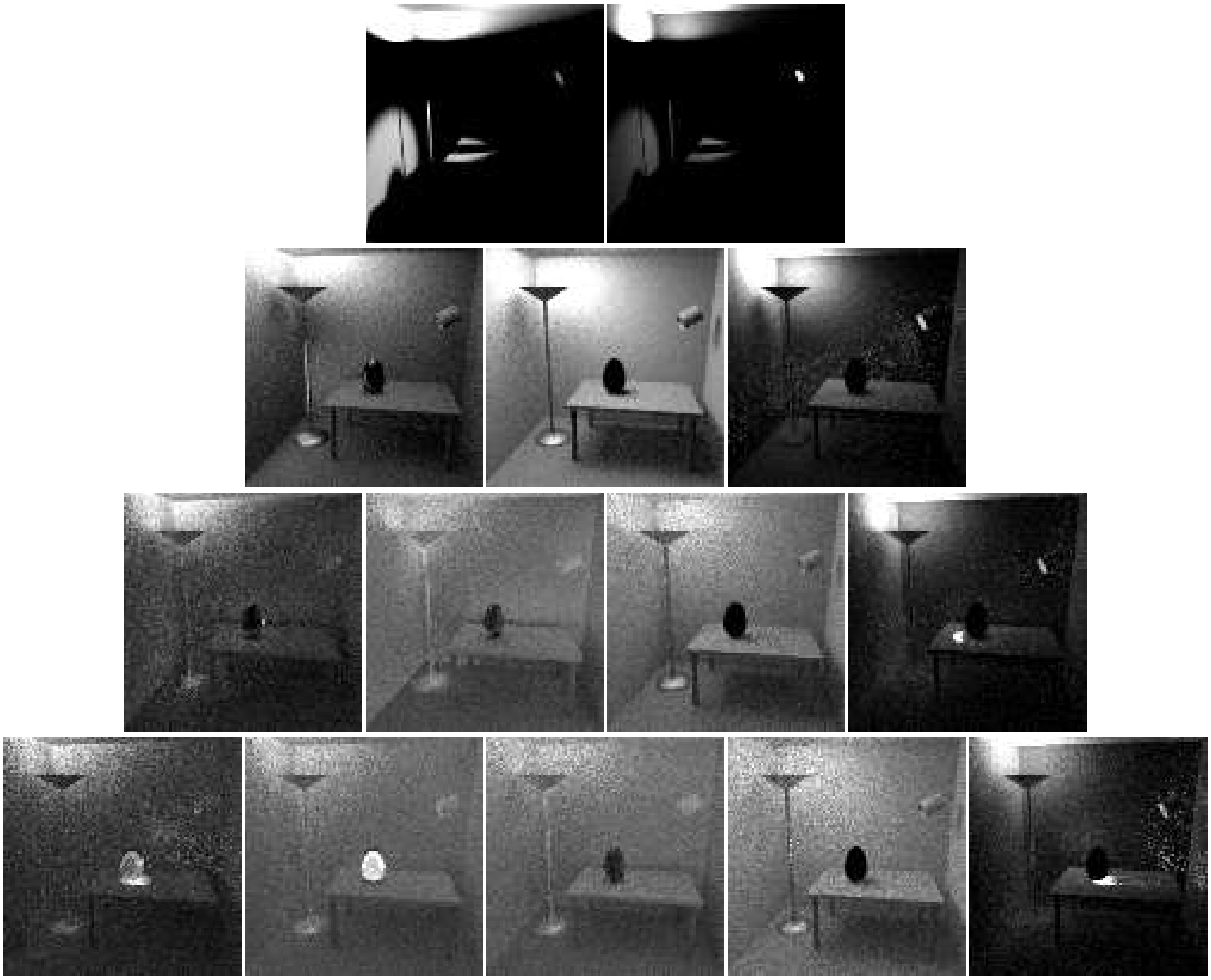
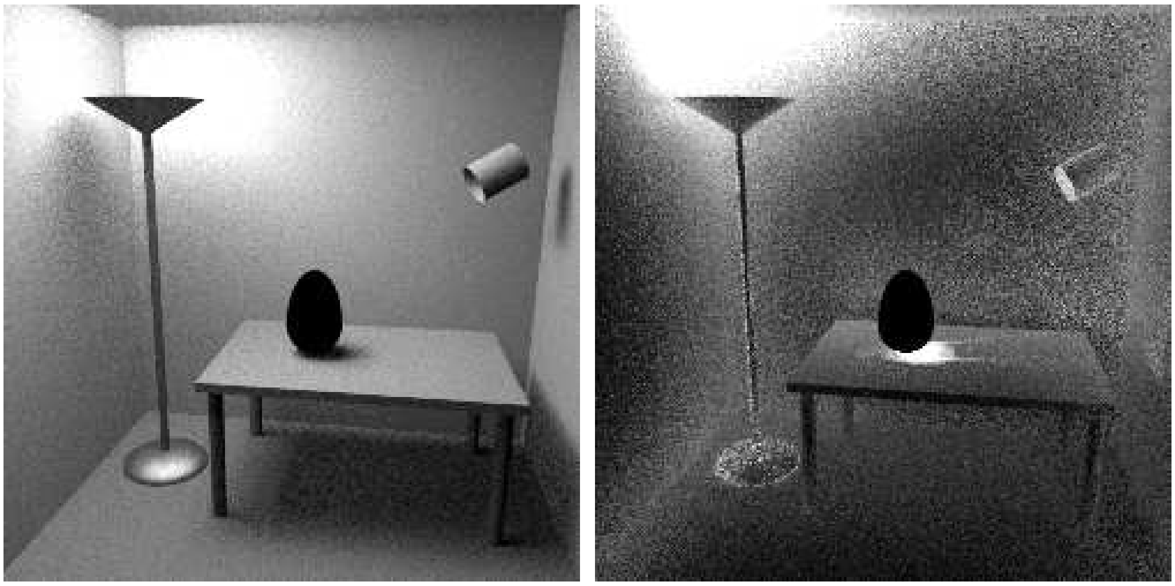


Figure 10.4: This figures shows the weighted contribution that each bidirectional sampling technique $p_{s,t}$ makes to Figure 10.3(a). Each row r shows the contributions of the sampling techniques for a particular path length $k = r + 1$. The position of each image in its row indicates how the paths were generated: the s-th image from the left in each row uses s light subpath vertices, while the t-th image from the right uses t eye subpath vertices. (For example, the top right image uses $s = 2$ light vertices and $t = 1$ eye vertex, while the bottom left image uses $s = 1$ light vertex and $t = 5$ eye vertices.) Note that these images have been over-exposed so that their details can be seen; specifically, the images in row r were over-exposed by r f-stops. The images were made by simply recording the contributions $C_{s,t}$ in a different image for each value of s and t.



(a) Two light vertices, two eye vertices ($p_{2,2}$). **(b)** Five light vertices, one eye vertex ($p_{5,1}$).

Figure 10.5: These are full-size images showing the weighted contributions to Figure 10.3(a) that are due to samples from two particular techniques ($p_{2,2}$ and $p_{5,1}$). These are enlarged versions of the images in Figure 10.4, where $p_{2,2}$ is the middle image of the second row, and $p_{5,1}$ is the rightmost image of the bottom row.

10.6 Comparison with related work

A similar bidirectional path tracing algorithm has been described independently by Lafortune & Willems [1993, 1994]. This section compares the two frameworks in detail, and discusses some possible extensions of the algorithms.

The most important difference between our algorithm and Lafortune's is that the samples are combined using a provably good strategy. This requires a substantially different theoretical basis for the algorithm, in order that multiple importance sampling can be applied. In particular, the path integral formulation of Chapter 8 makes two essential steps: it expresses light transport in the form of an integration problem, and it provides a well-defined basis for comparing the probabilities with which different sampling techniques generate the same path. On the other hand, Lafortune formulates bidirectional path tracing as a recursive evaluation of the *global reflectance distribution function* (GRDF).[10] This is certainly a valid theoretical basis for bidirectional path tracing; however, it does not express the problem in the form needed for multiple importance sampling.

Another difference is that our framework includes several important estimators that are missing from Lafortune's. These include the estimators where zero or one vertices are generated from the eye, and also the naive path tracing estimator where zero vertices are generated from the light source. These estimators are very important for generating caustics and other "difficult" transport paths, and help to make the calculations more robust. We have found that the estimator with one eye vertex ($t = 1$) is surprisingly useful for low-variance rendering in general (it is essentially a particle tracing technique where samples are recorded directly in the image). Also note that although Lafortune describes the estimator with one light vertex ($s = 1$), his framework does not allow the use of direct lighting techniques. This optimization is very important for making bidirectional path tracing competitive with standard path tracing on "normal" scenes, i.e. those where most surfaces are directly lit.

More generally, the two frameworks have a different conception of what bidirectional path tracing is. Lafortune describes it as a specific technique for generating a path from the eye, a path from the light sources, and connecting all pairs of vertices via shadow rays. On the other hand, we view bidirectional path tracing as a family of sampling techniques for paths. The samples from each technique can be generated in any way desired; the specific strategy of connecting every prefix of a light subpath to every suffix of an eye subpath is simply an optimization that allows these samples to be generated more efficiently. Any other desired method of generating the paths could be used instead, e.g. by connecting several different eye subpaths to the same light subpath, or by maintaining a "pool" of eye and light subpaths and making random connections between them, or by generating the paths in more than two pieces (by sampling one or more pieces starting from the middle).

A minor difference between the two frameworks is that Lafortune assumes that light sources are sampled according to emitted power, and that materials are sampled according to the BSDF (exactly). Our formulation of bidirectional path tracing allows the use of arbitrary probability distributions to choose each vertex. The direct lighting strategy applied

[10]The "GRDF" is simply a new name for the kernel of the solution operator $\mathbf{S}$ defined by equation (4.16).

to the case $s = 1$ is a simple example of why this is useful. Other possibilities include: selecting certain scene objects for extra sampling (e.g. portals between adjacent rooms, or small specular objects); using non-local sampling techniques to generate chains of specular vertices (see Section 8.3.4); or using an approximate radiance/importance solution to increase the sample densities in bright/important regions of the scene. Bidirectional path tracing is designed to be used in conjunction with these other sampling techniques, not to replace them.

Another minor difference is that our development is in terms of general linear measurements I_j, rather being limited to pixel estimates only. This means that bidirectional path tracing could be used to compute a view-independent solution, where the equilibrium radiance function L is represented as a linear combination of basis functions $\{B_1, \ldots, B_M\}$.[11] Each measurement I_j is simply the coefficient of B_j, and is defined by

$$I_j = \langle W_{\mathrm{e}}^{(j)}, L \rangle$$

where $W_{\mathrm{e}}^{(j)} = \tilde{B}_j$ is the corresponding dual basis function.[12] In this situation, each "eye subpath" starts from a surface of the scene rather than the camera lens. By using a fixed number of eye subpaths for each basis function, we can ensure that every coefficient receives at least some minimum number of samples. This bidirectional approach is an unexplored alternative to particle tracing for view-independent solutions, and may help to solve the problem of surface patches that do not receive enough particles. (Note that particle tracing itself corresponds to the case where $t = 0$, and is included as a special case of this framework.)

Lafortune & Willems [1995b] has described an alternative approach to reducing the number of visibility tests. His methods are based on standard Russian roulette and do not attempt to maximize efficiency. We have not made a detailed numerical comparison of the two approaches.

[11] Typically this representation is practical only when most surfaces are diffuse, so that the directional dependence of $L(\mathbf{x}, \omega)$ does not need to be represented.

[12] The dual basis functions satisfy $\langle \tilde{B}_i, B_j \rangle = 1$ when $i = j$, and $\langle \tilde{B}_i, B_j \rangle = 0$ otherwise. For example, when $\{B_1, \ldots, B_M\}$ is an orthonormal basis, then $\tilde{B}_j = B_j$.

10.7 Conclusions

Bidirectional path tracing is an effective rendering algorithm for many kinds of indoor scenes, with or without strong indirect lighting. By using a range of different sampling techniques that take into account different factors of the integrand, it can render a wide variety of lighting effects efficiently and robustly. The algorithm is unbiased, and supports the same range of geometry and materials as standard path tracing.

It is possible to construct scenes where bidirectional path tracing improves on the variance of standard path tracing by an arbitrary amount. To do so, it suffices to increase the intensity of the indirect illumination. In the test case of Figure 10.3, for example, the variance of path tracing increases dramatically as we reduce the size of the directly illuminated area on the ceiling, while bidirectional path tracing is relatively unaffected.

On the other hand, one weakness of the basic bidirectional path tracing algorithm is that there is no intelligent sampling of the light sources. For example, if we were to simulate the lighting in a single room of a large building, most of the light subpaths would start on a light source in a room far from the portion of the scene being rendered, and thus would not contribute. This suggests the idea of sampling light sources according to some estimate of their indirect lighting contribution. Note that methods have already been developed to accelerate the *direct* lighting component when there are many lights, for example by recording information in a spatial subdivision [Shirley et al. 1996]. However, these methods do not help with choosing the initial vertex of a light subpath. In general, we would like to choose a light source that is nearby physically, but is not necessarily directly visible to the viewer.

Similarly, bidirectional path tracing is not suitable for outdoor scenes, or for scenes where the light sources and the viewer are separated by difficult geometry (e.g. a door slightly ajar). In these cases the independently chosen eye and light subpaths will probably not be visible to each other.

Finally, note that bidirectional path tracing can miss the contributions of some paths if point light sources and perfectly specular surfaces are allowed. (This is true of standard path tracing as well.) For example, the algorithm is not capable of rendering caustics from a point source, when viewed indirectly through a mirror using a pinhole lens. This is because bidirectional path tracing is based on local path sampling techniques and thus it is will miss the contributions of paths that do not contain two adjacent non-specular vertices (see Section 8.3.3). However, recall that such paths cannot exist if a finite-aperture lens is used, or if only area light sources are used, or if there are no perfectly specular surfaces in the given scene. Thus bidirectional path tracing is unbiased for all physically valid scene models.