Clustering for Glossy Global Illumination[†]

Per H. Christensen *Dani Lischinski* *Eric J. Stollnitz* *David H. Salesin*

University of Washington

Abstract

We present a new clustering algorithm for global illumination in complex environments. The new algorithm extends previous work on clustering for radiosity to allow for nondiffuse (glossy) reflectors. We represent clusters as points with directional distributions of outgoing and incoming radiance and importance, and we derive an error bound for transfers between these clusters. The algorithm groups input surfaces into a hierarchy of clusters, and then permits clusters to interact only if the error bound is below an acceptable tolerance. We show that the algorithm is asymptotically more efficient than previous clustering algorithms even when restricted to ideally diffuse environments. Finally, we demonstrate the performance of our method on two complex glossy environments.

1 Introduction

Global illumination, the problem of computing an accurate solution for the illumination of a physical environment, is fundamental to computer graphics. A great deal of progress has been made on the special case of this problem known as "radiosity," in which all surfaces in the environment are ideally diffuse. While this assumption is a reasonable one for certain environments, such as interiors covered with latex paint, it is overly restrictive for many other situations that arise commonly in practice. It only takes a single glossy surface in a scene to violate the assumptions of radiosity.

The more general form of the global illumination problem, also known as "glossy global illumination," has considerably higher complexity. Whereas radiosity is concerned with how light reflecting from every surface point affects light reflecting from all other points, in glossy global illumination one must consider how light reflecting *in every direction* from every surface point affects light reflecting *in all other directions* from all other surface points. This higher dimensionality has made solving global illumination problems in complex glossy environments much more difficult than in complex diffuse environments.

In this paper we present a new algorithm for accelerating the simulation of global illumination in complex glossy environments. Our approach is inspired by recent clustering algorithms for radiosity. We introduce a new clustering scheme suitable for general glossy environments and show that this scheme is asymptotically more efficient than previous clustering algorithms for global illumination.

1.1 Previous Work

In part because of the high dimensionality of glossy global illumination, Monte Carlo methods [6, 19, 24, 29] have been the most widely used to date for simulating illumination in complex glossy scenes. However, Monte Carlo methods suffer from noisy artifacts and fairly slow convergence. One notable approach toward reducing these problems has been taken by Ward [36, 37], in which the diffuse interreflections, which are more expensive to compute with Monte Carlo, are sampled coarsely and cached.

A number of finite-element solutions have also been proposed for glossy global illumination. Immel *et al.* [18] extended the radiosity method to nondiffuse environments by uniformly discretizing the hemisphere of directions above each surface element in order to represent the directional distribution of reflected light. This uniform discretization resulted in very large systems of equations even for small environments. Sillion *et al.* [32] used spherical harmonics to represent the directional distributions of reflected light. This method used progressive radiosity, rather than a full matrix solution, and was thereby able to simulate more complex environments. However, progressive radiosity solutions typically simulate only a few bounces of light in the environment, and fully converged solutions for complex environments were still impractical.

The introduction of hierarchical radiosity [17] and wavelet radiosity [14] inspired several different hierarchical and wavelet-based algorithms for glossy global illumination [4, 8, 25, 28]. Two fundamentally different ways to parametrize the radiance function have been investigated. The "two-point transport" approach represents radiance as a function of four spatial variables, two on the source patch and two on the receiver [4, 25, 28]. Each interaction in this approach involves three patches: a source patch, a reflecting patch, and a receiver. Another approach represents radiance as a directional distribution at each point in the scene, using two spatial and two angular variables [8]. In this "radiance distribution" approach, each interaction involves only two patches, just like in the diffuse case.

The method described by Christensen *et al.* [8] can be thought of as the wavelet version of Immel's algorithm [18]: a four-dimensional wavelet representation is used to represent the spatially- and angularly-varying radiance distributions across surfaces in the environment. This representation is adaptive, sparse, and fast to evaluate. A wavelet decomposition of the transport operator results in a sparse linear system that can be solved very quickly, and importance-driven refinement [7, 35] is used to focus the computation on light transfers that have the greatest impact on the visible parts of the solution.

These hierarchical and wavelet-based algorithms are the most efficient to date for computing fully converged accurate simulations in small glossy environments. However, to obtain even the coarsest possible solution for p initial surfaces the method of Christensen *et al.* requires creating $O(p^2)$ pairs of initial interactions at the outset, while two-point transport methods start by creating $O(p^3)$ initial interactions. Because of the high complexity of the initial linking stage, no complex scenes have ever been rendered using these methods. In fact, to the best of our knowledge, the number of initial surfaces in any of the glossy environments rendered with hierarchical methods has not exceeded 152 [8].

Recently, several algorithms have been proposed for speeding up radiosity in complex diffuse environments, using some form of clustering [20, 27, 30, 33, 38]. The most recent of these algorithms [30, 33] have been motivated by efficient algorithms in the field of N-body simulation [1, 5, 15], the same algorithms that motivated hier-

[†] Per H. Christensen, Dani Lischinski, Eric J. Stollnitz, and David H. Salesin. Clustering for glossy global illumination. *ACM Transactions on Graphics*, January 1997 (to appear).

archical radiosity [17]. In many respects, the most successful clustering algorithm to date is that of Smits, Arvo, and Greenberg [33], which creates the clusters automatically, uses error bounds to guide the solution process, and has $O(p \log p)$ complexity.

1.2 Overview

In this paper, we describe a new clustering algorithm for accelerating glossy global illumination in complex environments. We extend the algorithm of Christensen *et al.* [8] in much the same way as Smits *et al.* [33] extended the original hierarchical radiosity algorithm [17]. We also improve the asymptotic complexity of the new clustering algorithm to $O(p)$, where p is the number of initial patches in the environment.

On an intuitive level, our algorithm works by representing a cluster of surfaces as a point source that emits and reflects light according to some directional distribution. Such an approximation is less accurate when considering interactions with other clusters that are nearby, but more accurate with clusters that are far away. By providing a bound on the error incurred, we can use this approximation to simplify the calculation of light transfer between clusters.

This approach is similar to the one taken in Greengard's fast multipole method [15], which represents the potential field due to a cluster of point masses by a multipole expansion about its center of mass. However, there are also several differences between the two methods, the most important being that in global illumination, the radiance distributions of the patches in a cluster do not superimpose linearly because of occlusion.

The idea of representing the light leaving complex geometry by a single directional distribution is not new. BRDFs are commonly used to model the reflection of light by complex microfaceted surfaces. Also, real light fixtures are specified in a similar way, with manufacturers providing goniometric diagrams from far-field measurements. Rushmeier *et al.* [27] used a directional representation of the reflectance of clusters of small surfaces. Representing clusters by means of directional distributions was also suggested by Sillion [30]. Part of the novelty of our approach is in providing bounds on the errors associated with such approximations. These bounds permit the use of approximations in a controlled fashion, when the errors are acceptable. Concurrently with this work, a similar approach has been independently investigated by Sillion *et al.* [31]. We will compare the two approaches in Section 7.

Our method not only handles glossy reflections; it also performs asymptotically faster than previous methods, even in diffuse environments. The applicability of our method to diffuse environments is not so surprising if one observes that a collection of diffuse surfaces, when considered as a cluster, is likely to have an overall reflectance that is highly directional. Thus, the directional distributions that our algorithm uses could be considered a natural representation for clusters, even in diffuse environments.

The rest of this paper is organized as follows. Section 2 describes in more detail the clustering method for diffuse environments of Smits *et al.* [33], which most closely resembles our own technique. In Section 3, we derive an importance-weighted bound on the contribution to the image of light transferred between two glossy clusters. Section 4 describes an algorithm that uses these bounds to compute a glossy global illumination solution. Section 5 covers the pre- and postprocessing stages of our algorithm, which are more implementation dependent. Section 6 discusses the results of our tests and provides comparisons with other methods. Finally, in Section 7, we give some conclusions and directions for future work.

2 Clustering for diffuse environments

Algorithms for diffuse global illumination attempt to solve the equation

$$L(x) = L^e(x) + \int f_r(x)\, G(x,y)\, L(y)\, dy, \qquad (1)$$

where L is the unknown equilibrium distribution of radiance, L^e is the emitted radiance, and $f_r(x)$ is the bidirectional reflectance distribution function (BRDF) of the surface at x, assumed to be independent of direction. The remaining part of the kernel, which only depends on geometry, is

$$G(x,y) = \frac{\cos\theta_x \, \cos\theta_y}{\|x-y\|^2}\, V(x,y),$$

where θ_x and θ_y are the angles between the surface normals and the line connecting x and y, and $V(x,y)$ is 1 if x and y are mutually visible and 0 otherwise.

Finite-element methods for solving this equation use a set of basis functions to represent the unknown radiance function. Once the influence of each basis function on every other basis function has been computed, the coefficients of the basis functions are found by solving a large linear system.

One of the difficulties associated with such methods is the sheer number of basis functions required to obtain a satisfactory solution for a typical scene. This problem is addressed in part by the hierarchical radiosity (HR) algorithm described by Hanrahan *et al.* [17]. HR uses a hierarchical set of basis functions, associating a single basis function with each of the input surfaces at the coarsest level. One basis function is then allowed to interact with another only if the error in the interaction falls below a given threshold. In general, coarse basis functions suffice for long distances and dim interactions, while finer basis functions are required for shorter distances and brighter interactions.

As pointed out by Smits *et al.* [33] and others, the main deficiency of HR is that the coarsest emitters and receivers are the input surfaces defining the environment. Thus, if p is the number of input surfaces, HR must begin by computing $O(p^2)$ interactions. For complex environments the cost of creating these initial links becomes dominant, and obtaining even the coarsest solution becomes prohibitively expensive.

A promising approach by Smits *et al.* [33] to the initial linking problem of HR is to group input surfaces together into clusters, thereby obtaining a smaller set of coarser entities to begin with. This technique extends the HR hierarchy upward toward a single cluster containing all surfaces. Two clusters R and S can then be linked if the interaction between them has a low enough error bound. Linking the two clusters allows the algorithm to avoid having to link all the input surfaces contained in R to those contained in S.

Note that in HR, m surfaces can be linked to n other surfaces using mn links. Smits *et al.* introduced two different types of links between clusters, α-links and β-links. These links correspond to two different bounds on an interaction, each of which can be computed in less than $O(mn)$ time, yielding an algorithm of improved asymptotic complexity.

The α-links transport light between two clusters by first estimating how much each patch in the source cluster S contributes to the receiving cluster R, taking into account the orientation of each source patch with respect to R. The sum of these contributions is then distributed to each patch in R, taking into account the orientation of each receiving patch with respect to S. Computing a bound for an α-link involves two sums: one over the patches in S and another over the patches in R. The time complexity of computing the error bound for an α-link is therefore $O(m{+}n)$, if S and R contain m and n patches, respectively. Smits *et al.* show that linking a hierarchy of clusters containing p initial patches to a fixed error tolerance using α-links results in $O(p)$ links with total time and space cost of $O(p \log p)$.

An asymptotically lower cost can be achieved using β-links, which ignore the orientations of the individual patches in each cluster. The bound on a β-link interaction can be computed in constant time, provided that each cluster stores the maximum radiance value of the patches it contains. Bounds on β-links are very coarse, because they are obtained by assuming that every source patch is directly facing and visible to every other receiving patch, that every receiving patch is highly reflective, and that all the patches in one cluster are as close as possible to the other cluster. Linking an entire hierarchy with p initial patches to a given tolerance using β-links results in $O(p)$ links, with only $O(p)$ total cost. However, because β-links have such crude bounds, Smits *et al.* use them to represent only the most negligible interactions between clusters. Most interactions require the use of α-links, and thus, the total cost of their clustering algorithm is $O(p \log p)$.

3 Clustering for glossy environments

Glossy global illumination algorithms attempt to solve a more general version of equation (1):

$$L(x,\vec{\omega}) = L^e(x,\vec{\omega}) + \int f_r(\vec{\omega}_{yx}, x, \vec{\omega})\, G(x,y)\, L(y,\vec{\omega}_{yx})\, dy, \qquad (2)$$

where $L(x,\vec{\omega})$ is the radiance leaving point x in direction $\vec{\omega}$, and $\vec{\omega}_{yx}$ denotes the direction from y to x. The units of radiance are watts per square meter per steradian [W/m^{2}sr]. The BRDF $f_r(\vec{\omega}_{yx}, x, \vec{\omega})$ is the ratio of the radiance reflected from x in direction $\vec{\omega}$ to the differential irradiance from the incident direction $\vec{\omega}_{yx}$.

One way to extend HR to solve equation (2) is by storing the directional distribution of radiance leaving each patch [8]. This approach to glossy global illumination suffers from the same initial linking problem that plagues HR. In order to apply clustering to hierarchical glossy global illumination we need to derive a method for efficiently bounding and approximating the transfer between clusters containing surfaces with arbitrary BRDFs and directional radiance distributions.

Smits *et al.* gave error bounds for the diffuse case using several norms. In the glossy case it is difficult to produce a useful ∞-norm bound, but in Sections 3.1 and 3.2 we will derive a bound for the importance-weighted 1-norm. (In fact, Smits *et al.* suggest that the 1-norm is more appropriate for computing a global illumination solution, as it gives an indication of the total "incorrect" energy in the environment, and assigns weights to surfaces according to their area.)

In Section 3.3 we introduce a cluster representation that allows us to bound an interaction between two clusters in constant time. Each cluster is represented as a point with directional distributions of outgoing and incoming radiance and importance. Collapsing clusters into points in this manner produces sufficiently accurate approximations if the interacting clusters are distant from each other. The bound can be computed in constant time because it does not require any knowledge of the geometry inside the clusters; all the necessary information is encoded in the directional distributions of the interacting clusters.

The cost of maintaining such a representation with a cluster depends on its directional resolution, but not on the number of links it has or on the number of patches it contains. Note that even when a cluster contains purely diffuse surfaces, it typically reflects different amounts of light in different directions. Thus, our representation is appropriate for both glossy and diffuse environments, yielding a much more accurate approximation in the diffuse case than either β-links or Sillion's density volumes [30], both of which treat clusters as isotropic entities.

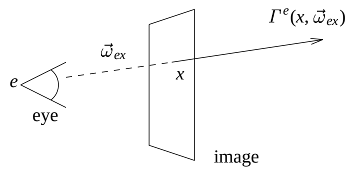


Figure 1 Emitted importance $\Gamma^e(x,\vec{\omega})$.

We shall refer to an interaction between a pair of clusters with directional distributions as an Ω-link. Because we can compute a bound for an Ω-link in constant space and time, our algorithm requires only $O(p)$ space and time for a hierarchy of clusters containing p input patches. Thus, the complexity of our algorithm is an improvement over the $O(p \log p)$ complexity of the diffuse-case algorithm of Smits *et al.*

3.1 Review of importance

Before we derive error bounds on an interaction between two clusters, we briefly review how importance can be used to measure the impact of an interaction on the image being created. Weighting our bounds by importance will enable us to focus on refining the transfers that contribute the most to the light reflected towards the eye. As demonstrated by Smits, Arvo, and Salesin [35], importance-driven refinement can yield very significant speedups in complex diffuse environments. Importance has also proved very effective in glossy environments [3, 8].

In earlier work, we introduced a directional variant of importance [7, 8]. Directional importance is defined as a unitless outgoing quantity that can be transported in exactly the same manner as radiance. Thus, the transport equation for directional importance is

$$\Gamma(x,\vec{\omega}) = \Gamma^e(x,\vec{\omega}) + \int f_r(\vec{\omega}_{yx}, x, \vec{\omega})\, G(x,y)\, \Gamma(y,\vec{\omega}_{yx})\, dy. \qquad (3)$$

The only difference between this equation and the transport equation for radiance (equation (2)) is that directional importance is typically emitted by the image plane, while radiance is emitted by the light sources. The emitted directional importance is defined as

$$\Gamma^e(x,\vec{\omega}_{ex}) = \begin{cases} 1, & \text{if } x \text{ is a point on the image;} \\ 0, & \text{otherwise.} \end{cases}$$

See Figure 1 for an illustration of this emission term. Under this definition of the emitted importance, $\Gamma(x,\vec{\omega})$ tells us how much of the light arriving at x from direction $-\vec{\omega}$ ultimately reaches the eye.

Christensen *et al.* [8] used the transport equation for directional importance (equation (3)) to derive the amount of power contributed to the image by a particular interaction. The amount of power that is transported directly from patch s to patch r and that ultimately reaches the image is given by

$$\Phi_{r,s} = \int_r \int_s \Gamma(x,\vec{\omega}_{xy})\, G(x,y)\, L(y,\vec{\omega}_{yx})\, dy\, dx. \qquad (4)$$

Importance is typically used for view-dependent solutions. If a view-independent solution is desired, importance $\Gamma(x,\vec{\omega})$ can be simply replaced by the directional-hemispherical reflectance $\rho_{\mathrm{dh}}(x,\vec{\omega})$ in the derivations that follow. This reflectance describes how much of the flux incident at x from direction $-\vec{\omega}$ is reflected to the rest of the environment [22].

3.2 Bounding glossy transfer between clusters

We will now extend the error bounds derived by Smits *et al.* [33] to incorporate directional radiance and directional importance. Smits *et al.* compute an error bound on an interaction by bounding it from above and below. Any approximation that lies within the upper and lower bounds clearly has an error less than the difference between them. Since lower bounds on visibility between clusters are hard to compute, Smits *et al.* set the lower bound on interactions between clusters to zero. As a result, the error bound becomes the upper bound on the interaction.

Following Smits *et al.* we denote the maximum value of a function f over some domain $A \times B$ by

$$\lceil f \rceil_{A,B} = \max_{x \in A,\, y \in B} f(x, y)$$

Consider first an interaction between two patches, r and s. Replacing the integration over patches r and s in equation (4) with the product of their areas times an upper bound on the integrand, we obtain the bound

$$\begin{aligned}\Phi_{r,s} &\leq A_r A_s \lceil \Gamma(x, \vec{\omega}_{xy})\, G(x, y)\, L(y, \vec{\omega}_{yx}) \rceil_{r,s} \\ &\leq A_r A_s \left\lceil \Gamma(x, \vec{\omega}_{xy})\, \frac{\cos\theta_x \, \cos\theta_y}{||x - y||^2}\, V(x, y)\, L(y, \vec{\omega}_{yx}) \right\rceil_{r,s}\end{aligned}$$

Splitting this upper bound into several parts yields

$$\begin{aligned}\Phi_{r,s} &\leq A_r A_s \lceil \Gamma(x, \vec{\omega}_{xy}) \, \cos\theta_x \rceil_{r,s} \left\lceil \frac{1}{||x - y||^2} \right\rceil_{r,s} \\ &\qquad \cdot \lceil V(x, y) \rceil_{r,s} \lceil L(y, \vec{\omega}_{yx}) \, \cos\theta_y \rceil_{r,s} \\ &\leq A_r \lceil \Gamma \cos\theta_x \rceil_{r,s} \frac{1}{(d_{r,s})^2} A_s \lceil L \cos\theta_y \rceil_{r,s}\end{aligned}$$

Here the upper bound on visibility $V(x, y)$ is simply set to 1, and $d_{r,s}$ denotes the minimum distance between the patches r and s.

This bound easily extends to a bound on the transfer between two clusters R and S. We only need to replace maxima over the areas of the patches with maxima over the bounding volumes of the clusters, and sum over the patches in each of the clusters:

$$\Phi_{R,S} \leq \left(\sum_{r \in R} A_r \lceil \Gamma \cos\theta_x \rceil_{r,S} \right) \frac{1}{(d_{R,S})^2} \left(\sum_{s \in S} A_s \lceil L \cos\theta_y \rceil_{R,s} \right) \tag{5}$$

The term $\lceil \Gamma \cos\theta_x \rceil_{r,S}$ is the importance (weighted by a cosine) of a receiving patch $r \in R$ maximized over all directions towards cluster S. The term $\lceil L \cos\theta_y \rceil_{R,s}$ is the radiance (weighted by a cosine) of a sending patch $s \in S$ maximized over all directions towards cluster R.

3.3 Bounds computable in constant time

We could use the bound in equation (5) as a generalization of the α-links of Smits *et al.* [33] to directional radiance and importance. This bound involves a sum over source patches and a sum over receiving patches, yielding a clustering algorithm of complexity $O(p \log p)$. We will now describe how storing directional information with each cluster can be used to yield bounds that are computable in constant time, leading to a clustering algorithm of complexity $O(p)$.

As mentioned earlier, we would like to treat clusters as point sources with angular distributions. With this goal in mind, we define a maximum outgoing radiant intensity distribution $\bar{I}(\vec{\omega})$, which gives an upper bound on the radiant intensity leaving a source cluster S in direction $\vec{\omega}$:

$$\bar{I}(\vec{\omega}) = \sum_{s \in S} A_s \max_{y \in s} \left[L(y, \vec{\omega}) \, \cos\theta_y \right]$$

The units of radiant intensity are watts per steradian [W/sr]. Similarly, we define a maximum outgoing "importance intensity" distribution $\overline{\Upsilon}(\vec{\omega})$, which gives the importance intensity leaving a receiving cluster R in direction $\vec{\omega}$:

$$\overline{\Upsilon}(\vec{\omega}) = \sum_{r \in R} A_r \max_{x \in r} \left[\Gamma(x, \vec{\omega}) \, \cos\theta_x \right]$$

Importance intensity has units of projected area [m^2]. It follows from these definitions that if we maximize over directions between clusters R and S we get the bounds

$$\sum_{s \in S} A_s \lceil L \cos\theta_y \rceil_{R,s} \leq \left\lceil \bar{I}(\vec{\omega}_{yx}) \right\rceil_{R,S}$$

and

$$\sum_{r \in R} A_r \lceil \Gamma \cos\theta_x \rceil_{r,S} \leq \left\lceil \overline{\Upsilon}(\vec{\omega}_{xy}) \right\rceil_{R,S}$$

Finally, substituting the previous two inequalities into equation (5), we get

$$\Phi_{R,S} \leq \left\lceil \overline{\Upsilon}(\vec{\omega}_{xy}) \right\rceil_{R,S} \frac{1}{(d_{R,S})^2} \left\lceil \bar{I}(\vec{\omega}_{yx}) \right\rceil_{R,S} \tag{6}$$

Thus, to estimate a bound on the transfer between R and S we need to approximate the minimum distance between clusters as well as the maxima of $\overline{\Upsilon}(\vec{\omega}_{xy})$ and $\bar{I}(\vec{\omega}_{yx})$ over the directions between the two clusters. Note that the time required to perform this computation will not depend on the number of patches within each of the participating clusters if we store a fixed-size precomputed representation of $\overline{\Upsilon}(\vec{\omega})$ and $\bar{I}(\vec{\omega})$ with each cluster.

4 The algorithm

In this section we describe a new clustering algorithm for glossy global illumination that uses the bounds on intercluster transfers derived in the previous section. The algorithm is an extension of a hierarchical algorithm for glossy global illumination that represents the light in the scene as a function of two spatial and two angular variables on each surface patch [8].

The algorithm starts by constructing a cluster hierarchy containing all the initial patches in the environment. Each cluster has several directional distributions describing its estimated outgoing and incoming radiance and importance, upper bounds on the radiance and importance leaving the cluster in each direction, and internal visibility in each direction inside its parent (see Section 5.3). The outgoing cluster distributions are initialized by pulling emitted radiance and initial importance from the patches towards the root of the cluster hierarchy. Initially, a single link is established from the root cluster to itself. The algorithm then alternates between refining the links and solving the resulting linear system, until reaching some final tolerance.

This process is summarized in the following pseudocode:

```
/* Preprocess: */
Construct cluster hierarchy
Initialize approximate visibility data structures
Pull emitted radiance and importance from patches to clusters
for a series of decreasing tolerances $\epsilon$ do
    /* Refine: */
    for each link $\ell$ do
        if ErrorBound($\ell$) $> \epsilon$ then Refine($\ell$)
    end for
    /* Solve: */
    for $k$ iterations do
        Transport radiance and importance along links
        Push radiance and importance from clusters to patches
        Pull radiance and importance from patches to clusters
    end for
end for
/* Render: */
Final gather
```

In our implementation, the series of refinement tolerances is specified by the user. The number of iterations k was set to three in all of the examples shown in Section 6.

In the rest of this section we concentrate on the main components of this algorithm: the representation of clusters, the refinement process, and the solution process. In Section 5, we describe the more implementation-dependent preprocessing and postprocessing stages of the algorithm, which include hierarchy construction, visibility preprocessing, and rendering. We also comment on the space and time complexity of each portion of the algorithm, both here and in Section 5.

4.1 Cluster representation of radiance and importance

In the transfer of radiance and importance between clusters, we will ignore each cluster's geometric extent, and treat a cluster as a point located at the center of its bounding volume. With each cluster, we store seven directional distributions. In describing these distributions, we will use r to denote a patch acting as a receiver and s to denote a patch acting as a source.

The quantities stored with each cluster include:

1. The radiant intensity distribution $I(\vec{\omega})$, which gives the radiant intensity [W/sr] leaving the cluster in the direction $\vec{\omega}$:

$$I(\vec{\omega}) = \sum_s \int_s L(y,\vec{\omega})\ \cos\theta_y\ V(y,\vec{\omega})\, dy,$$

   where V accounts for internal occlusion within the cluster, with $V(y,\vec{\omega})$ equal to 1 if the ray leaving y in direction $\vec{\omega}$ does not intersect any of the surfaces inside the cluster and 0 otherwise. (Section 5.3 describes how this internal visibility is computed.)

2. The directional irradiance distribution $L^{\text{in}}(\vec{\omega})$, which gives the flux per projected area [W/m^2] arriving at the cluster from direction $\vec{\omega}$.

3. The "importance intensity" distribution $\Upsilon(\vec{\omega})$, which gives the total importance [m^2] leaving the cluster in the direction $\vec{\omega}$:

$$\Upsilon(\vec{\omega}) = \sum_r \int_r \Gamma(x,\vec{\omega})\ \cos\theta_x\ V(x,\vec{\omega})\, dx.$$

4. The incoming directional importance distribution $\Gamma^{\text{in}}(\vec{\omega})$, which gives the directional importance [sr] arriving at the cluster from direction $\vec{\omega}$.

In order to efficiently bound the transfer between clusters as described in Section 3.3, each cluster must also store:

5. The maximum radiant intensity distribution $\bar{I}(\vec{\omega})$, which gives an upper bound on the radiant intensity [W/sr] leaving the point in the direction $\vec{\omega}$:

$$\bar{I}(\vec{\omega}) = \sum_s A_s \max_{y\in s} \left[L(y,\vec{\omega})\cos\theta_y\right]$$

6. The "maximum importance intensity" distribution $\overline{\Upsilon}(\vec{\omega})$, which gives an upper bound on the total importance [m^2] of the contained patches in direction $\vec{\omega}$:

$$\overline{\Upsilon}(\vec{\omega}) = \sum_r A_r \max_{x\in r} \left[\Gamma(x,\vec{\omega})\cos\theta_x\right]$$

Finally, in order to account for occlusion within clusters when pulling and pushing (Section 4.4), each cluster also stores:

7. The internal visibility distribution $V(\vec{\omega})$, which gives the probability that a ray leaving the cluster in direction $\vec{\omega}$ will exit the parent cluster without being occluded. The computation of $V(\vec{\omega})$ is described in Section 5.3.

A directional distribution can be approximated using any finite set of basis functions $[b_1(\vec{\omega}), \ldots, b_N(\vec{\omega})]$ defined over the sphere. Thus, each of the distributions described above is represented as an array of coefficients, with one coefficient for each basis function. For example, the outgoing radiant intensity $I(\vec{\omega})$ is represented as a linear combination

$$I(\vec{\omega}) = \sum_{i=1}^{N} I_i\, b_i(\vec{\omega}).$$

To simplify transfer computations, the supports of the basis functions should not overlap.

In our implementation we use thirty-two piecewise-constant basis functions. Each coefficient of a distribution describes the average magnitude of the represented quantity over the solid angle corresponding to the support of the basis function. For convenience, we divide the sphere of directions into two hemispheres and use the same method that we use to parameterize the hemisphere of directions for a patch [8]. This parameterization transforms the hemispherical domain to a unit square using a gnomonic projection followed by a "stretch," as shown in Figure 2.

Complexity analysis: The storage required for a cluster is proportional to the number of directional basis functions used, but does not depend on the number of patches in the cluster. Each cluster therefore requires a constant amount of storage, for a fixed directional resolution. Because the number of clusters depends linearly on the number of input patches p, the entire cluster hierarchy requires $O(p)$ storage.

4.2 Bounding and refining the transfer

It follows from equation (6) that in order to estimate a bound on the transfer between two clusters R and S we need to approximate the minimum distance between the clusters as well as the maxima of $\overline{\Upsilon}(\vec{\omega}_{xy})$ and $\bar{I}(\vec{\omega}_{yx})$. We first check to see if the bounding volumes

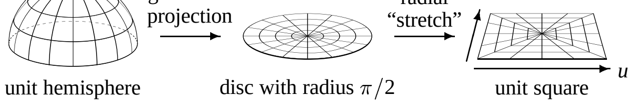


Figure 2 Angular projection: gnomonic projection and radial "stretch."

of the clusters overlap. If this is so, the bound is set to infinity. When the two clusters do not overlap, we estimate the bound similarly to Smits *et al.*: First we randomly choose a fixed number of pairs (x, y), where x is in the bounding volume of R and y is in the bounding volume of S. Then for each pair (x, y), we evaluate $\overline{\Upsilon}(\vec{\omega}_{xy})$, $\bar{I}(\vec{\omega}_{yx})$, and $||x - y||^{-2}$, recording the maximum value for each of these three quantities. The product of the three maxima is taken as an estimate of the bound in equation (6).

If the bound corresponding to a link exceeds the current tolerance, the link is refined. In our implementation, a link between two clusters is refined by splitting the "largest" cluster, linking its children to the "smaller" cluster, and recursively considering those new links for refinement. We define the size of a cluster as the sum of the areas of the patches contained within it. For a link between a patch and a cluster, the cluster end of the link is always refined. A link between two patches is refined as described by Christensen *et al.* [8]. New links that are established in the refinement process will be used later to transport energy, so an approximate visibility term is computed and stored with each link when it is created, as described in Section 5.2.

Another possible type of refinement is to increase the accuracy of the directional distributions stored with each cluster, rather than replacing links between two clusters with links between their subclusters. We avoid increasing the angular resolution of directional distributions in order to maintain constant space requirements for clusters (and time requirements for updating them). It is important to emphasize that using a fixed resolution for clusters does not limit the accuracy of the simulation: so long as the error in each transfer is bounded, all important and significant transfers will eventually be refined down to the patches. At this point, the algorithm of Christensen *et al.* [8] will resolve these transfers to the required accuracy.

Complexity analysis: Following Greengard [15], Hanrahan *et al.* [17], and Smits *et al.* [33], we assume that the number of links from each cluster to other clusters in the hierarchy is a constant that depends on the current tolerance, but not on the number of patches p. Because the hierarchy contains $O(p)$ clusters, there are $O(p)$ cluster-to-cluster links that can possibly be refined. The time spent in the refinement stage is therefore $O((t_b + t_v)p)$, where t_b is the time to compute a bound on the transfer between two clusters and t_v is the time to estimate the visibility between the clusters. As we mentioned earlier, we can bound the transfer in constant time. In Section 5.2, we describe a particular scheme for approximating visibility calculations that takes constant time as well. As a result, each refinement stage requires only $O(p)$ time. (The corresponding cost in the algorithm described by Smits *et al.* is $O(p \log p)$, because their α-links cannot be bounded in constant time.)

4.3 Transfer between clusters

To transfer light along a link from a source cluster S centered at y to a receiving cluster R centered at x, we need to convert the radiant intensity leaving y into an incoming quantity arriving at x.

Consider a differential receiving element centered at x. The differential flux $d\Phi$ arriving at x is given by the product of the radiant intensity of the source in the direction of the receiver, the visibility between the source and the receiver, and the solid angle $d\omega_x$ subtended by the receiver (as seen from the source):

$$d\Phi = I(\vec{\omega}_{yx})\, V(x, y)\, d\vec{\omega}_x = I(\vec{\omega}_{yx})\, V(x, y)\, \frac{\cos\theta_x\, dA_x}{||x - y||^2}$$

In order to convert this incoming flux into a quantity that is independent of the receiver's area and orientation, we divide by the projected area $\cos\theta_x\, dA_x$ to obtain

$$\frac{d\Phi}{\cos\theta_x\, dA_x} = \frac{I(\vec{\omega}_{yx})\, V(x, y)}{||x - y||^2}$$

The resulting incoming quantity is similar to irradiance, but has units of watts per *projected* area. We refer to it as *directional irradiance* and denote it by L^{in}. Thus, if the support of the i-th basis function contains the direction $\vec{\omega}_{yx}$, and the support of the j-th basis function contains the direction $\vec{\omega}_{xy}$, the transfer from cluster S to cluster R is performed by updating the j-th directional irradiance coefficient of R as follows:

$$L_j^{\text{in}} \leftarrow L_j^{\text{in}} + \frac{I_i\, V_{SR}}{||x - y||^2}$$

where the visibility V_{SR} is a fraction between 0 and 1 representing the average visibility between clusters S and R. Visibility between clusters is approximated as described in Section 5.2. This approximation is computed once when the link is established, and the result is stored with the link.

The transfer of importance from R to S is performed analogously:

$$\Gamma_i^{\text{in}} \leftarrow \Gamma_i^{\text{in}} + \frac{\Upsilon_j\, V_{SR}}{||x - y||^2}$$

Complexity analysis: Transporting light and importance along a link between two clusters takes a constant amount of time, since all the quantities involved in the updates are stored with the two clusters and the link between them.

4.4 Pulling and pushing

We use the terms "pull" and "push" in the same sense as in the hierarchical radiosity algorithm [17], except that pulling begins at the patch level and combines outgoing quantities to give values to coarser clusters in the hierarchy, while pushing distributes incoming quantities from the coarsest cluster down the hierarchy toward the patches.

When we pull radiant intensity up to a parent cluster from its children, we must first convert the outgoing radiance of each child patch s into a radiant intensity distribution I^s:

$$I^s(\vec{\omega}) = \int_s L(y, \vec{\omega})\, \cos\theta_y\, dy.$$

The coefficients I_i^s corresponding to the directional basis functions are obtained by sampling the integrand of this expression at a number of points y and directions $\vec{\omega}$. The parent cluster's radiant intensity is then given by summing the radiant intensities of its children, attenuated by any occlusion within the parent:

$$I_i^{\text{parent}} \leftarrow \sum_{\text{children}} I_i^{\text{child}}\, V_i^{\text{child}}$$

The internal visibility coefficients V_i are computed and stored with each child during a preprocessing stage, as described in Section 5.3.

Upper bounds on radiant intensity are also estimated by sampling points on the patches. The upper bounds are pulled up the cluster hierarchy in the same manner as radiant intensities, although they are not attenuated by internal occlusion.

To push the directional irradiance of a parent cluster to its children, we attenuate each coefficient L_i^{in} of the parent by the i-th internal visibility coefficient of each child. For a child that is itself a cluster, we merely add this attenuated value to the child's i-th directional irradiance coefficient. For a child that is a patch, however, we need to convert the directional irradiance of the parent cluster into radiance reflected off the patch. The radiance reflected from point x on the patch in direction $\vec{\omega}$ is given by the following integral over the hemisphere above x:

$$L(x, \vec{\omega}) = \int f_r(\vec{\omega}', x, \vec{\omega})\, V(x, \vec{\omega}')\, \cos\theta'\, dL^{\text{in}}(\vec{\omega}'),$$

where θ' is the angle between the incoming direction $\vec{\omega}'$ and the surface normal at x.

In our implementation, the directional irradiance that gets pushed down to a patch from its parent cluster is only used to update the coarsest basis function on the patch, which represents the average radiance of the patch over all locations and directions. In terms of directional distribution coefficients, this update is accomplished by increasing the coefficient of the patch's coarsest basis function by

$$\sum_i \rho_{\text{hc}}(2\pi \to \Delta\vec{\omega}_i)\, \frac{L_i^{\text{in}}}{\Delta\vec{\omega}_i}\, V_i.$$

Here ρ_{hc} is the hemispherical-conical reflectance of the patch [22] and $\Delta\vec{\omega}_i$ is the solid angle corresponding to the support of the i-th basis function on the sphere.

The quantities related to importance are pushed and pulled in exactly the same manner as those related to radiance.

Complexity analysis: Each cluster and each patch in the environment is updated exactly once per pull. The number of clusters is proportional to the number of input patches p, so the total number of updates per pull is $O(p)$. Since internal visibilities are precomputed, the cost of each update depends only on the angular resolution of directional distributions, which is fixed. Therefore, it takes $O(p)$ time to perform a pull. The argument for a push is completely analogous.

Each "gather" (transport, push, and pull) corresponds to a single Jacobi iteration. For global illumination problems, the number of iterations required for the solution to converge is typically small and independent of the number of input patches. Thus, the total time to solve the system represented by a fixed set of links among clusters and patches can be considered $O(p)$.

4.5 Discussion

Our approach to clustering, like the approach of Smits *et al.*, may raise certain concerns about the accuracy of the resulting solutions. In both clustering algorithms, transfers between clusters are approximated very coarsely. Furthermore, refining an interaction between two clusters by breaking it into several smaller pieces does not in itself guarantee an improvement in the accuracy of the resulting approximation: accuracy is increased only if the sum of the errors corresponding to these pieces is smaller than the error of the original interaction.

The philosophy behind these two clustering algorithms is that, in a typical complex environment, many of the transfers between clusters are very small because energy falls off with the square of the distance. These transfers have little or no impact on the solution, and even a very coarse approximation would suffice for them; however, we don't know *a priori* which of the transfers can be approximated in this way. Clustering provides a reliable means of determining where coarse approximations suffice. Significant transfers, on the other hand, are refined until all such interactions take place between patches, where transfers are treated more accurately.

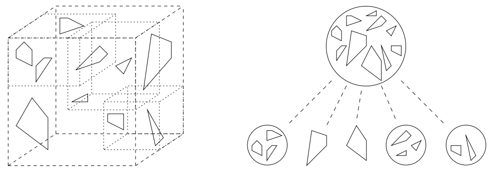

(a) (b)

Figure 3 A cluster with cluster children and patch children: (a) geometry; (b) schematic view.

5 Preprocessing and postprocessing

In this section, we describe the preprocessing and postprocessing stages of our method. Section 5.1 describes the particular technique we use to group input patches into clusters. Sections 5.2 and 5.3 discuss visibility preprocessing for visibility between clusters and for internal visibility within clusters, respectively. Finally, Section 5.4 explains how images are rendered from finite-element solutions using a final gathering pass.

5.1 Creation of the cluster hierarchy

To build the cluster hierarchy we use a top-down approach similar to that of Fournier and Poulin [9] and Glassner [11]. We start by creating a single root cluster containing all the initial surfaces in the scene. We compute the bounding box of the cluster and split it into eight octants. For each patch in the cluster, one of two things happens:

1. If the extent of the patch along all of the principal axes is less than that of an octant, the patch is assigned to the octant containing its centroid.
2. Otherwise, the patch becomes a direct child of the cluster.

Following the patch classification process, every octant containing more than a few patches is made into a child cluster. (Patches in the remaining nearly empty octants become direct children of the parent cluster.) The above process is repeated recursively on any newly created clusters. The recursion terminates when the number of patches in a cluster is sufficiently small.

The above algorithm assigns each patch to a single cluster, without splitting any of the patches. The construction results in a hierarchy of clusters whose bounding boxes may overlap. However, the bounding box of each child cluster is strictly contained within that of its parent. The children of a cluster can be smaller clusters as well as patches, as illustrated in Figure 3.

Complexity analysis: If we assume a relatively uniform distribution of p patches, the cluster construction described above will result in a hierarchy with $O(\log p)$ levels. At each level of the hierarchy, the construction has to consider $O(p)$ patches to distribute them into octants. The construction of the entire hierarchy therefore takes $O(p \log p)$ time. Note that a similar hierarchy could also be constructed from the bottom up, by first associating each input patch with a cell on some fine uniform grid. Then adjacent nonempty cells could be grouped together into larger cells. This approach would construct a hierarchy of depth $O(\log p)$ in $O(p)$ time. Our experience has been that the hierarchy construction takes only a small fraction of the total running time, so we have not found it necessary to improve the complexity of this stage.

5.2 Visibility between clusters

Each time the refinement stage described in Section 4.2 creates a new link, we need to estimate the visibility for that link. If the refinement stage is to spend only $O(p)$ time on links between clusters, we need to compute visibility in constant time. Unfortunately, using ray casting with a BSP-tree [10] or a hierarchy of bounding volumes [12] results in an expected cost of $O(\log p)$ per ray. More sophisticated acceleration schemes for ray tracing, such as the ray classification scheme proposed by Arvo and Kirk [2], or the ray coherence scheme of Ohta and Maekawa [23], reportedly exhibit near-constant expected time per ray, but these schemes are nontrivial to implement.

In our current implementation, we adopt a simple approximate visibility scheme that is based on the isotropic volume density approximation described by Sillion [30]. As a preprocessing step, we construct a uniform voxel grid of some fixed resolution over the scene's bounding box. The resolution of the grid is specified as a parameter to our method, and it does not depend on the number of objects in the scene. Each voxel in the grid is assigned a density (extinction coefficient) κ proportional to the total area of the patches the voxel contains.

Once the densities of the voxels have been assigned, the visibility between any two points x and y can be approximated by casting a ray from x to y and visiting each voxel intersected by the ray. The attenuation of the ray traveling a distance d through a voxel is given by $e^{-\kappa d}$. The visibility of the entire ray is approximated by the product of the attenuations inside each voxel. The average visibility for a link between clusters is estimated by tracing a fixed number of rays, as described above, and averaging their values.

For average visibility between a pair of patches, we need greater accuracy than the isotropic volume density approximation can provide. In this case, we use the conventional approach of shooting a fixed number of rays between the two patches, and intersecting each ray with the occluding geometry.

Complexity analysis: Assigning densities to voxels in the preprocessing stage involves computing the fraction of each patch's area that lies within each voxel. This can be accomplished in $O(p)$ time.

The estimation of average visibility for each new cluster-to-cluster or cluster-to-patch link takes constant time: since the total number of voxels is fixed, even the longest ray through the environment can intersect at most a fixed number of voxels.

5.3 Visibility within clusters

The pulling and pushing operations described in Section 4.4 redistribute quantities from a cluster to its children and vice versa. To account for internal occlusion within a cluster, we need to know the fraction of radiance leaving each child of the cluster that is unoccluded by other children of the cluster. There is no need to compute visibility within a cluster every time we pull or push, as the geometry and the hierarchy remain fixed. Therefore, once the cluster hierarchy has been created, we use a preprocessing step that computes the internal visibility distribution $V(\vec{\omega})$ for each child. $V(\vec{\omega})$ gives the probability that a ray leaving the child in direction $\vec{\omega}$ exits its parent cluster without occlusion. This probability is estimated by shooting a fixed number of rays for each directional basis function. Each ray is tested for intersection with all the siblings of the child.

Complexity analysis: Since the patches within each cluster are organized hierarchically, the expected time per ray is $O(\log n)$, for a parent cluster whose subtree contains a total of n patches. A cluster hierarchy containing p patches has $\log p$ levels. It follows that a cluster at level k (the root being level 0) is the root of a subtree with $\log p - k$ levels containing $8^{\log p-k}$ patches (assuming a branching factor of 8). Therefore, a ray intersection test takes time proportional to $\log(8^{\log p-k}) = \log p - k$. There are 8^k clusters at level k, so the total time for the visibility preprocessing within clusters is given by

$$c \sum_{k=0}^{\log p-1} 8^k(\log p - k) \le cp.$$

The constant c depends on the time it takes to perform a single ray–patch intersection test, on the number of children per cluster, on the number of rays shot per directional basis function (four in our implementation), and on the number of directional basis functions (thirty-two in our implementation). All of these quantities are independent of p, and therefore the total time taken by this stage of the algorithm is $O(p)$.

5.4 Final gather

To render an image from a global illumination solution one could simply assign each pixel the radiance value leaving the corresponding visible point towards the eye. However, since the solution is a piecewise-constant approximation to the radiance function, such a rendering would produce a blocky image. Furthermore, we want a rendering algorithm that would produce images of high visual quality even from very coarse finite-element solutions. Thus, following the ideas that Reichert [26], Lischinski *et al.* [21], and Smits [34] used for radiosity, we use a *final gather* step. In this step, the image is rendered by casting rays from the eye through the pixels. At each visible point in the scene we compute the radiance outgoing towards the eye by gathering radiance one last time through all the links contributing to the illumination at that point.

To perform a final gather to point x on patch r, we first evaluate the contribution from all the patches that are linked directly to r. The radiance coming from such a patch s that is reflected at x towards the eye e is given by

$$\int_s f_r(\vec{\omega}_{yx}, x, \vec{\omega}_{xe})\, G(x,y)\, L(y, \vec{\omega}_{yx})\, dy.$$

Since x and e are fixed, the integration is only over sending positions y on patch s. This integral is approximated by sampling s, with a number of samples proportional to the solid angle subtended by s as seen from x. The radiance $L(y, \vec{\omega}_{yx})$ is given by the finite-element solution, and visibility is evaluated using ray-casting.

Other links contributing to the illumination at x include patch-to-cluster links to the cluster containing patch r, cluster-to-patch links to patch r, and cluster-to-cluster links to the cluster containing r. For a patch-to-cluster link we perform the same final gather as for links between patches. However, in this case we use the average visibility value stored with the link, instead of casting rays.

For cluster-to-patch and cluster-to-cluster links we randomly select a fixed number of points y in the source cluster S. The radiance reflected from x towards the eye due to illumination by S is given by averaging

$$f_r(\vec{\omega}_{yx}, x, \vec{\omega}_{xe})\, I(\vec{\omega}_{yx})\, V(x,y)\, \frac{\cos\theta_{xy}}{||x-y||^2}$$

over the point samples y. Again, visibility is estimated by the average value stored with the link.

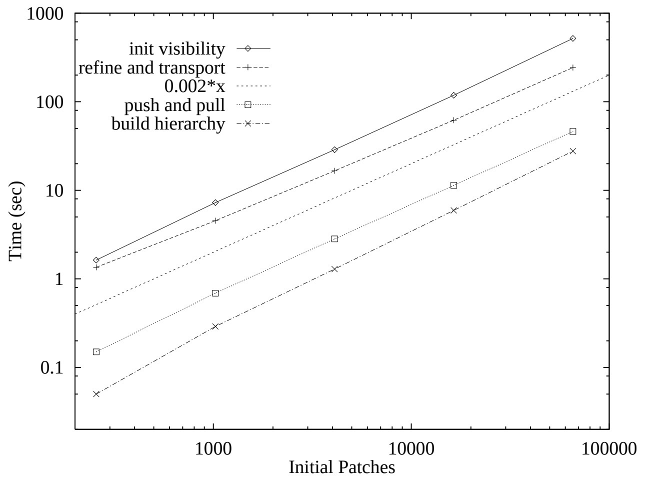


Figure 4 The time spent in various stages of the new method as a function of the number of initial patches.

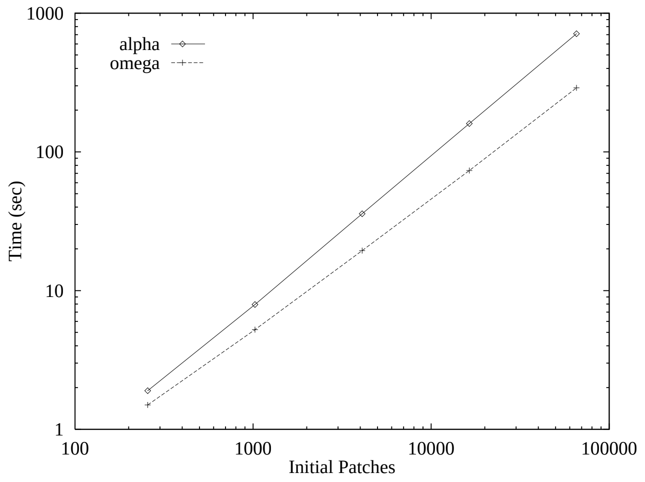


Figure 5 The time spent by different clustering algorithms as a function of the number of initial patches.

To produce antialiased images, we typically use sixteen samples per pixel. The direct illumination component is typically responsible for most of the variation in the radiance function across a pixel, so it is evaluated at each sample. Indirect illumination, on the other hand, often has low intensity and relatively little variation. At the same time, it is expensive to compute, so we save time by computing it only once per pixel, or even by re-using values over several pixels with low variation.

It is important to note that while the final gather considerably improves the visual quality of the solution, occasional visual artifacts may still remain. For instance, a boundary between adjacent patches may still be visible as a subtle shading discontinuity, if the link sets of the two patches differ too much. Some faint reflections might be missing if the corresponding energy transfer was too small to be sufficiently refined; other faint reflections that should not be present in the image because of occlusion might sometimes show through due to coarse visibility approximations over coarse links. All of these artifacts diminish as the finite-element solution becomes more refined.

Complexity analysis: Consider the cost for one ray cast from the eye. First, we need to find the first intersection with a patch (point x), which we assume takes $O(\log p)$ time on the average. Then we need to gather radiance across all the links that contribute to the basis functions whose support includes x. Each cluster and basis function has a constant number of links, so the total number of links to gather from is proportional to the depth of the hierarchy. There are $O(\log p)$ levels of the cluster hierarchy to gather from, and for these links visibility is computed in constant time. There are d levels of the patch basis function hierarchy to gather from, where d depends on the accuracy of the solution but not on p, and each link from a patch basis function requires $O(\log p)$ time for computing visibility. The total cost per ray from the eye is therefore $O(\log p + d \log p) = O(\log p)$. Note that the final gather, too, benefits from clustering: without clustering the solution at a surface point x could be influenced by as many as $O(p)$ links, requiring the final gather to spend $O(p \log p)$ time per ray from the eye.

6 Results

In this section we first experimentally verify the theoretical predictions regarding the $O(p)$ asymptotic complexity of our method, and compare its performance with that of Smits *et al.* Next, we examine the accuracy and effectiveness of our cluster representation. Then, we examine the results produced by our method on a highly glossy environment and make qualitative comparisons with Ward's RADIANCE system [36]. Finally, the effectiveness of the method for complex glossy environments is demonstrated using an architectural interior containing nearly 8000 initial surfaces.

6.1 Asymptotic behavior

Our first experiment is designed to examine the observed asymptotic time complexity of our method and to compare it to that of Smits *et al.* Since their algorithm was designed for diffuse global illumination, we perform the comparison using a diffuse environment. Our test environment consists of two concentric tessellated spheres: a hollow sphere of radius 2 containing a sphere of unit radius. Triangles on each sphere emit the same constant radiance. This test case is similar to the one used by Smits *et al.*, but we added the interior sphere to test the effects of occlusion.

We timed our method on tessellations containing 256, 1024, 4096, 16,384, and 65,536 triangles. Each of the runs involved constructing the cluster hierarchy, initializing internal visibilities for each cluster, pulling, refining to a specified tolerance, transporting energy through the links, and pushing. The graph in Figure 4 shows how the times spent in the different stages of our method grow with the complexity of the environment, for a fixed error tolerance. As a reference, we have also plotted the function $y = 0.002x$. This graph demonstrates that all the stages exhibit growth that is roughly linear in the environment size.

The graph in Figure 5 compares the total running time of our algorithm to that of Smits *et al.* Since their algorithm ignores internal occlusion within clusters, we omitted the time of the internal visibility precomputation from the total time for our algorithm. Both methods ran on the same machine, with the same tolerance, resulting in roughly the same number of links in each case. The difference in the asymptotic time complexities of the two methods is revealed by the difference in the slopes of the two log-log plots. Note that for the tessellation with 65,536 triangles, Ω-links exhibit a speedup by a factor of 2.5 over α-links. Since Ω-links are asymptotically more efficient than α-links, the improvement in performance should become even larger for more complex environments.

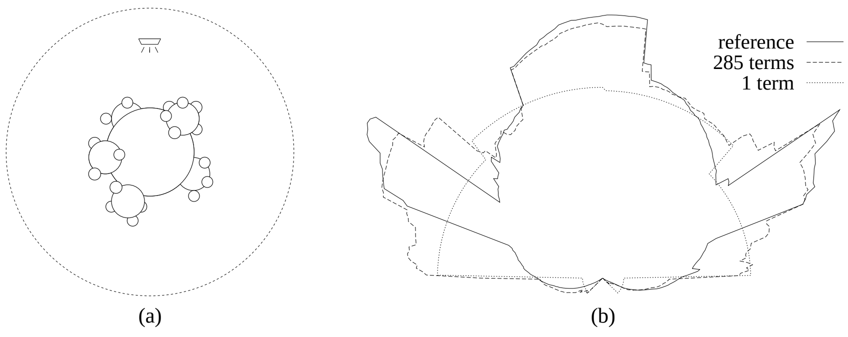


Figure 6 Glossy sphereflake: (a) a sketch of the geometry; (b) polar plots of cluster approximations. The solid line represents radiance evaluated on all 728 patches. The dashed line represents the evaluation of 285 terms: radiance from 189 patches, and radiant intensity from 96 clusters. The dotted line shows the coarsest approximation, the radiant intensity from a single cluster containing the entire sphereflake.

6.2 Accuracy of cluster approximation

To test the accuracy of the cluster approximation employed by our method we have used a “sphereflake” [16] made of 91 spheres and illuminated by a single square light source. A schematic view of the model is shown in Figure 6(a). An image of the same sphereflake (with a wood-textured polygon added behind it) is shown in Figure 7(b). The gray, cyan, magenta, and yellow spheres are diffuse; the red, green, and blue spheres are isotropically glossy; and the black, copper, and chrome spheres are anisotropically glossy. In our implementation each sphere is approximated by eight Bézier patches. In total, the sphereflake consists of 728 surface patches, of which 432 are glossy.

A cluster hierarchy containing all of the sphereflake’s patches was constructed as described in Section 5.1. The sphereflake was directly illuminated by a single light source, and the reflected radiance was pulled up through the cluster hierarchy. The reflected light was evaluated in several different ways at densely-spaced locations on an orbit around the sphereflake. The ratio of the orbit radius to the sphereflake radius was about 10:1. The results are shown in the form of polar plots in Figure 6(b). The solid curve corresponds to results obtained by directly integrating the radiance from each of the 728 patches. This curve serves as our reference solution, since no clustering approximations were used in its computation. The coarsest approximation is obtained by evaluating the radiant intensity distribution leaving the cluster that contains the entire sphereflake. Also plotted is an approximation of intermediate accuracy, where the transfer between the sphereflake and the receiver on the orbit is refined into 285 interactions: 189 with patches, and 96 with clusters.

As the polar plots in Figure 6 show, the approximation of the entire sphereflake as a single cluster is fairly coarse, yet it captures the most prominent characteristics of the radiant intensity distribution. As the approximation is refined by evaluating more terms, many more details are captured and the approximation becomes more accurate.

6.3 Results for a highly glossy environment

The sphereflake model described in the previous section was also used to test the ability of our method to correctly handle different types of interreflections in a scene with highly glossy surfaces. One solution computed by our method is shown in Figure 7. The finite-element solution shown in Figure 7(a) took 26 CPU minutes to compute on a IBM RS6000 workstation with a 100 MHz PowerPC processor. Of this time, 2 minutes were spent building the hierarchy and performing visibility preprocessing, and the remaining 24 minutes were spent refining and solving. The solution has 1,545 links between clusters, 4,789 links between a patch and a cluster, and 4,207 links between patches, which are further refined into 189,180 interactions between 104,458 wavelet coefficients representing the radiance in the environment.

The image in Figure 7(b) was computed using a final gather at a resolution of 3200×3200, and then was reduced to 800×800 pixels using a Gaussian filter. The final gather took 242 minutes in addition to the finite-element solution time, resulting in total computation time of 4.5 hours.

A careful examination of Figure 7 reveals that all of the possible combinations of diffuse and glossy transport mechanisms are present. Diffuse-to-diffuse transport creates subtle yellow, blue, and red color bleeding on the large gray sphere. Diffuse-to-specular transport creates the reflection of the diffuse spheres in the shiny ones. Specular-to-specular transport creates the reflections of highlights in the shiny spheres. Finally, specular-to-diffuse transport illuminates the base of the small yellow sphere near the top of the image.

6.4 Comparison with a Monte Carlo method

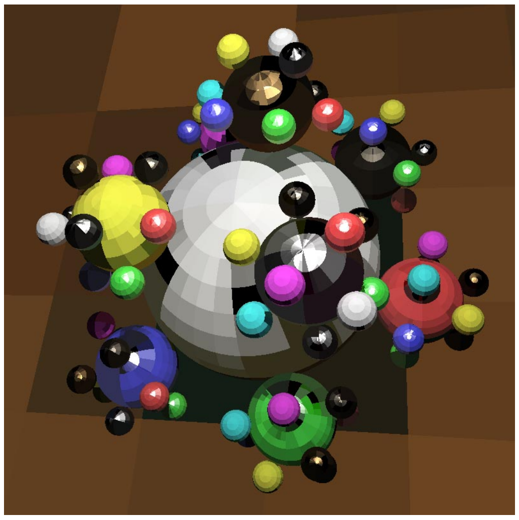
A qualitative comparison of our method and the RADIANCE system [36] was performed using the sphereflake scene described above. Both methods were used on the same machine to compute images at 3200×3200 resolution, and then the images were reduced to 800×800 pixels using a Gaussian filter. The RADIANCE system took 6.2 CPU hours, resulting in the image shown in Figure 8(a). After 4.7 CPU hours, our method produced the image in Figure 8(b)[1]. Note that the RADIANCE system had the advantage of performing all its computations with 91 spheres, while in our solution these spheres were represented as 728 Bézier patches. We did, however, use spheres for visibility rays.

Figure 8 clearly shows that our solution and that of RADIANCE converge toward the same final result. While RADIANCE is a mature product that has been debugged and optimized over the past decade, clustering and wavelet techniques for glossy global illumination are still in their infancy. Undoubtedly, our method could benefit enormously from further algorithmic refinement and fine-tuning, and our implementation could benefit from further debugging and optimization. Our conclusion from this experiment is that hierarchical finite-element methods are a viable and promising alternative to Monte Carlo for efficiently simulating glossy global illumination.

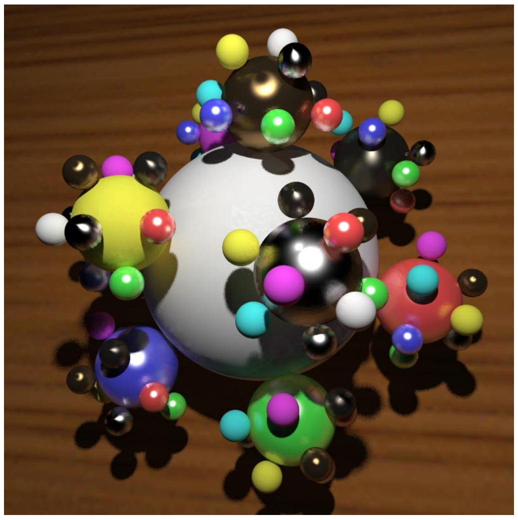
6.5 A complex interior

To test the effectiveness of our clustering method for complex glossy environments, we experimented with the architectural interior environment shown in Figure 9. This environment consists of

[1]This is the same image as in Figure 7(b).

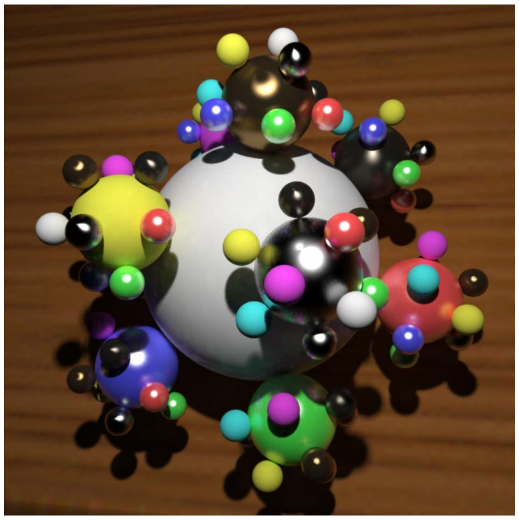


(a)

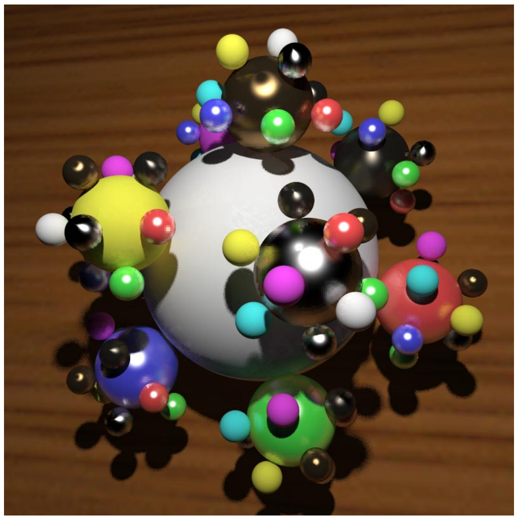


(b)

Figure 7 A sphereflake computed using our method: (a) finite-element solution; (b) after final gather.

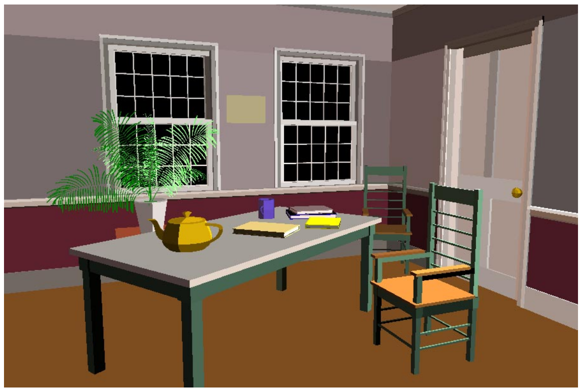


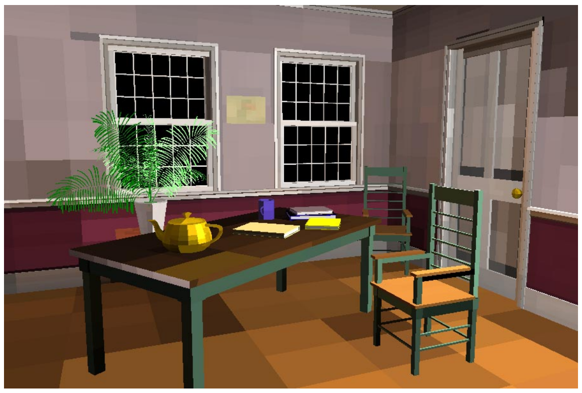
(a)

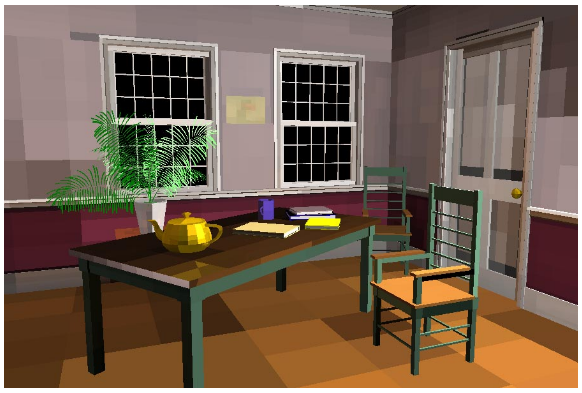


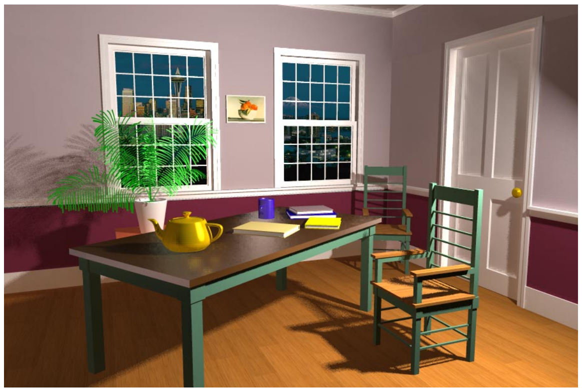
(b)

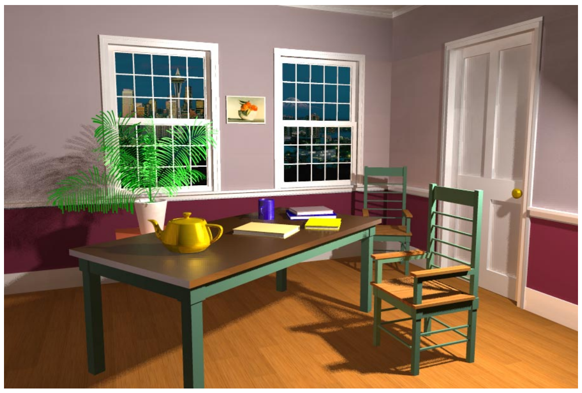
Figure 8 A comparison of sphereflake images: (a) image computed in 6.2 hours using RADIANCE; (b) image computed in 4.5 hours using our method.


(a)


(b)


(c)


(d)


(e)

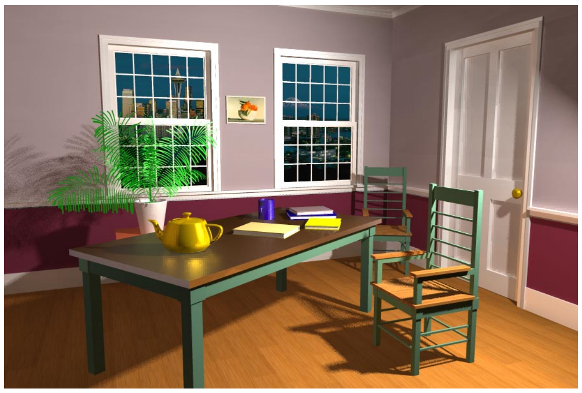

(f)

Figure 9 A room with both glossy and diffuse surfaces: (a–c) finite-element solutions computed by our method; (d–f) the same solutions after a final gather (local pass).

image	coefficients	interactions	solution time	final gather time
(a, d)	7,711	115,880	19 min	340 min
(b, e)	93,855	327,156	43 min	492 min
(c, f)	243,634	697,570	62 min	565 min

Table 1 Statistics for Figure 9. Times were measured on a DEC Alpha 3000/500X workstation.

7,711 patches: 7,643 quadrilaterals, and 68 Bézier patches. The teapot, mug, table top, door, doorknob, window frames, plant, and pot all have reflectances that are partially glossy and partially diffuse, giving a total of 6,629 glossy surfaces. The remaining surfaces are purely diffuse. The room is illuminated by a single diffuse area light source.

A solution without clustering requires $7,711^2 \approx 60,000,000$ potential interactions to be considered in the initial linking stage. In an attempt to compute such a solution, we ran out of virtual memory after 14 CPU hours, having created 1,850,176 initial links. We estimate that if we had enough memory (4 gigabytes) this computation would take 5 CPU days, just for the initial links and without any further link refinement. By contrast, a coarse solution was obtained after less than 19 minutes with our clustering algorithm. A cluster hierarchy consisting of 1629 clusters was constructed during the first 4 minutes, and the remaining 15 minutes were spent creating initial links and computing the solution shown in Figure 9(a). Of 115,880 total links in this solution, only 18,004 link two patches directly. Two more refined solutions are shown in Figures 9(b) and (c). Statistics for each of these solutions are reported in Table 1.

A final gather pass was performed on these solutions; the resulting images are shown in Figure 9(d-f). The images were rendered at a resolution of 900 $\times$ 600 pixels with sixteen rays per pixel. The final gather pass is by far the most time consuming stage of the simulation, due to the brute-force sampling strategies that our current implementation employs when gathering energy through links. We leave as future work the problem of optimizing this process; we believe that drastic improvements should be possible.

The final gather is able to produce an image of high quality even for the coarsest global solution: all the direct illumination, including shadows and specular highlights, and the textures are present. However, the coarsest solution does not yet have any direct links between the teapot's patches and the table top. The table top does receive light from the teapot, but this light is transferred via several cluster-to-patch links, as evidenced by the disjoint yellow reflections on the table top in Figure 9(d).

In the second solution, both the teapot and the table top have been refined, and direct interactions between the teapot's patches and the table top have been created. As a result, both the illumination on the teapot and the teapot's reflection on the table top are much more accurate in Figure 9(e). In the third solution, these surfaces and the transport between them are refined even further, and the reflection of the teapot on the table top is evident even before the final gather (Figure 9(c)). This progression of solutions demonstrates that our method lends itself to progressive image generation, with useful solutions available relatively early in the process.

7 Conclusion

We have presented a clustering method for efficiently simulating glossy global illumination in complex scenes. The algorithm approximates light leaving a collection of patches as radiant intensity emanating from a point, and estimates a bound on the error associated with this approximation. As in previous methods for clustering (in purely diffuse environments), the objective of this approximation is to avoid the quadratic cost of initial linking. Because our clustering algorithm has a time complexity that is only linear in the number of initial surfaces, it is arguably the first finite-element method capable of handling complex glossy environments. Furthermore, our algorithm's linear time complexity makes it asymptotically faster than previous clustering methods for diffuse environments [33], as confirmed by the results of our experiments.

A similar clustering approach has been concurrently and independently investigated by Sillion *et al.* [31]. Their algorithm also represents the radiance leaving a cluster by means of a directional distribution. One important difference between the two algorithms is that while ours uses a piecewise-constant basis to represent directional distributions, Sillion *et al.* use spherical harmonics, which provide a continuous representation, but are considerably more expensive. Another difference is that Sillion *et al.* only store outgoing radiance as a directional distribution; incoming light is pushed directly down the cluster hierarchy until it reaches the patches. This means that their algorithm's asymptotic complexity is still $O(p \log p)$. At this time we are unable to compare the actual performance of the two algorithms, as Sillion *et al.* describe a preliminary implementation of their algorithm, and do not report results on complex glossy scenes.

There are a number of aspects of our algorithm that require further research. The refinement strategy described in Section 4.2 is not very sophisticated. Ideally, we would like to predict which end of a cluster-to-cluster link to refine in order to get the most improvement in the solution. It may also be advantageous to create links from clusters directly to basis functions within individual patches, which our current implementation does not allow. In addition, the images in Figure 9 raise the question of how much work should be done in the global illumination pass before the final gather takes place.

Another natural extension to our algorithm would be to use a mutiresolution representation for the cluster's directional distributions, instead of the current uniresolution representation. So long as the size of this representation is bounded by a constant, the asymptotic complexity of our method should not be affected. However, more sophisticated refinement strategies would have to be devised for deciding whether to refine a cluster link by pushing the link to the cluster's children or by refining the cluster's directional representation.

In this paper, we have demonstrated that the finite-element solution method, in combination with hierarchical techniques such as wavelet bases, clustering, and importance-driven refinement, shows great promise for the glossy global illumination problem. Our clustering algorithm could also be used to improve other global illumination methods. For instance, it could be used as the first pass in the progressive multipass method of Chen *et al.* [6], or to provide a system like RADIANCE [36] with a function for importance sampling.

In summary, there has been a long progression of finite-element algorithms for global illumination over the last decade, starting with a dense matrix solution [13], and continuing through progressive, hierarchical, multiresolution, and finally clustering methods. Each of these algorithms has been described in detail, first for diffuse environments, and later for the more difficult nondiffuse case—with one notable exception: clustering. The work presented here fills this last gap, providing a hierarchical glossy global illumination algorithm with clustering for complex environments.

Acknowledgments

We would like to thank Tony DeRose for many helpful discussions during the course of this project. Thanks also to Jim Arvo for sharing his insights about clustering. The room model used to produce the images in Figure 9 was originally created by Matt Hyatt at the Cornell University Program of Computer Graphics, and the palm tree in the same images was created by Taweewan Siwadune. This work was supported by an NSF Graduate Research Fellowship, an NSF Postdoctoral Research Associates in Experimental Sciences award (CDA-9404959), NSF Presidential/Young Investigator awards (CCR-8957323 and CCR-9357790), an Alfred P. Sloan Research Fellowship (BR-3495), an ONR Young Investigator award (N00014-95-1-0728), and industrial gifts from IBM.

References

[1] Andrew W. Appel. An efficient program for many-body simulation. *SIAM Journal on Scientific and Statistical Computing*, 6(1):85–103, January 1985.

[2] James Arvo and David Kirk. Fast ray tracing by ray classification. In *Proceedings of SIGGRAPH '87*, pages 55–64. ACM, New York, 1987.

[3] Larry Aupperle and Pat Hanrahan. Importance and discrete three point transport. In *Proceedings of the Fourth Eurographics Workshop on Rendering* (Paris, June 1993), pages 85–94.

[4] Larry Aupperle and Pat Hanrahan. A hierarchical illumination algorithm for surfaces with glossy reflection. In *Proceedings of SIGGRAPH '93*, pages 155–162. ACM, New York, 1993.

[5] Josh Barnes and Piet Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324(4):446–449, December 1986.

[6] Shenchang Eric Chen, Holly E. Rushmeier, Gavin Miller, and Douglas Turner. A progressive multi-pass method for global illumination. In *Proceedings of SIGGRAPH '91*, pages 165–174. ACM, New York, 1991.

[7] Per H. Christensen, David H. Salesin, and Tony D. DeRose. A continuous adjoint formulation for radiance transport. In *Proceedings of the Fourth Eurographics Workshop on Rendering* (Paris, June 1993), pages 95–104.

[8] Per H. Christensen, Eric J. Stollnitz, David H. Salesin, and Tony D. DeRose. Global illumination of glossy environments using wavelets and importance. *ACM Transactions on Graphics*, 15(1):37–71, January 1996.

[9] Alain Fournier and Pierre Poulin. A ray tracing accelerator based on a hierarchy of 1D sorted lists. In *Proceedings of Graphics Interface '93*, pages 53–60. Canadian Information Processing Society, Toronto, Ontario, Canada, 1993.

[10] Henry Fuchs, Zvi M. Kedem, and Bruce Naylor. On visible surface generation by a priori tree structures. In *Proceedings of SIGGRAPH '80*, pages 175–181. ACM, New York, 1980.

[11] Andrew S. Glassner. Spacetime ray tracing for animation. *IEEE Computer Graphics and Applications*, 8(2):60–70, March 1988.

[12] Jeffrey Goldsmith and John Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications*, 7(5):14–20, May 1987.

[13] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. In *Proceedings of SIGGRAPH '84*, pages 213–222. ACM, New York, 1984.

[14] Steven J. Gortler, Peter Schröder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. In *Proceedings of SIGGRAPH '93*, pages 221–230. ACM, New York, 1993.

[15] Leslie F. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems*. MIT Press, Cambridge, MA, 1988.

[16] Eric A. Haines. A proposal for standard graphics environments. *IEEE Computer Graphics and Applications*, 7(11):3–5, November 1987.

[17] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. In *Proceedings of SIGGRAPH '91*, pages 197–206. ACM, New York, 1993.

[18] David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. In *Proceedings of SIGGRAPH '86*, pages 133–142. ACM, New York, 1986.

[19] James T. Kajiya. The rendering equation. In *Proceedings of SIGGRAPH '86*, pages 143–150. ACM, New York, 1986.

[20] A. J. Kok. Grouping of patches in progressive radiosity. In *Proceedings of the Fourth Eurographics Workshop on Rendering* (Paris, June 1993), pages 221–231.

[21] Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Combining hierarchical radiosity and discontinuity meshing. In *Proceedings of SIGGRAPH '93*, pages 199–208. ACM, New York, 1993.

[22] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometrical considertions and nomenclature for reflectance. NBS Monograph 160, National Bureau of Standards, October 1977.

[23] M. Ohta and M. Maekawa. Ray coherence theorem and constant time ray tracing algorithm. In *Proceedings of CG International '87*, pages 303–314. Springer-Verlag, Tokyo, 1987.

[24] S. N. Pattanaik and S. P. Mudur. Computation of global illumination by Monte Carlo simulation of the particle model of light. In *Proceedings of the Third Eurographics Workshop on Rendering*, pages 71–83. Consolidation Express, Bristol, UK, 1992.

[25] Sumanta N. Pattanaik and Kadi Bouatouch. Haar wavelet: A solution to global illumination with general surface properties. In G. Sakas, P. Shirley, and S. Müller, editors, *Photorealistic Rendering Techniques*, pages 281–294. Springer-Verlag, Berlin, 1995.

[26] Mark C. Reichert. A two-pass radiosity method driven by lights and viewer position. Master's thesis, Cornell University, 1992.

[27] Holly E. Rushmeier, C. Patterson, and A. Veerasamy. Geometric simplification for indirect illumination calculations. In *Proceedings of Graphics Interface '93*, pages 227–236. Canadian Information Processing Society, Toronto, Ontario, Canada, 1993.

[28] Peter Schröder and Pat Hanrahan. Wavelet methods for radiance computations. In G. Sakas, P. Shirley, and S. Müller, editors, *Photorealistic Rendering Techniques*, pages 310–326. Springer-Verlag, Berlin, 1995.

[29] Peter Shirley. A ray tracing method for illumination calculation in diffuse-specular scenes. In *Proceedings of Graphics Interface '90*, pages 205–212. Canadian Information Processing Society, Toronto, Ontario, Canada, 1990.

[30] François Sillion. Clustering and volume scattering for hierarchical radiosity calculations. In G. Sakas, P. Shirley, and S. Müller, editors, *Photorealistic Rendering Techniques*, pages 105–118. Springer-Verlag, Berlin, 1995.

[31] François Sillion, George Drettakis, and Cyril Soler. A clustering algorithm for radiance calculation in general environments. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95*, pages 196–205. Springer-Verlag, Vienna, 1995.

[32] François X. Sillion, James R. Arvo, Stephen H. Westin, and Donald P. Greenberg. A global illumination solution for general reflectance distributions. In *Proceedings of SIGGRAPH '91*, pages 187–196. ACM, New York, 1991.

[33] Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. In *Proceedings of SIGGRAPH '94*, pages 435–442. ACM, New York, 1994.

[34] Brian E. Smits. Efficient hierarchical radiosity in complex environments. Ph.D. thesis, Cornell University, 1994.

[35] Brian E. Smits, James R. Arvo, and David H. Salesin. An importance-driven radiosity algorithm. In *Proceedings of SIGGRAPH '92*, pages 273–282. ACM, New York, 1992.

[36] Greg Ward. The RADIANCE lighting simulation and rendering system. In *Proceedings of SIGGRAPH '94*, pages 459–472. ACM, New York, 1994.

[37] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *Proceedings of SIGGRAPH '88*, pages 85–92. ACM, New York, 1988.

[38] Hau Xu, Qun-Shang Peng, and You-Dong Liang. Accelerated radiosity method for complex environments. In *Proceedings of Eurographics '89*, pages 51–61. North-Holland, New York, 1989.