3 Irradiance Caching and Derived Methods

"Originality is merely an illusion."

—M.C. Escher, 1898–1972

SINCE its inception in 1988, irradiance caching [Ward et al., 1988] has become a widely-used technique for computing global illumination. In this chapter we explore this influential technique and the numerous extensions and improvements which have followed. We discuss the strengths of the algorithm that have led to its widespread use in academia as well as industry [Tabellion and Lamorlette, 2004; Kato, 2002]. We also present derivations needed to fully understand the approach, which are currently missing in the literature. Finally, we conclude by highlighting some shortcomings of the algorithm, which provides motivation for the contributions presented in Chapters 5 and 6 of this dissertation.

3.1 Algorithm Overview

Ray tracing techniques solve the rendering equation by means of recursive point-sampling. At each level of recursion, shading is evaluated by (1) computing the direct lighting from light sources, (2) computing specular reflection and refraction, and (3) computing diffuse indirect reflections. The lighting integral is effectively split up into disjoint components, each of which is approximated using the point-sampling processing of ray tracing. The cost of performing each of these operations is directly related to the number of rays that need to be traced. Contributions from (1) and (2) typically subtend a small solid angle and can therefore be effectively integrated using a relatively small number of sample rays. In contrast, (3) involves a lighting integral over

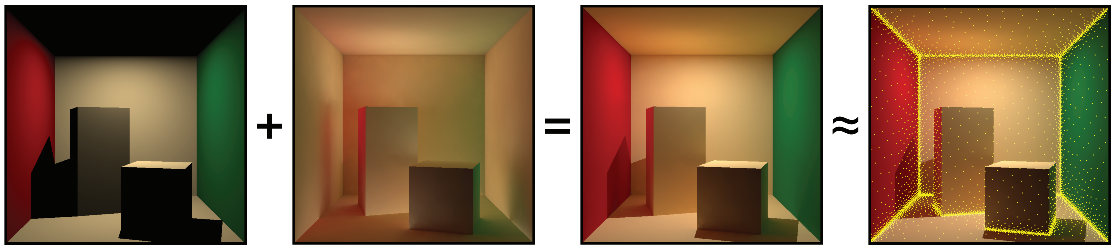

Figure 3.1: Irradiance caching decomposes the incident lighting into a *direct* component (far left) and an *indirect* component (left). The total lighting is obtained by summing these components (right). Even though the direct component may have sharp discontinuities, the indirect illumination tends to vary smoothly across Lambertian surfaces. Irradiance caching exploits this property by only computing this costly value at sparse locations (far right, shown in yellow) and using interpolation whenever possible.

the whole visible hemisphere, the approximation of which typically requires tracing hundreds or thousands of rays.

Irradiance caching is an acceleration technique for computing this diffuse indirect illumination component. Direct lighting tends to have sharp illumination discontinuities, for instance, near hard shadow boundaries. However, Ward et al. made the observation that, although diffuse indirect illumination is expensive to compute, the integration over the whole hemisphere causes this component to be extremely low frequency. This causes indirect illumination to change slowly across Lambertian surfaces. Irradiance caching exploits this property: instead of evaluating the costly illumination integral at each pixel on the image, a high-quality computation is performed only at sparse locations. These irradiance computations are cached and, when possible, reused through interpolation. This approach is illustrated in Figure 3.1.

Irradiance caching incorporates the beneficial properties of radiosity methods while retaining the generality of Monte Carlo approaches. If the cost of interpolating irradiance is negligible compared to a full integration, then the resolution of the rendered image has little impact on the computation time needed to simulate global illumination. However, unlike radiosity, irradiance caching gains further efficiency by concentrating computational effort only in regions of the scene which are directly visible by the camera. Furthermore, the algorithm uses a separate data structure to store the irradiance values. Hence, it does not artificially impose a coupling between the lighting simulation and the geometric representation of the scene. This is a major advantage, as there is no restriction on the kind of geometry that can be simulated with this technique. The algorithm originally described by Ward et al. was used to accelerate the computation of all indirect diffuse illumination within the RADIANCE Monte Carlo ray tracer [Ward, 1994]; however, since that time, irradiance caching has proven popular also as a high-quality final gather pass for other methods such as photon mapping, which we discuss in Chapter 7.

In the irradiance caching algorithm, indirect lighting values are cached in the following manner:

Algorithm 3.1: COMPUTEINDIRECTIRRADIANCE(**x**)

```
1 if there is at least one stored irradiance value near x then
2     Interpolate irradiance from the stored value(s).
3 else
4     Compute and store a new irradiance value at x.
```

The above procedure computes irradiance in a lazy, on-demand fashion. In order to implement this algorithm, a number of outstanding issues must first be addressed:

- How do we compute the irradiance values?
- What criterion is used to determine whether a cache point is “near”?
- How do we interpolate the nearby cached irradiance values?
- What data structure should be used to provide for efficient storage and query of the cache points?

We answer these questions in the following sections.

3.2 Computing Irradiance

In this section we describe the method used to compute an accurate irradiance estimate. This method is employed when no nearby cache points are found.

The irradiance at a point **x** can be expressed in terms of the incident radiance using

Equation 2.6:

$$E(\mathbf{x}) = \int_{\Omega} L(\mathbf{x} \leftarrow \vec{\omega})\,(\vec{\mathbf{n}} \cdot \vec{\omega})\; d\vec{\omega}. \tag{3.1}$$

This integral can be approximated using ray tracing by evaluating the Monte Carlo estimator:

$$E(\mathbf{x}) \approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{L(\mathbf{x} \leftarrow \vec{\omega})\,(\vec{\mathbf{n}} \cdot \vec{\omega})}{pdf(\vec{\omega})}. \tag{3.2}$$

Ward et al. use two variance reduction techniques to more efficiently evaluate this estimate. Choosing to distribute the sample rays with a cosine probability distribution, $pdf(\vec{\omega}) = (\vec{\mathbf{n}} \cdot \vec{\omega})/\pi$, provides variance-reduction through importance sampling by allowing the foreshortening term to cancel out:

$$E(\mathbf{x}) \approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{L(\mathbf{x} \leftarrow \vec{\omega})(\vec{\mathbf{n}} \cdot \vec{\omega})}{(\vec{\mathbf{n}} \cdot \vec{\omega})/\pi} \Rightarrow \frac{\pi}{N} \sum_{i=0}^{N-1} L(\mathbf{x} \leftarrow \vec{\omega}). \tag{3.3}$$

The evaluation is also stratified by converting to polar coordinates:

$$E(\mathbf{x}) \approx \frac{\pi}{MN} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} L(\mathbf{x}, \theta_j, \phi_k), \tag{3.4}$$

where

$$\theta_j = \sin^{-1}\left(\sqrt{\frac{j+\xi_1}{M}}\right), \qquad \phi_i = 2\pi \frac{i+\xi_2}{N}, \tag{3.5}$$

$(\xi_1, \xi_2) \in [0,1)^2$ are uniformly distributed random numbers, and the angles (θ_j, ϕ_k) are expressed with respect to the local coordinate frame at the evaluation location $\mathbf{x}$. The above expression uses a total of $M \times N$ samples[1]. Monte Carlo integration is covered in more detail in Appendix A.

[1]Generally M is chosen to be approximately πN

3.3 Interpolating Irradiance

For typical scenes, an accurate computation of Equation 3.4 requires hundreds or even thousands of sample rays. This is obviously a very costly operation; however, the goal of the irradiance caching algorithm is to compute these very accurate irradiance estimates at only a few locations within the scene and to perform interpolation or extrapolation whenever possible. Where the irradiance changes slowly we can compute the costly estimates very sparsely, whereas regions which exhibit sharp irradiance changes require more irradiance samples to faithfully capture the indirect illumination. Ward et al. developed a simplified heuristic which estimates an upper bound for the change of the irradiance $\varepsilon_i(\mathbf{x},\vec{\mathbf{n}})$. The inverse of this change in irradiance serves as a weight for interpolation between nearby cached irradiance values:

$$\mathrm{w}_i(\mathbf{x},\vec{\mathbf{n}}) = \frac{1}{\varepsilon_i(\mathbf{x},\vec{\mathbf{n}})}. \tag{3.6}$$

At any shading location, all previously stored cache points whose weight is above a user-specified threshold are used in the averaging. If no cache points have a weight above this threshold, then a new full irradiance estimate needs to be computed using Equation 3.4.

3.3.1 The "Split-Sphere" Model

In order to estimate an upper-bound on the maximal possible irradiance gradient, Ward et al. developed the "split-sphere" heuristic. In this model, a sample point is positioned at the center of a hypothetical spherical environment, which is half black and half white. The sample point is oriented to face the boundary between these two regions as illustrated in Figure 3.2. If we assume that concentrated light sources have been eliminated from the irradiance integral by separately accounting for direct lighting, then this *hypothetical* environment provides the largest possible irradiance gradient configuration.

In the split-sphere, irradiance is a function of the position, $\mathbf{x}$, and surface normal, $\vec{\mathbf{n}}$, of the evaluation location. Therefore, the maximum change in irradiance in the split-sphere is composed of the differential change with respect to change in both $\mathbf{x}$ and $\vec{\mathbf{n}}$. Ward et al. approximated this change by taking a first-order Taylor expansion of irradiance at $\mathbf{x}_0$. From this model, the estimated

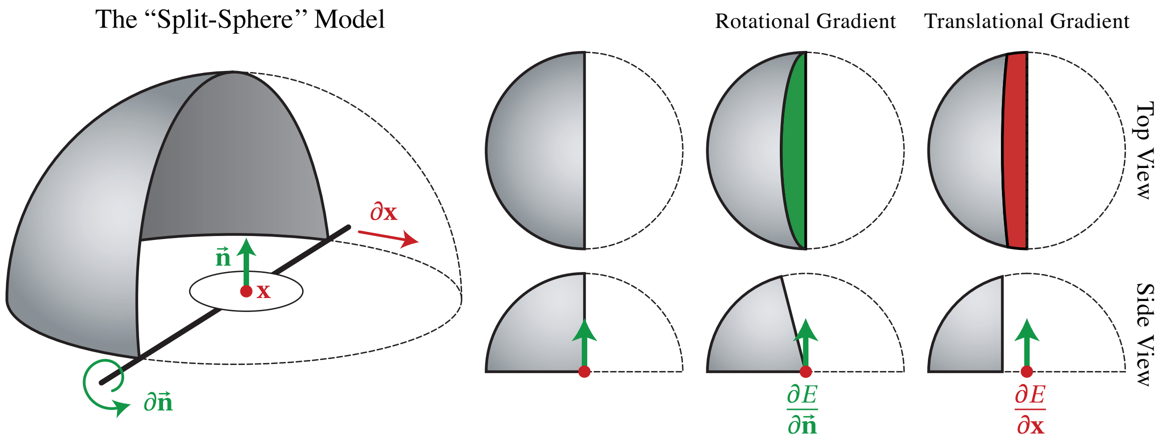

Figure 3.2: The "split-sphere" model. A surface element is positioned at the center of a hypothetical half-dark sphere. The induced change in irradiance is calculated with respect to change in translating the **x** (red) and rotating the orientation $\vec{\mathbf{n}}$ (green). This serves as an estimate of the maximum change in irradiance for *any* scene.

change in irradiance can be computed as:

$$\varepsilon_i(\mathbf{x}, \vec{\mathbf{n}}) = E_i \left(\frac{4}{\pi} \frac{\|\mathbf{x} - \mathbf{x}_i\|}{R_i} + \sqrt{1 - (\vec{\mathbf{n}} \cdot \vec{\mathbf{n}}_i)} \right), \tag{3.7}$$

where:

$$\mathbf{x}_i = \text{irradiance sample location},$$

$$\vec{\mathbf{n}}_i = \text{surface normal at } \mathbf{x}_i,$$

$$E_i = \text{irradiance at } \mathbf{x}_i,$$

$$R_i = \text{harmonic mean distance to surfaces from } \mathbf{x}_i.$$

The R_i quantity represents the "average" distance to visible surfaces. This value is computed using the harmonic mean of the sample distances $r_{j,k}$ as:

$$R_i = \frac{MN}{\sum_{j=0}^{M-1} \sum_{k=0}^{N-1} \frac{1}{r_{j,k}}}, \tag{3.8}$$

where each $r_{j,k}$ is attained from the lengths of the rays traced during the irradiance integration. A complete derivation of how this expression is obtained is provided in the next section.

The weight associated with each cache point is equal to the inverse of the estimated error for that cache point at the query location. The irradiance weighted averaging is therefore performed as follows:

$$E(\mathbf{x},\vec{\mathbf{n}}) \approx \frac{\sum_{i\in S} \mathrm{w}_i(\mathbf{x},\vec{\mathbf{n}})\,E_i}{\sum_{i\in S} \mathrm{w}_i(\mathbf{x},\vec{\mathbf{n}})}, \tag{3.9}$$

where:

$$a = \text{user selected error threshold parameter}$$
$$S = \left\{ i : \mathrm{w}_i(\mathbf{x},\vec{\mathbf{n}}) > \frac{1}{a} \right\}$$

In the final weighted interpolation, Ward et al. use a simplified expression to compute the weight by eliminating the constant terms in Equation 3.7

$$\mathrm{w}_i(\mathbf{x},\vec{\mathbf{n}}) = \frac{1}{\frac{\|\mathbf{x}-\mathbf{x}_i\|}{R_i} + \sqrt{1-\vec{\mathbf{n}}\cdot\vec{\mathbf{n}}_i}}. \tag{3.10}$$

Given this formulation of irradiance averaging, it is easy to compute the maximum distance at which the weight of a cache point will fall below the threshold $\frac{1}{a}$ by ignoring the change in surface normal and examining w_i as a function of only distance. This observation imposes a maximum possible valid radius of aR_i for each cache point. The finite spatial extent of cache points allowed Ward et al. to store them in an octree. The octree is used to efficiently find all cache points which overlap with a given shading location $\mathbf{x}$ in order to evaluate Equation 3.9. Since the valid radius depends on the harmonic mean distance R_i, cache points which “see” nearby geometry will have small valid radii, and cache points far away from visible geometry will have larger valid radii.

3.3.2 Derivation of the "Split-Sphere" Model

We now present a more rigorous derivation of the split-sphere irradiance gradient from Equation 3.7 than is currently available in the literature and illustrate the process in Figure 3.2. The change in irradiance depends on the translation of x and the rotation of the surface normal φ. This change can be approximated using a first order Taylor expansion about x_i:

$$\varepsilon(x,\varphi) = E_i\left(\frac{\partial E}{\partial x}(x-x_i) + \frac{\partial E}{\partial \varphi}(\varphi-\varphi_i)\right). \tag{3.11}$$

In the hypothetical split-sphere environment, the irradiance at a sample point is proportional to the projected area of the bright half of the hemisphere. The change in irradiance is equal to the change in ratio of the bright and dark parts. For the translational component $\frac{\partial E}{\partial x}$, the change in projected area of the bright half is shown in red in Figure 3.2. When Δx is small, the projection of the new region can be well approximated by a thin rectangle of size Δx by $2R$. Hence,

$$\frac{\partial E}{\partial x}\Delta x \approx \frac{2R\Delta x}{\frac{1}{2}\pi R^2} \Rightarrow \frac{4\Delta x}{\pi R}. \tag{3.12}$$

In order to examine the change in irradiance due to surface normal, we consider a rotation of the surface normal by a small angle φ. The axis of rotation that induces the most change in irradiance is parallel to the split plane between the bright and dark half of the environment. We can see in Figure 3.2 that as the sample point's orientation is rotated, the change in the projected bright region can be represented by half of an expanding ellipse (shown in green). The area of an ellipse with axes r_1 and r_2 is given by the formula $A_e = \pi r_1 r_2$. In our case, $r_1 = R$, and r_2 is dependent on the rotational offset $\Delta\varphi$. Examining the geometry in the bottom row of Figure 3.2(b), we see that $r_2 = R\sin(\Delta\varphi)$. This gives us

$$\frac{\partial E}{\partial \varphi}\Delta\varphi \approx \frac{\frac{1}{2}\pi R^2 \sin(\Delta\varphi)}{\frac{1}{2}\pi R^2} \Rightarrow \sin(\Delta\varphi). \tag{3.13}$$

The final Taylor expansion for the change in irradiance expands Equation 3.11 to:

$$\varepsilon(x,\varphi) = E_i \left(\frac{4}{\pi} \frac{|x - x_i|}{R} + \sin\left(\varphi - \varphi_0\right) \right). \tag{3.14}$$

In computing the upper bound on the irradiance gradient, we had assumed a fixed, imaginary split-sphere environment a constant distance R away from the sample location. However, we now need to apply this gradient estimate to the actual environment that is sampled using Equation 3.4. Generalizing the above gradient computation into vector-based quantities involves replacing the change in x into a distance between two points and the change in φ into the angle between two surface normals. This brings us to Equation 3.7.

3.4 Irradiance Gradients

Ward and Heckbert [1992] realized that it is possible to improve the quality of the irradiance estimate significantly by better utilizing the information provided during the sampling process. Specifically, it is possible to compute the *actual* gradient of the irradiance function from the information available in a sampled hemisphere. This is in contrast to the split-sphere derivation in the previous section which only computes a loose upper-bound on the gradient. If both the irradiance and its gradient are available, it is possible to perform higher-order interpolation of the irradiance samples.

The sampling of rays over the hemisphere (Equation 3.2) tells us not only the final approximation for irradiance, but also the brightness, direction, and distance for each individual sample ray. This extra data provides us with the information necessary to deduce a gradient of the sampled irradiance function. As with the split-sphere model, the irradiance gradient can be decomposed into two components: $\nabla_t E$, representing the gradient with respect to translation, and $\nabla_r E$, the gradient with respect to rotation. We present a full derivation of the irradiance gradient computation in the next section.

To take advantage of the additional gradient information in a higher-order interpolation,

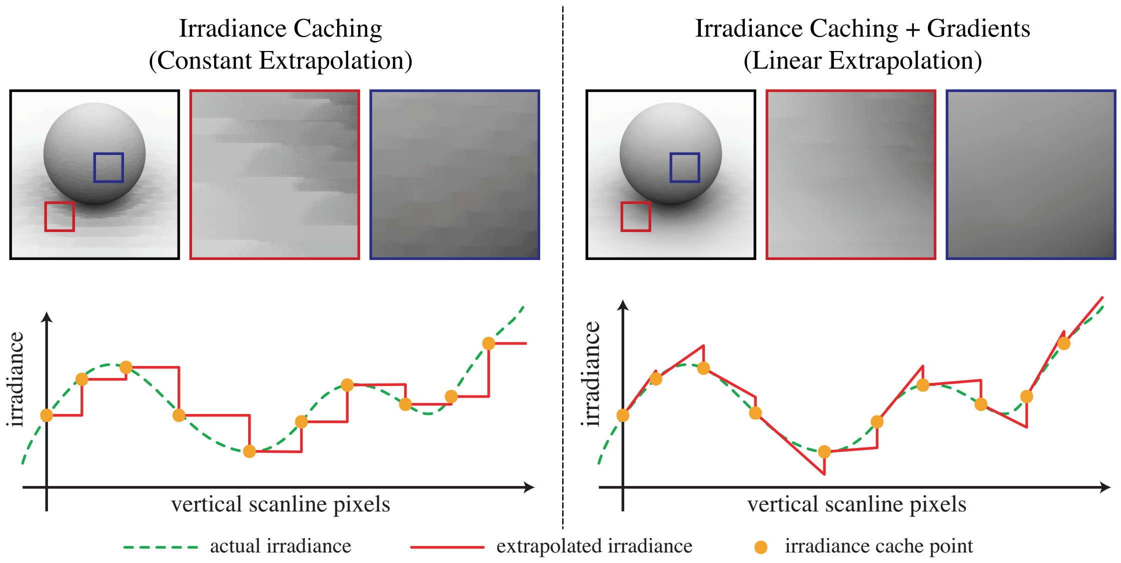

Figure 3.3: Render quality comparison between irradiance caching (left) and irradiance caching with gradients (right). The top row shows a simple scene rendered with both techniques. The two zoomed insets highlight the benefit of the translational (red) and rotational (blue) gradients. The graphs in the bottom image compare the extrapolated irradiance estimate for a single vertical scanline. New cache points are added on demand when the previous cache point's weight falls below a threshold. This results in constant extrapolation for irradiance caching (left), which produces significant errors. Incorporating gradients (right) results in a piecewise linear approximation, which follows the actual irradiance function more closely.

Equation 3.9 can be extended as

$$E(\mathbf{x},\vec{\mathbf{n}}) \approx \frac{\sum_{i\in S} w_i(\mathbf{x},\vec{\mathbf{n}})\,(E_i + (\vec{\mathbf{n}}_i \times \vec{\mathbf{n}})\cdot\nabla_r E_i + (\mathbf{x}-\mathbf{x}_i)\cdot\nabla_t E_i)}{\sum_{i\in S} w_i(\mathbf{x},\vec{\mathbf{n}})}. \tag{3.15}$$

This modification adds contribution from the translational gradient as we *move* away from a cache point and contribution from the rotational gradient as we *rotate* away from a cache point. This can be interpreted as performing a linear extrapolation from one cache point to the next, as opposed to a piecewise constant extrapolation (see Figure 3.3 for a one-dimensional analogy).

The incorporation of gradients into the irradiance caching scheme requires little extra computation but can provide vastly superior interpolation quality. A comparison rendering is provided in Figure 3.3.

3.5 Radiance Caching

One of the underlying observations justifying interpolation for irradiance caching is that reflected radiance due to indirect illumination changes slowly across Lambertian surfaces. Furthermore, since lighting on Lambertian surfaces is view-independent, a single irradiance value is sufficient for interpolating lighting across these surfaces. However, this compact representation is not possible with general surfaces since reflected light is, in general, view-dependent. In a series of papers, Křivánek et al. [2005b,a] extended the irradiance caching method to work with glossy surfaces. They observed that indirect illumination also changes slowly on surfaces with general low-frequency BRDFs. In order to interpolate indirect illumination on view-dependent glossy materials, Křivánek et al. cache the full directional *radiance* function instead of just a scalar irradiance value, and call their extension *radiance caching*.

3.5.1 Radiance Computation

Spherical functions, such as a full radiance field, can be efficiently stored using spherical harmonics. Since the radiance at surfaces is really only defined in the upper hemisphere, Křivánek et al. used hemispherical harmonics instead, which were introduced and derived in an earlier paper [Gautron et al., 2004]. Spherical (SH) and hemispherical (HSH) harmonics act in analogous ways, but HSH can more efficiently approximate functions over just a hemisphere. At a high level, these can be thought of as generalizations of a Fourier series applied to the spherical and hemispherical domains, respectively. In the context of radiance caching they can be interchanged except where explicitly noted. A more detailed introduction to spherical harmonics is provided in Appendix B.

Similarly to irradiance caching, the radiance caching algorithm uses a lazy caching scheme. When no cache records are found nearby, the incident lighting is evaluated accurately using Monte Carlo integration. During this process, the incoming radiance, $L(\mathbf{x} \leftarrow \vec{\omega})$, is projected onto (H)SH and stored as a vector of coefficients, $\Lambda = \{\lambda_l^m\}$, within a new cache record for subsequent interpolation. Each coefficient λ_l^m is computed using the Monte Carlo samples by integrating the product of the incident radiance and the corresponding basis function y_l^m:

$$\lambda_l^m = \frac{2\pi}{N}\sum_{k=0}^{N-1} L(\mathbf{x}\leftarrow\vec{\omega}_k)\, y_l^m(\vec{\omega}_k). \tag{3.16}$$

3.5.2 Radiance Interpolation

As in irradiance caching, efficiency is gained in radiance caching by attempting to interpolate from nearby cache points before computing a new cache record. Radiance caching uses a similar weighted averaging scheme as irradiance caching but interpolates the coefficient vectors instead of the irradiance values. However, the calculation of HSH coefficient vectors in Equation 3.16 was performed within the local coordinate frame of each cache point. Multiple HSH coefficient vectors, therefore, cannot be directly interpolated because they potentially lie in different coordinate frames. Fortunately, functions represented using (H)SH can be efficiently rotated [Gautron et al., 2004; Ivanic and Ruedenberg, 1996; Blanco et al., 1997; Choi et al., 1999]. Hence, before interpolation, each cache record's coefficient vector is aligned to a common coordinate frame to make interpolation possible[2]. This results in the following modification to Equation 3.9:

$$\Lambda(\mathbf{x},\vec{\mathbf{n}}) = \frac{\sum_{i\in S} \mathrm{w}_i(\mathbf{x},\vec{\mathbf{n}})\,\mathbf{M}_i(\Lambda_i)}{\sum_{i\in S} \mathrm{w}_i(\mathbf{x},\vec{\mathbf{n}})}, \tag{3.17}$$

where $\mathbf{M}_i$ is the HSH rotation matrix, which aligns the coordinate frame of cache point $\mathbf{x}_i$ to the frame at the query point $\mathbf{x}$. Computing the interpolated radiance field and the BRDFs as (H)SH allows us to integrate them at any scene location using a simple dot product of their coefficient vectors.

3.5.3 Translational Radiance Gradients

Ward and Heckbert [1992] found that interpolation quality can be substantially improved by calculating gradient information from the hemispherical samples. This same concept can be applied to the radiance caching algorithm. In irradiance caching with gradients, the gradient is

[2] If SH are used instead, this rotation is not necessary, as the radiance fields can be stored in global instead of local space.

composed of a rotational and translational component. Within the radiance caching framework, we already take rotational change into account by analytically rotating the coefficient vectors before interpolation. However, by incorporating a translational gradient we can further improve interpolation quality.

Křivánek et al. proposed two different methods for computing the translational gradient [2005b]. In radiance caching, the translational gradient expresses the change of each HSH coefficient λ_l^m with respect to movement in the local x- or y-axes at the cache location. As with traditional irradiance gradients, the radiance gradients are computed at the same time as the coefficients during the hemispherical sampling.

The first technique calculates the gradient numerically by translating the sample point $\mathbf{x}$ to $\mathbf{x}'$ along the local x-axis, and to $\mathbf{x}''$ along the local y-axis. It estimates the coefficient $\lambda_l^{m\prime}$ and $\lambda_l^{m\prime\prime}$ at these displaced locations and, finally, takes the finite differences:

$$\left[\frac{\partial\lambda_l^m}{\partial x}, \frac{\partial\lambda_l^m}{\partial y}, 0\right] = \left[\frac{\lambda_l^{m\prime} - \lambda_l^m}{\Delta x}, \frac{\lambda_l^{m\prime\prime} - \lambda_l^m}{\Delta y}, 0\right]. \tag{3.18}$$

The second technique proposed by Křivánek et al. [2005b], and also independently by Annen et al. [2004], calculates the gradient by analytically differentiating the terms in Equation 3.16. This calculation involves deriving partial derivatives for the hemispherical harmonics. We omit the full derivation here and refer the interested reader to the original paper [Křivánek et al., 2005b].

The above two gradient computations treat each sample ray independently. Therefore, any distribution of rays can be used, which allows for different sampling strategies such as Quasi-Monte Carlo sampling. However, because each sample is treated independently, neither of the techniques take changing occlusions into account. To address this limitation, Křivánek et al. [2005a] later proposed an improved gradient computation, which accounts for changing occlusions.

The improved gradient calculation takes the approach originally used by Ward and Heckbert [1992] by stratifying the samples and considering the marginal change in incoming radiance with respect to each cell boundary. To account for the fact that in radiance caching the values are projected onto hemispherical harmonic basis functions, Křivánek et al. [2005a] generalize the

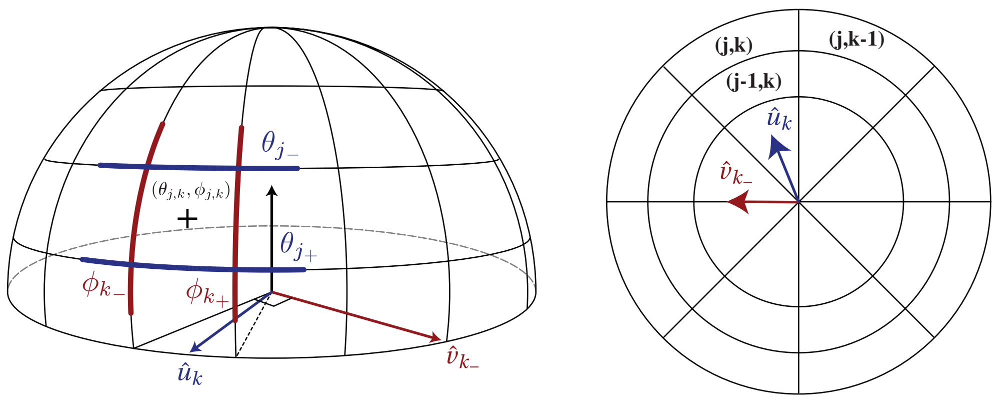

Figure 3.4: The stratified geometry used in the Ward and Heckbert [1992] gradient computation illustrated from the side (left) and above (right). The relevant quantities are described in Table 3.1.

calculations of Ward and Heckbert [1992] to work with an arbitrary weighting function applied to each sample. In Section 3.6 we explore this derivation in more detail and show how it relates to the original irradiance gradient formulation by Ward and Heckbert [1992].

Regardless of the technique used to calculate the gradient, Equation 3.17 can be extended as follows to incorporate the translational gradient information:

$$\Lambda(\mathbf{x},\vec{\mathbf{n}}) = \frac{\sum_{i\in S} \mathrm{w}_i(\mathbf{x},\vec{\mathbf{n}})\,\mathbf{M}_i\left(\Lambda_i + d_x\frac{\partial\Lambda_i}{\partial x} + d_y\frac{\partial\Lambda_i}{\partial y}\right)}{\sum_{i\in S}\mathrm{w}_i(\mathbf{x},\vec{\mathbf{n}})}, \tag{3.19}$$

where d_x and d_y are the displacements of $\mathbf{x}-\mathbf{x}_i$ along the local x- and y-axes.

3.6 Derivation of Irradiance Gradients

Since a complete derivation of the irradiance gradient computation is not included in the literature, we provide one here. The following discussion uses the notation described in Table 3.1 and illustrated in Figure 3.4.

At a high level, the irradiance computation in Equation 3.2 can be thought of as a weighted sum of the radiance. The contribution of each sample is the product of the radiance through the hemispherical cell $L(\mathbf{x}\leftarrow\vec{\omega}_{j,k})$, the solid angle or "area" of the cell $A_{j,k}$, and the cosine term

$(\vec{\mathbf{n}} \cdot \vec{\omega}_{j,k})$:

$$E(\mathbf{x}) \approx \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} A_{j,k} \, L(\mathbf{x} \leftarrow \vec{\omega}_{j,k}) \, (\vec{\mathbf{n}} \cdot \vec{\omega}_{j,k}). \tag{3.20}$$

The cell area is the integral over the bounds of each cell:

$$A_{j,k} = \int_{\phi_{k_-}}^{\phi_{k_+}} \int_{\theta_{j_-}}^{\theta_{j_+}} \sin\theta \, d\theta \, d\phi \implies (\cos\theta_{j_-} - \cos\theta_{j_+})(\phi_{k_+} - \phi_{k_-}). \tag{3.21}$$

The irradiance gradient considers how these terms change when translating the center of projection or rotating the surface normal by an infinitesimal amount. The rotational and translational gradients are obtained by differentiating Equation 3.20:

$$\begin{aligned}
\nabla_r E(\mathbf{x}) &\approx \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} \nabla_r (A_{j,k} \, L_{j,k} \cos\theta_j) \\
\nabla_t E(\mathbf{x}) &\approx \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} \nabla_t (A_{j,k} \, L_{j,k} \cos\theta_j),
\end{aligned}$$

where we have used the shorthand $L_{j,k} = L(\mathbf{x} \leftarrow \vec{\omega}_{j,k})$, and $\cos\theta_j = (\vec{\mathbf{n}} \cdot \vec{\omega}_{j,k})$. Since in irradiance caching surfaces are assumed to be Lambertian, the radiance in each cell remains constant, which

Table 3.1: Definitions of quantities used in the irradiance gradients derivation.

Symbol	Description
N	Number of divisions in azimuthal angle ϕ
M	Number of divisions in elevation angle θ
$\vec{\omega}_{j,k}$	Sample direction for cell (j,k), i.e.: $(\theta_{j,k}, \phi_{j,k})$
$A_{j,k}$	Area of cell (j,k), i.e.: $(MN)^{-1} pdf(\vec{\omega}_{j,k})^{-1}$
θ_j	Elevation angle at center of cells $(j,\cdot)$
ϕ_k	Azimuthal angle at center of cells $(\cdot,k)$
$(j_-,k),(j_+,k)$	Cell boundaries $(j \leftrightarrow j\text{-}1, k)$ and $(j \leftrightarrow j\text{+}1, k)$
$(j,k_-),(j,k_+)$	Cell boundaries $(j, k \leftrightarrow k\text{-}1)$ and $(j, k \leftrightarrow k\text{+}1)$
θ_{j_+}	Elevation angle at boundary $(j_+,\cdot)$
θ_{j_-}	Elevation angle at boundary $(j_-,\cdot)$
θ_{j_+}	Elevation angle at boundary $(j_+,\cdot)$
ϕ_{k_-}	Azimuthal angle at boundary $(\cdot,k_-)$
ϕ_{k_+}	Elevation angle at boundary $(\cdot,k_+)$
$\hat{u}_k$	Tangent vector in the ϕ_k direction
$\hat{v}_k$	Tangent vector in the $\phi_k + \frac{\pi}{2}$ direction
$\hat{v}_{k_-}$	Tangent vector in the $\phi_{k_-} + \frac{\pi}{2}$ direction

simplifies the above expressions to:

$$\nabla_r E(\mathbf{x}) \approx \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} L_{j,k} \nabla_r (A_{j,k} \cos\theta_j) \tag{3.22}$$

$$\nabla_t E(\mathbf{x}) \approx \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} L_{j,k} \nabla_t (A_{j,k} \cos\theta_j). \tag{3.23}$$

In the following sections we derive expressions for $\nabla_r E(\mathbf{x})$ and $\nabla_t E(\mathbf{x})$.

3.6.1 The Rotational Gradient

The rotational gradient needs to express the change in the projection solid angle $A_{j,k} \cos\theta_j$ of each hemisphere sample cell as it is rotated in any direction. We make the observation that under a rotation of the surface normal, the stratified cell areas $A_{j,k}$ do not change. Hence,

$$\nabla_r (A_{j,k} \cos\theta_j) = A_{j,k} \nabla_r \cos\theta_j, \tag{3.24}$$

and Equation 3.22 reduces to

$$\nabla_r E(\mathbf{x}) \approx \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} L_{j,k} A_{j,k} \nabla_r \cos\theta_j. \tag{3.25}$$

The only factor that influences the rotational irradiance gradient is the cosine foreshortening term $\cos\theta$, and $\frac{d}{d\theta}\cos\theta_j = -\sin\theta_j$. In order to compute the gradient of the whole stratified hemisphere, we need to sum up all of the marginal cell differentials multiplied by their corresponding rotation axes:

$$\nabla_r E = -\sum_{k=0}^{N-1} \left\{ \hat{v}_k \sum_{j=0}^{M-1} L_{j,k} A_{j,k} \sin\theta_j \right\}. \tag{3.26}$$

In the irradiance caching algorithm the hemispherical samples are constructed using a cosine-weighted distribution. This simplifies Equation 3.21 to

$$A_{j,k} = \frac{\pi}{M N \cos\theta_j}, \tag{3.27}$$

which reduces Equation 3.26 to

$$\nabla_r E = -\frac{\pi}{MN}\sum_{k=0}^{N-1}\left\{\hat{v}_k \sum_{j=0}^{M-1} L_{j,k}\frac{\sin\theta_j}{\cos\theta_j}\right\}, \tag{3.28}$$

$$= -\frac{\pi}{MN}\sum_{k=0}^{N-1}\left\{\hat{v}_k \sum_{j=0}^{M-1} L_{j,k}\tan\theta_j\right\}. \tag{3.29}$$

3.6.2 The Translational Gradient

In order to compute the gradient due to translation, we use a derivation similar to that provided by Křivánek et al. [2005a]. We observe that the cosine foreshortening term does not change under a translation, therefore, $\nabla_t E(\mathbf{x})$ simplifies to computing the change in cell area:

$$\nabla_t E(\mathbf{x}) \approx \sum_{j=0}^{M-1}\sum_{k=0}^{N-1} \nabla_t A_{j,k} L_{j,k}\cos\theta_j. \tag{3.30}$$

Since we are only interested in translational gradients along the surface, the change in cell area can be reduced to computing the marginal change in the cell walls with respect to translation along the base plane. For each cell (j,k), we decompose the gradient into directional derivatives of the four neighboring cell walls:

$$\nabla_t A_{j,k} = \hat{u}_k \cdot \nabla_{\hat{u}_k} A_{j_-,k} - \hat{u}_k \cdot \nabla_{\hat{u}_k} A_{j_+,k} + \hat{v}_{k_-} \cdot \nabla_{\hat{v}_{k_-}} A_{j,k_-} - \hat{v}_{k_+} \cdot \nabla_{\hat{v}_{k_+}} A_{j,k_+}, \tag{3.31}$$

where $\nabla_{\hat{u}_k} A_{j_-,k}$ computes the scalar directional derivative of area due to movement of boundary (j_-,k) along the $\hat{u}_k$ direction, and the other quantities are defined analogously (see Table 3.1). These directional derivatives can be interpreted as computing the length of the cell wall multiplied by the rate of motion of the wall due to movement in the perpendicular direction. This is illustrated in Figure 3.5.

Since neighboring cells share cell boundaries, it is more convenient to express the translational gradient in Equation 3.30 as a summation over all the cell boundaries instead of the cells. The movement of a cell wall will increase the contribution from one adjacent cell and induce a corresponding decrease from the other adjacent cell. Because of this, we consider the difference

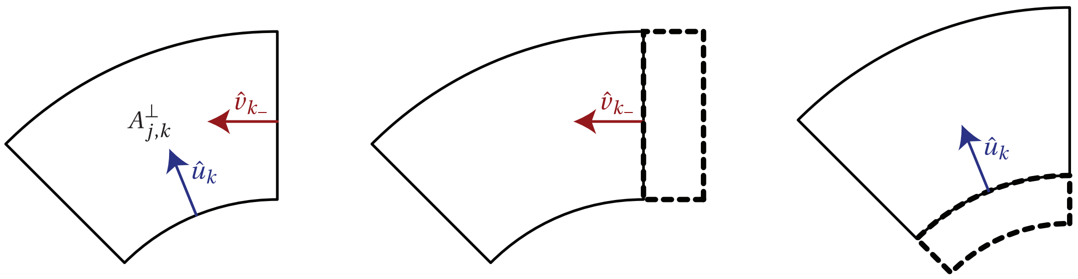

Figure 3.5: The change in cell area due to translation is decomposed into the movement of each cell wall. For both the azimuthal and polar directions, the induced change in cell area is equal to the length of the wall multiplied by the rate of motion of the wall.

in radiance between the two adjacent cells to determine the effect on the overall irradiance sum[3]:

$$\nabla_t E(\mathbf{x}) = \sum_{k=0}^{N-1} \left(\hat{u}_k \sum_{j=1}^{M-1} \nabla_{\hat{u}_k} A_{j_-,k} (L_{j,k} - L_{j\text{-}1,k}) \cos\theta_{j_-} + \hat{v}_{k_-} \sum_{j=0}^{M-1} \nabla_{\hat{v}_{k_-}} A_{j,k_-} (L_{j,k} - L_{j,k\text{-}1}) \cos\theta_j \right). \quad (3.32)$$

This formulation is similar to that provided by Křivánek et al. [2005a] but with the correction that the first term is weighted by the cosine at the cell boundary, $\cos\theta_{j_-}$, instead of at the cell center, $\cos\theta_j$. We derive the rate of motion of the two types of cell walls in the following sections.

Displacement Along $\hat{u}_k$

The induced change in cell area by moving cell wall (j_-, k) along the base plane vector $\hat{u}_k$ is:

$$\nabla_{\hat{u}_k} A_{j_-,k} = \nabla_{\hat{u}_k} \theta_{j_-} \cdot \frac{\partial A_{j,k}}{\partial \theta_{j_-}}, \quad (3.33)$$

$$= \nabla_{\hat{u}_k} \theta_{j_-} \cdot (\phi_{k_+} - \phi_{k_-}) \sin\theta_{j_-}, \quad (3.34)$$

where the $(\phi_{k_+} - \phi_{k_-}) \sin\theta_{j_-}$ arises directly from differentiating the definition of $A_{j,k}$ in Equation 3.21. To determine the change in elevation angle as we move along direction $\hat{u}_k$, we instead

[3]Note that the degenerate cell wall at the pole (where $\theta = 0$) is excluded in the loop by starting at $j = 1$.

consider the analogous problem within the canonical hemispherical parametrization:

$$x = r\cos\phi\sin\theta, \tag{3.35}$$

$$y = r\sin\phi\sin\theta, \tag{3.36}$$

$$z = r\cos\theta, \tag{3.37}$$

$$r = \sqrt{x^2+y^2+z^2}, \tag{3.38}$$

$$\phi = \tan^{-1}\left(\frac{y}{x}\right), \tag{3.39}$$

$$\theta = \cos^{-1}\left(\frac{z}{r}\right), \tag{3.40}$$

and determine the change in θ as we translate along x:

$$\begin{aligned}
\frac{\partial\theta}{\partial x} &= \frac{zx}{r^3\left(1-\frac{z^2}{r^2}\right)^{\frac{1}{2}}}, && \text{by differentiating Equation 3.40,} \\
&= \frac{\cos\theta x}{r^2\left(1-\cos^2\theta\right)^{\frac{1}{2}}}, && \text{because } \frac{z}{r} = \cos\theta, \\
&= \frac{\cos\theta x}{r^2\sin\theta}, && \text{because } \sin^2\theta+\cos^2\theta = 1, \\
&= \frac{\cos\theta\cos\phi\sin\theta}{r\sin\theta}, && \text{because } \frac{x}{r} = \cos\phi\sin\theta, \\
&= \frac{\cos\theta}{r}, && \text{because we evaluate at } y = 0\text{, where } \cos\phi = 1.
\end{aligned} \tag{3.41}$$

In order to apply this derivation to the hemispherical samples, Ward and Heckbert [1992] observed that as the sample location moves, the rate of motion of the boundary between two cells is always determined by the closer of the two sample directions. This observation takes changing occlusions into account. Therefore, the radius in the denominator should use the minimum of the distances for the two neighboring cells, resulting in:

$$\nabla_{\hat{u}_k} A_{j_-,k} = \frac{(\phi_{k_+}-\phi_{k_-})\sin\theta_{j_-}\cos\theta_{j_-}}{\min\left(r_{j,k}, r_{j-1,k}\right)}. \tag{3.42}$$

Displacement Along $\hat{v}_{k_-}$

We also consider a displacement along $\hat{v}_{k_-}$ for cell walls of the form (j, k_-). The induced change in area is:

$$\nabla_{\hat{v}_{k_-}} A_{j,k_-} = \nabla_{\hat{v}_{k_-}} \phi_{k_-} \cdot \frac{\partial A_{j,k}}{\partial \phi_{k_-}}, \tag{3.43}$$

$$= \nabla_{\hat{v}_{k_-}} \phi_{k_-} \cdot (\cos\theta_{j_+} - \cos\theta_{j_-}). \tag{3.44}$$

In order to compute $\nabla_{\hat{v}_{k_-}} \phi_{k_-}$ we again turn to the canonical parametrization and consider the change of ϕ as we translate along y:

$$\begin{aligned}
\frac{\partial\phi}{\partial y} &= \frac{x}{x^2+y^2}, && \text{by differentiating Equation 3.39,}\\
&= \frac{1}{x}, && \text{because we evaluate at } y = 0,\\
&= \frac{1}{r\cos\phi\sin\theta}, && \text{from Equation 3.35,}\\
&= \frac{1}{r\sin\theta}, && \text{because } \cos\phi = 1 \text{ at } y = 0.
\end{aligned} \tag{3.45}$$

By again using the minimum of the neighboring cell distances we arrive at the following induced change by translating along $\hat{v}_{k_-}$:

$$\nabla_{\hat{v}_{k_-}} A_{j,k_-} = \frac{(\cos\theta_{j_+} - \cos\theta_{j_-})}{\sin\theta_j \min\left(r_{j,k}, r_{j-1,k}\right)}. \tag{3.46}$$

Putting it Together

Plugging the values for the cell wall differentials from Equations 3.42 and 3.46 into Equation 3.32, the final translation gradient is:

$$\begin{aligned}
\nabla_t E(\mathbf{x}) = \sum_{k=0}^{N-1} \Bigg(&\hat{u}_k \sum_{j=1}^{M-1} \frac{(\phi_{k_+} - \phi_{k_-})\sin\theta_{j_-}\cos\theta_{j_-}}{\min\left(r_{j,k}, r_{j-1,k}\right)} \left(L_{j,k} - L_{j-1,k}\right)\cos\theta_{j_-} + \\
&\hat{v}_{k_-} \sum_{j=0}^{M-1} \frac{(\cos\theta_{j_+} - \cos\theta_{j_-})}{\sin\theta_j \min\left(r_{j,k}, r_{j-1,k}\right)} \left(L_{j,k} - L_{j,k-1}\right)\cos\theta_j \Bigg).
\end{aligned} \tag{3.47}$$

3.6.3 Equivalence of the Gradient Formulations

Ward and Heckbert [1992] compute the translational gradient by considering the movement of the *projected* cell boundaries. Their original formulation is:

$$\nabla_t E(\mathbf{x}) = \sum_{k=0}^{N-1} \left(\hat{u}_k \frac{2\pi}{N} \sum_{j=1}^{M-1} \frac{\sin\theta_{j_-} \cos^2\theta_{j_-}}{\min\left(r_{j,k}, r_{j-1,k}\right)} \left(L_{j,k} - L_{j-1,k}\right) + \right.$$
$$\left. \hat{v}_{k_-} \sum_{j=0}^{M-1} \frac{(\sin\theta_{j_+} - \sin\theta_{j_-})}{\min\left(r_{j,k}, r_{j-1,k}\right)} \left(L_{j,k} - L_{j,k-1}\right) \right). \tag{3.48}$$

The translational gradients we derived in the previous section differ from this formulation. We instead followed the approach of Křivánek et al. [2005a] and considered the movement of the *unprojected* cell boundaries weighted by a cosine term. As Křivánek et al. showed, this approach has extra flexibility since it allows the use of any weighting function in place of the cosine term.

Křivánek et al. [2005a] stated that their "formula yields numerically very similar results to that of Ward and Heckbert." Unfortunately, the derivation by Křivánek et al. [2005a] contained a few minor errors which obscured the fact that these gradient formulations are in fact *analytically* equivalent. In this section we prove this equivalence.

Equivalence Along $\hat{u}_k$

We first consider the gradient contribution along $\hat{u}_k$. In our formulation, the cell radiance difference, $\left(L_{j,k} - L_{j-1,k}\right)$, is weighted by:

$$\frac{(\phi_{k_+} - \phi_{k_-}) \sin\theta_{j_-} \cos\theta_{j_-}}{\min\left(r_{j,k}, r_{j-1,k}\right)} \cdot \cos\theta_{j_-}, \tag{3.49}$$

whereas in the original Ward and Heckbert formulation it is:

$$\frac{2\pi}{N} \frac{\sin\theta_{j_-} \cos^2\theta_{j_-}}{\min\left(r_{j,k}, r_{j-1,k}\right)}. \tag{3.50}$$

For a uniform or cosine-weighted stratification, $(\phi_{k_+} - \phi_{k_-}) = \frac{2\pi}{N}$, which makes these two formulations equivalent along $\hat{u}_k$.

Equivalence Along $\hat{v}_{k_-}$

Showing that the gradient formulations along $\hat{v}_{k_-}$ are equivalent is a bit more involved. Our formulation weights each cell radiance difference, $\left(L_{j,k} - L_{j,k-1}\right)$, by:

$$\frac{\cos\theta_{j_+} - \cos\theta_{j_-}}{\sin\theta_j \min\left(r_{j,k}, r_{j-1,k}\right)} \cdot \cos\theta_j \Longrightarrow \frac{\cos\theta_{j_+} - \cos\theta_{j_-}}{\tan\theta_j \min\left(r_{j,k}, r_{j-1,k}\right)} \tag{3.51}$$

and Ward and Heckbert weight it by:

$$\frac{\sin\theta_{j_+} - \sin\theta_{j_-}}{\min\left(r_{j,k}, r_{j-1,k}\right)}. \tag{3.52}$$

To show that these are equivalent, we start by expressing the boundary angles, θ_{j_-} and θ_{j_+}, as offsets from the angle at the cell center, θ_j:

$$\theta_{j_-} = \theta_j - \frac{\Delta\theta}{2}, \tag{3.53}$$

$$\theta_{j_+} = \theta_j + \frac{\Delta\theta}{2}, \tag{3.54}$$

where $\frac{\Delta\theta}{2}$ is half the cell-width along the θ direction. Using this decomposition, we can rewrite Equations 3.51 and 3.52 as:

$$\frac{\cos\left(\theta_j - \frac{\Delta\theta}{2}\right) - \cos\left(\theta_j + \frac{\Delta\theta}{2}\right)}{\tan\theta_j \min\left(r_{j,k}, r_{j-1,k}\right)}, \qquad \text{and} \qquad \frac{\sin\left(\theta_j + \frac{\Delta\theta}{2}\right) - \sin\left(\theta_j - \frac{\Delta\theta}{2}\right)}{\min\left(r_{j,k}, r_{j-1,k}\right)}.$$

By using the angle sum and difference trigonometric identities:

$$\sin\left(\theta_j \pm \frac{\Delta\theta}{2}\right) = \sin\theta_j \cos\frac{\Delta\theta}{2} \pm \cos\theta_j \sin\frac{\Delta\theta}{2}, \tag{3.55}$$

$$\cos\left(\theta_j \pm \frac{\Delta\theta}{2}\right) = \cos\theta_j \cos\frac{\Delta\theta}{2} \mp \sin\theta_j \sin\frac{\Delta\theta}{2}, \tag{3.56}$$

these two formulations simplify to equivalent expressions:

$$\frac{\left(\cancel{\cos\theta_j \cos\tfrac{\Delta\theta}{2}} + \sin\theta_j \sin\tfrac{\Delta\theta}{2}\right) - \left(\cancel{\cos\theta_j \cos\tfrac{\Delta\theta}{2}} - \sin\theta_j \sin\tfrac{\Delta\theta}{2}\right)}{\tan\theta_j \min\left(r_{j,k}, r_{j-1,k}\right)} \Longrightarrow \frac{2\cancel{\sin\theta_j} \sin\tfrac{\Delta\theta}{2}}{\cancel{\tan\theta_j} \min\left(r_{j,k}, r_{j-1,k}\right)},$$

$$\Longrightarrow \frac{2\cos\theta_j \sin\tfrac{\Delta\theta}{2}}{\min\left(r_{j,k}, r_{j-1,k}\right)}, \qquad (3.57)$$

$$\frac{\left(\cancel{\sin\theta_j \cos\tfrac{\Delta\theta}{2}} + \cos\theta_j \sin\tfrac{\Delta\theta}{2}\right) - \left(\cancel{\sin\theta_j \cos\tfrac{\Delta\theta}{2}} - \cos\theta_j \sin\tfrac{\Delta\theta}{2}\right)}{\min\left(r_{j,k}, r_{j-1,k}\right)} \Longrightarrow \frac{2\cos\theta_j \sin\tfrac{\Delta\theta}{2}}{\min\left(r_{j,k}, r_{j-1,k}\right)}. \qquad (3.58)$$

3.7 Other Extensions

Several other extensions to the irradiance caching method have been proposed in the literature. We briefly summarize some of the most significant contributions in this section.

3.7.1 Approximate Global Illumination

Several papers have explored the topic of accelerating the global illumination computation by making physically inaccurate approximations during the irradiance caching algorithm. These methods loosen the physical restrictions of the computation to gain efficiency, while attempting to minimize the approximation errors and artifacts introduced. We will cover two different approaches to this problem. The first uses simplified scene geometry during the indirect lighting calculation, while the second decomposes the indirect light into near and far components in order to exploit their individual advantages.

Using Simplified Geometry

The algorithm proposed in Tabellion and Lamorlette [2004] was developed within the context of a production rendering system for computer animated films. The primary concerns involved were efficiency and artistic control, as opposed to physically accurate simulation. One of the approximations employed to make the global illumination calculation more computationally tractable was the use of simplified scene geometry during the indirect lighting calculation.

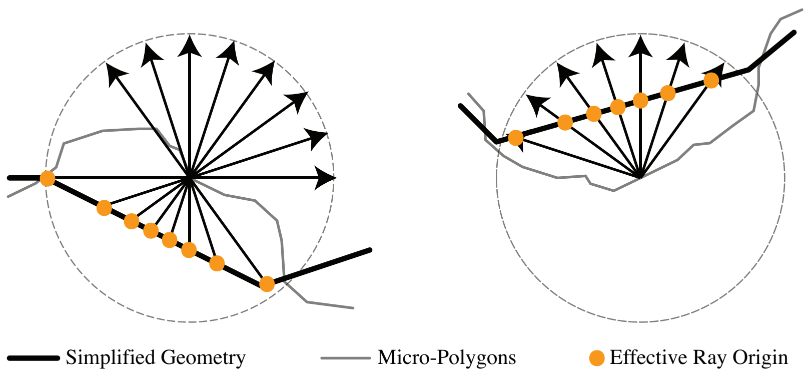

Figure 3.6: Tabellion and Lamorlette adjusted the ray origins in the irradiance estimate in order to compensate for simplified geometry.

The main bottleneck of the irradiance caching algorithm is the time spent tracing rays. To mitigate this, the system proposed by Tabellion and Lamorlette ray traces coarsely tessellated geometry. Since the underlying rendering system was a micro-polygon renderer, coarse representations of the scene geometry were readily available. Using simplified geometry had been proposed in the past [Rushmeier et al., 1994]; however, Tabellion and Lamorlette's system used coarse geometry even near the ray origin. Since rays originate at positions on the fine-scale geometry, problems can occur from incorrect "self-intersections" with the coarse geometry. This problem, and its proposed solution, is illustrated in Figure 3.6.

When tracing an indirect ray, the actual ray origin may need to be adjusted to compensate for the geometric inconsistency. The new ray origin is found by tracing within a user-defined distance both before and after the origin. The closest hit to the ray origin becomes the new ray origin.

Using simplified geometry for the indirect rays allowed Tabellion and Lamorlette to gain an order of magnitude improvement in performance.

Error Metric. Tabellion and Lamorlette also introduced a new error metric used during the weighted averaging of the cached irradiance samples. They argued that the new error metric improves cache point distribution for scenes with high geometric detail by reducing clumping in corners. To compute the error of using a cached sample i at position $\mathbf{x}$, they proposed the following formula:

$$\varepsilon_i = \kappa \max\left(\varepsilon_{pi}(\mathbf{x}), \varepsilon_{ni}(\vec{\mathbf{n}})\right), \qquad \text{with} \tag{3.59}$$

$$\varepsilon_{pi} = \frac{\|\mathbf{x} - \mathbf{x}_i\|}{\max\left(\min\left(\frac{R_i}{2}, R_+\right), R_-\right)}, \quad \text{and} \tag{3.60}$$

$$\varepsilon_{ni} = \frac{\sqrt{1 - \vec{\mathbf{n}} \cdot \vec{\mathbf{n}}_i}}{\sqrt{1 - \cos(10°)}}. \tag{3.61}$$

Here, κ is the only user-adjustable quality parameter and is by default set to 1. R_i is the closest distance to surfaces seen from $\mathbf{x}_i$. R_- and R_+ are 1.5 and 10 times the square root of the projected pixel area, respectively. These act like minimum and maximum valid radii constraints on the cache points. Their inclusion brings about the reduced clumping properties of the metric.

A weighted average of cache points whose weights are non-negative are included in the interpolated irradiance. The new error metric changes the weighting function used to

$$\mathrm{w}_i(\mathbf{x}, \vec{\mathbf{n}}) = 1 - \varepsilon_i(\mathbf{x}, \vec{\mathbf{n}}). \tag{3.62}$$

Irradiance Decomposition

Arikan et al. [2005] noted that, overall, irradiance caching is an excellent technique for efficiently computing global illumination but suffers from over-sampling around areas of high geometric complexity due to the potentially rapid change in irradiance in these areas. Especially with the trend of increasing geometric complexity, these areas can easily dominate the global illumination computation. The authors attempted to directly attack this problem by decomposing the incoming irradiance into two parts and exploiting the properties of each part individually.

The incident hemisphere Ω is partitioned into Ω_D and Ω_N based on whether a distant or near point is visible in that direction. The distinction of near and far is based on a user-defined distance threshold. This also implies a partitioning of incoming radiance: L_{i_D}, incident radiance from distant surfaces, and L_{i_N}, incident radiance from nearby surfaces.

By construction, L_{i_D} changes very slowly across surfaces and is therefore an excellent candidate for interpolation. In contrast, L_{i_N} varies rapidly and is handled separately.

Arikan et al. cached radiance instead of irradiance to attack the over-sampling problem. The irradiance due to distant surfaces is computed using Monte Carlo integration, and the resulting radiance field is stored as nine SH coefficients. Instead of sampling the hemispherical radiance at specific cache points, Arikan et al. distributed the origin of the hemispherical sample rays over multiple surface points. This allowed them better coverage of the whole radiance field while using fewer cache points. The specifics of this procedure are explained in detail in the original paper [Arikan et al., 2005].

The incident irradiance due to nearby surfaces is calculated using analytic point-to-triangle form factors assuming perfect visibility. This assumption is not always true and is the source of most of the approximation error. Since this quantity can change rapidly it is computed per pixel.

These two irradiance values cannot be combined directly, since the geometry from the nearby irradiance might block the irradiance from distant surfaces. Therefore, when computing the analytic form factors, the distant irradiance is sampled in the direction of the triangle and is assumed to be constant over the whole triangle. These quantities are subtracted to account for the occlusion.

3.7.2 Distributed Global Illumination

The *Kilauea* renderer was designed to handle extreme geometric complexity within a production environment while producing images with high-quality global illumination [Kato, 2002]. Kilauea used photon mapping for global illumination in combination with an irradiance cache-like algorithm for efficient final gather. To satisfy the primary goal of high image quality, the renderer traces $65 \times 65 = 4,225$ rays per irradiance sample. This large amount of gather rays guarantees smooth and accurate results for all but the most grueling scenes, even during animation. Kilauea handles this ray tracing complexity by employing massively parallel rendering. However, as a distributed renderer based on message passing, global illumination within Kilauea has a different performance profile to address. Tracing rays within such a context incurs an even larger overhead than in a traditional ray tracer. Thus, a method which minimizes the total number of final gather rays is essential.

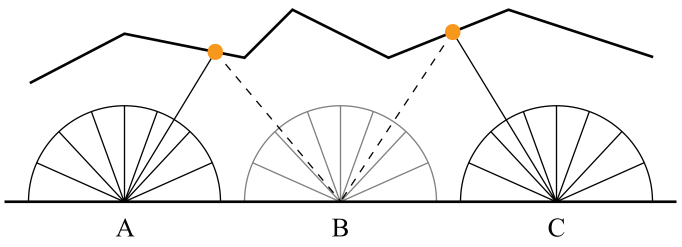

Figure 3.7: Hit point reprojection. We can estimate the radiance strata at location B by reprojecting the hit points from both A and C.

Hit Point Reprojection

The approach taken by Kato [2002] employed final gather reprojection to reduce the number of necessary rays. Instead of simply storing an irradiance value, the whole stratified radiance hemisphere is stored as a 2D grid of 65 × 65 colors and depths. Given this information from neighboring points, we can reproject each strata's hit points onto a different irradiance location, as in Figure 3.7. To minimize the amount of unresolved pixels, four neighboring cache points are used in the reprojection. Nonetheless, gaps could still occur. If a missing stratum is surrounded by valid strata, these values are just interpolated. All remaining gaps are computed by tracing rays. The authors found that this method can drastically reduce the number of necessary final gather rays.

The memory requirements for storing all the hemispherical sample information for each cache point might seem overwhelming, but the pixels in the rendered image are processed in small independent buckets. As a result, this information never needs to be stored for a whole image at once.

Adaptive Sample Placement

In addition to the reprojection, Kilauea employs a hybrid screen/object-space irradiance sample placement algorithm. The pixels in the rendered image are processed in small, independent, rectangular buckets. One bucket at a time, the algorithm first computes the hit location and normal for each pixel. By comparing the location and orientation of neighboring pixel hit points, "critical pixels" are identified. These critical pixels roughly correspond to silhouettes and creases in the scene and, for accuracy, require un-interpolated irradiance computations. For all remaining "safe pixels," a precise irradiance estimate is computed (using Equation 3.4 with hit point reprojection) sparsely over the image plane, typically on a grid of every 16×16 pixels. This grid imposes a tiling over which interpolation or subdivision can occur. If no critical pixels overlap a tile, then that tile is a candidate for interpolation. The harmonic mean distance (Equation 3.9) of the corner irradiance samples is evaluated. If the means are found to be large, the irradiance is predicted to vary smoothly, and, consequently, a simple bilinear interpolation of the corner irradiance samples is computed over the entire tile. If critical pixels do overlap the tile, or if the harmonic mean distances are too small, the tile is uniformly subdivided. This process continues down to a user-defined minimum tile size.

Using both adaptive sample placement and hit point reprojection, the authors were able to reduce the number of irradiance gather rays per pixel to the range of $3 - -25$ for typical scenes, in contrast to the $4,225$ samples that would be needed to compute a full 65×65 estimate at each pixel.

3.7.3 Animation

Temporal Aliasing and Coherence

In a naïve extension of irradiance caching to animation, a new irradiance cache would be computed per frame. This new cache would not only have different irradiance values for each cache point, but the location of the irradiance cache points would be different for every frame. Due to the stochastic sampling of the irradiance and the lack of inter-frame coherence of cache point locations, significant differences would arise between frames. The approximation errors due to interpolation and sparse sampling might be unnoticeable for an individual frame, but when these artifacts change drastically from frame to frame, the illusion is ruined. This phenomenon results in what appears as "popping" or "flickering" of the indirect illumination.

Age-based Invalidation. The observation made by Tawara et al. [2004] is that, for many regions of a scene, the irradiance changes slowly and minimally over time when objects are animated. It would be fruitful to exploit this feature, if only these regions could somehow be identified. Instead of explicitly identifying these regions, however, the authors decided to probabilistically re-evaluate only a small subset of all global illumination rays using a simple age-based heuristic.

In order to accommodate this, the irradiance caching algorithm is extended by storing radiance values instead of irradiance. The full stratified hemisphere of samples is stored, caching the radiance and the distance to the nearest hit point for each stratum. In addition, each stratum is assigned an age, which is initialized to 0. As the animation progresses, the age of each stratum is incremented. A random subset of the oldest radiance strata is selected to be retraced each frame.

Updating a subset of the strata has many advantages. Firstly, by choosing only 10% of all strata per frame, the render speed improves by approximately a factor of 10. Furthermore, in this example, each radiance strata can be expected to be updated, on average, every 10 frames. Moreover, reusing previous radiance cache locations reduces the temporal flickering problem described above.

The authors also suggested an improved heuristic which takes the visibility of dynamic objects into account. A flag is included to indicate whether each stratum intersected a static or dynamic object. When randomly updating strata, more weight is given to strata viewing dynamic objects. This adapts the process to update more vigorously in scene regions near moving objects.

Two issues remain: how do we handle irradiance cache points on dynamic objects, and how do we deal with the fact that the optimal density and distribution of cache point locations changes over time? To address the first issue, the authors simply attached the irradiance samples to dynamic objects. The cache point locations are transformed with their underlying object. This is a gross approximation since the radiance directions will no longer be valid once transformed, but the authors claimed that the artifacts are negligible for slowly moving objects. The second problem can be addressed by searching within the local neighborhood of each cache point and eliminating a point if the number of nearby neighbors is larger than a given threshold (such as 10). This technique showed good results in adapting the cache point distribution to the changing needs of the animated environment.

Separate Static and Dynamic Caches. An alternate approach to age-based invalidation was proposed by Tawara et al. [2002]. Here, the proposed technique separates the irradiance cache into a static and dynamic cache. The motivation behind this is much the same as with irradiance decomposition by Arikan et al. [2005]. In this case, however, we decompose the irradiance with respect to motion, and not distance, to exploit the properties of the static and dynamic irradiance caches independently.

The static cache contains only irradiance from non-moving objects and hence does not flicker. Static objects can be determined with respect to the whole animation sequence, or the animation can be divided into shorter segments and static objects identified within each time segment. Dynamic objects are handled separately. A distinct dynamic irradiance cache is created which contains contributions from moving objects.

These two caches cannot be directly combined because changing occlusion and inter-reflection from dynamic objects needs to be properly integrated into the final irradiance. The proposed algorithm uses irradiance caching in conjunction with photon mapping [Jensen, 2001]. Changes in occlusions from dynamic objects are handled by photons with negative energy in the same spirit as “shadow photons” [Jensen, 2001]. Utilizing information in this dynamic photon map, the two irradiance caches can be effectively combined.

Exploiting the Human Visual System

The sensitivity of the Human Visual System (HVS) changes with respect to both spatial contrast and temporal effects. For instance, human observers are most sensitive to low frequency spatial patterns and less so to high frequencies. Furthermore, people can discern moving objects with different levels of accuracy with respect to speed. Fast moving objects are more noticeable than static or slow moving objects, but very fast movement can interfere with the eye’s tracking ability. Peak temporal sensitivity lies somewhere in between. These spatiotemporal sensitivity disparities allow for an increased level of error tolerance when generating synthetic images.

Using these properties of the HVS, Yee [2000] improved the efficiency of the global illumination computation by computing a spatiotemporal error tolerance map, which determines how important each pixel is for an individual frame in an animation. This map, the Aleph map, is used as an oracle that guides perceptually-driven rendering of the global illumination solution.

The algorithm requires the computation of estimated frames for the animation. These frames can be generated using graphics hardware or by ray tracing with direct illumination only.

The frame estimates serve as input for constructing the spatiotemporal sensitivity map. Though the details of the algorithm are beyond the scope of this paper, the Aleph map computation incorporates the motion of each pixel, through motion estimation, as well as the varying intensity, color, and orientation content of the image. This results in a single grayscale image which identifies areas of increased attention based on the HVS.

The perceptual information is applied to the irradiance caching algorithm by modifying the error threshold “a” in Equation 3.9 based on the Aleph map value of the underlying pixel. Recall that the error threshold parameter determines the search radius for nearby cache locations. This parameter is increased for pixels with a larger error tolerance as governed by the Aleph map. This allows for performing fewer irradiance computations and more irradiance interpolations in areas of low interest.

Yee was able to achieve a 6x to 11x speedup over the base irradiance caching algorithm by using spatiotemporal information. It is important to note that this method also works for static scenes and, by using the spatial importance information alone, a speedup of around 2x was observed.

3.8 Limitations

Over the past 20 years, irradiance caching methods have proven to be a practical solution to the global illumination problem. The many extensions in the literature speak to the versatility of the approach. Irradiance caching and derived methods can be used to effectively simulate lighting in scenes containing diffuse as well as glossy materials, and the algorithm has been successfully applied to animation and the production of computer generated films. However, one underlying assumption limits the applicability of all of these extensions.

All of the previous work has assumed that light travels unobstructed between surfaces. None of the previous work has addressed the issue of volumetric scattering. In the next chapter we extend our theoretical model of light to include effects from scattering media. Light transport in scattering media is a potentially fruitful related problem, which could benefit from the same approach that has worked so well for indirect surface illumination. Though the high-level ideas could very well be applied within participating media, most of the underlying derivations in irradiance caching explicitly make assumptions which are only valid at surfaces. Therefore, in Chapter 5 we develop a completely new algorithm for caching illumination within participating media, which is inspired by irradiance caching techniques.

Unfortunately, scenes containing participating media also pose a problem for irradiance caching at surfaces since these scenes invalidate many underlying assumptions. Indirect lighting from participating media affects the irradiance gradient at surfaces, invalidating the gradient derivations presented in this chapter. In Chapter 6 we extend the irradiance gradient formulation to properly include effects from participating media.