CHAPTER 6

ZONAL SOLUTION METHODS

In zonal methods, the radiances of a scene are computed in advance of rendering in a view-independent process. In this chapter, zonal methods and their relation to image methods are discussed. In Section 6.1, the zonal method for diffuse environments is discussed, both in terms of linear systems of equations and in terms of physical simulation. Section 6.2 outlines optimization strategies for zonal environments, and speculates that the $O(N^2)$ time complexity of zonal methods can be beat. That section includes a proof that the expected number of rays needed for a zonal solution is $O(N)$. In Section 6.3, zonal methods for specular and glossy environments are discussed, and zonal and image-based methods are combined in a general way. This approach has the advantage over previous approaches to glossy environments that only the storage needed for each reflection type is required. Section 6.4 summarizes the content of this chapter.

6.1 Zonal Methods for Diffuse Environments

The simplest zonal methods assume all surfaces are diffuse reflectors[40]. First the environment is subdivided into N discrete patches that are assumed to be constant in reflectance, reflected

power, and emitted power. The reflectance (R_i) and emitted power (Φ_i^e) are known, and the reflected power (Φ_i^r) is unknown. If we solve for Φ_i^r, then we can find Φ_i, the total power coming from the ith patch.

Once the total power of each patch is found, it can be converted to radiance using Equation 4.1. These radiance values can then be interpolated to form a smooth appearance[23]. The next several sections show methods of solving for Φ_i.

6.1.1 Diffuse Zonal Methods as Linear Algebraic Equations

The total power coming from the ith surface is the sum of emitted and reflected power: $\Phi_i = \Phi_i^e + \Phi_i^r$. The reflected power is the reflectivity times the incoming power. The incoming power is a fraction of the outgoing power of the other surfaces. The fraction of the outgoing power from surface *source* that hits surface *target* is called a form-factor (or view-factor or configuration factor), and is denoted $f_{source \rightarrow target}$. This yields an expression for the total power coming from surface i:

$$\Phi_i = \Phi_i^e + R_i \sum_{j=0}^{N} f_{j \rightarrow i} \; \Phi_j \tag{6.1}$$

Conservation of energy implies:

$$\sum_{i=0}^{N} f_{j \rightarrow i} \; \Phi_j \leq 1$$

with equality if the system is closed. Equation 6.1 can be written down in matrix form:

$$\mathbf{A}\Phi = \Phi^e \tag{6.2}$$

Where the matrix $\mathbf{A}$ is:

$$\mathbf{A} = \begin{bmatrix} (1 - R_1 f_{1\to 1}) & -R_1 f_{2\to 1} & -R_1 f_{3\to 1} & \cdots & -R_1 f_{N\to 1} \\ -R_2 f_{1\to 2} & (1 - R_2 f_{2\to 2}) & -R_2 f_{3\to 2} & \cdots & -R_2 f_{N\to 2} \\ -R_3 f_{1\to 3} & -R_3 f_{2\to 3} & (1 - R_3 f_{3\to 3}) & \cdots & -R_3 f_{N\to 3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -R_N f_{1\to N} & -R_N f_{2\to N} & -R_N f_{3\to N} & \cdots & (1 - R_N f_{N\to N}) \end{bmatrix} \tag{6.3}$$

It can be shown that the system rewritten in terms of radiance is diagonally dominant (though not sparse), so a Gauss-Seidel iterative method can be used to solve for Φ[23]. The Gauss-Seidel method will require $O(N^2)$ solution time, and the matrix $\mathbf{A}$ will require $O(N^2 f)$ initialization time, where f is the average time to calculate a form-factor. The storage requirement is $O(N^2)$ because $\mathbf{A}$ has N^2 elements.

6.1.2 Diffuse Zonal Methods as Light Transport Simulation

Another way to look at solving for Φ is to use direct simulation. We first set our estimate of Φ_i to be Φ_i^e for all i. For each surface i that has non-zero Φ_i^e, we can shoot a set of n_i energy packets each carrying a power of Φ_i^e / n_i. When a packet with power Φ hits a surface j, we can add $R_j \Phi$ for our estimate of Φ_j, and reflect a new energy packet with power $R_j \Phi$. This energy packet will bounce around the environment until it is depleted to a point where truncation is used. This basic energy packet tracing technique has been used in Heat Transfer[55, 28, 113], Illumination Engineering[108], and Physics[105, 58].

This method, which I call *reflection simulation*, has the problem that each reflection is followed by a ray intersection test to find the next surface hit. The later reflections will carry a relatively small amount of power, so tracing these later rays is somewhat wasteful in the sense that we have bad 'load-balancing': some rays do more work than others. One solution to this

problem is to replace the reflection model with a model where light is absorbed and immediately reemitted (after attenuation by the reflectance). A scene where light is absorbed and reemitted in this way looks exactly like a scene where light is reflected, so solving for the transport in either model will yield the same solution.

To solve for the absorb and reemit model, we can again send power in bundles from light sources. When a bundle carrying power Φ hits a surface j, the absorbed power that will later be reemitted by surface j can be increased by $R_j\Phi$. After each light source emits its power, reflective surfaces can, in turn, emit their absorbed power. The efficiency of this method is best if surfaces with the greatest amount of power send their power first. This method, which I call *absorb and reemit simulation*, is used in computer graphics, where it is called *progressive refinement radiosity*[22]. The form factors needed to send energy from a given patch are usually calculated on the fly, so there is no $O(N^2)$ storage requirement. This space optimization could also be done in the Gauss-Seidel solution, since only one row of the matrix is used at a time.

6.1.3 Form Factor Calculation

If the absorb and reemit simulation method is used, the crucial step occurs when the designated source patch sends its power into the environment. The most straightforward method of sending this power is the Monte Carlo method[69, 1, 2, 100], where a random set of energy bundles is emitted (as rays) in a diffuse distribution, and these power carrying rays are sent to the other surfaces (generating rays in a diffuse distribution is discussed in Appendix B). This method is shown in Figure 6.1, where the grey source patch is sending many rays into the environment. Figure 6.2 shows a simple environment, similar to that used by [40, 72], with radiances calculated by the Monte Carlo method.

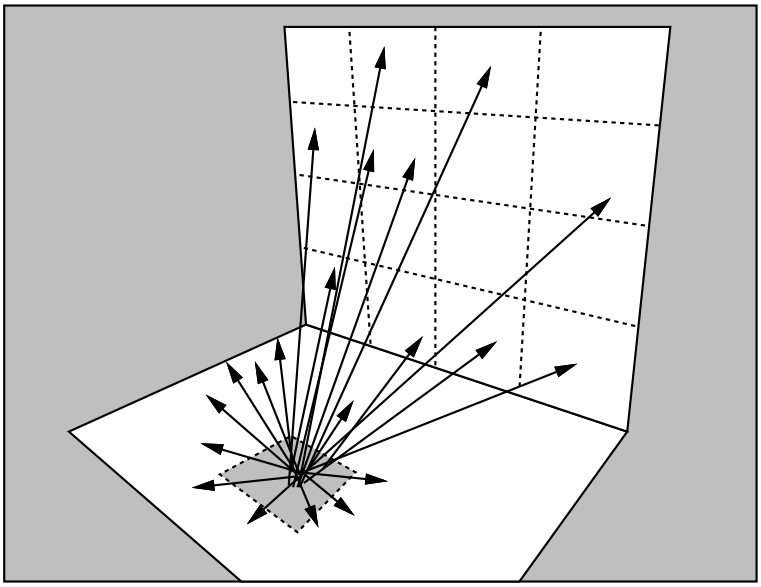

Figure 6.1: Monte Carlo emission of energy.

Figure 6.2: Zonal solution for diffuse scene.

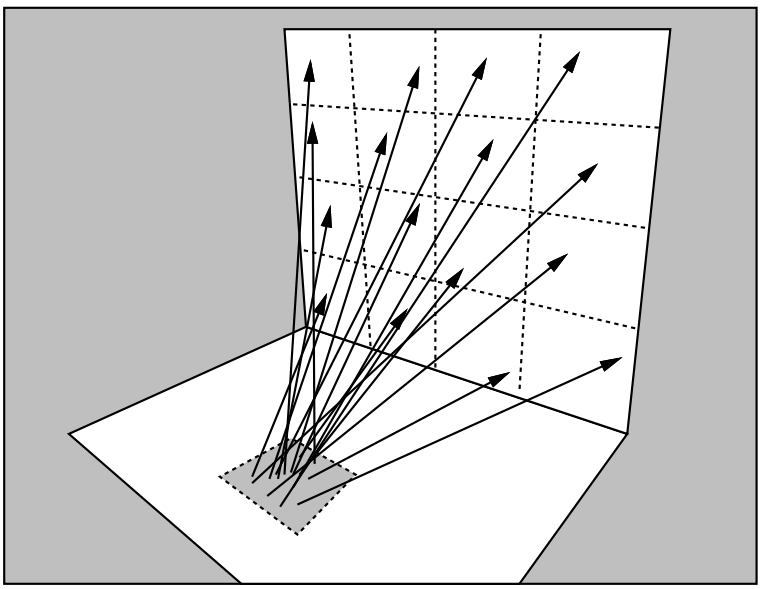

Figure 6.3: Analytic emission of energy.

Another way to send power is to explicitly calculate the energy sent from the source zone to every other zone, as shown in Figure 6.3. I call this way of transporting power an *analytic method*. If a ray between two patches is interrupted, then no power is sent between that pair. Wallace used this basic method combined with some optimizations and vertex oriented energy transport[118]. Another analytic method was used by Nishita and Nakamae who used shadow volumes to test for visibility[79]. A ray tracing-based analytic method has also been used in Illumination Engineering, though the method was restricted to rectangular zones aligned with the coordinate planes[15].

The classic way to send energy is by using a *Hemicube method*[23]. This method, shown in Figure 6.4, sends power into directional bins on the surface of a cube surrounding the zone. All of the energy sent through a directional zone goes to whatever patch is first seen through the center of the zone. Because of this the Hemicube is prone to aliasing. Artifacts arising from aliasing of the Hemicube can be lessened by increasing the Hemicube's directional resolu-

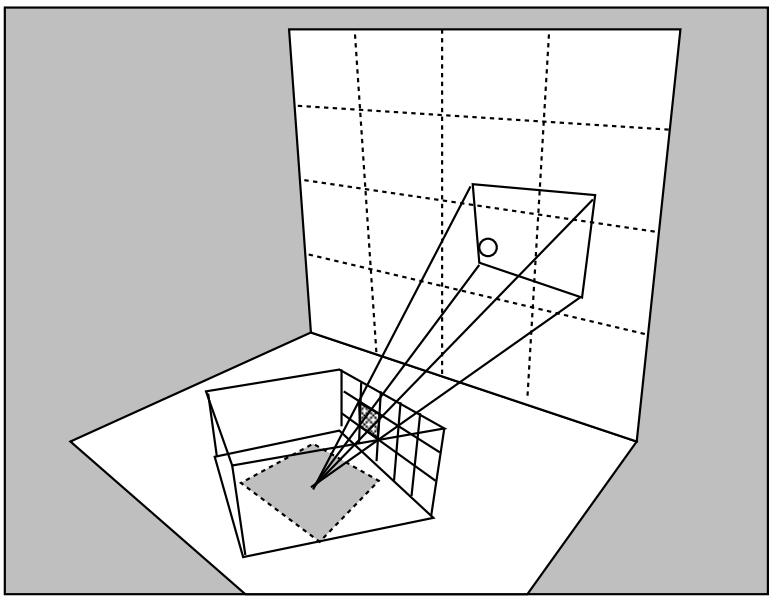

Figure 6.4: Hemicube emission of energy.

tion, rotating the hemicube by a random angle about the surface normal of the source patch, or employing correction techniques such as those presented by Baum et al.[8]. Methods of accelerating the Hemicube method by using hardware features, spatial coherence, and pixel coherence, are discussed by Rushmeier et al.[90]. One problem with the Hemicube method is that it approximates the parent patch as a point, so if a 'sending' scheme is used, shadows will be sharp.

The Hemicube and Monte Carlo methods of transporting power can be said to be in a family of methods that divide energy into angular bins. Other methods that do this are the ray tracing method of Sillion and Puech[103], and the Hemisphere method of Spencer[107]. Rather than sending power in directions, the analytic methods send power explicitly between each pair of zones. The advantage of the directional methods is that the amount of precision they employ is proportional to the solid angle subtended by the target patch. This avoids wasting much time

Figure 6.5: Zonal calculation with diffusely transmitting lampshade.

on small, far away patches. On the other hand, the error is not nearly as easy to predict as it is with analytic methods.

All of these methods could be used for diffuse transmission, as done by Rushmeier and Torrance[94]. An image with a diffusely transmitting lampshade is shown in Figure 6.5.

6.2 Optimizations for Diffuse Zonal Methods

Generating an image by the simulation methods of the last section require $O(Ns)$ where N is the number of emitting zones, and s is the amount of time it takes for one zone to send it accumulated power. This is because the solution will have an acceptable average error after a set number of reflections of light (usually 4 to 20 depending on average reflectivity in the scene), and each full set of reflections is approximated by all N zones firing their power once. Two basic optimizations are to reduce the number of patches N that send power, and to reduce the time s spent sending the power.

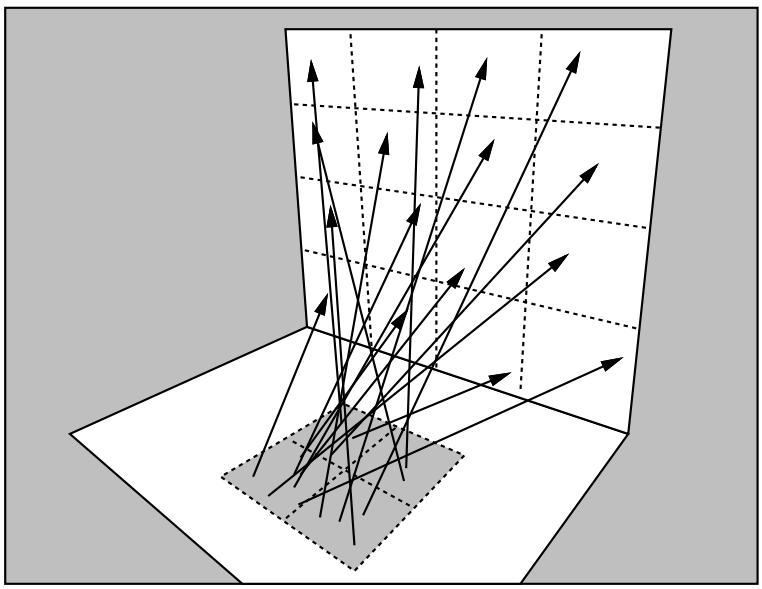

Figure 6.6: Four elements are collected into one patch before sending power.

6.2.1 Patch and Element Substructuring

The oldest optimization in zonal methods is *patch and element substructuring*[24]. Because indirect lighting is often soft, i.e. it does not change much in character over a distance, we can calculate some of this lighting with decreased accuracy. One way to do this is to collect several small zones, or *elements*, into one large *patch* which emits the accumulated power of the group of elements. The softness of the indirect component is shown in Figure 6.7, while the direct lighting can be hard, as shown in Figure 6.8. An emitting patch made up of four elements is shown in Figure 6.6. If zones can be constructed out of sets of e elements, and N is the total number of zones, then the time complexity can be reduced from the naive case of $O(Ns)$ to $O(Ns/e)$. Another speedup is to only use the elements for direct lighting, so the indirect lighting will have a reduced number of receiving zones[100]. Hanrahan and Salzmann have recently introduced a generalization of this idea where the patches are divided hierarchically into elements, and the level of size used to account for transport between two patches is based

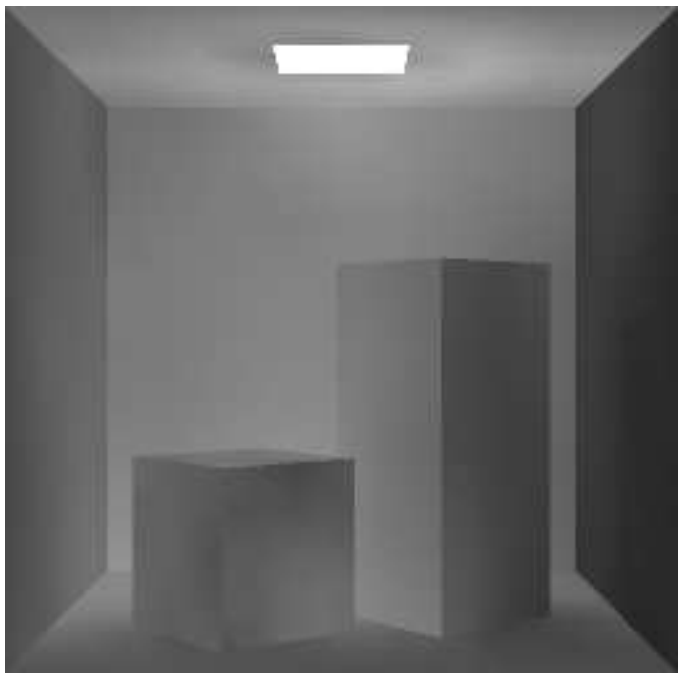

Figure 6.7: Indirect illumination.

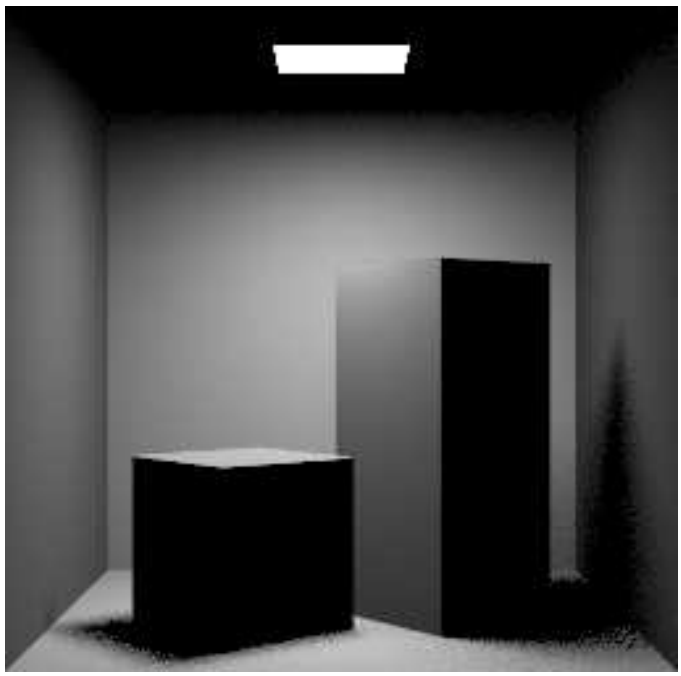

Figure 6.8: Direct illumination.

on the amount of power they exchange[48]. Though they have not yet extended their method to environments with occlusion, their initial results are very promising. Campbell and Fussell have extended adaptive meshing to non-quadtree data structures[17], and their initial results are accurate for shadows.

To get some insight into why the substructuring idea works, imagine that we are figuring the radiance at a point $\mathbf{x}$ that is due to a collection of elements. This is simply:

$$L(\mathbf{x}) = \int_{elements} \frac{R(\mathbf{x})}{\pi} L_{in}(\mathbf{x}, \psi) \cos\theta d\omega$$

We approximate this with a patch of radiance L_{ave} with the same solid angle as the elements. This amounts to approximating the cosine term with $\cos\theta_0$, where θ_0 is the angle to the center of the patch. If the patch is reasonable small, the maximum error will be small. In practice, if there are few regularities in the element radiances, the average error will be even lower.

A possible problem of all substructuring techniques is that if the initial discretization into patches is very fine, no subdivision may be needed ($e = 1$), so no speedup is attained.

6.2.2 Speeding Up the Emission of Power

The other way to speed up the zonal method is to reduce the amount of time it takes for a patch to emit power. The main way this is done is to reduce the accuracy of the solution if not much energy is being sent. Baum et al. did this by using a lower resolution Hemicube for indirect lighting[8]. Airey and Ouh-young used a Hemicube for direct lighting and then switched to ray tracing with the number of rays being proportional to the energy a patch has to send[1]. I used a strictly Monte Carlo method with the number of rays sent being set proportional to unsent power[100].

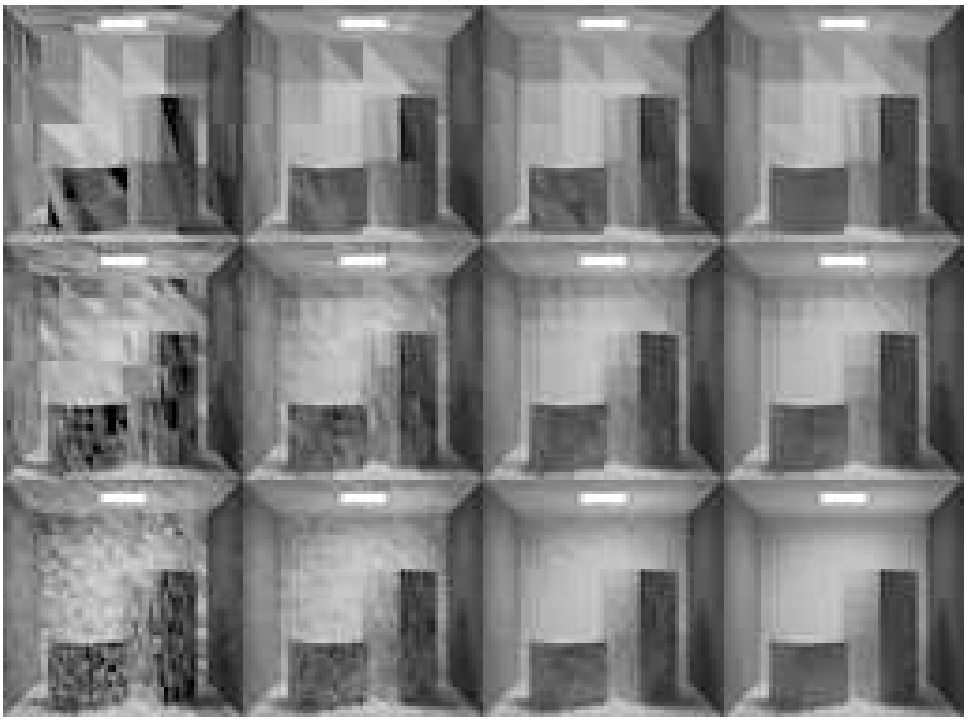

Figure 6.9: Rooms where each column has a number of rays proportional to the number of zones.

6.2.3 Optimal Time Complexity

Both Airey and I have empirically observed that the initial number of rays needed in the Monte Carlo approach is approximately proportional to the number of zones N. An example set of figures using this heuristic is shown in Figure 6.9, where the error does seem to go down consistently as the number of rays is kept proportional to the number of zones. This has the surprising implication that we can generate a zonal solution with $O(N)$ rays, so the solution time is approximately $O(\log N)$. This assumes that the average time to trace a ray is $O(N \log N)$, which is often true for divide and conquer search strategies in well-behaved scenes. However, the worst case behavior of ray tracing may be quite poor. Devillers[29] has done some initial work on the time needed to trace a ray, but it is still largely an unexplored topic. One unfortunate thing about sending rays in numbers proportional to power is that patch and