The Direct Lighting Computation in Global Illumination Methods

Changyaw Wang

Thanks to my committee members, Randall Bramley, Dennis Gannon, Frank Prosser, and Robert T. Glassey. To my advisor, Peter Shirley, who initiated this research, contributed many key ideas, and proofread the whole document. Thanks to my fellow students, Paulo Maciel, Kenneth Chiu, Rajesh K. Sharma, Shankar Swamy, Eric Wernert, and Kurt Zimmerman. Thanks to the people in the graphics community Eric Chen, Holly Rushmeier, Gregory Ward, Pheng Heng, and Kelvin Sung. Thanks to my friends, Jim Kim and Chang-shu Liu. Finally, thanks to my wife, my son, and my families in Taiwan. To my wife, Minru Wang, who took care of everything; to my son, Eugene Wang, who gave me enumerous fun to help me continuing my work; to my families for their endless support.

Contents

List of Figures

List of Tables

Table of Notation

Q: Radiant energy.

Φ: Radiant flux.

M: Radiant exitance.

L: Radiance.

I: Radiant intensity.

E: irradiance.

ρ: reflectance.

g: Visibility. $g = 1$ if visible and $g = 0$ is not.

θ: Angle between the normal vector and the direction of an incident light.

θ': Angle between the normal vector and the direction of an emitted light.

ψ: Direction of an emitted/reflected radiance.

ψ': Direction of an incident radiance.

dA': Differential area on the luminaire.

x: Point of interest.

x': Point on a luminaire.

$\|x' - x\|$: Distance between x' and x.

Ω: Integration domain.

L_{direct}: Direct lighting.

$L_{indirect}$: Indirect lighting.

$p(x)$: Probability density function.

$p(x|y)$: Conditional probability density function.

ξ: Random number in $[0, 1]$.

$p_1(x)$: $\frac{1}{A'}$ where A' is the total area of a luminaire.

$p_2(x)$: $\frac{1}{A''}$ where A'' is the total visible area of a lumianire.

$p_3(x)$: $\frac{1}{\omega'}$ where ω' is the solid angle covered by a luminaire.

$p_4(x)$: $\frac{cos\theta}{\omega'}$.

p_{total}: Projected solid angle or $\int_\Omega cos\theta d\omega$.

D_N: L^∞-discrepancy.

T_N: L^2-discrepancy.

p.d.f.: Probability density function.

Solid Angle or ω: Projected area over a unit sphere.

BRDF or f_r: Bidirectional reflectance distribution function.

1 Introduction

Creating realistic looking images is the goal of image synthesis in computer graphics. This document addresses several important issues regarding image synthesis for complex scenes. It pays particular attention to the "direct lighting computation", where the brightness of an object that is due to light that comes directly from the source (without reflection) is calculated as in Figure 1.1.

Generating an image involves three major steps. The initial step, scene specification, defines geometry, material, lighting, texture, movement, camera, etc. This step sets up a complete scene in a virtual space. The second step, rendering, produces numerical values for physical quantities that a viewer can sense. This step usually is accomplished by simulating light transport within the scene. The final step, scaling, transforms the rendered image so

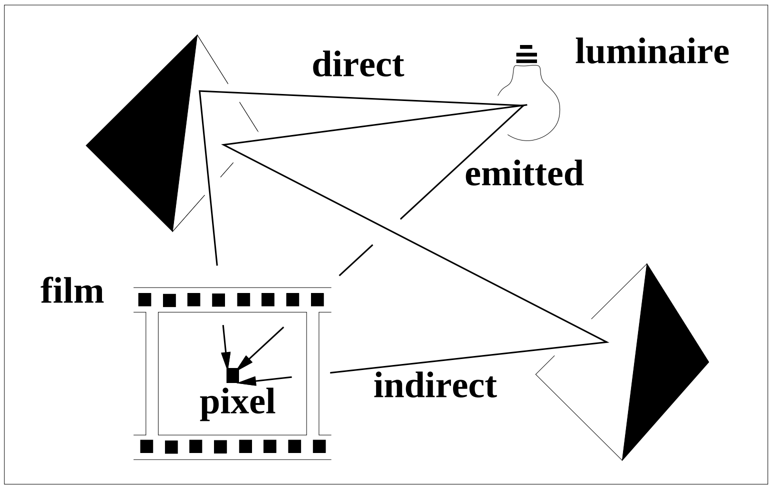


Figure 1.1: Direct lighting, indirect lighting and emitted lighting.

that it can be displayed on a device such as a cathode ray tube (CRT). This step creates an image on the CRT which gives an impression similar to the rendered image displayed on an ideal CRT with an infinite dynamic range[1].

This document focuses on the rendering process. Specifically it covers new sample point generation methods [8], new warping schemes that transform samples to various two dimensional luminaire domains [77], and a complete supersampling framework.

Given an accurate scene database, lighting is the key issue in realistic rendering. The earliest lighting simulation used local illumination [21], which only considers the luminaire[2] and the object whose color is being calculated. Although local illumination gives a more vivid shading variation than constant shading, it fails to mimic shadows which are an

[1]Dynamic range is the ratio between the maximum and minimum intensities.

[2]A *luminaire* is an object that products light. In this document, a luminaire doesn't include the surrounding fixtures.

extremely important visual cue in a realistic image [48].

Global illumination [84, 15, 25, 51], on the other hand, accounts for interreflection among objects. Not surprisingly, global illumination takes much more computation time than local illumination. One major speed up strategy for global illumination is to treat direct lighting and indirect lighting[3] differently [51, 36, 82, 60, 58]. In general, the direct lighting can cause rapid shading change and is simulated by ray tracing [84, 23], which starts from the eye or the camera film and reversely traces photons back to the luminaire as shown in Figure 1.2. The indirect lighting in a diffuse environment produces smooth shading variation due to multiple and recursive interactions among objects, and is commonly solved by zonal methods with a discretized environment [25, 63, 29, 6, 72] or ray tracing with cached indirect lighting [82] as discussed in Chapter 2. In a scene with specular or transparent objects, the situation gets complicated because indirect lighting can also produce rapid shading variation [31, 6]. It is also possible to solve direct and indirect lighting all at once [69].

In a natural scene, it is estimated that 80 million polygons are required to give a realistic look [74]. In real examples, Ward used over 3 million surfaces (mostly cones) to generate tens of pine tress over a field of snow [81], and Snyder and Barr used 2 billion objects to generate a scene with hundreds of trees over a grass field [21]. Due to such high scene complexities, we choose to handle direct lighting and indirect lighting separately in order to avoid the further tessellation required to capture the rapid radiance change caused by

[3]Direct lighting is the light from a luminaire to the observer after exactly one bounce off an object while indirect lighting is the light after more than one bounce.

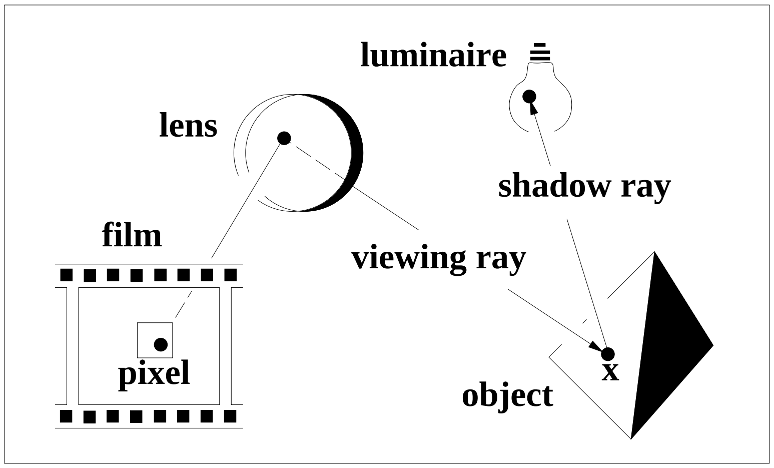


Figure 1.2: Ray Tracing with a Camera Model.

direct lighting. In our implementation, distribution ray tracing is used for the direct lighting computation and the geometric simplification method of Rushmeier et al. [58] is used for the indirect lighting computation. Rushmeier's method is a modified zonal method which stores the radiant intensity at a resolution lower than the resolution of the objects/surfaces. This reduces the memory requirements as well as the calculation time for energy exchange. This method can also prevent complete expansion of a procedural geometric object, where a function generates the object instead of storing the explicit geometry of the object.

In Chapter 2, we introduce some basic radiometric and photometric quantities and then give a mathematical formulation and explanation of the integrals we need to solve. Finally, we address several numerical solution methods for this integral.

In Chapter 3, we discuss Monte Carlo methods for the direct lighting computation. The key issue with Monte Carlo methods is the choice of a probability density function or p.d.f., which decides the distribution of the samples. We will apply four different p.d.f.s to various luminaire geometries. One of the newly proposed p.d.f.s can give the optimal sampling distribution in a diffuse environment if the whole luminaire is unblocked [77]. This method gives the analytical solution when the whole luminaire is visible. When the luminaire is blocked, it can generate stochastic samples according to the direct lighting contribution from an non-occluded luminaire. The other new p.d.f. has the samples evenly distributed within the solid angle[4] covered by the luminaire [76].

The direct lighting from more than one luminaire is the sum of the combined direct lighting. The straightforward summation gives a valid result [15, 14], but it requires consulting all luminaires, including those dim and far away. It can be time consuming when the number of luminaires is over a few tens. Culling methods [39, 78, 81], consult all luminaires, although they calculate contributions only from dominant luminaires. Unlike culling methods, the Monte Carlo methods in Chapter 4 will give an unbiased estimate and has a lower time complexity than current culling methods [65, 60, 61].

Sample point generation is a key issue in Monte Carlo methods, and is addressed in Chapter 5. Traditional Monte Carlo methods with random samples are inefficient because the samples are not evenly distributed[5] within the domain. Quasi-Monte Carlo methods with quasi-random samples are usually used [15, 14, 62, 60, 44]. Commonly used quasi-random samples include jittered samples, Poisson disk samples, and N-rooks samples. Two

[4]A two dimensional angle is the covered arc length of a unit circle whereas a three dimensional angle or solid angle is the covered area over a unit sphere.

[5]Ideal evenly distributed samples means the fraction of the samples covered in a subdomain equals the fraction of the domain size covered by the subdomain.

dimensional, jittered samples and Poisson disk samples are nearly equidistributed on the two dimensional domain whereas N-rooks samples are equidistributed in all one dimensional sub-domains. However it is currently difficult to predict the behavior of quasi-random samples. Although mathematicians have studied quasi-random samples and related them to equidistribution and its measure [1, 28, 71, 83, 50, 4] the application of these mathematical results in computer graphics is still limited. In this chapter, we also generalize the idea of equidistribution and propose a new multi-jittered sampling method [7], whose samples are equidistributed in any sub-domain. In a two dimensional square domain, the multi-jittered sampling method superimposes the jittered cells and N-rooks cells, and generate the samples that are jittered as well as N-rooks in order to have equidistribution in two dimensions and in either of the single dimensions. The complete seven-dimensional integral, which computes the pixel color, will be revisited in Chapter 5 to show the importance of equidistribution in dimensions higher than two and the motivation for multi-jittered sampling methods. The number of samples used to estimate the integral is another major efficiency consideration. Although adaptive supersampling, which uses more than one sample within a pixel for antialiasing, has often been proposed, the prediction of the number of samples needed is still a challenging topic. Methods based on criteria from statistics [43], signal processing [18], or perceptual models [44] have been suggested, but the justification of the criteria involves the not fully understood relationship between the estimation error of the pixel color, the display algorithm, and the human perceptual system. Without incorporating this missing information, all the proposed methods are limited in one way or the other. Instead we suggest a more complete supersampling method architecture based on the human perceptual system.

The new results in the document include:

- Two new sampling strategies for direct lighting from single luminaire (Chapter 3).
- A new sampling strategy with spatial subdivision for direct lighting from multiple luminaires (Section 4.4).
- A new multi-jittered sampling scheme (Section 5.2.3 and Section 5.2.5).
- A new numerical method to determine the number of initial samples (Section 5.3.2).
- A new adaptive supersampling framework (Section 5.3).

2

Global Illumination Problem Formulation and Solution Methods

In global illumination we compute the spectral radiance of each pixel, which can be represented as an integral in eight dimensions, $Wavelength \times Pixel \times Lens \times Time \times Luminaire \times Direction$. In some parts of the solution, the $Direction$ is replaced by the set of all surfaces points. Both sets are two dimensional ((θ, ϕ) and (u, v)).

This integral is too complicated to solve analytically, and is difficult to approximate numerically because of its high dimension. Quasi-Monte Carlo integration methods, which are designed for high dimension problems, are suitable for estimating the pixel color.

In this chapter, we give an intuitive explanation and show the exact formula of the energy/light transportation equation, which is then expanded into an integral for the pixel color and an integral for direct lighting. In the second half of this chapter we discuss the numerical solutions for global illumination, with a special focus on Monte Carlo methods.

2.1 Rendering Equation

Radiance is a quantity that is closely related to the color perceived by the human visual system. It is defined as the radiant power transferred per unit area per unit projected solid angle. The definitions of radiance, power, solid angle, many other common radiometric quantities and the corresponding photometric quantities are given in Appendix A.

One way to characterize energy transport is to relate the incident radiance and the outgoing radiance change at a point x on a surface, i.e. $dL_{out} \propto L_{in}$, as in radiative heat transfer [68]. This relation also involves the bidirectional reflectance-distribution function f_r to account for the material property and a cosine term to account for the projected area of the incident light beam, e.g. the projected area is minimal if the light is perpendicular to the surface, and increases as the light goes down to the horizon [21]. The complete form is $dL_{out_r} = f_r(in, out_r)\cos(N, in)L_{in}$, where N is the normal vector at x. Note that the transmission is omitted here for notational convenience, but it can be mechanically incorporated into the formula because it is simply $dL_{out_t} = f_t(in, out_t)\cos(N, in)L_{in}$.

The rendering equation which expresses the outgoing radiance in terms of all incident

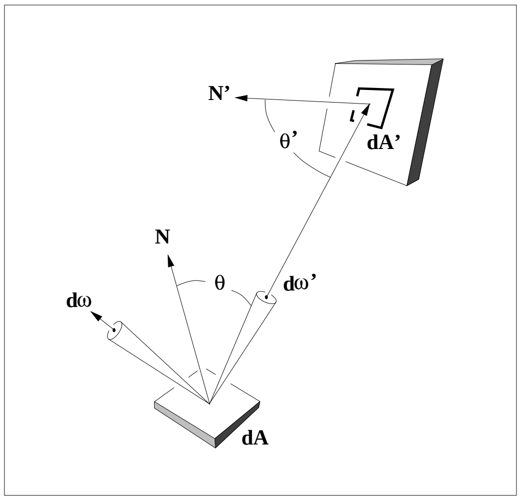


Figure 2.1: Rendering Equation

radiance is commonly represented in one of two different forms. The first one shows that the outgoing radiance toward $d\omega$ is the sum of the emitted radiance and the reflected radiance coming from every possible direction ψ' as in Figure 2.1, and in Equation 2.1 (from Immel) [34]:

$$L_\lambda(x,\psi) = E_\lambda(x,\psi) + \int_{\forall\psi' \in \Omega_{\psi'}} f_{r\lambda}(x,\psi,\psi') L_\lambda(x,\psi') \cos\theta d\omega_{\psi'} \tag{2.1}$$

where $E(x,\psi)$ is the emitted radiance in direction ψ at x, $L(x,\psi')$ the radiance from direction ψ' incident at x, $d\omega'$ is the differential solid angle subtended by ψ'.

The second form of the rendering equation shows that the outgoing radiance is the sum of the emitted radiance and the radiance from all visible surfaces, which is an integral over all surfaces (as used by Kajiya [36]):

$$L_\lambda(x,\psi) = E_\lambda(x,\psi) + \int_{\forall x' \in \Omega_{x'}} g(x,x') f_{r\lambda}(x,\psi,\psi') L_\lambda(x,\psi') \cos\theta \frac{dA' \cos\theta'}{\|x'-x\|^2} \tag{2.2}$$

where $g(x,x')$ is the *geometry term*, which is zero if there is an obstruction between x and x', and one otherwise.

The rendering equation represents only geometric optics and does not consider the light's properties in terms of the electromagnetic waves [30, 49]. This simplification ignores the effects of polarization, interference, and diffraction; but this is a minor limitation for most applications since these phenomena are not very noticeable in most scenes.

The rendering equations in Equation 2.1 and Equation 2.2 show the radiance toward one direction ψ only. To represent the pixel color, we need to integrate all the radiance toward the pixel of interest, or sometimes also the nearby pixels, within a time interval, and over the visible spectrum. The complete integral becomes :

$$L(i,j) = \int_{Wavelength} \int_{Pixels} \varpi(A_p) \int_{Lens} \int_{Time} L_\lambda(x,\psi)\ dt\ dA_l\ dA_p\ d\lambda \tag{2.3}$$

where (i,j) is the index of the pixel of interest. $\varpi(A_p)$ is the weight function (filter) on the pixel space and $Pixels$ is the support (nonzero domain) of $\varpi(A_p)$. $Lens$ is the area of the lens and $Time$ is the time interval over which the camera shutter is open. For ease of discussion, we assume a box filter of unit width, i.e. $\varpi(A_p) = 1$, thus we use $Pixel$

instead of $Pixels$. For animation, a non-uniform weight function would be used to avoid the regualrity in error, i.e. aliasing.

For efficiency, $Wavelength$ is generally treated using traditional quadrature and $L(i,j)$ becomes:

$$L_\lambda(i,j) = \int_{Pixel} \int_{Lens} \int_{Time} L_\lambda(x,\psi)\ dt\ dA_l\ dA_p \tag{2.4}$$

This modification means that a ray which carries a spectrum does not spatially diverge even when the ray enters an object such as a prism. This is not physically correct because the color dispersion[1] [30, 49] is not simulated, but this is not a severe limitation for most applications. If dispersion effects are needed, the integral should be retained in the form of Equation 2.3.

2.1.1 Direct Lighting

The rendering equation can also be cast as multiple terms, with each term representing a certain number of reflections between the emission and the exposure:

$$L = E^{[0]} + L^{[1]}_{direct} + L^{[2,\cdots,\infty]}_{indirect}$$

where the superscript means the number of bounces contained in the corresponding term.

[1]Dispersion is the separation of light into its constituent wavelengths, by refraction or diffraction with formation of a spectrum.

To derive L_{direct} and $L_{indirect}$, we expand the recursive rendering equations in Equation 2.1 and Equation 2.2. If we replace $L_\lambda(x,\psi')$ at the right hand side of Equation 2.1 by the same equation with $L_\lambda(x,\psi')$ at the left hand side, we have,

$$L_{direct} = \int_{\forall\psi' \in \Omega_{\psi'}} f_{r\lambda}(x,\psi,\psi') E_\lambda(x,\psi') \cos\theta d\omega_{\psi'} \tag{2.5}$$

else, when L_{direct} is derived from Equation 2.2, we have,

$$L_{direct} = \int_{\forall x' \in \Omega_{x'}} g(x,x') f_{r\lambda}(x,\psi,\psi') E_\lambda(x,\psi') \cos\theta \frac{dA' \cos\theta'}{\|x' - x\|^2} \tag{2.6}$$

Since non-emitting surfaces do not contribute to L_{direct}, the integration domain $\Omega_{\psi'}$ can be simplified to the directions toward luminaires only and $\Omega_{x'}$ to the surfaces of luminaires.

L_{direct} is not a recursive function. It contains the details of the scene. On the other hand, $L_{indirect}$ is a recursive function with a few details (an exception is caustics caused by focusing effects). The separation of L_{direct} and $L_{indirect}$ gives the opportunity to solve these two integrals with different numerical methods and different precisions. Chapter 3 is devoted to the direct lighting computation from one luminaire and the whole of Chapter 4 is devoted to the direct lighting calculation from multiple luminaires.

2.1.2 Indirect Lighting

Indirect lighting is commonly solved by zonal methods (finite element methods). In these methods objects are tessellated into elements or patches. Each element of the tessellation is assumed small enough that it can be characterized by one bidirectional reflectance-distribution function $f_r(x, \psi, \psi')$ [63, 6, 29, 69, 72]. Then energy is transported among elements and patches. In the *shooting approach*, a patch distributes its energy to the visible elements while in the *gathering approach*, an element collects its energy from all the other patches and elements. Common zonal methods require storage of the energy information on each element. This is overkill in a complex environment because most objects have little contribution to the image and do not need high accuracy.

In our implementation, the indirect lighting is computed by geometry simplification [58], where the energy is distributed over a coarsely tessellated environment. After the reflected energy of every element is calculated in the zonal phase, we treat the tessellated environment as luminaires with zero reflectivity. Then in the viewing phase the reflected ray is used to compute the indirect lighting in a manner similar to collecting direct lighting. If the reflected ray hits an object very close by, the ray is further reflected to avoid error amplification due to the large solid angle covered by the nearby object.

Indirect lighting can also be solved by ray tracing as is done by Ward [82], where the radiance values at points in three dimensional space is stored instead of on objects. Existing information is reused if the radiance at a nearby point with similar normal vector is already stored. Storing the radiance at points in space can avoid the length computation time in path tracing, and the meshing problems of the zonal methods. The only major concern in this approach is the density of the samples. In a complex environment, coherence exists only within a very small volume and hence we need many samples. On the other hand, since the number of viewing rays increases, the importance of each ray is decreased, and this in turn increases the tolerance for coarser estimation of indirect lighting.

Although having the full solution for indirect lighting is commonly thought of as the ultimate goal, there are reasons to believe that full solution might be an overkill. One reason is that a fast but coarse approximation can produce an image without a significant degradation of quality [58]. The other reasons relate to the non-linear transformation of a real-world image to a screen image which only allows a small intensity range [73, 7]. The non-linear transformation of Tumblin and Rushmeier suggests that the full intensity range of the screen image should be a nonlinear function to the actual intensity range of the real-world image, i.e. a real-world image is mapped only to a portion of the intensity range on the screen. The benefit of this method is the ability to distinguish different lighting conditions, for example two rooms with the same geometry setting but one with a 60 *watt* lamp and the other with a 200 *watt* lamp. However, this transformation further limits the intensity resolution. The other non-linear transformation by Chiu et al. [7] suggests that a real-world radiance can be mapped to a range of screen intensities instead of a single intensity as long as the relative contrast is maintained. Both transformations decrease the accuracy of the real-world image and can make the smooth transition of radiance from

$L_{indirect}$ even less noticeable.

If the full solution for indirect lighting is not necessary, the ambient light[2] might be sufficient to give a reasonable approximation [10, 60]. To make the ambient light a good approximation of indirect lighting, we still need to consult the lighting condition and the environment. For example in an empty room with a 100 *watt* luminaire and reflectance 0.5 on the walls, a good guess of the ambient light can be made from the knowledge that the total energy is cut in half after each bounce, i.e. the ambient light $\approx (0.5 + 0.5^2 + 0.5^3 + \cdots) \times 100$ $watts = 100\ watts$. Images using ambient light usually look somewhat flat, so there is some indication that the indirect lighting should be computed, but not with an extreme accuracy.

2.2 Numerical Integration over A Finite Interval

L_{direct} and $L(i, j)$ can be approximated by many numerical methods [5, 17], which fall into four categories: traditional quadrature, Monte Carlo methods [37, 9, 57], weighted Monte Carlo methods [24], and quasi-Monte Carlo methods [71, 62, 19]. This classification emphasizes various Monte Carlo methods because their important properties which have made them a practical choice for computer graphics:

[2]Ambient light is a simple model which produces constant illumination on all surfaces regardless of their position or orientation.

1. Unlike some traditional quadrature methods, the number of samples is not proportional to 2^d for d-dimensional problems.
2. Their stochastic nature makes the estimation error appear as noise, which is more visually acceptable than aliasing artifacts created by deterministic process [15, 14].
3. The simplicity of the solution makes implementation fairly painless.

2.2.1 Traditional Quadrature Formulas

The basic idea of numerical quadrature, which approximates a definite integral by a weighted sum of finite samples, can be expressed in the following equation:

$$\int_{\Omega} f(x)dx = \sum_{i=1}^{n} \int_{\Omega_i} f(x)dx \approx \sum_{i=1}^{n} \omega_i f(x_i)$$

This formula uses ω_i to estimate $\int_{\Omega_i} dx$ and $f(x_i)$ to estimate $f(x)$ within Ω_i. In order to have a good approximation, $f(x_i)$ should be representative of $f(x)$ in Ω_i and this is true if $f(x)$ is smooth in ω_i, which is generally true when Ω_i is compact and small.

There are three common methods used to generate the ω_i and x_i [20]. Gaussian quadrature uses deterministic ω_i and x_i. Monte Carlo methods and quasi-Monte Carlo methods start with a partition of the domain and the weight of each cell, and then a sample is assigned in each cell. Weighted Monte Carlo methods assign samples before constructing the partition and computing the weights.

2.2.2 Monte Carlo Methods

Monte Carlo methods represent an integral as a statistical expected value, which is then approximated by an average [33, 57, 38, 89]:

$$\begin{aligned}\int_{\forall x \in S} f(x) d\mu(x) &= \int_{\forall x \in S} \frac{f(x)}{p(x)} p(x) d\mu(x) \\ &= E[\frac{f(X)}{p(X)}] \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \end{aligned} \tag{2.7}$$

where X is a random variable having a probability density function $p(x)$ (denoted $X \sim p$), i.e. $p(x) > 0$ when $f(x) \neq 0$, and $\int_{\forall x \in S} p(x) d\mu(x) = 1$. $d\mu(x)$ is the differential measure of point x, and x_i is a sample chosen according to $p(x)$. $\frac{f(x_i)}{p(x_i)}$ is called the primary estimate, the average of primary estimates is the secondary estimate.

Monte Carlo methods involve two steps [57]. The first step is the choice of a valid $p(x)$. In general, the more similar $p(x)$ is to $f(x)$, the smaller the error is [60]. If $p(x) \propto f(x)$, then $\int_{\forall x \in S} f(x) d\mu(x) = \frac{f(x_i)}{p(x_i)}$ for any x_i such that $f(x_i) > 0$. However, the rendering equations in Equation 2.1 and Equation 2.2, and Equation 2.4 are so complicated that $p(x) \propto f(x)$ is still impossible. The second step is to find x_i, which is distributed according to $p(x)$. In one dimension, suppose that $P(x)$ is the probability distribution function of $p(x)$, i.e. $P(x) = \int_a^x p(x) dx$, then $x_i = P^{-1}(\xi)$ where ξ is a random number in $[0, 1]$. When the analytical form of $P^{-1}(x)$ does not exist, numerical inversion should be used. If $p(x)$ is non-uniform, then the sampling strategy of finding x_i is called *importance sampling*.

The main drawback of Monte Carlo methods is slow convergence. The variance of the estimate is $var[\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}]$ or $\frac{1}{N}var[\frac{f(x_i)}{p(x_i)}]$. Since the estimation error behaves similar to the square root of the variance, we need to quadruple the samples in order to halve the error.

2.2.3 Weighted Monte Carlo Methods

Weighted Monte Carlo methods first select samples at random locations, and then construct a partition which is used to decide the weights for the samples. A special advantage of the weighted Monte Carlo methods is that samples can be placed where they are most needed. This requires some knowledge of the function which is possible in many, but not all, instances.

In a two dimensional domain, Glassner used the Voronoi diagram as the partition and the area covered by each cell as the weight [24]. The Voronoi cell is a good choice because it is compact and has the sample near the center of each cell. The process of generating a Voronoi diagram is illustrated in Figure 2.2.

Weighted Monte Carlo methods become less attractive as the dimension goes up because of the dimensional effect, where the number of partition cells increases at the same order as the dimension of the domain [17].

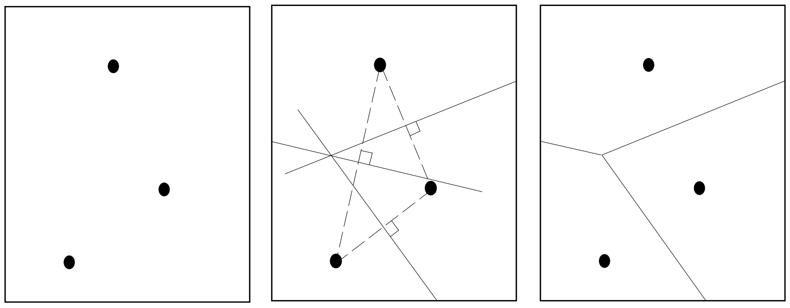


Figure 2.2: Voronoi Diagram Construction. Left: given samples. Middle: drawing the perpendicular line at the middle of the line segment connecting every pair of samples. Right: defining each Voronoi cell by the perpendicular lines related to the sample.

2.2.4 Quasi-Monte Carlo Methods

Quasi-Monte Carlo methods are Monte Carlo methods with quasi-random samples, which have some, but not all, properties of random samples [62, 44, 47, 45].

Number theory has been used to study quasi-Monte Carlo methods for many years, and has yielded some interesting results. In Chapter 5, we will discuss these results, and relate them to computer graphics.

2.3 Summary

Global illumination can be mapped to an integration problem, which is extremely difficult to compute because of recursion, high dimension, and the continuity and the shape of the integration domain. Quasi-Monte Carlo methods are attractive for approximating this integral. The sampling issue is the major concern in quasi-Monte Carlo methods and knowledge about sampling from both number theory and computer graphics may be useful in speeding up the calculation of this approximation.

3

Direct Lighting from One Luminaire

In this chapter we present four sampling methods which can be used with quasi-Monte Carlo methods to compute direct lighting, L_{direct} in Equation 2.5 or Equation 2.6 [77].

It is possible to apply methods other than Monte Carlo methods to estimate L_{direct}. One possible method is to simply combine L_{direct} with $L_{indirect}$ and solve everything via naive zonal methods [25, 75]. However, this approach requires an extremely fine mesh size to capture rapid radiance changes caused by shadowing, projected solid angle of the luminaire, the intensity distribution of the luminaire, and reflectivity. The second method discretizes the luminaire into point luminaires at specific positions [42], but unfortunately the regularity of luminaire positions might cause aliasing artifacts. The third method first calculates the visible boundary and then uses the exact formula of L_{direct} from a perfectly diffuse polygonal luminaire [51]. The limitation of this method is that only polygonal or polyhedral objects are allowed in the environment. The fourth method is to divide the (rectangular) luminaire into quadtree cells of various sizes, and estimate L_{direct} from each piece of the luminaire by Monte Carlo methods [81]. The problem with this approach is that the artificial division may result in more samples than actually needed.

The quasi-Monte Carlo methods we present in this chapter do not have the restrictions and limitations of the other methods, do not subdivide the luminaire, and most importantly they are extremely effective for computing direct lighting on diffuse objects. If an object is perfectly specular like a mirror, one reflected ray is all we need to compute the direct lighting and indirect lighting, and calculating L_{direct} independently is unnecessary. If a surface is not perfectly specular, it is unclear which calculation method converges more rapidly [66].

3.1 Luminaire Sampling Strategies for Diffuse Environment

The full integral of L_{direct} can be written in two forms. First, expanding the Equation 2.5 using spherical coordinates :

$$L_{direct} = \int_{\forall\hat{\theta}\in\Omega_{\hat{\theta}}} \int_{\forall\hat{\phi}\in\Omega_{\hat{\phi}}} f_{r\lambda}(x, \psi, \psi') E_\lambda(x', -\psi') \cos\theta d\omega_{\psi'} \sin\hat{\theta} d\hat{\phi} d\hat{\theta} \tag{3.1}$$

where $\hat{\theta}$ is the angle between Z axis and ψ'. $\hat{\phi}$ is the angle from the XZ plane to ψ' in a counterclockwise direction. Second, L_{direct} can be derived from Equation 2.6 :

$$L_{direct} = \int_{\forall v \in \Omega_v} \int_{\forall u \in \Omega_u} f_{r\lambda}(x, \psi, \psi') E_{\lambda}(x', -\psi') \cos\theta d\omega_{\psi'} \frac{\cos\theta_i'}{\|x_i' - x\|^2} du dv \tag{3.2}$$

The application of the Monte Carlo methods presented in Section C to L_{direct} is straightforward when the integration domain can be analytically defined. This Chapter applies four different probability density functions, that we consider to be of increasing desirability in a diffuse environment, to luminaires with a simple geometry. Each p.d.f. will provide a valid estimate for any BRDF and will provide a low or zero variance estimate in a diffuse environment. L_{direct} can be computed using either Equation 3.1 or Equation 3.2, but often only one of them gives a natural coordinate system or parameterized integration boundary to work with. For example, while sampling a polygon, it is usually easier to perform calculations on the surface of the polygon, but while sampling a sphere spherical coordinates is a more convenient calculation space.

In this chapter, x' means a point and ψ' means a direction.

Case 1: $p_1(x') = \dfrac{1}{A'}$ where A' is the total area of the luminaire.

$p_1(x') = \dfrac{1}{A'}$ is easy to use and can give a quick estimate. This p.d.f. is particular useful for debugging, but it requires many samples to render realistic image because of its potentially high variance. The primary estimate is $L(x, \psi) \approx g(x, x') f_r(x, \psi, \psi') E(x', \psi') \cos\theta A'$

$\dfrac{\cos\theta'}{\|x'-x\|^2}$.

Case 2: $p_2(x') = \dfrac{1}{A''}$ where A'' is the visible area on the luminaire when seen from x.

Due to the difficulty of parameterizing the intersection of the luminaire and the tangent plane at x, A'' reduces to the region not blocked by the luminaire itself. The only situation where case 1 and case 2 can be different is when the luminaire has curved surfaces.

This case can be much more efficient than case 1 which wastes some samples in the invisible region. In the situation of a spherical luminaire or a cylindrical luminaire, case 1 will produce points that are invisible more than half the time while case 2 will always produce visible points.

The primary estimate is $L(x,\psi) \approx g(x,x')f_r(x,\psi,\psi')E(x',\psi')\cos\theta\dfrac{\cos\theta'}{\|x'-x\|^2}A''$.

Case 3: $p_3(\psi') = \dfrac{1}{\omega'}$ where ω' is the solid angle covered by the luminaire seen from x'.

The primary estimate is $L(x,\psi) \approx f_r(x,\psi,\psi')L(x',\psi')\cos\theta\,\omega'$. Since $d\omega' = \dfrac{\cos\theta' dA'}{\|x'-x\|^2}$, $p_3(\psi') = \dfrac{1}{\omega'}$ is equivalent to $p_3(x') = \dfrac{\cos\theta'}{\omega'\|x'-x\|^2}$. So the primary estimate can also be $L(x,\psi) \approx g(x,x')f_r(x,\psi,\psi')L(x',\psi')\cos\theta\displaystyle\int_{\forall x'\in\Omega_{x'}}\dfrac{\cos\theta'}{\|x'-x\|^2}dA'$.

Deciding between the use of $p_3(\psi')$ or $p_3(x')$ depends on which domain can define the solid angle more easily. Naturally $p_3(\psi')$ is good for spherical luminaires, whereas $p_3(x')$ is good for planar luminaires.

This case improves case 2 by putting more samples at the region with larger solid angle. The difference between case 3 and case 2 is most obvious if x is close to the luminaire.

Case 4: $p_4(\psi') \propto \cos\theta$ when the whole luminaire is above the tangent plane at x'.

If any part of the luminaire is below the tangent plane at x', then $p_4(\psi')$ is not a valid p.d.f. since $\cos\theta$ is negative. To avoid invalid $p_4(\psi')$, in these situations we can switch to the methods from case 1, case 2, or case 3.

The primary estimate is $L(x,\psi) \approx f_r(x,\psi,\psi')E(x',\psi') \int_{\forall\psi'\in\Omega_{\psi'}} \cos\theta d\omega'$ and again $p_4(\psi') \propto \cos\theta$ is equivalent to $p_4(x') \propto \dfrac{\cos\theta\cos\theta'}{\|x'-x\|^2}$ or

$$p_4(x') = \frac{\frac{\cos\theta\cos\theta'}{\|x'-x\|^2}}{\int_{\forall x'\in\Omega_{x'}} \cos\theta \frac{\cos\theta'}{\|x'-x\|^2} dA'} \tag{3.3}$$

and the primary estimate of $L(x,\psi)$ can also be $L(x,\psi) \approx g(x,x')f_r(x,\psi,\psi')E(x',\psi')$ $\displaystyle\int_{\forall x'\in\Omega_{x'}} \cos\theta \frac{\cos\theta'}{\|x'-x\|^2} dA'$.

Case 4 is an improvement over case 3 in a diffuse environment because $\cos\theta$ can characterize the integrand in L_{direct} very well. Case 4 can also be worse than any other case in an environment with non-diffuse objects or luminaires if p_4 happens to have higher values in the unimportant regions.

To use Monte Carlo methods, we need to know either $p(\dot{\psi})$ and the sampled direction $\dot{\psi}$, or $p(\dot{x})$ and the sampled position $\dot{x}$ where $\dot{x}$ is the intersection of the luminaire with $\dot{\psi}$

Luminaire	$p_1(x') = 1/A'$	$p_2(x') = 1/A''$	$p_3(\psi') = 1/\omega'$	$p_4(\psi') \propto \cos\theta$
Sphere	Yes	Yes	Yes	Num.
Rectangle	Yes	N.A.	Num.	Num.
Triangle	Yes	N.A.	Approx.	Num.
Polygon	Yes	N.A.	Approx.	Num.
Disk	Yes	N.A.	Approx.	Approx.
Cylinder	Yes	Yes	Approx.	Approx.

Table 3.1: Results of applying four sampling methods in a single luminaire environment. N.A. means doesn't exist, Num. means numeric function inversion is needed in the solution, and Approx. means an approximation is presented.

and $p(\dot{x}) = p(\dot{\psi})\dfrac{\cos\theta'}{\|x - \dot{x}\|^2}$. Finding such a p.d.f. is a geometric, or an algebraic problem that varies in complexity for different sampling methods and luminaire geometries. We have tried spheres, rectangles, triangles, polygons, disks, cylinders, and the results are shown in Table 3.1 after applying the four p.d.f.'s to various luminaires. The specifics of calculating these p.d.f.'s is dealt with in Section 3.1.1 through Section 3.1.6.

The coordinate systems used in the following sections are chosen specifically to simplify the derivation. This does not effect the generality of these methods, but implies some computational expense for coordinate transformation.

3.1.1 Spherical Luminaires

Suppose the luminaire with radius r is centered at the origin and x is at $(0, 0, d)$.

Case 1 : $p_1(x') = 1/A' = 1/(4\pi r^2)$, choose $\dot{x} = (r, \theta, \phi)$:

$$\dot{x} = (r, \arccos(1 - 2\xi_1), 2\pi\xi_2)$$

in the local spherical coordinate system centered at $(0, 0, d)$.

Case 2 : $p_2(x') = \dfrac{1}{A''} = \dfrac{1}{2\pi r^2(1 - \frac{r}{d})}$.

$$\dot{x} = (r, \arccos(1 - \xi_1 + \xi_1 \frac{r}{d}), 2\pi\xi_2)$$

where $\dot{x}$ is in spherical coordinates.

The next two cases use the coordinate system UVW as in Fig. 3.1. W is from x to the center of the luminaire. V is $W \times N$ and U is $V \times W$. x is at the origin. The luminaire centers at $(0, 0, d)$ and has radius r. $N = (N_u, 0, N_w)$ is a constant vector. If $N = W = (0, 0, 1)$, V can be any vector perpendicular to W.

Case 3 : $p_3(\psi') = \dfrac{1}{\omega'} = \dfrac{1}{2\pi(1 - \frac{\sqrt{d^2-r^2}}{d})}$.

$\dot{\psi}$ is a unit vector from the origin to $(1, \dot{\theta}, \dot{\phi})$ in spherical coordinates. $\dot{\theta} = \arccos(1 - \xi_1 + \xi_1 \frac{\sqrt{d^2-r^2}}{d})$ and $\dot{\phi} = 2\pi\xi_2$ as derived in [76].

Case 4 : $p_4(\psi') = \dfrac{\cos\theta}{p_{total}}$, where

$$p_{total} = \int_{\forall\psi' \in \Omega_{\psi'}} \cos\theta d\omega'$$

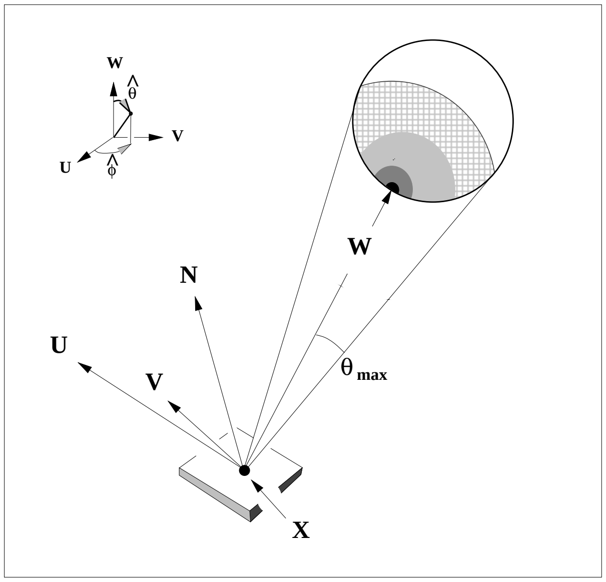


Figure 3.1: Spherical Luminaire

$$
\begin{aligned}
&= \int_0^{\theta_{max}} \int_0^{2\pi} \cos\theta \sin\hat{\theta} d\hat{\phi} d\hat{\theta} \\
&= \int_0^{\theta_{max}} \int_0^{2\pi} N \cdot (\sin\hat{\theta}\cos\hat{\phi}, \sin\hat{\theta}\sin\hat{\phi}, \cos\hat{\theta}) \sin\hat{\theta} d\hat{\phi} d\hat{\theta} \\
&= \int_0^{\theta_{max}} 2\pi N_w \sin\hat{\theta}\cos\hat{\theta} d\hat{\theta} \\
&= \pi N_w \sin^2\theta_{max}
\end{aligned}
$$

where $\theta_{max} = \arcsin\dfrac{r}{d}$. The angle $\dot{\psi} = (1, \dot{\theta}, \dot{\phi})$ can be computed by the numerical inversion from the following two formulas:

$$
\begin{aligned}
\xi_1 &= \int_0^{\dot{\theta}} p(\hat{\theta}) d\hat{\theta} \\
&= \int_0^{\dot{\theta}} \int_0^{2\pi} p(\hat{\theta}, \hat{\phi}) d\hat{\phi} d\hat{\theta}
\end{aligned}
$$

$$\begin{aligned}
&= \int_0^{\dot\theta}\int_0^{2\pi} p(\psi')\sin\hat\theta d\hat\phi d\hat\theta \\
&= \frac{1}{p_{total}}\int_0^{\dot\theta}\int_0^{2\pi}\cos\theta\sin\hat\theta d\hat\phi d\hat\theta \\
&= \frac{1}{p_{total}}\int_0^{\dot\theta} 2\pi N_w\cos\hat\theta\sin\hat\theta d\hat\theta \\
&= \frac{1}{p_{total}}\pi N_w\sin^2\dot\theta
\end{aligned}$$

So, $\dot\theta = \arcsin\sqrt{\frac{\xi_1 p_{total}}{\pi N w}} = \arcsin\sqrt{\xi_1}\sin\theta_{max}$.

$$\begin{aligned}
\xi_2 &= \int_0^{\dot\phi} p(\hat\phi|\dot\theta)d\hat\phi \\
&= \frac{\int_0^{\dot\phi} p(\dot\theta,\hat\phi)d\hat\phi}{p(\dot\theta)} \\
&= \frac{\int_0^{\dot\phi} p(\dot\theta,\hat\phi)d\hat\phi}{\int_0^{2\pi} p(\dot\theta,\hat\phi)d\hat\phi} \\
&= \frac{\int_0^{\dot\phi} N\cdot(\sin\dot\theta\cos\hat\phi,\sin\dot\theta\sin\hat\phi,\cos\dot\theta)d\hat\phi}{\int_0^{2\pi} N\cdot(\sin\dot\theta\cos\hat\phi,\sin\dot\theta\sin\hat\phi,\cos\dot\theta)d\hat\phi} \\
&= \frac{\int_0^{\dot\phi} N_u\sin\dot\theta\cos\hat\phi + N_w\cos\dot\theta d\hat\phi}{\int_0^{2\pi} N_u\sin\dot\theta\cos\hat\phi + N_w\cos\dot\theta d\hat\phi} \\
&= \frac{N_u\sin\dot\theta\sin\dot\phi + N_w\dot\phi\cos\dot\theta}{2\pi N_w\cos\dot\theta}
\end{aligned}$$

We need the numerical inversion to find out $\dot\phi$.

Figure 3.2 has one viewing ray for each of the 200^2 pixels. The average execution time and the fraction of the time taken compared to case 1 from 10 trials are in Table 3.1.1.

For the same test scene, a visual comparison shows that case 1 needs 81 jittered samples to achieve shading as smooth as case 4, case 2 needs 36 jittered samples, and case 3 needs 9

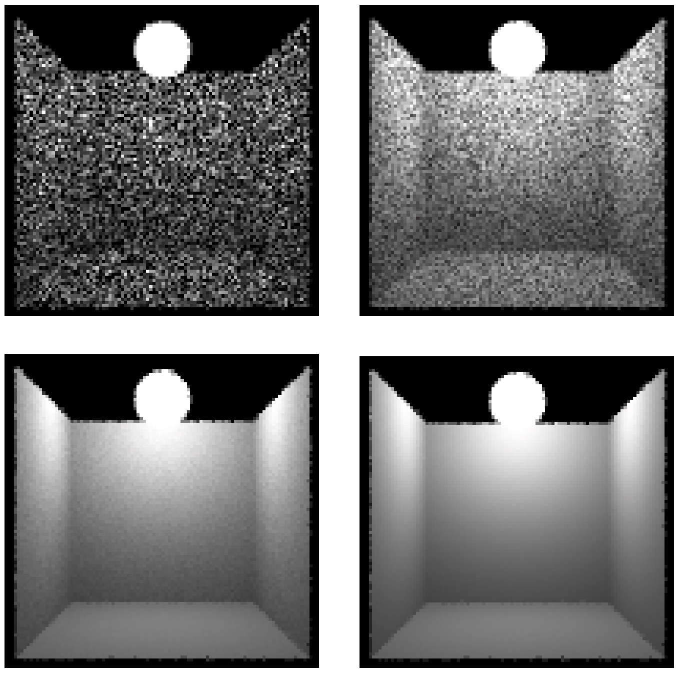


Figure 3.2: Sampling a spherical luminaire with 1 sample per pixel. Top left: $p_1(x') = \frac{1}{A'}$. Top right: $p_2(x') = \frac{1}{A''}$. Bottom left: $p_3(\psi') = \frac{1}{\omega'}$. Bottom right: $p_4(\psi') = \frac{\cos\theta}{p_{total}}$

Spherical Luminaire	Case 1	Case 2	Case 3	Case 4
Time (sec)	149.08	156.16	162.96	173.74
Time / Time(Case 1)	1.00	1.05	1.09	1.16

Table 3.2: Execution time of spherical luminaire and $\frac{Time}{Time(Case1)}$.

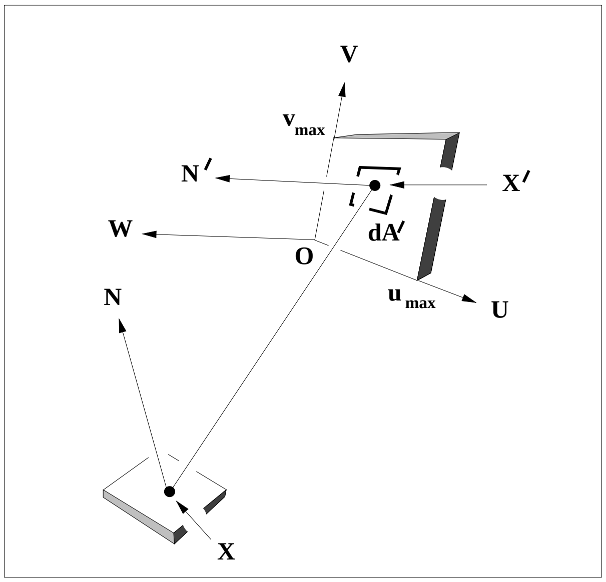


Figure 3.3: Rectangular Luminaire

jittered samples. Note that we only perform the test with N^2 jittered samples where N is a positive integer, because jittered samples work best at N^2. The jittered sampling method will be discussed in Chapter 5.

3.1.2 Rectangular Luminaires

Consider the coordinate system UVW in Figure 3.3, where the luminaire lies on the UV plane within the range $(0 \leq u \leq u_{max}, 0 \leq v \leq v_{max})$, and $x = (c_1, c_2, c_3)$, $x' = (u, v, 0)$, $N = (N_u, N_v, N_w)$, $N' = (0, 0, 1)$. The vector from x to x' is $(u - c_1, v - c_2, -c_3)$.

Case 1 : $p_1(x') = \frac{1}{A'} = \frac{1}{u_{max} v_{max}}$.

$$\dot{x} = (\xi_1 u_{max}, \xi_2 v_{max}, 0)$$

Case 3 : $p_3(x') = \frac{\frac{\cos\theta'}{\|x'-x\|^2}}{p_{total}}$

$$\begin{aligned}
p_{total} &= \int_{\forall x' \in \Omega_{x'}} \frac{\cos\theta'}{\|x'-x\|^2} dA' \\
&= \int_0^{v_{max}} \int_0^{u_{max}} \frac{N' \cdot -(u-c_1, v-c_2, -c_3)}{\|x'-x\|^3} du dv \\
&= \int_0^{v_{max}} \int_0^{u_{max}} \frac{c_3}{((u-c_1)^2 + (v-c_2)^2 + c_3^2)^{\frac{3}{2}}} du dv \\
&= \int_{-c_2}^{v_{max}-c_2} \int_{-c_1}^{u_{max}-c_1} \frac{c_3}{(u^2+v^2+c_3^2)^{\frac{3}{2}}} du dv \\
&= \arctan \frac{uv}{c_3\sqrt{u^2+v^2+c_3^2}} \Big|_{u=-c_1}^{u=u_{max}-c_1} \Big|_{v=-c_2}^{v=v_{max}-c_2}
\end{aligned}$$

$\dot{x} = (\dot{u}, \dot{v}, 0)$ can be computed by the numerical inversion of the following two formulas.

$$\begin{aligned}
\xi_1 &= \int_0^{\dot{v}} p(v) dy \\
&= \int_0^{\dot{v}} \int_0^{u_{max}} p(u,v) du dv \\
&= \int_0^{\dot{v}} \int_0^{u_{max}} p(x') du dv \\
&= \frac{1}{p_{total}} \int_{-c_2}^{\dot{v}-c_2} \int_{-c_1}^{u_{max}-c_1} \frac{c_3}{(u^2+v^2+c_3^2)^{\frac{3}{2}}} du dv \\
&= \frac{1}{p_{total}} \arctan \frac{uv}{c_3\sqrt{u^2+v^2+c_3^2}} \Big|_{u=-c_1}^{u=u_{max}-c_1} \Big|_{v=-c_2}^{v=\dot{v}-c_2}
\end{aligned}$$

$$
\begin{aligned}
\xi_2 &= \int_0^{\dot{u}} p(u|\dot{v})du \\
&= \frac{\int_0^{\dot{u}} p(u,\dot{v})du}{p(\dot{v})} \\
&= \frac{\int_0^{\dot{u}} p(u,\dot{v})du}{\int_0^{u_{max}} p(u,\dot{v})du} \\
&= \frac{\int_{-c_1}^{\dot{u}-c_1} \frac{c_3}{(u^2+(\dot{v}-c_2)^2+c_3^2)^{\frac{3}{2}}}du}{\int_{-c_1}^{u_{max}-c_1} \frac{c_3}{(u^2+(\dot{v}-c_2)^2+c_3^2)^{\frac{3}{2}}}du} \\
&= \frac{\frac{c_3 u}{((\dot{u}-c_1)^2+c_3^2)\sqrt{(\dot{u}-c_1)^2+(\dot{v}-c_2)^2+c_3^2}}\Big|_{u=-c_1}^{u=\dot{u}-c_1}}{\frac{c_3 u}{((\dot{u}-c_1)^2+c_3^2)\sqrt{(\dot{u}-c_1)^2+(\dot{v}-c_2)^2+c_3^2}}\Big|_{u=-c_1}^{u=u_{max}-c_1}}
\end{aligned}
$$

Case 4 : $p_4(x') = \frac{\frac{\cos\theta\cos\theta'}{\|x'-x\|^2}}{p_{total}}$

$$
\begin{aligned}
p_{total} &= \int_{\forall x'\in\Omega_{x'}} \cos\theta \frac{\cos\theta'}{\|x'-x\|^2} dA' \\
&= \int_0^{v_{max}}\int_0^{u_{max}} \frac{(N\cdot(u-c_1, v-c_2, -c_3))(N'\cdot -(u-c_1, v-c_2, -c_3))}{\|x'-x\|^4} dudv \\
&= \int_0^{v_{max}}\int_0^{u_{max}} \frac{(N_u(u-c_1)+N_v(v-c_2)-N_w c_3)c_3}{((u-c_1)^2+(v-c_2)^2+c_3^2)^2} dudv \\
&= \int_{-c_2}^{v_{max}-c_2}\int_{-c_1}^{u_{max}-c_1} \frac{(uN_u+vN_v-c_3N_w)c_3}{(u^2+v^2+c_3^2)^2} dudv \\
&= -\frac{(c_3N_u+uN_w)\arctan\frac{v}{\sqrt{u^2+c_3^2}}}{2\sqrt{u^2+c_3^2}} - \\
&\quad \frac{(c_3N_v+vN_w)\arctan\frac{u}{\sqrt{v^2+c_3^2}}}{2\sqrt{v^2+c_3^2}}\Big|_{u=-c_1}^{u=u_{max}-c_1}\Big|_{v=-c_2}^{v=v_{max}-c_2}
\end{aligned}
$$

$\dot{x}(=(\dot{u},\dot{v},0))$ can be computed by the numerical inversion of the following two formulas.

$$
\begin{aligned}
\xi_1 &= \int_0^{\dot{v}} p(v)dv \\
&= \int_0^{\dot{v}}\int_0^{u_{max}} p(u,v)dudv
\end{aligned}
$$

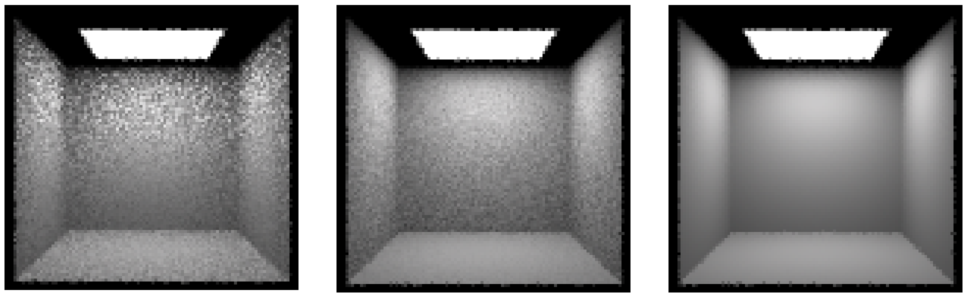


Figure 3.4: Sampling a rectangular luminaire with 1 sample per pixel. Left: $p_1(x') = \frac{1}{A'}$. Middle: $p_3(x') = \frac{\frac{\cos\theta'}{\|x'-x\|^2}}{p_{total}}$. Right: $p_4(x') = \frac{\frac{\cos\theta\cos\theta'}{\|x'-x\|^2}}{p_{total}}$.

$$
\begin{aligned}
&= \int_0^{\dot{v}} \int_0^{u_{max}} p(x') du dv \\
&= \frac{1}{p_{total}} \int_{-c_2}^{\dot{v}-c_2} \int_{-c_1}^{u_{max}-c_1} \frac{(uN_u + vN_v - c_3N_w)c_3}{(u^2+v^2+c_3^2)^2} du dv \\
&= \frac{1}{p_{total}} \{ -\frac{(c_3N_u + uN_w)\arctan\frac{v}{\sqrt{u^2+c_3^2}}}{2\sqrt{u^2+c_3^2}} - \frac{(c_3N_v + vN_w)\arctan\frac{u}{\sqrt{v^2+c_3^2}}}{2\sqrt{v^2+c_3^2}} \} \Big|_{u=-c_1}^{u=u_{max}-c_1} \Big|_{v=-c_2}^{v=\dot{v}-c_2}
\end{aligned}
$$

$$
\begin{aligned}
\xi_2 &= \int_0^{\dot{u}} p(u|\dot{v}) du \\
&= \frac{\int_0^{\dot{u}} p(u,\dot{v}) du}{p(\dot{v})} \\
&= \frac{\int_0^{\dot{u}} p(u,\dot{v}) du}{\int_0^{u_{max}} p(u,\dot{v}) du} \\
&= \frac{\int_{-c_1}^{\dot{u}-c_1} \frac{(uN_u + (\dot{v}-c_2)N_v - c_3N_w)c_3}{(u^2+(\dot{v}-c_2)^2+c_3^2)^2} du}{\int_{-c_1}^{u_{max}-c_1} \frac{(uN_u + (\dot{v}-c_2)N_v - c_3N_w)c_3}{(u^2+(\dot{v}-c_2)^2+c_3^2)^2} du} \\
&= \frac{\frac{-c_3^3N_u - c_3^2uN_w + c_3u(\dot{v}-c_2)N_v - c_3(\dot{v}-c_2)^2N_u}{2((\dot{v}-c_2)^2+c_3^2)(u^2+(\dot{v}-c_2)^2+c_3^2)} + \frac{c_3(-c_3N_w + (\dot{v}-c_2)N_v)\arctan\frac{u}{\sqrt{(\dot{v}-c_2)^2+c_3^2}}}{2((\dot{v}-c_2)^2+c_3^2)^{1.5}} \Big|_{u=-c_1}^{u=\dot{u}-c_1}}{\frac{-c_3^3N_u - c_3^2uN_w + c_3u(\dot{v}-c_2)N_v - c_3(\dot{v}-c_2)^2N_u}{2((\dot{v}-c_2)^2+c_3^2)(u^2+(\dot{v}-c_2)^2+c_3^2)} + \frac{c_3(-c_3N_w + (\dot{v}-c_2)N_v)\arctan\frac{u}{\sqrt{(\dot{v}-c_2)^2+c_3^2}}}{2((\dot{v}-c_2)^2+c_3^2)^{1.5}} \Big|_{u=-c_1}^{u=u_{max}-c_1}}
\end{aligned}
$$

Figure 3.4 has one viewing ray for each of the 200^2 pixels. The average execution time and

Rectangular Luminaire	Case 1	Case 3	Case 4
Time (sec)	133.82	218.6	246.04
Time / Time(Case 1)	1.00	1.63	1.84

Table 3.3: Execution time of rectangular luminaire and $\frac{Time}{Time(Case1)}$.

the time as a fraction of the time taken by case 1 from 10 trials are in Table 3.1.2.

To achieve the same visual smoothness as case 4, case 1 seems to require more than 100 jittered samples and case 3 needs 9 jittered samples.

3.1.3 Triangular Luminaires

Consider the coordinate system UVW as in Fig. 3.3. Suppose the triangular luminaire is bounded by $u = 0$ at bottom, $u = av$ at left, and $u = cv + d$ at right. Also, $u = av$ and $u = cv + d$ meet at $(u_{max}, v_{max}, 0)$ where $u_{max} = \frac{ad}{a-c}$ and $v_{max} = \frac{d}{a-c}$.

Case 1 : $p_1(x') = \dfrac{1}{A'} = \dfrac{1}{\frac{v_{max}d}{2}}$.

$$\dot{x} = (1 - \sqrt{1-\xi_1})(d, 0, 0) + \xi_2\sqrt{1-\xi_1}(u_{max}, v_{max}, 0)$$

where $\dot{x}$ is in vector form.

Case 3 : $p_3(x') = \dfrac{\frac{\cos\theta'}{\|x'-x\|^2}}{p_{total}}$

We have not been able to compute case 3 exactly, so instead the solid angle covered by the triangle luminaire, it is approximated by a triangle whose vertices are the intersections of the unit sphere centered at x and the three lines connecting x and the actual luminaire vertices [76].

Case 4 : $p_4(x') = \dfrac{\frac{\cos\theta\cos\theta'}{\|x'-x\|^2}}{p_{total}}$

$$\begin{aligned}
p_{total} &= \int_{\forall x' \in \Omega_{x'}} \cos\theta \frac{\cos\theta'}{\|x'-x\|^2} dA' \\
&= \int_0^{v_{max}} \int_{av}^{cv+d} \frac{(N\cdot(u-c_1, v-c_2, -c_3))(N'\cdot -(u-c_1, v-c_2, -c_3))}{\|x'-x\|^4} dudv \\
&= \int_0^{v_{max}} \int_{av}^{cv+d} \frac{(N_u(u-c_1)+N_v(v-c_2)-N_w c_3)c_3}{((u-c_1)^2+(v-c_2)^2+c_3^2)^2} dudv \\
&= \int_{-c_2}^{v_{max}-c_2} \int_{a(v+c_2)-c_1}^{c(v+c_2)+d-c_1} \frac{(uN_u+vN_v-c_3N_w)c_3}{(u^2+v^2+c_3^2)^2} dudv \\
&= \frac{(c_3N_u - ac_3N_v + (ac_2-c_1)N_w)\arctan\frac{a(ac_2-c_1)+v+a^2v}{\sqrt{a^2c_3^2+c_3^2+(ac_2-c_1)^2}}}{2\sqrt{a^2c_3^2+c_3^2+(ac_2-c_1)^2}} - \\
&\quad \frac{(c_3N_u - cc_3N_v + (d+cc_2-c_1)N_w)\arctan\frac{c(d+cc_2-c_1)+v+c^2v}{\sqrt{c^2c_3^2+c_3^2+(d+cc_2-c_1)^2}}}{2\sqrt{c^2c_3^2+c_3^2+(d+cc_2-c_1)^2}} + \\
&\quad \frac{(c_3N_v+vN_w)\arctan\frac{a(v+c_2)-c_1}{\sqrt{v^2+c_3^2}}}{2\sqrt{v^2+c_3^2}} - \frac{(c_3N_v+vN_w)\arctan\frac{c(v+c_2)+d-c_1}{\sqrt{v^2+c_3^2}}}{2\sqrt{v^2+c_3^2}} \Big|_{v=-c_2}^{v=v_{max}-c_2}
\end{aligned}$$

$\dot{x}(=(\dot{u},\dot{v},0))$ can be computed by the numerical inversion of the following two formulas.

$$\begin{aligned}
\xi_1 &= \int_0^{\dot{v}} p(v)dv \\
&= \int_0^{\dot{v}} \int_{av}^{cv+d} p(u,v)dudv \\
&= \int_0^{\dot{v}} \int_{av}^{cv+d} p(x')dudv
\end{aligned}$$

$$\begin{aligned}
&= \frac{1}{p_{total}} \int_{-c_2}^{\dot{v}-c_2} \int_{a(v+c_2)-c_1}^{c(v+c_2)+d-c_1} \frac{(uN_u + vN_v - c_3N_w)c_3}{(u^2+v^2+c_3^2)^2} du dv \\
&= \frac{1}{p_{total}} \{ \frac{(c_3N_u - ac_3N_v + (ac_2-c_1)N_w) \arctan \frac{a(ac_2-c_1)+v+a^2v}{\sqrt{a^2c_3^2+c_3^2+(ac_2-c_1)^2}}}{2\sqrt{a^2c_3^2+c_3^2+(ac_2-c_1)^2}} - \\
&\frac{(c_3N_u - cc_3N_v + (d+cc_2-c_1)N_w) \arctan \frac{c(d+cc_2-c_1)+v+c^2v}{\sqrt{c^2c_3^2+c_3^2+(d+cc_2-c_1)^2}}}{2\sqrt{c^2c_3^2+c_3^2+(d+cc_2-c_1)^2}} + \\
&\frac{(c_3N_v + vN_w) \arctan \frac{a(v+c_2)-c_1}{\sqrt{v^2+c_3^2}}}{2\sqrt{v^2+c_3^2}} - \frac{(c_3N_v + vN_w) \arctan \frac{c(v+c_2)+d-c_1}{\sqrt{v^2+c_3^2}}}{2\sqrt{v^2+c_3^2}} \}\Big|_{v=-c_2}^{v=\dot{v}-c_2}
\end{aligned}$$

$$\begin{aligned}
\xi_2 &= \int_{a\dot{v}}^{\dot{u}} p(u|\dot{v}) du \\
&= \frac{\int_{a\dot{v}}^{\dot{u}} p(u,\dot{v}) du}{p(\dot{v})} \\
&= \frac{\int_{a\dot{v}}^{\dot{u}} p(u,\dot{v}) du}{\int_{a\dot{v}}^{c\dot{v}+d} p(u,\dot{v}) du} \\
&= \frac{\int_{a\dot{v}-c_1}^{\dot{u}-c_1} \frac{(uN_u+(\dot{v}-c_2)N_v-c_3N_w)c_3}{(u^2+(\dot{v}-c_2)^2+c_3^2)^2} du}{\int_{a\dot{v}-c_1}^{c\dot{v}+d-c_1} \frac{(uN_u+(\dot{v}-c_2)N_v-c_3N_w)c_3}{(u^2+(\dot{v}-c_2)^2+c_3^2)^2} du} \\
&= \frac{\frac{-c_3^3N_u-c_3^2uN_w+c_3u(\dot{v}-c_2)N_v-c_3(\dot{v}-c_2)^2N_u}{2((\dot{v}-c_2)^2+c_3^2)(u^2+(\dot{v}-c_2)^2+c_3^2)} + \frac{c_3(-c_3N_w+(\dot{v}-c_2)N_v)\arctan\frac{u}{\sqrt{(\dot{v}-c_2)^2+c_3^2}}}{2((\dot{v}-c_2)^2+c_3^2)^{1.5}}\Big|_{u=a\dot{v}-c_1}^{u=\dot{u}-c_1}}{\frac{-c_3^3N_u-c_3^2uN_w+c_3u(\dot{v}-c_2)N_v-c_3(\dot{v}-c_2)^2N_u}{2((\dot{v}-c_2)^2+c_3^2)(u^2+(\dot{v}-c_2)^2+c_3^2)} + \frac{c_3(-c_3N_w+(\dot{v}-c_2)N_v)\arctan\frac{u}{\sqrt{(\dot{v}-c_2)^2+c_3^2}}}{2((\dot{v}-c_2)^2+c_3^2)^{1.5}}\Big|_{u=a\dot{v}-c_1}^{u=c\dot{v}+d-c_1}}
\end{aligned}$$

Figure 3.5 has one viewing ray for each of the 200^2 pixels. The average execution time and the time as a fraction of the time taken by case 1 from 10 trials are in Table 3.1.3.

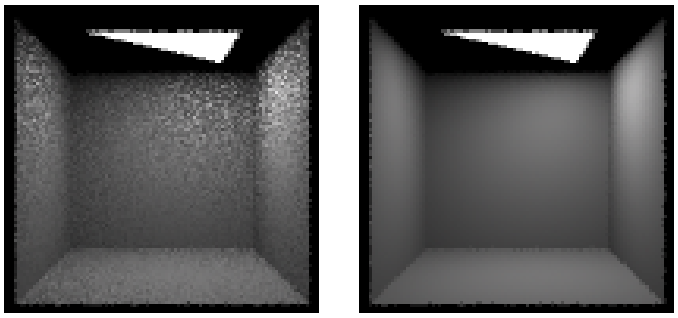


Figure 3.5: Sampling a triangular luminaire with 1 sample per pixel. Left: $p_1(x') = \frac{1}{A'}$. Right: $p_4(x') = \frac{\frac{\cos\theta\cos\theta'}{\|x'-x\|^2}}{p_{total}}$.

Triangular Luminaire	Case 1	Case 4
Time (sec)	153.64	276.84
Time / Time(Case 1)	1.00	1.8

Table 3.4: Execution time of triangular luminaire and $\frac{Time}{Time(Case1)}$.

3.1.4 Polygonal Luminaires

Suppose a polygon is combined of n triangles, T_1,. . .,T_n.

Case 1 : $p_1(x') = \frac{1}{A'}$.

$A' = \sum_{i=1}^{n} A'_{T_i}$ or if the polygon has been projected onto XY plane, then $A' = \frac{1}{2}\,(\sum_{i=1}^{n} x_i\, y_{(i+1 \bmod i)} - \sum_{i=1}^{n} y_i\, x_{(i+1 \bmod i)})$, where (x_i, y_i), $i+1, \ldots, n$ is the counter-clockwise enumeration of the vertices of P [56].

Finding $\dot{x}$ with a given random number pair (ξ_1, ξ_2) is equivalent to finding T_i such that $\frac{1}{A'}\sum_{k=1}^{i-1} A'_{T_i} \le \xi_1 < \frac{1}{A'}\sum_{k=1}^{i} A'_{T_i}$ and then finding a point on triangle T_i with ξ_2, as discussed in Section 3.1.3.

Case 3 : $p_3(x') = \frac{\frac{\cos\theta'}{\|x'-x\|^2}}{p_{total}}$

First we find a rectangle R, that contains the polygon and is as small as possible. Choose a sample on R as discussed in Section 3.1.2. If the sample is outside the polygon, then the sample is a waste. Note that throwing away redundant samples is cheap because a shadow ray is not sent unless the sample is inside the polygon.

Case 4 : $p_4(x') = \frac{\frac{\cos\theta\cos\theta'}{\|x'-x\|^2}}{p_{total}}$

p_{total} is the sum of the p_{total} from T_1,. . .,T_n as discussed in Section 3.1.3. A better way to compute p_{total} is in [51, 3]. Suppose the polygonal luminaire has n vertices, p_1,. . .,p_n in clockwise enumeration as seen in x in Figure 3.3,

$$P_{total} = \frac{1}{2} \sum_{j=1}^{i} \theta_j (N \cdot N_j) \tag{3.4}$$

where θ_j is the angle in radians, between the two vectors x to p_j and x to $p_{(j+1 \bmod i)}$. N_j is the unit normal vector of the triangle $x\,p_j\,p_{(j+1 \bmod i)}$ - the cross product of the two vectors, x to p_j and x to $p_{(j+1 \bmod i)}$.

Finding $\dot{x}$ with a given random number pair (ξ_1, ξ_2) can be accomplished in two steps. The first step is to choose T_i such that $\frac{P_{total}\ of\ T_1, \ldots, T_{i-1}}{P_{total}\ of\ the\ polygon} \leq \xi_1 < \frac{P_{total}\ of\ T_1, \ldots, T_i}{P_{total}\ of\ the\ polygon}$. P_{total} of T_1,...,T_i can be incrementally computed by Equation 3.4 when T_1,...,T_i form a convex polygon. Note if T_1,...,T_i form a concave polygon, then Equation 3.4 can't be used because P_{total} might be negative. The second step is to choose a point on T_i with random number ξ_2.

3.1.5 Disk Luminaires

Suppose the disk luminaire with radius r is centered at the origin and that $N' = (0, 0, 1)$.

Case 1 : $p_1(x') = \frac{1}{A'} = \frac{1}{\pi r^2}$.

$$\dot{x} = (r\sqrt{\xi_1}, 2\pi\xi_2, 0)$$

where $\dot{x}$ is in the local spherical coordinate system centered at the disk. The disk luminaire

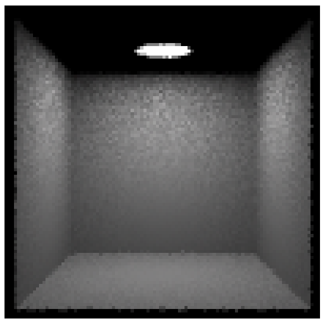


Figure 3.6: Sampling a disk luminaire with 1 sample per pixel. $p_1(x') = \dfrac{1}{A'}$.

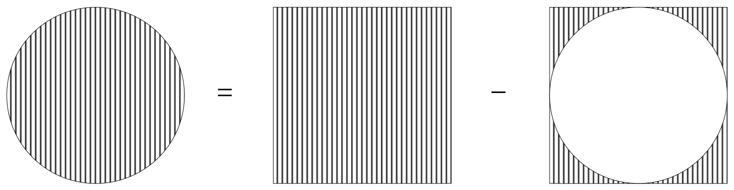


Figure 3.7: Decomposition of a Disk Luminaire

sampled by case 1 is shown in Figure 3.6.

Case 3 and case 4 : decomposition of the visible region A''.

The disk can be constructed as Figure 3.7 and instead of sampling a disk, we sample the square as discussed in Section 3.1.2. Although the samples outside the disk are wasted, the computation cause by these redundant samples is relatively cheap since the shadow ray is guaranteed to miss the luminaire and hence can save the expensive computation of shadow ray intersection.

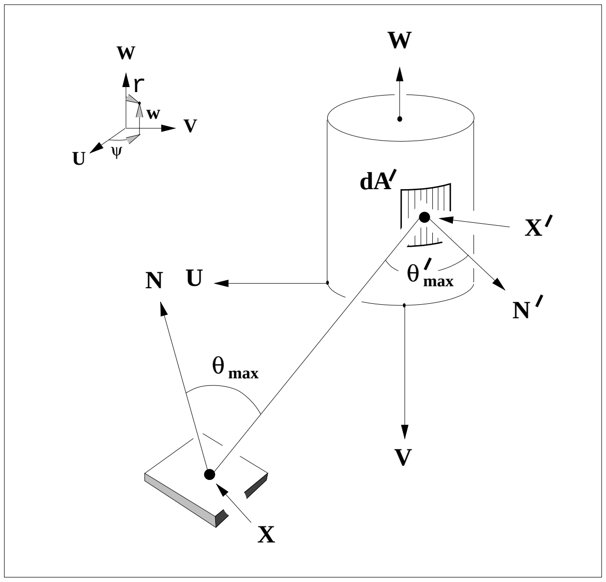


Figure 3.8: Cylindrical Luminaire

3.1.6 Cylindrical Luminaires

Consider the cylindrical coordinate system UVW as in Fig. 3.8. The cylinder can be defined by $(0 \leq \psi \leq 2\pi, 0 \leq w \leq w_{max}, r)$ in cylindrical coordinate system and the visible region is $(0 \leq \psi \leq \psi_{max}, 0 \leq w \leq w_{max}, r)$.

Case 1 : $p_1(x') = \dfrac{1}{A'} = \dfrac{1}{2\pi r w_{max}}$.

$$\dot{x} = (2\pi\xi_1, w_{max}\xi_2, r)$$

where $\dot{x}$ is in cylindrical coordinate system.

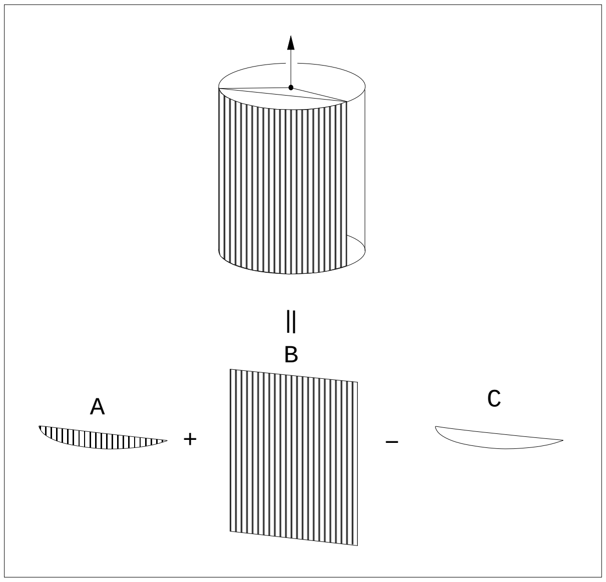


Figure 3.9: Decomposition of Cylindrical Luminaire

Case 2 : $p_2(x') = \dfrac{1}{A''} = \dfrac{1}{\psi_{max} r w_{max}}$.

$$\dot{x} = (\psi_{max}\xi_1, w_{max}\xi_2, r)$$

Case 3 and case 4 : decomposition of the visible region A''.

The visible region of a cylindrical luminaire in Figure 3.9 can only be A+(B-C) or A+B+C. If (B-C), we just sample B and waste the sample when it is in C. Note that the shadow ray to a sample in C is not necessary because we know in advance the ray will miss the luminaire.

A and C are regions within a disk bounded by $-\psi'_{max} \le \psi \le \psi'_{max}$. Assume the disk with radius r is on UV, centered at the origin. Then A is bounded by $\cos\psi'_{max} \leq u \leq 1$ and $-\sqrt{r^2-u^2} \leq v\sqrt{r^2-u^2}$. To sample A, we can use $p_1(x') = \frac{1}{A'}$,

$$\begin{aligned} A' &= \int_{\cos\psi'_{max}}^{1} \int_{-\sqrt{r^2-u^2}}^{\sqrt{r^2-u^2}} dv du \\ &= u\sqrt{r^2-u^2} + r^2 \arcsin\frac{u}{r}\Big|_{u=\cos\psi'_{max}}^{u=1} \end{aligned}$$

$\dot{x}(=(\dot{u},\dot{v},0))$ can be computed by the numerical inversion of the following two formulas.

$$\begin{aligned} \xi_1 &= \int_{\cos\psi'_{max}}^{\dot{u}} p(u)du \\ &= \int_{\cos\psi'_{max}}^{\dot{u}} \int_{-\sqrt{r^2-u^2}}^{\sqrt{r^2-u^2}} p(u,v)dvdu \\ &= \int_{\cos\psi'_{max}}^{\dot{u}} \int_{-\sqrt{r^2-u^2}}^{\sqrt{r^2-u^2}} p(x')dvdu \\ &= \frac{1}{A'}u\sqrt{r^2-u^2} + r^2 \arcsin\frac{u}{r}\Big|_{u=\cos\psi'_{max}}^{u=\dot{u}} \end{aligned}$$

$$\begin{aligned} \xi_2 &= \frac{\int_{-\sqrt{r^2-\dot{u}^2}}^{\dot{v}} dv}{\int_{-\sqrt{r^2-\dot{u}^2}}^{\sqrt{r^2-\dot{u}^2}} dv} \\ &= \frac{\dot{v}+\sqrt{r^2-\dot{u}^2}}{2\sqrt{r^2-\dot{u}^2}} \end{aligned}$$

$\dot{u}$ can be solved by the numerical inversion and $\dot{v} = (2\xi_2 - 1)\sqrt{r^2-\dot{u}^2}$.

Figure 3.10 has one viewing ray for each of the 200^2 pixels. The average execution time and the time as a fraction of the time taken by case 1 from 10 trials are in Table 3.1.6.

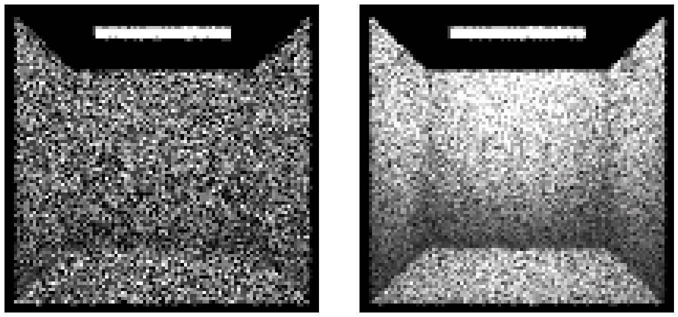


Figure 3.10: Sampling a cylindrical luminaire with 1 sample per pixel. Left: $p_1(x') = \frac{1}{A'} = \frac{1}{2\pi r w_{max}}$. Right: $p_2(x') = \frac{1}{A''} = \frac{1}{\psi_{max} r w_{max}}$.

Cylindrical Luminaire	Case 1	Case 2
Time (sec)	162.8	174.32
Time / Time(Case 1)	1.00	1.07

Table 3.5: Execution time of cylindrical luminaire and $\frac{Time}{Time(Case1)}$.

3.1.7 Tabular Luminaires

A TV screen can not always be treated as one big rectangular luminaire since the color change has an impact on the nearby object. Similarly, a clear sky dome, with maximum radiance contrast over 30 [53], should not be simulated as a luminaire with constant radiance.

When computing the direct lighting, a TV screen can be treated as a collection of pixel luminaires while a clear sky can be discretized and treated as a group of luminaires, but obviously this is not efficient because the coherence among individual luminaires is not considered.

A better approach is to construct a hierarchy such as a Mip map [85] or a summed-area table [16], and use the coarsest representation whenever possible. Note that the solid angle and the projected solid angle from a rectangular luminaire can be analytically computed as in Section 3.1.2, which makes possible of close estimate of the direct lighting from each Mip map cell for a TV screen.

3.2 Sampling Strategies for Non-diffuse Environments

If $f_{r\lambda}(x, \psi, \psi')$ is the only dominant term in $f_{r\lambda}(x, \psi, \psi')E_\lambda(x', -\psi')\cos\theta$ and it is represented by a Phong BRDF [41, 66] or a Gaussian Model [78], then an effective $p(\psi') \sim f_{r\lambda}(x, \psi, \psi')$ can be derived.

If only one term dominates each time, the weighted quasi-Monte Carlo methods in Section 2.2.3 can choose samples from each term in its local coordinate system. However, generally in a non-diffuse environment choosing a $p(\psi')$ with fast convergence rate is very difficult because the different local coordinate systems make the combined behavior of $f_{r\lambda}(x, \psi, \psi')E_\lambda(x', -\psi')\cos\theta d\omega_{\psi'}$ hard to predict [66].

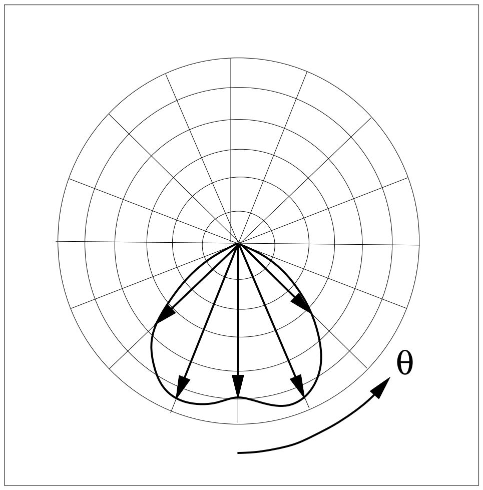


Figure 3.11: Polar graph of luminous intensity.

3.3 Luminaires and Imposters

Due to complexity and efficiency considerations, objects are sometimes modified and often simplified. The simplified object can be thought of as an “imposter”. An example of an imposter is an object with zero reflectivity that is shaped like the original object and emits light in exactly the same distribution and intensity as is reflected from the original non-emitting object [67]. This light emitting imposter would be indistinguishable from the original light reflecting object. The application of imposter objects appears frequently in computer graphics as special cases but the use of imposters have not been formally justified. In this section, we present several examples of imposters, paying special attention to imposter objects for luminaires.

A luminaire is often specified by its shape and luminous intensity (which defines the luminous flux per unit solid angle as explained in Appendix A) [52, 35, 32]. The luminous intensity of a luminaire can be represented as a polar graph (luminous intensity distribution curve), which shows a typical distribution curve of a point on the luminaire at a particular plane, in Figure 3.11.

In this section we discuss imposter objects for luminaires which are used to make direct and indirect lighting computations more efficient.

3.3.1 Imposters with Simplified Luminaire Specification

In computer graphics the lighting computation often assumes the luminaire has a simple geometry such as a point, a sphere, a rectangle, and has a luminous intensity distribution similar to a Phong distribution: $k \cos^N \theta$, where k is a constant and θ is as illustrated in Figure 3.11. If $N = 1$, the luminaire is perfectly diffuse [74, 67, 66]. These luminaire imposters can sometimes be very different from a real luminaire, but their use has been widely accepted in realistic rendering. Figure 3.12 replaces a luminaire with two emitting cylinders and fixtures by a rectangular luminaire imposter in the lighting computation. In the geometry simplification method suggested by Rushmeimer et al. [58], groups of objects are replaced by a box.

The luminaire imposter helps speed up the lighting calculation, but the luminaire displayed in the image must be in its original representation. For example, if a incandescent lamp is in an image, the lamp should be seen in its true shape with the filament in order

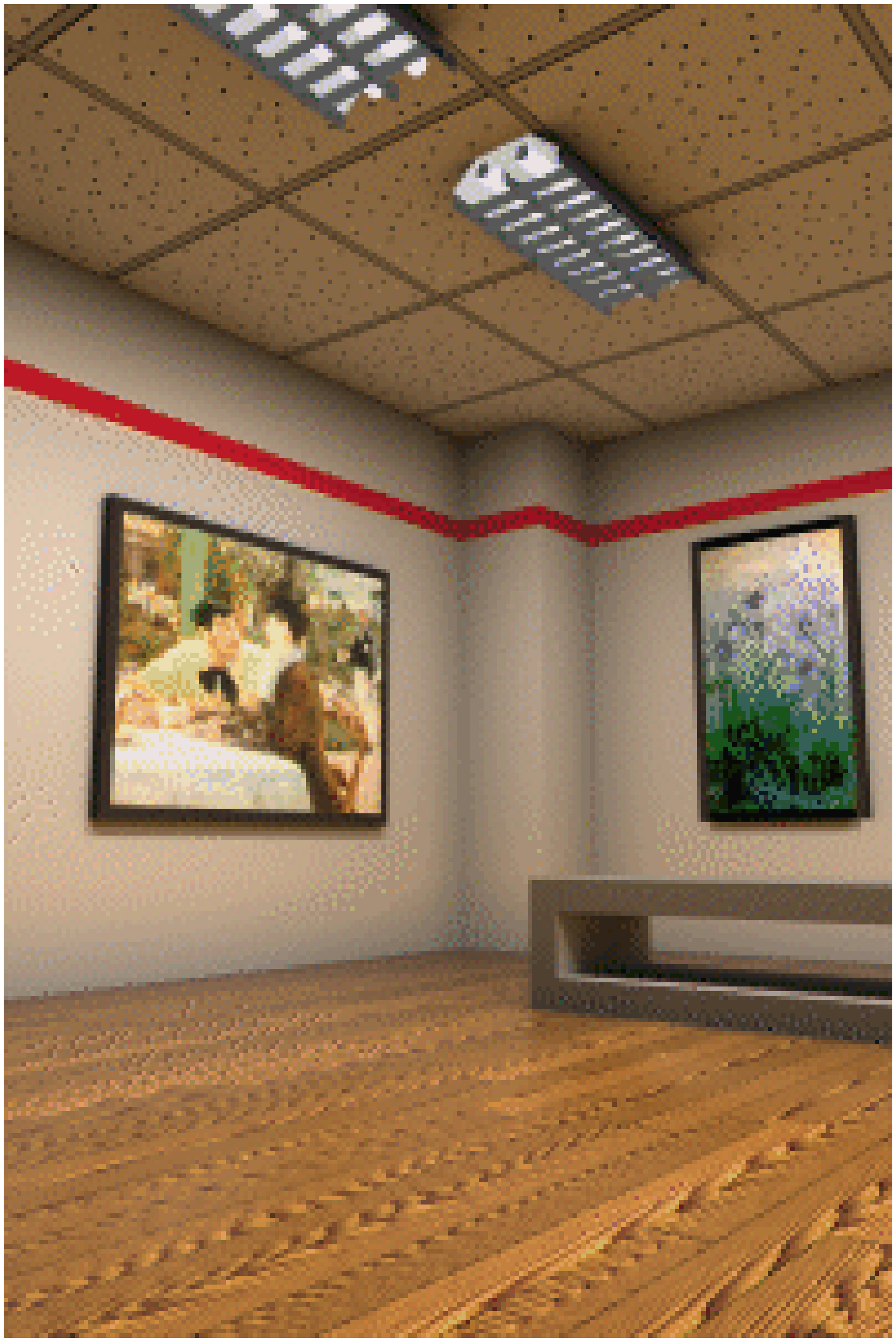


Figure 3.12: An imposter of rectangular luminaire for lighting computation.

to be realistic. In an object-oriented design, this is achieved by associating two intersection tests with a luminaire, one for the viewing ray and the other for the shadow ray [79]. However, the shading on the fixtures in this approach would be wrong because the imposter can result in noticeable shading difference on nearby objects. To fix this, we ask the shadow ray to see the true luminaire when an object is close to the luminaire. This is similar to Rushmeier's strategy of not using the simple environment inside a threshold radius [58]. This method can be generalized to incorporate multiple representation for an object. For example, the hierarchy suggested for a TV screen or the sky in Section 3.1.7. In zonal methods, combining (energy-receiving) elments and (energy-releasing) patches is a typical example of two-level representation [13, 12] while the hierarchical algorithm is an example of multiple resolution [29, 26, 2].

To compute the color change due to an imposter at a point or in a pixel is difficult since it involves the computation of the difference between two high dimensional integrals with different domains (the shape of the luminaire) and integrands (the luminous intensity of the luminaire). Yet, to decide the accuracy of an imposter is more complicated than computing the color difference, because human visual system does not see the absolute radiance [40, 7]. Furthermore, since images are not compared side-by-side, the "accuracy" of an imposter depends on how humans interpret the visual cues to verify the validity of an image. Because of this complexity in human perception, an effective objective measure of the validity of an imposter will be difficult to derive.

Modification schemes similar to luminaire imposters are widely used in image synthesis, for example objects with simplified reflectance functions and objects stored at multiple levels of resolution [21, 27, 80]. Another typical imposter example is to change the physical behavior. For example, non-emitting objects in zonal methods can emit the same amount of light as they would reflect [25, 63, 29].

3.3.2 Imposters with Extra Information

In a daytime indoor scene, the dominant luminaire would be the sun (assuming it is visible). Without predetermined knowledge of visibility in the environment, the direct lighting is carried out by sending shadow rays to the sun. Instead, if we use the luminaire imposter which includes the sun and the knowledge of the window frame, we can screen an object to avoid sending shadow rays if the object cannot see the sun through the window. A similar example is a luminaire imposter for a lamp with built-in knowledge of its surrounding lamp shade as in [59]. This extra information in an environment with multiple luminaires can help to derive a p.d.f. whose samples converge faster since a luminaire, which is invisible to the object of interest, will not be selected.

3.3.3 Imposters on a virtual domain

In lighting computation, a new geometric shape in a luminaire imposter corresponds to a new domain in surface area. Similarly, the modification can be performed on the domain

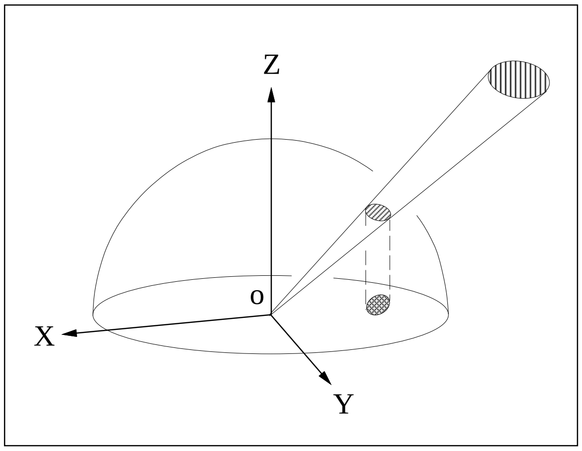


Figure 3.13: Nusselt analogy - the projected solid angle of an object is equal to the area being projected on the plane perpendicular to the normal vector at the illuminated point.

of solid angle, i.e. $\int_{\Omega} f(x)dx \approx \sum_{\Omega'} f(x)$. A special subdivision scheme to transform the curved surface of solid angle into planes is the hemicube [11], i.e. replacing a unit sphere by a unit box. An element in a hemicube is the smallest unit in approximation; if the projected luminaire covers the center of a element, the whole element is assumed to be covered; otherwise, the element is assumed to be totally uncovered.

The third integration domain in lighting computation is integrated over $\cos\theta d\omega$. One special subdivision scheme which can transform this non-intuitive integration domain into a plane is the rectangular grid suggested by Sillion [70]. In this subdivision, the projected solid angle $\int_{\Omega} \cos\theta d\omega$ of a luminaire is approximated by the product of the number of elements and $\frac{\pi}{N}$, where N is the total number of elements.

A different subdivision scheme for $\cos\theta d\omega$ is to apply Nusselt analogy [68, 11], i.e. $\cos\theta d\omega$ equals the projected area of $d\omega$ on the plane P perpendicular to the normal vector

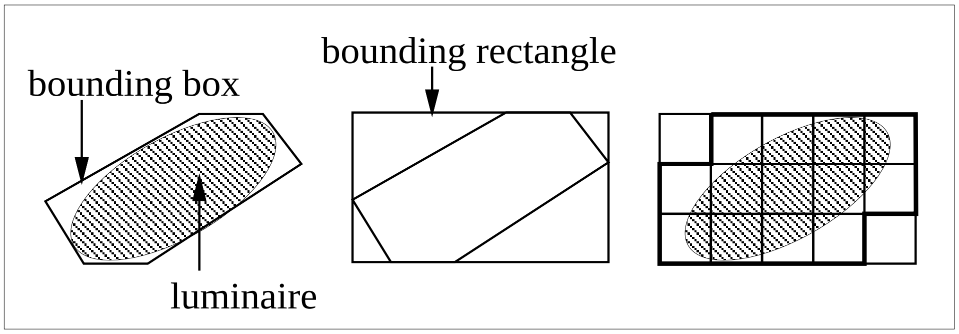


Figure 3.14: Subdivision Scheme for the domain over $\cos\theta d\omega$.

at the illuminated point as shown in Figure 3.13. Note the projected area can never exceed the unit circle. The subdivision scheme for this plane is illustrated in Figure 3.14 and is as follows.

1. Give a bounding box of the luminaire. The bounding box can be a volume for a $3D$ luminaire or a $2D$ shape for a planar luminaire.
2. Project the bounding box onto the plane P.
3. Compute the axis-aligning bounding rectangle of the projected bounding box.
4. Divide the bounding rectangle into square grids and treat each grid cell as the smallest element in the approximation.

Lighting estimation in the integration domain over $\cos\theta d\omega$ is effective in a diffuse environment because of the dominant term $\cos\theta$. The subdivision over the plane P is also suitable for specular surfaces because adaptive subdivision in a square cell is straightforward and the corresponding bidirectional reflectance-distribution function f_r on P can be computed quickly, $f_r(\theta_i, \phi_i; \theta_r, \phi_r) = f(\theta_i, \phi_i; \cos^{-1}\sqrt{1 - x^2 - y^2}, \tan^{-1}\frac{y}{x})$. To compute the corresponding incident radiance from a luminaire requires sending rays from the illuminated point toward $(\theta_r, \phi_r, 1)$ in spherical coordinations.

3.4 Summary

The major contribution of this chapter is a thorough discussion of direct lighting computation in a diffuse environment. The shapes of real-world luminaires are more complicated than the simple geometries discussed here, but the shading caused by these simplified luminaires is often indistinguishable from the real ones. So, it is plausible to accelerate the direct lighting computation by replacing the real luminaires by simplified ones (imposters). Note that the viewing ray should still see the complicated luminaire.

The most complicated integrals in Section 3.1.2 and Section 3.1.3 were solved using the symbolic algebra package Mathematica[87]. We also tried to apply the same sampling strategy in case 3 and case 4 to other types of luminaires, but the integrals were too difficult for Mathematica, so we have chosen approximation schemes. The implementation of these methods requires special caution, especially when the implementation of a formula takes tens of lines. Our experience tells that a typing error in this sort of code is easy to make, and difficult to find.

Which sampling strategy among the four discussed in this chapter should be used? In general, the case 4 with $p(\psi') = \dfrac{\cos\theta}{p_{total}}$ is the best in a diffuse environment because of its fast convergence rate and small overhead, i.e. ray intersection becomes more expensive. In a non-diffuse environment, the sampling methods suggested here are valid but can be inappropriate because they can cause high variance and require many samples. However, finding a $p \sim f_{r\lambda}(x, \psi, \psi')E_\lambda(x', -\psi')\cos\theta$ for general environment is very difficult and remains a research topic.

In path tracing [36], a viewing ray generates one shadow ray for direct lighting and one reflected/transmitted ray for indirect lighting. In a diffuse environment, this becomes very inefficient because the number of shadow rays needed dramatically decreases when $p(\psi') = \dfrac{1}{\omega'}$ or, when $p(\psi') = \dfrac{\cos\theta}{p_{total}}$, while the number of reflected/transmitted rays needed remains the same. This suggests a flexible branching scheme, with an uneven number of shadow rays and reflected/transmitted rays. Such a branching scheme opens two new questions. The first one is how to predict two numbers n and m - one for shadow rays and the other for viewing rays. The second question is how to choose m well-distributed rays among n viewing rays (assuming $m < n$).

Luminaire imposters which simplify the geometry shape or the luminous intensity distribution of the replaced luminaire can speed up the lighting computation while still generating convincing images. The simple luminaires discussed in Section 3.1 are examples of imposters.

4

Direct Lighting from Multiple Luminaires

The direct lighting from multiple luminaires is the sum of the direct lighting from each individual luminaire, i.e. $L_{direct} = \sum_{i=1}^{N} L_{direct}^{[i]}$. In traditional ray tracing programs the direct lighting computation is carried out by explicit summation [15], which requires one shadow ray for each luminaire. This approach can be computationally expensive because too many shadow rays are sent to unimportant luminaires which have little contribution.

One possible improvement over the straightforward summation is to predict the contribution from each luminaire, then send shadow rays to the dominant luminaires, and finally add together the direct lighting from the dominant luminaires and a guessed direct lighting from the unimportant luminaires [78]. Predicting the importance of a luminaire in a non-diffuse environment can be complicated because it should account for the luminaire intensity, reflectance of the illuminated surface, solid angle of the luminaire, and the visibility [39]. In this chapter we formalize an approach of designating important/unimportant luminaires, and present three specific methods that follow this approach [65, 66, 77]. These methods are aimed not only at efficiently classifying luminaires, but also at efficiently calculating the contribution from each luminaire.

4.1 Monte Carlo Methods for Multiple Luminaires

The direct lighting from multiple luminaires is the sum of the direct lighting from each luminaire as in Equation 4.1. The Monte Carlo methods that are used to approximate an integral can also be applied to approximate summations. A formal argument is given in the Appendix B to verify the Monte Carlo summation

$$L_{direct} = \sum_{i=1}^{N} L_{direct}^{[i]} \approx \frac{L_{direct}^{[j]}}{p_j} \tag{4.1}$$

where $L_{direct}^{[i]}$ is the direct lighting from the ith luminaire and can be approximated by the methods in Chapter 3. p_j is the probability of choosing the jth luminaire, and hence $\sum_{i=1}^{N} p_i = 1$ and $p_j > 0$ if $L_{direct}^{[j]} > 0$.

Equation 4.1 means that the primary estimate of the direct lighting from multiple luminaires only needs one shadow ray. This gives us the ability to easily manage the number of shadow rays and thus to compute the direct lighting to any desired precision.

Any valid p.d.f. can be used to approximate L_{direct} in Equation 4.1, but a good p.d.f. should be inexpensive to compute and generate samples with fast convergence rate. An ideal p.d.f. will have its distribution proportional to luminaires' contribution, i.e. the effort(number of samples) being put on each luminaire is proportional to its contribution. In the remaining sections, we examine three different strategies for constructing a p.d.f.

4.2 Constant Method

The simplest p.d.f. is the constant one, with $p_i = \frac{1}{N}$ for all $i \in [0, N]$ [41]. Since this method is both simple and fast, it is useful for debugging and testing, but its high variance requires too many samples for rendering a realistic scene.

If we use stratified samples[1] in the constant method and send N shadow rays for each viewing ray, then this special case of the constant method is equivalent to the approach of sending one shadow ray to each luminaire in traditional ray tracing methods.

[1]Stratified sampling first divides the sampling domain and then chooses a sample from each subdomain in order to guarantee equidistributed samples. Jittering is an example of stratified samples as discussed in Section 5.2.2.

4.3 Linear Method

To minimize the noise caused by a bad choice of p_i, we can consult each luminaire to get a good estimate of $L_{direct}^{[i]}$ for all $i \in [1, N]$. We call this the linear method since its execution time is linearly proportional to the number of luminaires. This method of setting p_i was first used in [61], and was first theoretically justified in [65].

In a diffuse environment, we can estimate $L_{direct}^{[i]}$ by assuming the entire luminaire is visible. For example if the luminaire a contributes L_a and the luminaire b contributes L_b when unblocked, then the probability of choosing the luminaire a is $\dfrac{L_a}{L_a + L_b}$ whereas choosing the luminaire b is $\dfrac{L_b}{L_a + L_b}$. If $L_a \gg L_b$, luminaire a is much more likely to be chosen.

The linear method is similar to a generalized form of Ward's method [78]. In Ward's method, separation of important and unimportant luminaires means using two summations to approximate L_{direct}, and the important luminaires are equivalent to the luminaires with large p.d.f.

Figure 4.1 has 100 rectangular luminaires sampled by the linear method, where the direct lighting from a luminaire is estimated by $f_r(x, \psi, \psi')E(x', \psi') \int_{\forall \psi' \in \Omega_{\psi'}} \cos\theta d\omega'$ as the case 4 in Section 3.1 and each pixel has between 3 and 10 samples in this case.

Implementing the linear p_i code was trickier than we expected. We implemented a method for each type of luminaire that estimated $L_{direct}^{[i]}$ for a particular x and f_r. If the

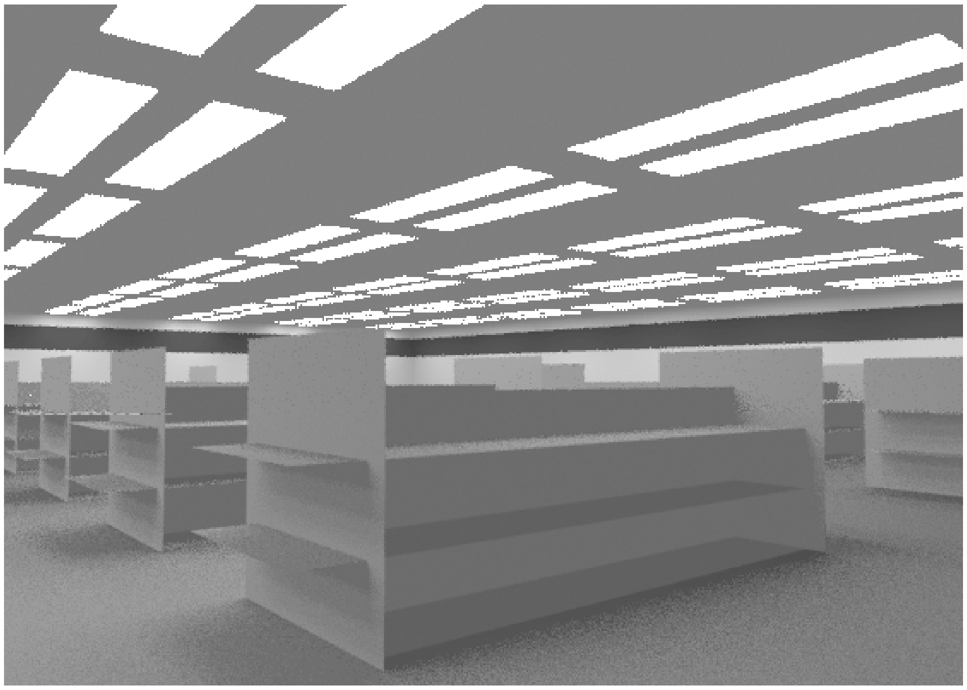


Figure 4.1: A demonstration of the linear method.

entire luminaire is below the tangent plane at x, then the estimate for $L_{direct}^{[i]}$ should be zero. An easy mistake to make (which we made), is to set $L_{direct}^{[i]}$ to zero if the center of the luminaire is below the horizon. This will make p_i take the one value that is not allowed: zero. Such a bug will become obvious in pictures of spheres illuminated by luminaires that subtend large solid angles as in Figure 4.2, but in many scenes such errors are not noticeable (the figures in [61] had this bug, but it was not noticeable).

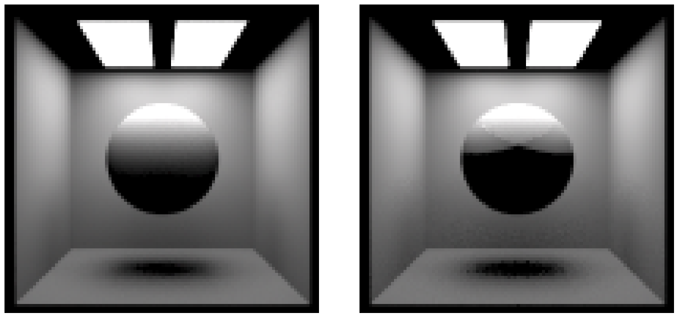


Figure 4.2: Demonstration of the artifact caused by incorrect p_i. Left: correct image. Right: wrong image with discontinuous shading transition due to $p_i = 0$ when $L_{direct}^{[i]} > 0$.

4.4 Spatial Subdivision Method

In the linear method, choosing a p.d.f. based on estimated contribution requires querying every luminaire in the scene. This is acceptable for many scenes, but if N is large (thousands or millions), even that might be too slow. In such scenes at most a few hundred luminaires (but usually only tens of them) will contribute significantly to the radiance at any particular point.

A possible way to speed up the linear method, while still maintaining a good p.d.f. is to estimate the contribution from unimportant luminaires coarsely. Suppose the luminaires are already divided into two groups according to their potential contribution. Then the estimated direct lighting from the more important group can be accurately approximated by the linear method and that from the other by the constant method with a few queries to estimate the average brightness of the group. Coarse estimation of the less important group is sufficient because this group has little to contribute and the estimation error it can introduce is limited.

To decide which luminaires are important for a particular x is a difficult task. As pointed out by Kok and Jansen [39], a luminaire that is more important to the color of x than most other luminaires is likely to be more important to the neighboring points of x. This implies we can use a spatial subdivision scheme to precompute a group of more important luminaires for each spatial cell in the spatial subdivision structure. For a particular cell a luminaire is put in the more important group if it might contribute more than a threshold average spectral radiance to a diffuse surface within the cell. A conservative way to determine whether the maximum potential contribution of a luminaire is above the threshold is to evaluate Equation 3.1 or Equation 3.2 with g, and $\cos\theta$ being set to one and f_r to f_{max}, which equals to $\frac{\rho_{max}}{\pi}$ in a diffuse environment where ρ_{max} is the maximum reflectance among all objects and $0 \leq \rho_{max} \leq 1$. If f_{max} is much above one as on a specular surface, a more involved strategy would be needed. Figure 4.3 illustrates the basic idea with a set of luminaires and their regions of importance. The stratification of the scene into boxes is arbitrary.

An easy way to choose the subdivision cells is simply to use the leaf cells of the conventional subdivision structure (e.g. the octree leaves of a Glassner style octree used for ray intersection acceleration [22]), and maintain a separate important group at each leaf. This is a finer than needed subdivision for scenes where the number of objects is much greater than the number of luminaires. An advantage of this is that no luminaire lists need to be constructed for empty cells, and that the characteristics of reflective objects can be used to construct the lists. Instead, we have implemented a separate *light octree* that

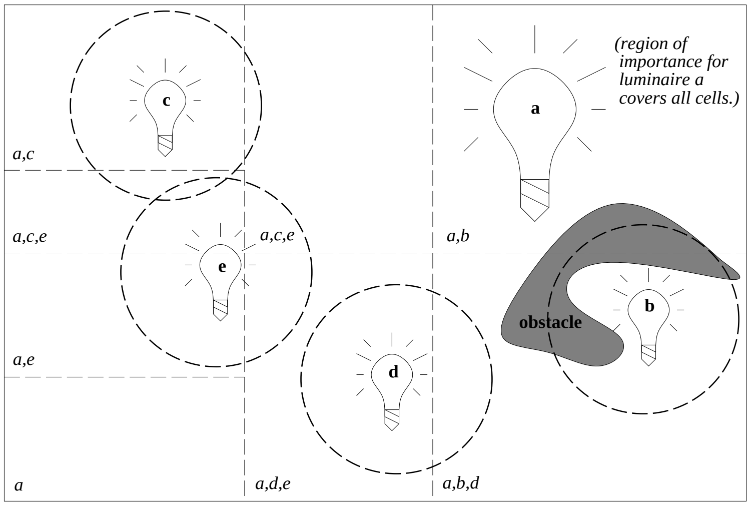


Figure 4.3: Subdivision of space. A luminaire is a member if it has the potential to contribute to a region.

recursively subdivides itself until each leaf is at a maximum allowed depth or when the size of the important group for that cell is below a specified limit. The depth and size limits are similar to those in a conventional octree, and their values are even less well understood so far. To avoid excessive subdivision, we check to see if the minimum contribution of an important luminaire to a cell is above the threshold. If all important luminaires are thus determined to be in the more important group for any possible descendant of that cell, we do not subdivide.

Another difficulty in building the light octree is what to use for the average radiance threshold. For our program, we assume that we know what spectral radiance distribution, L_{white}, would map to white on the display device. We set the radiance threshold to be some fraction of L_{white}[2]. Because our display device has eight bits per channel, we usually make

[2] L_{white} should be chosen according to a perceptual viewer model, such as the model implemented by Tumblin and Rushmeier [73, 7]. Such models will become increasingly important as physically based rendering

the threshold a few percent of L_{white}. Such a threshold will ensure that any luminaire that can change the pixel color more than a few intensity steps will be included in important group. If L_{white} is chosen correctly, then the sum of contributions of luminaires should be no more than L_{white}, and thus the number of important luminaires should be less than one hundred for most scenes.

To characterize important versus unimportant luminaires, we put a intensity bounding box around each luminaire and compare the box against the cell; if the box overlaps with the cell, then the luminaire is treated as an important luminaire. To avoid an important luminaire being overlooked, the intensity bounding box must contains the isosurface defined by $\int f_r L \cos\vartheta d\omega' = T$ where T is a predefined threshold. In the absence of any knowledge of the material and viewing direction, we must assume $f_r = f_{max}$, $\cos\vartheta = 1$, and that the whole luminaire is visible as in the worst case. f_{max} is $\frac{\rho_{max}}{\pi}$ in a diffuse environment where ρ_{max} is the maximum reflectance in the environment. In a non-diffuse environment, f_{max} is the maximum f_r whose direct lighting is computed separatly from indirect lighting [66].

Figure 4.4 shows four pictures of a pair of sphereflakes. Each sphereflake is composed of 7381 spheres. In the three images with 7381 luminaires, the light octree was used. As expected, the one sample case has a large variance, and the forty sample case is fairly smooth. Notice that both of the 10 sample cases (with 1 and 7381 luminaires) have visible noise. This implies that both cosine and visibility errors cause noise in the 7381 luminaire case. The 7381 luminaire figure with ten samples took less than twice as long as the one

becomes more popular.

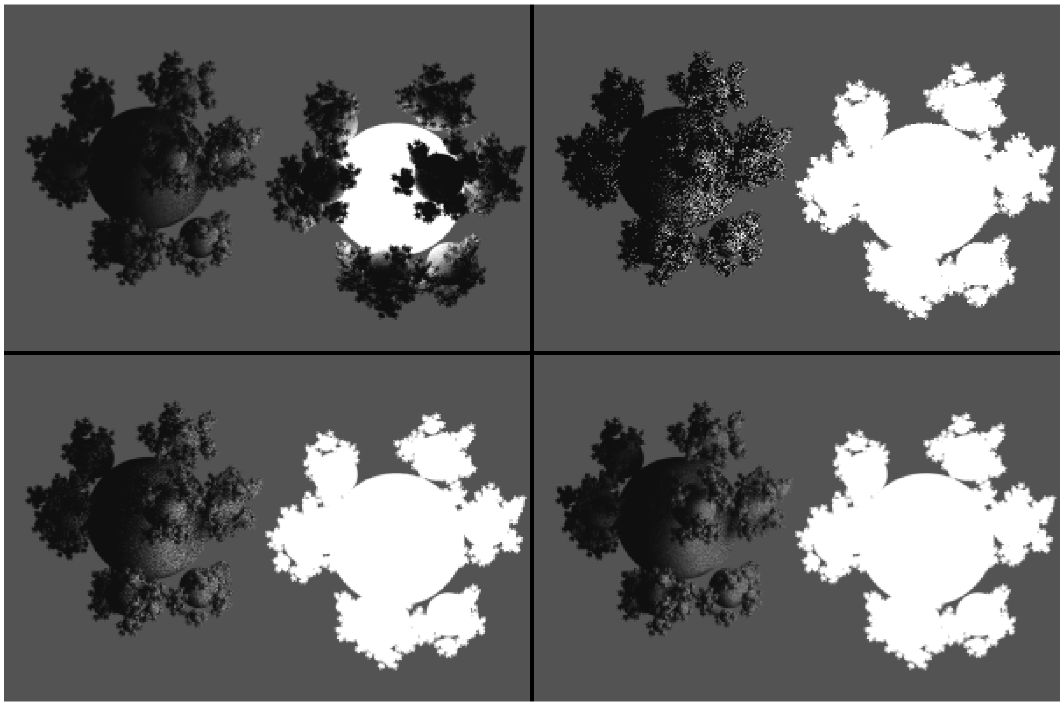


Figure 4.4: Top left: One luminaire, 10 samples per pixel. Top right: 7381 luminaires, 1 sample. Bottom left: 7381 luminaires, 10 samples. Bottom right: 7381 luminaires, 40 samples.

luminaire case with ten samples. This means the overhead of choosing the ray using the light octree does not swamp the cost of sending the shadow ray, even in this extreme case where half of the objects are luminaires. In less pathological scenes, the light octree should be smaller relative to the number of primitives, and performance should get even better.

4.4.1 Intensity Bounding Box Construction for Spherical Luminaires

The isosurface $f_{max} \int L d\omega' = T$ from a perfectly diffuse spherical luminaire with radius r is a sphere with radius d as in Figure 4.5. d satisfies $f_{max} L 2\pi (1 - \sqrt{\frac{d^2 - r^2}{d}}) = T$ and

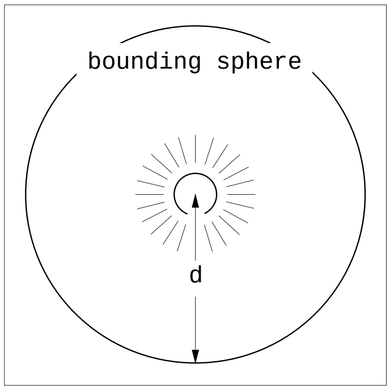


Figure 4.5: Intensity bounding box of a spherical luminaire.

thus, $d = \sqrt{\dfrac{r^2}{1-(1-\frac{T}{2\pi L f_{max}})^2}}$.

A perfectly diffuse luminaire has a constant radiance L (so radiant intensity $L\cos\theta'$) in a direction ϕ', where θ' is the angle between the surface normal of the luminaire and ϕ'. For a non-diffuse luminaire, we have the radiant intensity I, and from which we can derive a perfectly diffuse luminaire whose radiant intensity is larger or equal to I, i.e. $I(\phi') \leq L\cos\theta'$. Since I is smooth, continuous, and often symmetric, L can be approximated closely by the maximum of $\dfrac{I(\phi')}{\cos\theta'}$ at a small number of sample directions. This approach can effective approximate the incandescent lamp, because which is generally not very directional.

4.4.2 Intensity Bounding Box Construction for Planar Luminaires

A planar luminaire can be diffuse like a typical flourescent ceiling light. Suppose $\dfrac{\cos\theta'}{\|x_i' - x\|^2}$ in Equation 3.2 is a constant, which is a reasonable assumption because the

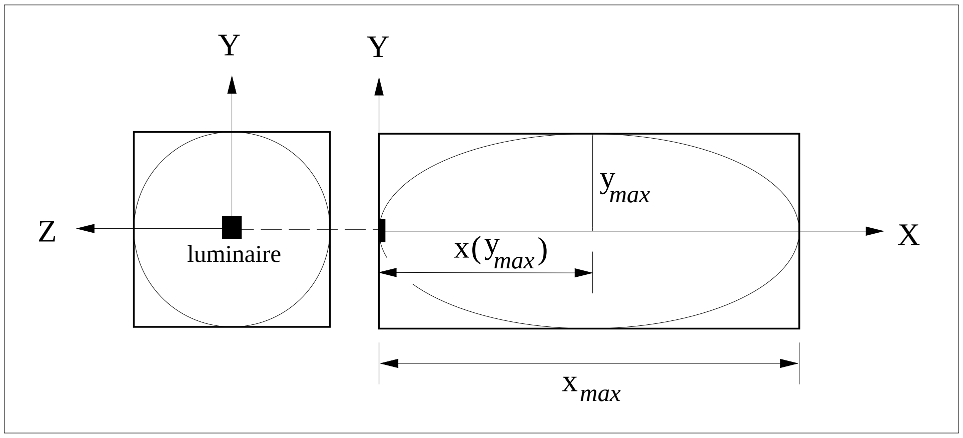


Figure 4.6: Intensity bounding box of a planar luminaire which is centered at the origin and facing X axis.

intensity bounding box is usually much larger than the luminaire, then the isosurface $f_{max}L\int d\omega' = T$ becomes $f_{max}L\cos\theta' A = Td^2$, where A is the area of the luminaire and $d^2 = \|x_i' - x\|^2$. Suppose the luminaire is centered at the origin and faces the positive X axis as in Figure 4.6. We can construct a intensity bounding box defined by $0 \le x \le x_{max}$ and $-y_{max} \le y, z \le y_{max}$ that contains the isosurface.

Since $d = \sqrt{\frac{f_{max}LA}{T}\cos\theta'}$, x_{max} occurs when $\theta' = 0$, and $x_{max} = \sqrt{\frac{f_{max}LA}{T}}$. y_{max} occurs when $d\sin\theta'$ is maximum or

$$\frac{d}{d\theta'}\sqrt{\frac{f_{max}LA}{T}\cos\theta'}\sin\theta' = 0$$

$$\sqrt{\frac{f_{max}LA}{T}}(-\tfrac{1}{2}\cos^{-\frac{1}{2}}\theta'\sin^2\theta' + \cos^{\frac{3}{2}}\theta') = 0$$

$$\theta' = \tan^{-1}\sqrt{2}$$

So, $\theta' = 0.304$ and $y_{max} = 0.299\sqrt{\frac{0.954 f_{max}LA}{T}}$. If the luminaire is close to perfectly diffuse,

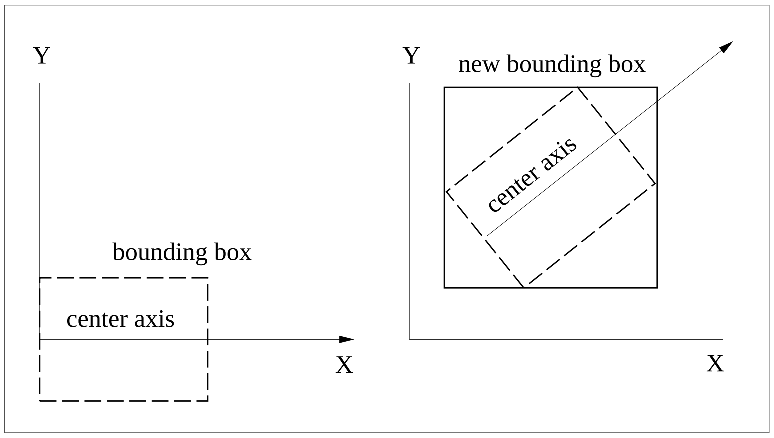


Figure 4.7: Creating a new intensity bounding box after rotating and translating the original intensity bounding box.

then L can be approximated by the maximum of $\dfrac{I(\phi')}{\cos\theta'}$ at some sample directions.

If the luminaire is not at the desired orientation, we rotate the eight corners of the intensity bounding box such that the imaginary luminaire agrees with the real luminaire and then create the actual intensity bounding box aligning with the coordinate system from the rotated corners as in Figure 4.7.

A planar luminaire can also be directional like a car headlight. In this case we assume that they are Phong luminaires, i.e. $I = k\cos^N\vartheta'$ (so $L = k\cos^{N-1}\vartheta'$), where k is a constant. Again assume that the luminaire is small relative to the intensity bounding box and the isosurface is defined by $d = \sqrt{\dfrac{f_{max}A}{T}k\cos^N\theta'}$. Then $x_{max} = \sqrt{\dfrac{f_{max}Ak}{T}}$ occurs when $\theta' = 0$ and after the similar derivation as in the diffuse case, y_{max} occurs as $\theta' = \tan^{-1}\sqrt{\dfrac{2}{N}}$.

4.4.3 Intensity Bounding Box Construction for Cylindrical Luminaires

Suppose that the cylinder is defined by $-y_c \leq y \leq y_c$ and radius r. Approximate the cylinder by its cross section at $x = 0$. Then x_{max} and y_{max} have the same formulas as for the planar luminaire in the previous section, and $z_{max} = x_{max}$ because of the symmetry.

4.5 Summary

The basic rationale behind the direct lighting calculation methods in this chapter is that direct lighting should not be calculated to a higher accuracy than necessary. This is very similar in concept to Kajiya's argument that we should not expend much work for deep parts of the ray tree [36]. It is not always true that one shadow ray per viewing ray is optimal. For example, the 100 luminaire case, one shadow ray is better than 100 shadow rays, but two or three might be better still. This issue requires further investigation.

Equation 4.1 represents L_{direct} with one summation, but this is not necessarily the optimal solution. If luminaires are grouped together, it is desirable that they have visibility coherence because the result from a shadow ray represents all of them. Since the visibility is too expensive to precompute, the best we can probably do is to make the maximum possible error in the estimation of a luminaire proportional to its potential contribution. In the Section 4.3, L_{direct} is divided into two integrals, one for important luminaires and, the other for unimportant luminaires - to avoid the large estimation error caused by the unimportant luminaires.

We have presented techniques for constructing probability spaces on luminaire surfaces for direct lighting calculations. We have found these techniques very useful for complicated scenes. The major limitation of the work is that it does not take into account the geometry term or the BRDF. Extending the methods to include these terms is possible, but is not easy.

The linear and the light octree methods are useful only for scenes with a large number of luminaires. Such scenes are becoming increasingly important. In outdoor scenes, especially in urban settings, scenes with thousands and even hundreds of thousands of luminaires are commonplace. In opera and theater applications, hundreds or thousands of luminaires are typical [69]. In infrared scenes, almost all surfaces are luminaires, so something such as the light octree is crucial. These methods also make it easy to use luminaires defined by many polygons or parametric patches. No matter how many patches define a light bulb surface, it will receive only one shadow ray.

The objects which have strong reflected light are suggested to be incorporated with luminaires into direct lighting computation because they can also cause sharp radiance change on other objects [39, 6]. The implementation is generally achieved by first categorizing bright emitting luminaires and reflected objects as light sources, after the zonal methods has computed indirect lighting. Then in the viewing phase, the direct lighting from the light sources are computed. The methods presented in this chapter are very helpful for this approach because the number of light sources can easily go up to the range of hundreds or thousands.

5

Sample Point Generation Strategies

Distribution ray tracing is formalized as a high dimensional integral and solved by quasi-Monte Carlo methods as discussed in Chapter 2. The number of quasi-random samples required in distribution ray tracing to reduce the error to a visually acceptable level may range from about ten to a few thousands per pixel depending on the complexity of the image. In this chapter we discuss the sampling issues which are closely related to the efficiency of quasi-Monte Carlo methods. These cover some applications of integration theory, existing sampling methods, desired sample set properties, a new sampling scheme which can generate samples equidistributed in any subdomain, and a supersampling framework which accounts for the complexity of the pixel, the limited dynamic range of the display medium, and human perception.

5.1 Analysis of Sample Behavior

The effectiveness of a set of samples in estimating a pixel color depends on the spatial distribution of the samples. For example, a pattern with samples that had several "clumps" might provide a poor estimate because the clumps happen to occur in regions with values far from the expected value. Since distribution ray tracing is computationally expensive, reducing the number of samples needed for a target accuracy is desirable. Tools from statistics, signal processing, and integration theory have contributed to the understanding of what properties a set of sample would have. People in mathematics treat all the quasi-random sample methods and their convergence as in the field of number theory. While each of these tools has its unique value in adding to our understanding of sampling, none of them makes this understanding complete.

5.1.1 Statistics and Sampling

The Monte Carlo methods in Section 2.2.2 transform an integral into a statistical expected value, which is then estimated using a finite number of random samples. Quasi-Monte Carlo methods use the same formula as Monte Carlo methods but with quasi-random samples. The behavior of Monte Carlo methods are well understood from statistics, but statistics can not be straightforwardly applied to quasi-Monte Carlo methods because the quasi-random samples are not random variables.

5.1.2 Signal Processing and Sampling

Estimating the pixel color via quadrature formulas can also be realized as the reconstruction of an image signal via sampling. Theoretically, a continuous signal with band-limited frequency can be properly reconstructed with a sample rate greater than twice the highest frequency component in its spectrum [23]. In computer graphics, edge and texture can make the pixel color a discrete continuous signal with unbounded frequency. The nature of high frequency in image synthesis explains the occurrence of aliasing artifacts, such as jaggies and Moire patterns [21, 23]. Signal processing also predicts that the aliasing can be reduced by pushing the error into high frequencies as being used by stochastic samples [15]. However, signal processing does not help us predict the absolute quality of estimation to be expected from a given sampling pattern.

5.1.3 Discrepancy

Integration theory suggests that the convergence rate from a set of samples is closely related to the discrepancy [71, 62, 19], which is a measure of equidistribution. Ideal equidistribution means that the fraction of the samples in each subdomain is equal to the fractional measure of that subdomain. With a finite number of samples, ideal equidistribution is impossible.

Quadrature formula shows that an integral over a domain equals to the integral over the sum of all the subdomains, i.e. $\int_{\Omega} f(x)dx = \sum_{i=1}^{n} \int_{\Omega_i} f(x)dx$. If all samples have the same weight, then samples are equidistributed if there exist a subdivision such that all Ω_i are compact and close to $\frac{1}{n}$. Otherwise, the samples are not equidistributed.

On the unit square, two widely used definitions of discrepancy [71, 19] are:

1. L^{∞}-discrepancy $D_N =$ maximum of $|\frac{n}{N} - xy|$ for $\forall (x, y) \in [0, 1]^2$
2. L^2-discrepancy $T_N = \sqrt{\int_0^1 \int_0^1 |\frac{n}{N} - xy|^2 dxdy}$ for $\forall (x, y) \in [0, 1]^2$

where N is the total number of samples and n is the number of samples inside the region $[0, x] \times [0, y]$. The term $|\frac{n}{N} - xy|$ is called the local discrepancy at (x, y). The discrepancy, which is within $[0, 1]$, becomes smaller as the samples are more equidistributed. The axis-aligned rectangle in local discrepancy test has been extended to disks [46] and random-edges [19] in an effort to make discrepancy more valuable for graphics application. The local discrepancy in random-edge discrepancy is the difference between the area of a unit square above a random line and the fraction of sample points above the same line.

The estimation error from a set of samples has been shown to be bounded by a linear combination of the discrepancy and the variation (defined below) of the estimated function [17, 71, 1]. This suggests that discrepancy might be used to rate the quality of sample sets. In one dimension, we have the following theorem:

Theorem 1 (Koksma) *If f is a function of bounded (finite) variation $V(f)$ on the unit* *interval I and $x_1, \cdots, x_N$ are points in I with L^∞-discrepancy D_N, then*

$$|\frac{1}{N}\sum_{i=1}^{N} f(x_i) - \int_0^1 f(t)dt| \leq V(f)D_N \tag{5.1}$$

where the variation $V(f)$ is the least upper bound of $\sum_{i=1}^{N} |f(\xi_i) - f(\xi_{i-1})|$ for any partition $\xi_0, \xi_1, \cdots, \xi_N$ such that $0 = \xi_0 < \xi_1 < \cdots < \xi_N = 1$. Basically, the variation $V(f)$ is the maximum total (absolute) change of a function after being discretized into $N + 2$ different points, including the end points.

In two dimensions, we have the following theorem:

Theorem 2 (Zaremba) *If f is a function of two dimensional bounded variation $V_{X,Y}(f)$ and of one dimensional bounded variations $V_X(f_{y=1})$ and $V_Y(f_{x=1})$ on the unit square I, and $(x_1, y_1), \cdots, (x_N, y_N)$ are points in I with L^∞-discrepancy D_N, then*

$$|\frac{1}{N}\sum_{i=1}^{N} f(x_i, y_i) - \int_0^1 \int_0^1 f(u,v)dudv| \ \leq \ V_{X,Y}(f)D_{N,X,Y} + V_X(f_{y=1})D_{N,X} + V_Y(f_{x=1})D_{N,Y} \tag{5.2}$$

where $D_{N,X,Y}$ is D_N in XY domain. $D_{N,X}$ and $D_{N,Y}$ are D_N in X and Y, respectively. $V_{X,Y}(f)$ is the least upper bound of $\sum_{i=1}^{N}\sum_{j=1}^{M} |f(\xi_i, \eta_i) - f(\xi_{i-1}, \eta_i) - f(\xi_i, \eta_{i-1}) + f(\xi_{i-1}, \eta_{i-1})|$ among all possible partitions $0 = \xi_0 < \xi_1 < \cdots < \xi_N = 1$ and $0 = \eta_0 < \eta_1 < \cdots < \eta_N = 1$.

Similar formulas for both one dimension and two dimensions can also be established with T_N [28].

Theorem 2 cannot be directly applied to predict error in image synthesis because the image function in computer graphics contains edges and thus has unbounded variation $V_{X,Y}$ in Equation 5.2. However, the discrepancy might still be helpful in predicting the convergence behavior from a sample pattern [62]. Discrepancy gives an objective measure of the ability of a sample set to calculate images with features similar to rectangles. This gives motivation for Mitchell's introduction of edge based discrepancy [46].

5.2 Sampling Methods

A sampling method should generate samples that estimate an integral with as small an error as possible. Sampling methods from integration theory have developed different numerical rules to generate sample sets of extremely low discrepancy [83, 89, 1], but these rules create deterministic sample sets which are prone to aliasing artifacts when used in image synthesis, because each pixel is sampled using the same pattern. On the other hand, methods from computer graphics generate stochastic samples with higher discrepancy.

Theorem 2 shows that $1D$ discrepancy contributes to the upper bound of the estimation error in two dimensions. This implies that perhaps a good $2D$ sample pattern should also be effective in one dimension, and helps to explain the behavior of N-rooks samples as discussed in Section 5.2.2. A natural extension to higher dimension of the same idea suggests the importance of the low discrepancy in any lower dimensional subset of the domain and motivates us to the new multi-jittered sampling methods described in Section 5.2.3.

5.2.1 Methods from Integration Theory

Three representative sampling methods from integration theory are shown here. Since the methods from integration theory are meant to approximate a function once, these methods give only one sample pattern for a given number of samples.

Fibonacci Number Sequence : The Fibonacci sequence is defined as $F_1 = 1$, $F_2 = 1$, and $F_m = F_{m-1} + F_{m-2}$ for $m \geq 3$. In a square domain, the kth sample is

$$(x, y) = (\frac{k-1}{N}, \{\frac{(k-1)F_{m-1}}{N}\})$$

where N is the total number of samples, $1 \leq k \leq N$, and $\{s\}$ is the fractional part of s.

Irrational Number Sequence : Suppose $\theta_1, \cdots, \theta_d$ are d irrational numbers that are linearly independent over the rational numbers, i.e. $\alpha_1\theta_1 + \cdots + \alpha_d\theta_d \neq 0$ for any rational numbers α_1,$\cdots$,α_d. The kth sample over the hypercube $0 \leq x_i \leq 1$, for all $i \in [1, d]$ is

$$(x_1, \cdots, x_d) = (\{k\theta_1\}, \cdots, \{k\theta_d\})$$

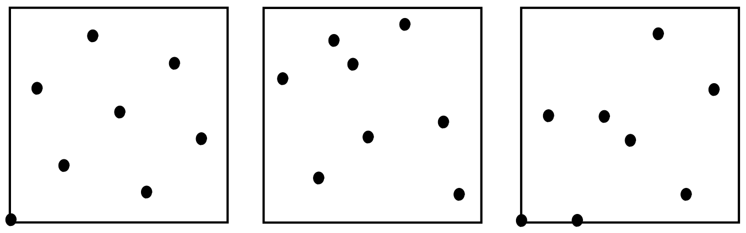


Figure 5.1: From left to right : Fibonacci number sequence, irrational number sequence, Zaremba-Hammersley sequence.

$\theta_1 = \sqrt{2}$ and $\theta_2 = \sqrt{3}$ are one possible choice in a square domain.

Zaremba-Hammersley sequence : The kth sample over the hypercube $0 \leq x_i \leq 1$, for all $i \in [1, d]$ is

$$(x_1, \cdots, x_d) = (\frac{k-1}{N}, \psi_{P_1}(k-1), \cdots, \psi_{P_{d-1}}(k-1))$$

where P_i is the ith prime, e.g. $P_1 = 2$ and $P_2 = 3$. If the representation of m base r is $m = a_n r^n + \cdots + a_i r^i + \cdots + a_0$ where $a_n \neq 0$, then

$$\psi_r(m) = (a_0 \bmod r)\frac{1}{r} + \cdots + ((a_i + i) \bmod r)\frac{1}{r^{i+1}} + \cdots + ((a_n + n) \bmod r)\frac{1}{r^{n+1}}$$

The numerical experiments have shown that the Zaremba-Hammersley sequence has the lowest L^2-discrepancy T_N among many commonly suggested methods from integration theory, including the other two methods in this Section [83, 89, 1]. The sample patterns with 9 samples from each of these sampling methods are in Figure 5.1.

The major obstacle when repeatedly applying these methods is the aliasing artifact

Number of Samples	*5*	*13*	*55*	*144*	*610*	*1597*
Fibonacci Sequence	0.0701	0.0273	0.0052	0.0008	0.0006	0.0002
FS After Operations	0.1121	0.0650	0.0484	0.0352	0.0432	0.0321

Table 5.1: Average L^2-discrepancy from Fibonacci Sequence with and without geometry operations from 100 trials.

caused by the deterministic samples. In order to eliminate aliasing, we introduced randomness into the existing patterns via simple geometry operations such as shifting, rotation, and reflection. Ideally, these operations preserve low discrepancy from maintaining the relative position among most samples. Unfortunately, experiments with the Fibonacci number sequence shows that although the aliasing is removed, the discrepancy increases many times, as in Table 5.2.1.

It is also possible to randomly and repeatedly use a set of sample patterns which have low discrepancy values, but to arrange these patterns in order to decrease the number of patterns required rely on further research.

5.2.2 Previous Methods in Rectangular Domains

Sampling methods for square or rectangular domains, e.g. a pixel or a rectangular luminaire, have been heavily studied in computer graphics [15, 44, 60, 8]. The most widely used strategies are random sampling, regular sampling, jittered sampling, Poisson disk sampling and N-rooks sampling as illustrated in Figure 5.2.

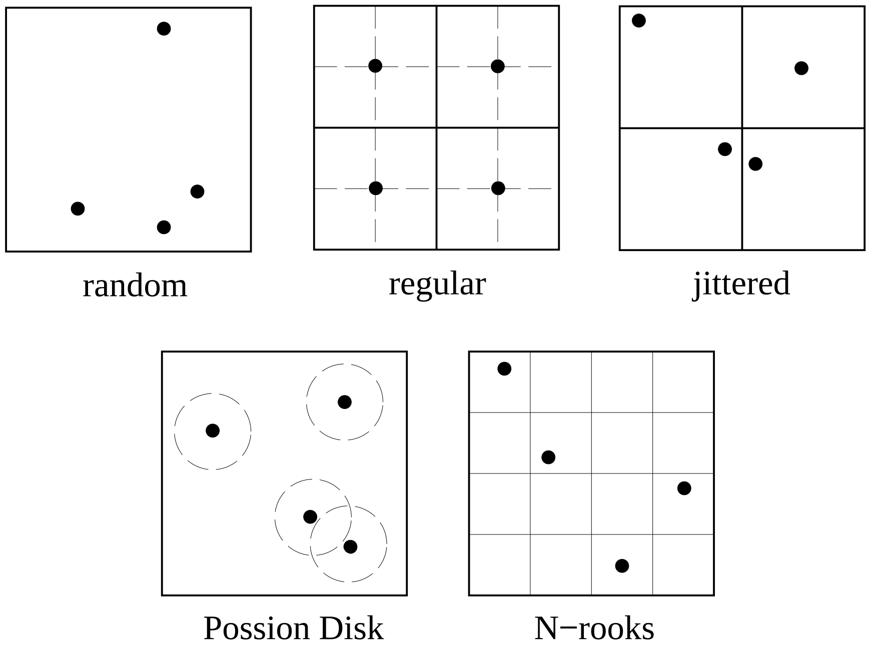


Figure 5.2: Random samples, regular samples, jittered samples, Poisson disk samples, and N-rooks samples with 4 samples.

The random sampling method randomly chooses a sample within the domain. Its simplicity makes it theoretically interesting, but such samples have high variance because of the possibility of random samples to clump together.

The regular sampling method places samples evenly in a regular grid pattern. The regular samples are equidistributed in $2D$, but its regularity can cause aliasing artifacts.

Jittered sampling divides the domain into a set of equal-size cells, and a point is randomly chosen from each jittered cell [14]. The jittered samples are equidistributed in two dimensions. but are not much better than random samples in one dimension projection.

The Poisson disk sampling method places samples apart by, at least, a predefined distance r [15, 18]. To generate Poisson disk samples, we can sequentially create random samples, and reject a sample if its distance to any previous sample is within r. The Poisson disk samples are designed to avoid clumping, which will improve equidistribution in two dimensions.

The N-rooks sampling method randomly associates rows and columns such that each row and each column have exactly one sample [62]. The N-rooks samples are evenly distributed in both single dimensions separately.

5.2.3 Multi-jittered Sampling Methods in Rectangular Domains

Multi-jittered sampling methods attempt to minimize the discrepancy in every subdomain. In a square or a rectangular domain, the multi-jittered samples have one sample in each bin (like N-rooks samples) and have one sample in each jittered cell (like jittered samples). An example with 4 multi-jittered samples is shown in Figure 5.3. The multi-jittered samples can be generated using either of two schemes. One constrains jittered samples to be N-rooks, and the other constrains N-rooks samples to be jittered.

5.2.3.1 Jittered Oriented Method. This method is capable of creating N by M multi-jittered samples on a rectangular domain incrementally. It first divides the X domain into N cells, each with M bins. Similarly it divides the Y domain into M cells, each with N

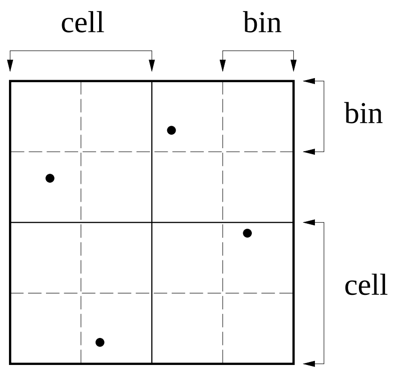


Figure 5.3: Multi-jittered samples.

bins. This subdivision creates N by M cells and each cell has M by N bins. Sequentially one sample is placed in each cell as in jittered sampling such that each bin on X, and each bin on Y have exactly one sample.

A faster way to generate multi-jittered samples is to first permute bins in each cell, and then for the (i, j) cell, we put the sample at the jth permuted bin inside the ith cell on X, and the ith permuted bin inside the jth cell on Y [8].

5.2.3.2 N-rooks Oriented Method. To generate N N-rooks samples, we divide the domain into N by N cells. Then we independently permute the list $(1, 2, \cdots, N)$ twice into L_1 and L_2. The ith N-rooks sample is a random point inside the cell $(L_1[i], L_2[i])$, where $L_j[i]$ is the ith element in the list L_j. If the order of the samples doesn't need to be stochastic, then the ith cell can be simplified to $(i, L_1[i])$.

Multi-jittered samples are N-rooks samples with an extra constraint, i.e. a different permutation scheme that makes them jittered. Suppose that NM samples are needed. We

Number of Samples	*16*	*64*	*100*	*400*	*1600*
random samples	0.0828	0.0433	0.0361	0.0183	0.0086
regular samples	0.0578	0.0256	0.0234	0.0114	0.0051
N-rooks samples	0.0498	0.0201	0.0162	0.0083	0.0043
jittered samples	0.0497	0.0186	0.0127	0.0046	0.0016
multi-jittered samples	0.0380	0.0146	0.0083	0.0030	0.0011

Table 5.2: $2D$ L^2-discrepancy from the average of 100 trials. Each discrepancy needs $10,000$ local discrepancy computations.

Number of Samples	*16*	*64*	*100*	*400*	*1600*
random samples	0.0863	0.0439	0.03470	0.01739	0.00886
regular samples	0.0301	0.0106	0.00766	0.00247	0.00055
N-rooks samples	0.0490	0.0237	0.01943	0.00946	0.00467
jittered samples	0.0445	0.0160	0.01144	0.00404	0.00142
multi-jittered samples	0.0394	0.0146	0.01056	0.00384	0.00138

Table 5.3: $2D$ L^2 edge discrepancy from the average of 100 trials. Each discrepancy needs $10,000$ local discrepancy computations.

first divide $1,\ldots,NM$ into N lists, $(1,\ldots,M)$, $\cdots$, $(i+1,\ldots,i+M)$, $\cdots$,$((N-1)M+1,\ldots,NM)$. Choose one number from each list, permute them, and concatenate the new list to the end of the list L. Continue the procedure until all numbers are exhausted. Then the ith multi-jittered sample is in the cell $(i, L[i])$.

The average L^2-discrepancy values of 100 trials from various sample patterns are shown in Table 5.2. Experiments show that the discrepancy has approximately one percent error if the local discrepancy is computed at the $10,000$ evenly spaced positions, and that this number decreases as the number of samples increases.

The results in Table 5.2 and in Table 5.3 confirm that multi-jittered samples have

Sampling Methods	random	N-rooks	jittered	multi-jittered
L^2-discrepancy	0.0539/0.203	0.0333/0.0778	0.0375/0.0674	0.0318/0.0410
L^2 edge discrepancy	0.0526/0.146	0.0378/0.0741	0.0384/0.0544	0.0352/0.0435

Table 5.4: The minimum and maximum discrepancy values for 16 samples.

low L^2-discrepancy and L^2 edge discrepancy. The instability of discrepancy from regular samples in these experiments can help to explain that although deterministic samples can be equidistributed and are effective in approximating an image function, the aliasing artifact can happen when the image function is correlated to the distribution of the samples.

Every entry in Table 5.2 and in Table 5.3 is an average of 100 trials. The values of individual trials can vary. For example, in the case of 16 multi-jittered samples, the average of L^2-discrepancy is 0.038 but the actual values are in the range $[0.0318, 0.0410]$. The ranges from stochastic sampling methods in this section are in Table 5.4. A smaller range means a sampling method is more stable, and this stability from the worst to the best is random, N-rooks, jittered, multi-jittered.

In a square domain, the multi-jittered samples, like jittered samples, are most effective when the number of samples is a perfect square, N^2. In the case where the number of samples x satisfies $(N-1)^2 < x < N^2$, we can randomly choose x samples from a pattern of N^2 samples. However, this can leave a hole in a sample pattern and produce poorly distributed samples. Instead we decompose x into $n_1^2 + \cdots + n_m^2$ and use a combination of patterns with n_1^2, $\cdots$, n_m^2 samples. To compute n_1, $\cdots$, n_m, we first find the largest number n_1 satisfying $n_1^2 < x$, i.e. $n_1 = \lfloor\sqrt{x}\rfloor$, then the largest number n_2 satisfying $n_1^2 + n_2^2 < x$, i.e. $n_2 = \lfloor\sqrt{x - n_1^2}\rfloor$, and continue the process until $n_1^2 + \cdots + n_m^2 = x$.

5.2.4 Methods for Other Two Dimensional Domains

Domains other than the unit square appear frequently in distribution ray tracing. For example, the circular lens and the spherical luminaire both need to be sampled. Simple rejection methods can generate samples on these domains, but these can be inefficient because some samples are wasted, and it is difficult to keep samples stratified.

An alternative method is to generate equidistributed samples on a unit square, and then stretch the unit square to the desired shape/domain. This is sometimes called warping [86, 64] and does not have the drawback of the rejection method. However, a compact cell in the unit square might become a strip and the degree of equidistribution might not be conserved after warping, although new samples may still be uniform. Whether samples generated directly in the space they will be sampling can improve equidistribution properties is still a research issue.

5.2.5 Methods for Three Dimensional Domains

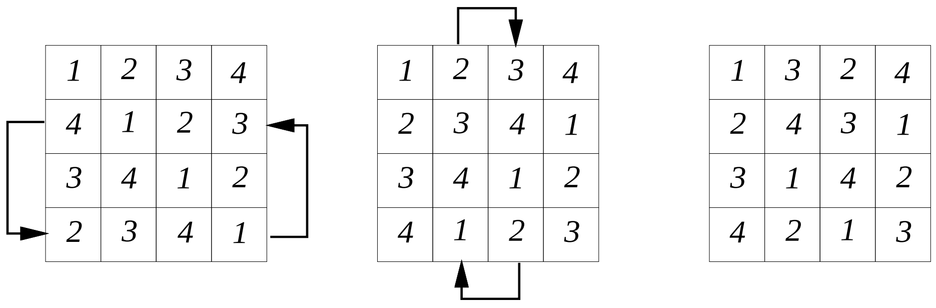
In a $3D$ domain such as $Pixel \times Time$, Cook associates jittered samples on $Pixel$ with jittered/stratified samples on $Time$ [14]. Mitchell uses a jittered pattern on $Pixel$ and then sequentially assign a time sample to each pixel sample such that the new time sample has (nearly) the maximum distance to all the existing time samples [45]. Note these two methods only guarantee equidistribution on $Pixel$ and on $Time$.

In $3D$, full multi-jittered samples would be jittered in $3D$ and projections in XY, YZ, XZ, X, Y, Z would also be jittered. Although full multi-jittered samples in three dimensions is probably desirable, we have been able to create only sample sets which are partially multi-jittered. Suppose that XYZ is the $3D$ domain of interest, for example, XY is $Pixel$ and Z is $Time$. Then the following method can generate the samples equidistributed on X, Y, Z, XY, XZ, YZ, but not equidistributed on XYZ. Suppose that N^2 samples are required.

1. Generate N^2 multi-jittered samples on XY. This gives samples equidistributed on X, Y, and XY.

2. Divide Z into N cells and each of which into N bins; place a sample in each bin and achieve the equidistribution on Z.

3. Create a N by N table with X and Y axes. The integer in each cell represents the corresponding Z cell. Initialize the table like the left most one in Figure 5.4. This gives samples equidistributed on XZ and YZ, because, for any given column or row, the Z cells are all different. To create this table we start with the first row and shift it toward the right hand side by one cell every time we move down a row.



1	2	3	4
4	1	2	3
3	4	1	2
2	3	4	1

1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

1	3	2	4
2	4	3	1
3	1	4	2
4	2	1	3

Figure 5.4: The initial table at left gives a set of nearly multi-jittered samples. The middle table is the initial table after switching two rows whereas the right table is the middle table after another switch.

Number of Samples	*16*	*256*	*1600*
random samples	0.0716	0.01753	0.00733
regular samples	0.0757	0.02546	0.01390
jittered samples	0.0603	0.01136	0.00320
N-rooks samples	0.0480	0.01117	0.00444
Cook samples	0.0480	0.01051	0.00390
Mitchell samples	0.0473	0.00721	0.00235
multi-jittered samples	0.0409	0.00630	0.00197

Table 5.5: $3D$ L^2-discrepancy from the average of 100 trials.

4. Permute columns and rows. Note that the equidistribution is invariant under permutation.

5. Assign a unique bin to the samples with the same Z cell.

The result in Table 5.5 shows that multi-jittered still has the lowest average $3D$ L^2 discrepancy, where $T_N = \sqrt{\int_0^1 \int_0^1 \int_0^1 |\frac{n}{N} - xyz|^2 dxdydz}$ for $\forall(x, y, z) \in [0, 1]^3$.

The N-rooks oriented approach can also be used to generate $3D$ multi-jittered samples. Suppose N samples are needed. We start with a the sequence $1, \cdots, N$ and make it into two lists L_1 and L_2, where L_1 is the same sequence being divided into groups of two numbers, i.e. $(1,2),\cdots,(i,i+1),\cdots$, and L_2 is the same sequence being divided into groups of three numbers, i.e. $(1,2,3),\cdots,(i,i+1,i+2),\cdots$. Choosing one number from each group in L_1, permute them, and concatenate the new list to the end of L'_1. Similarly we can get L'_2 from L_2 and the ith multi-jittered sample would be in the cell $(i, L'_1[i], L'_2[i])$. Note that 2 and 3 are chosen because they are prime numbers. This method can be easily extended to higher dimension, but it guarantees equidistribution only in one and two dimensions.

5.2.6 Samples in More Than Three Dimensions

In our implementation, the pixel color in Equation 2.4 is modified to:

$$\int_{Pixel}\int_{Lens}\int_{Time}(E(x,\psi) + L_{direct} + L_{indirect})\ dt\ dA_l\ dA_p$$

or in full expansion:

$$\begin{aligned}\int_{Pixel}\int_{Lens}\int_{Time}\Bigg[E(x,\psi)\ +\sum_{Luminaire}\int_{Direction}E(x,\psi')\ d\omega_{\psi'}+ \\ \int_{Direction}L(x,\psi'')\ d\omega_{\psi''}\Bigg]\ dt\ dA_l\ dA_p\end{aligned} \tag{5.3}$$

where ψ' is the direction to luminaire and ψ'' is the direction to the hemisphere above x centered at the normal vector at x. $\sum\limits_{Luminaire}$ means the direct lighting from multiple luminaires is the sum of the direct lighting from each of them. Several terms in the integrand share the same outer integrals, which means that computationally they share the same viewing ray.

The direct lighting computation needs $8D$ samples in $Pixel \times Lens \times Time \times Luminaire \times Direction$ whereas the indirect lighting computation needs $7D$ samples in $Pixel \times Lens \times Time \times Direction$. Traditionally, samples in different domains, $Pixel$, $Lens$, $Time$, and $Direction$, are chosen independently, but this approach does not guarantee that samples are equidistributed in any lower dimensional projection of the domain [14, 45].

Figure 5.5 shows three cases. Case 1 shows that the effectiveness in detecting a feature boundary perpendicular to X relies on the sample equidistribution on X; case 2 shows that the effectiveness in detecting a feature boundary parallel to X relies on the sample equidistribution on Y; and case 3 shows that effectiveness in detecting a feature boundary parallel to the diagonal relies on the sample equidistribution on XY. Here X and Y can also be thought of as a multi-dimensional projection of domain. The following list gives several practical examples to demonstrate this idea.

1. $X = Pixel$ and $Y = Lens$

   Case 1 happens when the object is in focus, case 2 happens when the size of the pixel is relative small compared to the focal length, and case 3 happens otherwise.

2. $X = Pixel \times Lens$ and $Y = Time$

   Case 1 happens when a pixel only covers stationary objects, and case 3 happens

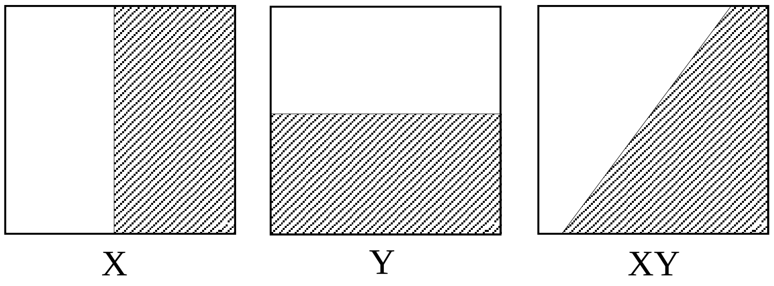


Figure 5.5: The projection of domain whose equidistribution is desired. Case 1 (left): equidistribution in X is important for a feature boundary perpendicular to X. Case 2(Middle): equidistribution in Y is important for a feature boundary parallel to X. Case 3 (Right): equidistribution in XY is important for a feature boundary not parallel to X or Y.

otherwise.

3. $X = Luminaire$ and $Y = Direction$

   Case 1 happens when luminaires cover a small solid angle, case 2 happens if there is only one luminaire, and case 3 happens otherwise.

4. $X = Luminaire$ and $Y = Time$

   Case 1 happens when the lighting condition remains unchanged, case 2 happens when there is only one small luminaire whose intensity changes over time, case 3 happens otherwise.

5. $X = Pixel \times Lens$ and $Y = Luminaire \times Direction$

   Case 1 happens when all luminaires cover small solid angle and all are blocked or unblocked, and case 3 happens otherwise.

In order to examine equidistribution in all projects of the domain, we choose samples from a five (emission) dimensional cube, a seven (indirect lighting) dimensional cube, and a eight (direct lighting) dimensional cube, which are then warped to the actual domains. Note that these three domains share the same five dimensional domain $Pixel \times Lens \times Time$. Ideally we want to generate multi-jittered samples in these high dimensional domains, but the number of constraints in multi-jittered sampling explode exponentially, i.e. the number of constraints is $2^n - 1$ in n dimensions. Instead we can only make sure some particular projections of subdomain have equidistribution by repeatedly applying the $2D$ and $3D$ multi-jittered sampling methods.

5.3 Adaptive Supersampling

Supersampling means taking more than one sample within a pixel, and adaptive supersampling means taking extra samples based on the result from previous samples. Adaptive supersampling methods can be with subdivision, which places extra samples only in regions with high variance [84, 36, 54, 23, 24], or without subdivision, which evenly distributes extra samples over the pixel [43, 18, 55, 35, 44, 38].

In subdivision supersampling, if the variation of the estimates in a region is over a predefined threshold, the region is further sampled. Subdivision supersampling methods are effective if the initial samples can reliably detect regions that require extra sampling. On the other hand, supersampling without subdivision uses the initial samples only to predict the overall pixel complexity, instead of subpixel complexity as in subdivision supersampling. The conservative approach makes non-subdivision supersampling the choice in rendering complex realistic scenes because the information delivered by a few samples is too sparse to decide where to concentrate samples in the $5-8$ dimensional space.

The prediction of the total number of samples relies on several factors, namely the pixel complexity, the sample convergence rate, the nonlinear response of human vision, and the limitation of the medium. All the existing methods are based on contrast or on the pixel complexity such as variance or the absolute estimation error [43, 55, 18, 44]. In this section, we discuss the existing non-adaptive supersampling methods and their weaknesses, and then present a supersampling framework which can take all these factors into account.

5.3.1 Existing Adaptive Supersampling Methods

Statistical methods have been used to control adaptive supersampling. A radiance returned from a viewing ray is an estimate of the pixel color. In statistics, the pixel color is the expected value of a random variable X with a unique distribution formed by all possible estimates within a pixel. If we use a sufficient number of viewing rays generated from random samples, then $L = \frac{1}{n}\sum_{k=1}^{n} x_k$ is a good approximation of the pixel color, and the standard derivation σ is a good approximation of the estimation error $(X - E[X])$ if X has a normal distribution, where $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - E[X])^2$ and x_i is an estimate of $E[X]$ [33].

Let $s_n^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - L)^2$. Then the Chi-square test [43]:

$$p(\frac{ns_n^2}{\sigma^2} < \chi_\beta^2(n-1)) = \beta$$

can be used to predict N, the number of samples needed in supersampling, from n initial samples. Similarly, the t-test [55] is suggested to compute N.

However, applying these statistical tests to supersampling has two weaknesses. The first weakness is that the fundamental assumption of a normal distribution of sample values is questionable. For example, if a pixel has only two constant colors, then X has a binomial distribution, which can be very different from normal distribution when n is small, say less than 30 [33]. The other weakness is that the quasi-random samples behave differently from random samples [46], and the validity of using the same formulas is questionable.

Signal processing suggests that $|X - E[X]| \propto \beta$, where β is the sampling rate, which changes with the number of samples as well as the sampling methods [18]. If we repeat the estimation process twice at different number of samples, say i and j, then we have two equations with two unknowns, $E[X]$ and C:

$$|\frac{1}{N}\sum_{k=1}^{i} X_k - E[X]| = C\beta_i$$
$$|\frac{1}{N}\sum_{k=1}^{j} X_k - E[X]| = C\beta_j$$

After solving for C, we can compute N from β_N by letting $C\beta_N = T$ where T is the maximum acceptable estimation error. A difficulty with this approach is selecting i and j such that they are small enough to be practical and large enough to give stable estimation.

Contrast is closely related to human perception and is the fundamental measure that other other visual behaviors are derived from. One common form of luminance contrast, $\frac{L_{max} - L_{min}}{L_{max} + L_{min}}$, where L_{max} and L_{min} are the maximum and minimum radiance respectively [35], is suggested to predict the number of samples, i.e. the number of samples taken is proportional to the contrast [44]. This method is plausible because it considers the nonlinear response of human perception; but it still has some weaknesses. The first problem is that the contrast is measured within a pixel, but human eyes see the contrast from surrounding pixels and the relationship between these two values is not clear. The second problem arises when two very different pixels have the same contrast, for example one has fine texture which may require hundreds of samples while the other has a simple edge which requires tens of samples.

5.3.2 A Complete Adaptive Supersampling Framework

The framework below is based on the observation about sampling covered earlier in this chapter.

1. Decide the number of initial samples, n, to be taken over each pixel.

2. Render the initial image with n initial samples.

3. Transform the initial image into the displayable image for the media of choice. If the media is a CRT, then a good transformation would be an extension combining Tumblin [73] and Chiu [7].

4. Decide the accuracy function $S(x, y)$, where $S(i, j)$ is the accuracy requirement at pixel (i, j). The accuracy function should be based on the transformed image and a perceptual model because its value is proportional to the sensitivity of human eyes to a displayed image.

5. Estimate the error caused by initial samples. This error can be approximated via different measures, for instance the standard derivation or the estimated variation of a function as defined in Theorem 2. The traditional estimation error is a function of $L_1,\cdots,L_n$. This should probably be extended to a function of all the available information such as the object, the texture, the normal vector, etc.

6. Compute the effectiveness of the initial samples. Discrepancy relates to the effectiveness.

7. Compute the number of samples required, N, which is a function of a predefined tolerance, the required accuracy, the estimation error, the effectiveness, the expected effectiveness of a sample pattern with N samples, and probably a predefined confidence index.

In this framework, there are many missing elements that require further research. We present here a new supersampling method based on our partial understanding of the missing elements.

1. The number of initial samples in existing supersampling methods assume 4 to 9 samples without justification [43, 36, 44, 23]. Instead, we assume that the initial samples are expected to detect the case of two features fully occupying a pixel when the projected area of either feature on the pixel is larger than 5% of the pixel size. Missing a feature with projected area less than 5% generally introduces little error. To verify if 5% is appropriate, we need to know the color difference ΔC of the two features and the error tolerance T, i.e. checking $5\% \leq T\Delta C$. If the boundary is an edge, then to successfully detect both features means the samples must fall on both sides of an random edge. This can be done by slightly modifying Mitchell's method for random edge discrepancy [46]. Instead of counting the number of samples below a line, we examine if all samples are below a line. Experiments show that to achieve 93% or 99% successful rate, we need at least 9 multi-jittered samples or 16 multi-jittered samples, respectively.

2. Render the initial image with 16 initial samples.

3. Transform the initial image by Chiu's method [7].

4. Set ϵ to be the maximum of $\dfrac{|L_{neighbor} - L|}{L_{neighbor} + L}$, where neighbor means an adjacent pixel and L is the radiance at the pixel of interest.

5. Compute the standard deviation σ, which is the sum of the standard deviations from each group, i.e. $\sigma = \sum\limits_{i \in group} \sigma_i$ where σ_i is as in the standard definition for standard

deviation. The new variance definition separates samples into groups because the standard variance is a good measure for continuous function, but not necessarily for discrete function. When a group has only one sample, we approximate its standard derivation by the average of other groups. If there are over 3 groups, then it is a complex pixel and we will let N to be the maximum number of samples allowed in a pixel.

6. Find N by solving $\sigma_N = k\ \epsilon$, where σ_N is the expected standard deviation with N samples and k is a predefined constant for accuracy tuning. If the sampling strategy has $O(N^{-d})$ accuracy, then $\sigma_N = \sigma_n\ (\frac{n}{N})^d$. Random sampling has $O(N^{-\frac{1}{2}})$ [46]; other quasi-random samples which converge faster are likely to have d smaller than 0.5.

5.4 Summary

In this chapter, we emphasize the importance of equidistribution across projections of subdomain and derived new multi-jittered sampling methods which have the lowest discrepancy and the highest stability among the commonly used stochastic sampling methods. As we take a global view of the sampling issue and hope to create samples in a 7 or 8 dimensional domain, we face two problems. The first one is explosion in the number of samples when generalizing the existing methods; for example, two jittered cells in each domain would require at least 2^8 samples. The second problem is the explosion in the number of constraints; for example, multi-jittered sampling methods need to simultaneously satisfy equidistribution in $2^8 - 1$ projections of subdomain. How to get around these problems in order to generate good samples is still a research topic.

Quasi-Monte Carlo methods approximate an integral via quasi-random samples, whose behaviors have been studied in integration theory. However, the discontinuity and the high frequency in an image limit the application of integration theory because the image function is ill-conditioned and has extremely large bounded variation. The other characteristic of image synthesis that makes applications of number theoretic methods difficult is that instead of estimating one function with each sample pattern, we need to estimate each image function with many sample patterns each of which estimates only a portion of the image function. This introduces aliasing artifacts, which is a problem going across the pixel(integral) boundary and is not addressed in integration theory.

Supersampling reduces error by taking more than one sample, but to predict the total number of samples required is difficult. Existing methods solve this problem from the view points of statistics, signal processing, and human vision individually. However, a successful supersampling method requires the knowledge from all of them. In this chapter, we presented a framework that accounts for the image complexity, sample convergence rate, nonlinear response of human eyes, and the limitation of the media. Many elements in this framework require further investigation, so a framework that is not as general but is practical now is also presented.

6 Conclusion

In this document, we formalized the pixel color in image synthesis to a sum of three integrals which include the emitted lighting in a five dimensional integral, the direct lighting in an eight dimensional integral, and the indirect lighting in a seven dimensional integral. In distribution ray tracing, these integrals can be approximated by sequentially applying quasi-Monte Carlo methods. The effectiveness of this method depends heavily on the spatial distribution of samples, which is the major focus of this work.

When evaluating the lighting integrals, the equidistribution of samples across lower dimensional projections is also important, especially when the number of samples is small as in the case of distribution ray tracing. To minimize the variance caused by the undesired sample correlation in different subdomains, we suggested multi-jittered sampling which generates samples that are equidistributed in any combination of subdomains. Our experiments showed that the multi-jittered samples have the lowest discrepancy as well as the smallest discrepancy variation among the most commonly used methods in computer graphics. So far we can only generate multi-jittered sampling whose samples are equidistributed in one dimensional domains and two dimensional domains, because the difficulty of generating equidistributed samples increases with the dimension of the sampling space, so the number of constraints in multi-jittered sampling increase exponentially. To effectively generate multi-jittered samples in 7 and 8 dimensional domains is desirable, but relies on further investigation.

Effectively predicting the total number of samples in adaptive supersampling can increase the efficiency of a distribution ray tracing program. We proposed a supersampling framework which takes into account the human visual system, the limited dynamic range of media, the estimation error, and convergence rate of sampling method. However, almost all of these elements are not well understood and require more research in human vision, transformation schemes from a radiance image to a displayed image, statistics, signal processing, and integration theory. Without such knowledge, supersampling methods can only have limited success.

In computer graphics, the average discrepancy is often related to the convergence rate of a particular sampling method. However, within one sampling method the discrepancy can vary a great deal and can form a non-symmetric distribution, which opens the question of whether average discrepancy is able to characterize a sampling method.

Discrepancy is guaranteed to be a good and useful measure of sample equidistribution only when applied to a function of bounded variation. Unfortunately, the image function contains edge and textures that can cause discontinuity in the function and thus unbounded variation. Also, the discrepancy can not predict the occurrence of aliasing artifact. Thus the discrepancy must be modified to be helpful in image synthesis.

One of the major purposes of initial samples is feature detection. In this document we assume that pixel has two features and the features are divided by a random edge. With slight modification of Mitchell's method for computing random edge discrepancy [46], we are able to numerically justify that approximately 10 multi-jittered samples are sufficient to detect both features if their projected areas are not smaller than 5% of the pixel area. Missing a feature less than 5% of a pixel is generally acceptable unless the missing feature is very bright while the other feature is not.

Lighting is computed at the innermost integral (rendering equation) of pixel color. Direct lighting is separated from indirect lighting because of the sharp shading change direct lighting can cause on other objects. In this document, we have investigated the single luminaire environment by applying four different sampling strategies to six primitive luminaires. Among these strategies, $p \sim \cos\theta$ gives the optimal estimation in a perfectly diffuse environment; $p \sim \cos\theta$ and $p \sim \dfrac{1}{\omega'}$ generate samples with fast convergence rate in a diffuse environment. None of these four strategies is suitable for a general environment with directional luminaires and specular surfaces. Future work is required to handle three different local coordinate systems in order to develop a p.d.f. efficient for a general environment.

Imposters which represent luminaires in simplified forms can reduce the rendering complexity while still generating images that are "indistinguishable" from the image of the original objects. The application of imposters is common in computer graphics, but each application has been treated as special case. This document characterized the use of imposters into several categories: simplification of representation, extra information carriage, summation in a different domain, and physical behavior simplification. The primitive luminaires discussed in direct lighting are imposters of simplified representation.

Direct lighting from multiple luminaires is formalized as a Monte Carlo summation which is a generalizated form of the previous methods. Using the same formula from Monte Carlo summation with different p.d.f.s, we have derived three different schemes, each useful in different situations. The constant method is useful for debugging. The linear method is useful for domains with tens of luminaires. Spatial subdivision method is useful for extremely large amount of luminaires.

The most important extension of this work that should be done is the generalization of our sampling schemes to environments that are neither specular nor diffuse. This will not improve our images, but will make some of them much faster to compute.

A

Terms and Units for Radiometric and Photometric Quantities

Terminologies for global illumination in computer graphics have been borrowed from several different fields, such as photometry, radiometry, colorimetry, etc. To avoid unnecessary confusion, we will follow the definition from American National Standard and Illuminating Engineering Society [35] as shown in Figure A.1, along with some supplementary information from Wyszecki and Stiles [88].

A quantity with a subscript λ means a spectral concentration, i.e. a quantity corresponding to a narrow wavelength interval. A quantity with a λ in parentheses means a function of wavelength. The symbols for photometric quantities are the same as those for the corresponding radiometric quantities. When it is necessary to differentiate them the subscripts v for photometry and e for radiometry should be used.

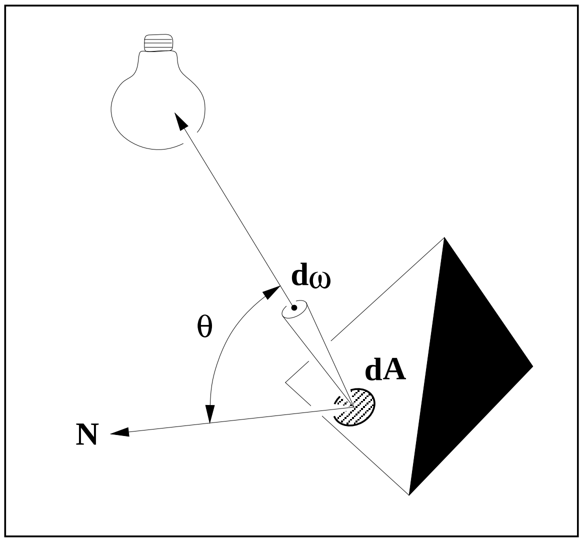


Figure A.1: Geometry for radiometric and photometric quantities.

A.1 Radiometric Quantities

Radiometry is the measurement of quantities associated with radiant energy.

radiant energy : Q, *joul*.

Energy traveling in the form of electromagnetic waves.

radiant flux or **radiant power** : Φ, *watt*.

The radiant energy emitted, transferred, or received through a surface in unit time interval, i.e. $\Phi = \frac{dQ}{dt}$.

radiant exitance or **radiant emittance** : M, $\frac{watt}{m^2}$.

The radiant flux per unit area of the energy-emitting surface, i.e. $M = \frac{d\Phi}{dA}$. In zonal methods, this term is called radiosity and has the symbol B.

irradiance : E, $\frac{watt}{m^2}$.

The radiant flux per unit area of the energy-receiving surface, i.e. $E = \frac{d\Phi}{dA}$.

solid angle : ω, *steradian* or sr.

Solid angle is the projected area of an object onto the unit sphere centered about a point of interest. It combines into a single number the projected area and the inverse-square law of illumination, $E = \frac{I}{d^2}$.

radiant intensity : I, $\frac{watt}{sr}$.

The radiant flux per unit solid angle, i.e. $I = \frac{d\Phi}{d\omega}$.

radiance : L, $\frac{watt}{sr \times m^2}$.

The radiant flux per unit solid angle per projected area, i.e. $L = \frac{d^2\Phi}{d\omega \times dA \times \cos\theta}$. Radiance behaves like color perceived by human eyes and is probably the most heavily used term in global illumination.

A.2 Photometric Quantities

Photometry is the measurement of quantities associated with light.

light or **luminous energy** : Q, $lumen - hour$.

Light is the radiant energy which can excite the retina and produce a visual sensation. The electromagnetic spectrum of visible wavelength are from about 380 to 770 nm

where 1 $nm = 10^{-9}\ m$.

spectral luminous efficacy of radiant flux : $K(\lambda)$, $\dfrac{lumen}{watt}$.

Spectral luminous efficacy defines the relationship between radiant flux and luminous flux as $K(\lambda) = \dfrac{\Phi_{v\lambda}}{\Phi_{e\lambda}}$. $K(\lambda)$ is commonly expressed as a product of maximum luminous efficacy K_m ($= 683\ \dfrac{lumen}{watt}$), and spectral luminous efficiency function $V(\lambda)$, which is normalized in $[0,1]$. $K(\lambda)$ basically corresponds to photopic vision, mediated by the cones and similarly there is a relationship for scotopic vision or rod vision, i.e. $K'(\lambda) = K'_m V'(\lambda)$. Unless otherwise indicated, photometric quantities relate to photopic vision.

luminous flux or **luminous power** : Φ, *lumen*.

$$\Phi_v = \int_{380}^{770} \Phi_{e\lambda} K(\lambda) d\lambda = K_m \int_{380}^{770} \Phi_{e\lambda} V(\lambda) d\lambda$$

luminous intensity : I, *candela* $(= \dfrac{lumen}{sr})$.

The luminous flux per unit solid angle, i.e. $\dfrac{d\Phi}{d\omega}$. Luminous intensity has been the primary standard of light instead of luminous flux because it best serves the purpose in early photometric practice when the visual matching is used to compare the brightness of different luminaires. It is also the quantity to specify the distribution curve of a luminaire.

luminous exitance or **luminous emittance** : M, *footcandle* $(= \dfrac{lumen}{ft^2})$.

The luminous flux per unit area of the light-emitting surface, i.e. $M = \dfrac{d\Phi}{dA}$.

illuminance or **illumination** : E, *footcandle* $(= \dfrac{lumen}{ft^2})$.

The luminous flux per unit area of the energy-receiving surface, i.e. $E = \frac{d\Phi}{dA}$.

luminance : L, $footlambert$ $(= \frac{candela}{\pi \times ft^2})$.

The radiant intensity per projected area, i.e. $L = \frac{dI}{dA \times \cos\theta}$.

A.3 Material Properties

diffusing surface .

The surface scatters the incident flux in all directions. A perfectly diffusing surface (Lambertian surface) uniformly scatters such that the radiance is the same in all directions.

specular surface or **regular surface** .

The surface reflects the incident flux predominately at the specular angle or angle of reflection [23].

reflectance : ρ, *none.*

The reflected flux per incident flux, i.e. $\rho = \frac{\Phi_r}{\Phi_i}$.

bidirectional reflectance-distribution function : $f_r(\theta_i, \phi_i; \theta_r, \phi_r)$, sr^{-1}.

The reflected radiance per unit irradiance, i.e.
$f_r(\theta_i, \phi_i; \theta_r, \phi_r) = \frac{dL_r(\theta_r, \phi_r)}{dE_i(\theta_i, \phi_i)} = \frac{dL_r(\theta_r, \phi_r)}{L_i(\theta_i, \phi_i) \cos\theta d\omega}$.
In a lambertian surface, $f_r(\theta_i, \phi_i; \theta_r, \phi_r) = \frac{\rho}{\pi}$.

For the reason of simplicity, the direction of the incident angle (θ_i, ϕ_i) will be replaced by ψ in this document whereas the direction of the reflected angle (θ_r, ϕ_r) will be replaced by ψ'.

B

Monte Carlo Summation

The direct lighting from multiple luminaires is a sum of the direct lighting from each luminaire, but the sum can be transformed to an integral which is approximated by the quasi-Monte Carlo methods.

$$\begin{aligned}
L_{direct} &= L_{direct}^{[1]} + \cdots + L_{direct}^{[i]} + \cdots + L_{direct}^{[N]} \\
&= \int_0^1 L_{direct}^{[1]} d\tau + \cdots + \int_{i-1}^{i} L_{direct}^{[i]} d\tau + \cdots + \int_{N-1}^{N} L_{direct}^{[N]} d\tau \\
&= \int_0^N L_{direct}^{[1]} u_1(\tau) d\tau + \cdots + \int_0^N L_{direct}^{[i]} u_i(\tau) d\tau + \cdots + \int_0^N L_{direct}^{[N]} u_N(\tau) d\tau \\
&= \int_0^N L_{direct}^{[1]} u_1(\tau) + \cdots + L_{direct}^{[N]} u_N(\tau) d\tau
\end{aligned}$$

where $u_i(\tau) = 1$ when $i - 1 < \tau \leq i$ and $u_i(\tau) = 0$ otherwise. Now, we can apply the Monte Carlo methods in Section 2.2.2 and Section 2.2.4 to this integral.

$$L_{direct} \approx \frac{L_{direct}^{[1]} u_1(\dot{\tau}) + \cdots + L_{direct}^{[N]} u_N(\dot{\tau})}{p(\dot{\tau})} = \frac{L_{direct}^{[\lceil\dot{\tau}\rceil]}}{p(\dot{\tau})} \tag{B.1}$$

The ceiling function $\lceil\tau\rceil = i$ when $i - 1 < \tau \leq i$. The integrand $L_{direct}^{[1]} u_1(\tau) + \cdots + L_{direct}^{[N]} u_N(\tau)$ is a step function and thus it is reasonable to have the discrete probability density function $p(\tau)$ equal to $p(\lceil\tau\rceil)$ for all τ, and $L_{direct} \approx \dfrac{L_{direct}^{[\lceil\dot{\tau}\rceil]}}{p(\lceil\dot{\tau}\rceil)}$.

C

Monte Carlo Methods for Double Integrals

If a double integral has a separable integrand, the double integral can be written as a product of two one dimensional integrals, which can be estimated by the Monte Carlo methods addressed in Section 2.2.2. If the integrand is not separable as L_{direct} in Equation 2.5 or Equation 2.6, applying Monte Carlo methods becomes complicated.

Without loss of generality, assume $L_{direct} = \int_{\Omega_\nu} \int_{\Omega_\mu} f(\mu, \nu) d\mu d\nu$ and we can apply the Monte Carlo methods to the outer integral. First we need to choose a p.d.f. $p(\dot{\mu}, \dot{\nu})$, and from which we can derive $p_\nu(\nu) = \int_{\Omega_\mu} p(\mu, nu) d\mu$. Then after chosing a sample$\dot{\mu}$ to estimate the outer integral, we can use the conditional p.d.f. $p_\mu(\mu|\dot{\nu}) = \dfrac{p(\mu, \dot{\nu})}{\int_{\Omega_\nu} p_u(\mu, \dot{\nu}) d\mu}$ to estimate the inner integral.

$$\begin{aligned}
L_{direct} &= \int_{\Omega_\nu} \int_{\Omega_\mu} f(\mu, \nu) d\mu d\nu \\
&= \int_{\Omega_\nu} \int_{\Omega_\mu} \frac{f(\mu, \nu)}{p_\nu(\nu)} p_\nu(\nu) d\mu d\nu \\
&\approx \frac{1}{p_\nu(\dot{\nu})} \int_{\Omega_\mu} f(\mu, \dot{\nu}) d\mu
\end{aligned}$$

$$\begin{aligned} &= \frac{1}{p_\nu(\dot\nu)} \int_{\Omega_\mu} \frac{f(\mu,\dot\nu)}{p_\mu(\mu|\dot\nu)} p_\mu(\mu|\dot\nu) d\mu \\ &\approx \frac{f(\dot\mu,\dot\nu)}{p_\nu(\dot\nu) p_\mu(\dot\mu|\dot\nu)} \\ &= \frac{f(\dot\mu,\dot\nu)}{p(\dot\mu,\dot\nu)} \end{aligned}$$

If given two random numbers ξ_1 and ξ_2 in $[0,1]$, then $\dot\mu$ and $\dot\nu$ can be derived by inverting the following two equations.

$$\xi_1 = \int_{\nu_{min}}^{\dot\nu} p_\nu(\nu) d\nu = \int_{\nu_{min}}^{\dot\nu} \int_{\Omega_\mu} p(\mu,\nu) d\mu d\nu \tag{C.1}$$

$$\xi_2 = \int_{\mu_{min}(\dot\nu)}^{\dot\mu(\dot\nu)} p_\mu(\mu|\dot\nu) d\mu = \frac{\int_{\mu_{min}(\dot\nu)}^{\dot\mu(\dot\nu)} p(\mu,\dot\nu) d\mu}{\int_{\Omega_\mu} p(\mu,\dot\nu) d\mu} \tag{C.2}$$

In order to implement this inversion, the antiderivative of $p(\mu,\nu)$ with respect to μ must be known. If we must start with a function $g(\mu,\nu)$ such that $p(\mu,\nu) \propto g(\mu,\nu)$, then $p(\mu,\nu) = \frac{g(\mu,\nu)}{P_{total}}$ where $P_{total} = \int_{\Omega_\nu} \int_{\Omega_\mu} g(\mu,\nu) d\mu d\nu$, and we must be able to compute $\int_{\Omega_\nu} \int_{\Omega_\mu} g(\mu,\nu) d\mu d\nu$ analytically.

So, if Equation C.1, Equation C.2, and P_{total} are known for any given $p(\dot\mu,\dot\nu)$, then computing the primary estimate $\frac{f(\dot\mu,\dot\nu)}{p(\dot\mu,\dot\nu)}$, or the secondary estimate $\frac{1}{N}\sum_{i=1}^{N} \frac{f(\dot\mu_i,\dot\nu_i)}{p(\dot\mu_i,\dot\nu_i)}$ is straightforward.

Bibliography

[1] A.H.Stroud. *Approximate Calculation of Multiple Integrals.* Prentice-Hall, 1971.

[2] Larry Aupperle and Pat Hanrahan. A hierarchical illumination algorithm for surfaces with glossy reflection. *Computer Graphics*, 6(1):155–162, August 1993. ACM Siggraph '93 Conference Proceedings.

[3] Daniel R. Baum, Holly E. Rushmeier, and James M. Winget. Improving radiosity solutions through the use of analytically determined form-factors. *Computer Graphics*, 23(3):325–334, July 1989. ACM Siggraph '89 Conference Proceedings.

[4] Jozsef Beck and William W.L. Chen. *Irregularities of distribution.* Cambridge University Press, Cambridge, 1987.

[5] Richard L. Burden, J. Douglas Faires, and Albert C. Reynolds. *Numerical Analysis.* Prindle, Weber & Schmidt, Boston, Massachusetts, 1978.

[6] Shenchang Eric Chen, Holly Rushmeier, Gavin Miller, and Douglass Turner. A progressive multi-pass method for global illumination. *Computer Graphics*, 25(4):165–174, July 1991. ACM Siggraph '91 Conference Proceedings.

[7] K. Chiu, M. Herf, P. Shirley, S. Swamy, C. Wang, and K. Zimmerman. Spatially nonuniform scaling functions for high contrast images. *Graphics Interface*, 1993.

[8] Ken Chiu, Peter Shirley, and Changyaw Wang. Multi-jittered sampling methods. In Paul S. Heckbert, editor, *Graphics Gems 4*. Academic Press, New York, NY, 1993. (to appear).

[9] William Gemmell Cochran. *Sampling Techniques.* Wiley, New York, 1977.

[10] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A progressive refinement approach to fast radiosity image generation. *Computer Graphics*, 22(4):75–84, August 1988. ACM Siggraph '88 Conference Proceedings.

[11] Michael F. Cohen and Donald P. Greenberg. The hemi-cube: a radiosity solution for complex environments. *Computer Graphics*, 19(3):31–40, July 1985. ACM Siggraph '85 Conference Proceedings.

[12] Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis.* Academic Press, 1993.

[13] Micheal F. Cohen, Donald P. Greenberg, David S. Immel, and Philip J. Brock. An efficient radioisty approach for realistic image synthesis. *IEEE Computer Graphics and Applications*, 6(2):26–35, 1986.

[14] Robert L. Cook. Stochastic sampling and distributed ray tracing. In Andrew S. Glassner, editor, *An Introduction to Ray Tracing*. Academic Press, San Diego, CA, 1989.

[15] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. *Computer Graphics*, 18(4):165–174, July 1984. ACM Siggraph '84 Conference Proceedings.

[16] Frank Crow. Summed-area tables for texture mapping. *Computer Graphics*, 18(4):207–212, July 1984. ACM Siggraph '84 Conference Proceedings.

[17] Philip J. Davis and Philip Rabinowitz. *Methods of Numerical Integration*. Academic Press, second edition, 1984.

[18] Mark Z. Dippe and Erling Henry Wold. Antialiasing through stochastic sampling. *Computer Graphics*, 19(3):69–78, July 1985. ACM Siggraph '85 Conference Proceedings.

[19] David P. Dobkin and Don P. Mitchell. Random-edge discrepancy of supersampling patterns. In *Proceedings of Graphics Interface '93*, pages 62–69, 1993.

[20] H. Engels. *Numerical Quadrature and Cubature*. Academic Press, New York, 1980.

[21] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Reading, MA, second edition, 1990.

[22] Andrew S. Glassner. Space subdivision for fast ray tracing. *IEEE Computer Graphics and Applications*, 4(10):15–22, 1984.

[23] Andrew S. Glassner, editor. *An Introduction to Ray Tracing.* Academic Press, San Diego, CA, 1989.

[24] Andrew S. Glassner. Dynamic stratification. In *Graphics Interface '93*, 1993.

[25] Cindy M. Goral, Kenneth E. Torrance, and Donald P. Greenberg. Modeling the interaction of light between diffuse surfaces. *Computer Graphics*, 18(4):213–222, July 1984. ACM Siggraph '84 Conference Proceedings.

[26] Steven J. Gortler, Peter Schroder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. *Computer Graphics*, 6(1):221–230, August 1993. ACM Siggraph '93 Conference Proceedings.

[27] Roy Hall. *Illumination and Color in Computer Generated Imagery.* Springer-Verlag, New York, N.Y., 1988.

[28] John H. Halton. Estimating the accuracy of quasi-monte carlo integration. In S.K.Zaremba, editor, *Applications of Number Theory to Numerical Analysis*, pages 345–360. Academic Press, 1972.

[29] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, July 1991. ACM Siggraph '91 Conference Proceedings.

[30] Eugene Hecht and Alfred Zajac. *Optics.* Addison-Wesley, Reading, MA, 1974.

[31] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *Computer Graphics*, 24(3):145–154, August 1990. ACM Siggraph '90 Conference Proceedings.

[32] Ronald N. Helms and M. Clay Belcher. *lighting for energy-efficienct luminous environments.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1991.

[33] Robert V. Hogg and Allen T. Craig. *Introduction to Mathematical Statistics.* Macmillan, New York, 1978.

[34] David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. *Computer Graphics*, 20(4):133–142, August 1986. ACM Siggraph '86 Conference Proceedings.

[35] American National Standard Institude. Nomenclature and definitions for illuminating engineering. ANSI Report, 1986. ANSI/IES RP-16-1986.

[36] James T. Kajiya. The rendering equation. *Computer Graphics*, 20(4):143–150, August 1986. ACM Siggraph '86 Conference Proceedings.

[37] Malvin H. Kalos and Paula A. Whitlock. *Monte Carlo Methods.* John Wiley and Sons, New York, N.Y., 1986.

[38] David Kirk and James Arvo. Unbiased sampling techniques for image sysnthesis. *Computer Graphics*, 25(4):153–156, July 1991. ACM Siggraph '91 Conference Proceedings.

[39] A. Kok and F. Jansen. Source selection for the direct lighting calculation in global illumination. In *Proceedings of the Second Eurographics Workshop on Rendering*, 1991.

[40] Edwin H. Land. The retinex theory of color vision. *(The) Scientific American*, (12):108–128, December 1977.

[41] Brigitta Lange. The simulation of radiant light transfer with stochastic ray-tracing. In *Proceedings of the Second Eurographics Workshop on Rendering*, 1991.

[42] Eric Languenou and Pierre Tellier. Including physical light sources and daylight in a global illumination model. *Computer Graphics Forum*, 1992. Eurographics '92.

[43] Mark E. Lee, Richard A. Redner, and Samuel P. Uselton. Statistically optimized sampling for distributed ray tracing. *Computer Graphics*, 19(3):61–68, July 1985. ACM Siggraph '85 Conference Proceedings.

[44] Don P. Mitchell. Generating antialiased images at low sampling densities. *Computer Graphics*, 21(4):65–72, July 1987. ACM Siggraph '87 Conference Proceedings.

[45] Don P. Mitchell. Spectrally optimal sampling for distribution ray tracing. *Computer Graphics*, 25(4), July 1991. ACM Siggraph '91 Conference Proceedings.

[46] Don P. Mitchell. Ray tracing and irregularities of distribution. In *Proceedings of Third Eurographics Workshop on Rendering*, pages 61–69, 1992.

[47] Don P. Mitchell and Arun N. Netravali. Reconstruction filters in computer graphics. *Computer Graphics*, 22(4):221–228, August 1988. ACM Siggraph '88 Conference Proceedings.

[48] Earl N. Mitchell. *Photographic Science.* John Wiley and Sons, 1984.

[49] Karl Dieter Moller. *Optics.* University Science Books, Mill Valley, Calif, 1988.

[50] Harald Niederreiter. Quasi-monte carlo methods and pseudo-random numbers. *Bulletin of the American Mathematical Society*, 84(4):957–1029, November 1978.

[51] Tomoyuki Nishita and Eihachiro Nakamae. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. *Computer Graphics*, 19(3):23–30, July 1985. ACM Siggraph '85 Conference Proceedings.

[52] James L. Nuckolls. *Interior Lighting for Environmental Designers.* Wiley-interscience, New York, second edition, 1983.

[53] Commission Internationale De L'Eclairage International Commission on Illumination Internationale Beleuchtungskommission. Standardization of luminance distribution on clear skies, 1973. C.I.E. No, 22 (TC-4.2.).

[54] James Painter and Kenneth Sloan. Antialiased ray tracing by adaptive progressive refinement. *Computer Graphics*, 23(3):281–288, July 1989. ACM Siggraph '89 Conference Proceedings.

[55] Werner Purgathofer. A statistical method for adaptive stochastic sampling. *Computer Graphics forum*, 1986. Eurographics '92, Also appears in Computers & Graphics, 1987, Vol.11, Number 2, 157-162.

[56] Jon Rokne. The area of a simple polygon. In James Arvo, editor, *Graphics Gems 2*. Academic Press, New York, NY, 1991.

[57] Reuven Y. Rubinstein. *Simulation and the Monte Carlo method.* New York : Wiley, 1981.

[58] Holly Rushmeimer, Charles Patterson, and Aravindan Veerasamy. Geometric simplification for indirect illumination calculations. *Graphics Interface*, 1993.

[59] Peter Shirley. Personal Communication.

[60] Peter Shirley. *Physically Based Lighting Calculations for Computer Graphics.* PhD thesis, University of Illinois at Urbana-Champaign, November 1990.

[61] Peter Shirley. A ray tracing algorithm for global illumination. *Graphics Interface '90*, pages 205–212, May 1990.

[62] Peter Shirley. Discrepancy as a quality measure for sampling distributions. In *Eurographics '91*, pages 183–193, September 1991.

[63] Peter Shirley. Radiosity via ray tracing. In James Arvo, editor, *Graphics Gems 2.* Academic Press, New York, NY, 1991.

[64] Peter Shirley. Warping transformations for sampling. In David Kirk, editor, *Graphics Gems 3.* Academic Press, New York, NY, 1992.

[65] Peter Shirley and Changyaw Wang. Direct lighting by monte carlo integration. In *Proceedings of the Second Eurographics Workshop on Rendering*, 1991.

[66] Peter Shirley and Changyaw Wang. Distribution ray tracing theory and practice. *Computer Graphics forum*, 1992. Eurographics '92.

[67] Peter Shirley and Changyaw Wang. Luminaire sampling in distribution ray tracing. Technical Report 343, Department of Computer Science,Indiana University, January 1992.

[68] Robert Siegel and John R. Howell. *Thermal Radiation Heat Transfer.* McGraw-Hill, New York, N.Y., 1981.

[69] Francois Sillion, James Arvo, Stephen Westin, and Donald Greenberg. A global illumination algorithm for general reflection distributions. *Computer Graphics*, 25(4):187–196, July 1991. ACM Siggraph '91 Conference Proceedings.

[70] Francois Sillion and Claude Puech. A general two-pass method integrating specular and diffuse reflection. *Computer Graphics*, 23(3):335–344, July 1989. ACM Siggraph '89 Conference Proceedings.

[71] S.K.Zaremba, editor. *Applications of Number Theory to Numerical Analysis.* Academic Press, New York and London, 1972.

[72] Brian E. Smits, James R. Arvo, and David H. Salesin. An importance-driven radiosity algorithm. *Computer Graphics*, 26(2):273–282, July 1992. ACM Siggraph '92 Conference Proceedings.

[73] Jack Tumblin and Holly Rushmeier. Tone reproduction for realistic computer generated images. *IEEE Computer Graphics and Applications*, November 1993.

[74] Steve Upstill. *The RenderMan Companion.* Addison-Wesley Publishing Company, 1990.

[75] John R. Wallace, Kells A. Elmquist, and Eric A. Haines. A ray tracing algorithm for progressive radiosity. *Computer Graphics*, 23(3):335–344, July 1989. ACM Siggraph '89 Conference Proceedings.

[76] Changyaw Wang. Physically correct direct lighting for distribution ray tracing. In David Kirk, editor, *Graphics Gems 3*. Academic Press, New York, NY, 1992.

[77] Changyaw Wang and Peter Shirley. Monte carlo techniques for the direct lighting calculation. *ACM Transaction on Computer Graphics*, 1993. (submitted), Also appears in ACM Siggraph '93 Course Notes, Global Illumination.

[78] Greg Ward. Adaptive shadow testing for ray tracing. In *Proceedings of the Second Eurographics Workshop on Rendering*, 1991.

[79] Gregory J. Ward. Personal communication.

[80] Gregory J. Ward. Measuring and modeling anisotropic reflection. *Computer Graphics*, 26(2):265–272, July 1992. ACM Siggraph '92 Conference Proceedings.

[81] Gregory J. Ward. The radiance lighting simulation system. *Global Illumination*, pages 1–16, 1992. ACM Siggraph '92 Course Notes.

[82] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. *Computer Graphics*, 22(4):85–92, August 1988. ACM Siggraph '88 Conference Proceedings.

[83] Tony T. Warnock. Low-discrepancy point sets. In S.K.Zaremba, editor, *Applications of Number Theory to Numerical Analysis*. Academic Press, New York, NY, 1972.

[84] Turner Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, June 1980.

[85] Lance Willams. Pyramidal parametrics. *Computer Graphics*, 17(3), July 1983. ACM Siggraph '83 Conference Proceedings.

[86] Grorge Wolberg. *Digital image warping.* IEEE Computer Society Press, 1990.

[87] Stephen Wolfram. *Mathematica.* Addison-Wesley Publishing Company, Inc., 1991.

[88] Gunter Wyszecki and W.S. Stiles. *Color Science : Concepts and Methods, Quantitative Data and Formulae.* John Wiley and Sons, second edition, 1982.

[89] Daniel Zwillinger. *Handbook of Integration.* Jones and Bartlett, Boston, 1992.