EUROGRAPHICS '98 / N. Ferreira and M. Göbel
(Guest Editors)

Volume 17, (*1998*), *Number 3*

A Light Hierarchy for Fast Rendering of Scenes with Many Lights

Eric Paquette†, Pierre Poulin†, and George Drettakis‡

† Département d'informatique et de recherche opérationnelle
Université de Montréal. E-mail: {paquette|poulin}@iro.umontreal.ca

‡ iMAGIS-GRAVIR/IMAG-INRIA§
BP 53, F-38041 Grenoble Cedex 9, FRANCE. E-mail: George.Drettakis@imag.fr

Abstract

We introduce a new data structure in the form of a light hierarchy for efficiently ray-tracing scenes with many light sources. An octree is constructed with the point light sources in a scene. Each node represents all the light sources it contains by means of a virtual light source. We determine bounds on the error committed with this approximation to shade a point, both for the cases of diffuse and specular reflections. These bounds are then used to guide a hierarchical shading algorithm. If the current level of the light hierarchy provides shading of sufficient quality, the approximation is used, thus avoiding the cost of shading for all the light sources contained below this level. Otherwise the descent into the light hierarchy continues.

Our approach has been implemented for scenes without occlusion. The results show important acceleration compared to standard ray-tracing (up to 90 times faster) and an important improvement compared to Ward's adaptive shadow testing.

Keywords: Image synthesis, rendering, ray-tracing, hierarchy, illumination, reflection, Phong, bounds, clustering, octree.

1. Introduction

Realistic rendering has been a major goal of computer graphics from its very outset. Many powerful rendering approaches such as ray-tracing[1], radiosity-based methods[2, 3], and stochastic ray-tracing or Monte Carlo methods[4] have been presented over the last two decades, resulting in images of impressive realism. For most existing commercial rendering systems (for animations, film special effects, post-production, advertising, etc.), ray-tracing remains the rendering algorithm of choice. In such environments, scenes containing a large number of geometric primitives as well as a large number of light sources are common. Everyday scenes with a large number of light sources include shopping malls, chandeliers, city streets at night, etc. In addition, more complex light sources such as extended area sources, shades over a light source, or special sources to simulate flames are often required; they are typically approximated as a large collection of simpler point light sources.

The rendering of these scenes is a challenge to computer graphics, due to the number of light sources and the complex illumination that results, especially when we consider their combined rather than individual effect. In this paper we address the problem of efficient rendering of such scenes by introducing a new data structure, representing the light sources by a light hierarchy.

1.1. Motivation

Despite advances in the treatment of scenes containing a large number of geometric primitives using spatial subdivision techniques (octrees[5], grids[6], hierarchies of bounding volumes[7], etc.), little has been done to accelerate ray-tracing of scenes with many light sources.

For scenes with hundreds or thousands of light sources, shading and shadowing calculations (rays sent to the light

§ iMAGIS is a joint research project of CNRS/INRIA/UJF/INPG.

sources) quickly become the dominant cost of the rendering process. As a result, lighting designers and other users of rendering systems are forced to crudely approximate interesting and complex lighting behavior with only a very small number of simple light sources.

The work on spatial subdivision and hierarchical algorithms for lighting calculations[8], hierarchical approaches coupled with clustering [9, 10], and spatial subdivision have allowed the acceleration of lighting calculation for scenes containing a large number of polygons. A natural application of the same hierarchical concepts is the development and use of the light hierarchy for ray-tracing which we present in this paper.

1.2. Contributions

The goal of our approach is twofold: to provide efficient ray-tracing of scenes with many light sources with minimum loss of image quality, and to provide an intuitive quality parameter based on consistent error bounds for all the approximations made.

To achieve this goal in a comprehensive manner, we have restricted our attention to direct illumination from point light sources using lambertian (diffuse) and Phong[11] (specular) reflection models. Such assumptions are common in most commercial uses of ray-tracing systems.

Our solution involves the development of error bounds to evaluate the maximum potential error produced by the approximation. In this paper, we develop an algorithm which exploits the hierarchy and the bounds for the shading under direct illumination of scenes without occlusion. The results of our algorithm in this context (see Section 5) show that our light hierarchy significantly speeds up shading for scenes with many light sources.

It is interesting to note that in computer graphics research, an initial solution to an illumination problem is often presented for the unoccluded case (e.g., for radiosity [12] or hierarchical radiosity [13]), which then led to complete solutions including shadows (hemi-cube radiosity [14] and hierarchical radiosity with shadows[8]). The study and introduction of a comprehensive solution without occlusion is an important step in the necessary understanding of the problem, leading subsequently to an algorithm including shadows.

For a given 3D point being shaded, traversal of the light hierarchy allows us to determine the effect on shading of intermediate nodes (representing potentially large numbers of light sources contained beneath the current level of the light hierarchy). As a consequence, we can completely avoid shadowing calculations for certain such nodes (i.e., avoiding shadowing for many light sources), or identify those light sources or sets of light sources which have the largest impact on final shading. To better underline the potential of our approach, after presenting the algorithm and the results of the implementation, we will also discuss our ideas on the treatment of shadows in Section 6.

2. Previous and Related Work

There is an extensive literature on the research dedicated to speeding up shadow calculations using spatial coherence and subdivision.[15] Most of these approaches however are highly dependent on the number of light sources, and are thus unsuitable for scenes with many light sources.

Some algorithms have nonetheless addressed the case of scenes with many light sources. Bergeron[16] defines a sphere of influence around each point light source. The radius of each sphere is related to its light source intensity. Any object outside a sphere of influence can ignore the contribution of this point light source for both shading and shadowing. This method is efficient in many cases, but will fail for numerous light sources of low intensity because of the smaller radius of the spheres. Objects outside these spheres will ignore them all, even though the combined contribution of all light sources together may produce an important illumination effect. Another such situation occurs when several point light sources are used to simulate a complex light source. Improving the approximation typically requires increasing the number of point light sources, each of them individually emitting a smaller proportion of the complex light global intensity. With spheres of influence, the radii would then reduce, and the overall scene illumination would decrease as the complex light representation would be improved. This is illustrated in Fig. 1.

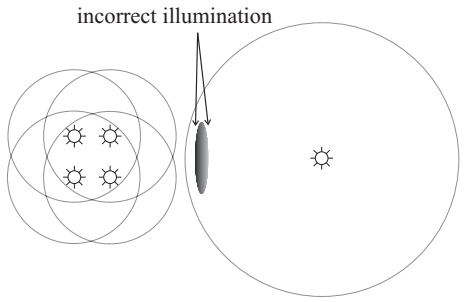


Figure 1: *Problem associated with the spheres of influence*

Ward[17] presents a different approach where a sorted list of light source contributions is maintained. The main idea is to calculate the potential contribution of each light source at every point to shade (without considering visibility), and to use this estimation to sort the list of light sources. The ordered list is traversed and thus the real contribution (including visibility calculation) of the most important light sources is computed first. If the sum of the potential contributions of the remaining light sources is smaller than a predetermined percentage of the sum of all real contributions computed so far, the traversal stops. This method performs

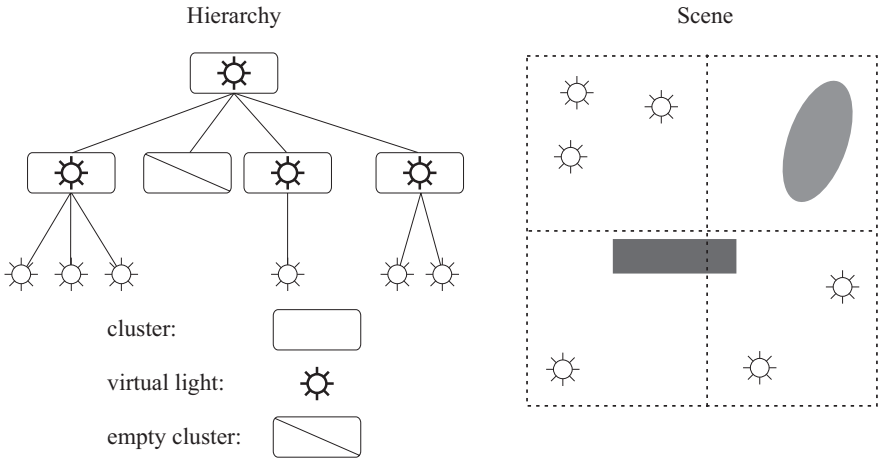


Figure 2: *Example of light hierarchy*

well for a moderate number of light sources, and is the most suitable algorithm to date for the treatment of scenes with many sources. However as the number of light sources increases, the cost of sorting the contributions (at each pixel) of all these light sources can become an important factor of the total rendering cost for scenes where the geometrical complexity is smaller than the illumination complexity. An extensive comparison of our approach to that of Ward's is presented in Section 5.

Shirley *et al.*[4] divide light sources into two categories: bright (important) and dim (less important). This selection is performed as a preprocess, and is based on an approach similar to the sphere of influence. A sampling probability is then assigned to each bright light source, and a unique probability is assigned to all the dim light sources. If a large number of rays are shot per pixel, this method can be very effective. However, as with all Monte Carlo approaches, noise due to insufficient sampling can appear in the rendered images. Moreover, since the dim light source to be sampled is chosen randomly, an unsuitable partitioning into dim and bright light sources can greatly increase the amount of noise.

In radiosity-based methods, work has been performed in clustering objects for light-transfer calculations [9, 10]. These methods do not treat light sources separately, since they attempt to treat global illumination, and thus any surface is a light source in later iterations. As a consequence, these methods typically will not perform very well, since (primary) light sources are often clustered with the other objects of the scene, and no light-specific hierarchy is actually built.

Houle and Fiume[18] store an emission map in a multiresolution structure (quadtree) to efficiently resample a continuously varying emission distribution. However this structure is only valid over a 2D planar surface, and as such cannot easily be extended for independent light sources arbitrarily distributed in 3D space without losing its hierarchical nature.

Stam and Fiume[19] simulate flames with a set of multiresolution particles. To shade a point illuminated by their flames, the illumination is computed at two adjacent levels of the flame hierarchy, starting from the top. When the difference in illumination is larger than a preset threshold, the process continues at the immediately superior resolution. This hierarchical structure is efficient for flames consisting of a large number of particles. An early stop can however occur when the difference between two adjacent levels is small, but would be significant with a higher resolution. This is due to the fact that no bound on the approximation is provided.

3. Hierarchy of Point Light Sources

As mentioned in the introduction, a hierarchical data structure representing the point light sources in the scene is required to achieve our goals of efficient and consistent treatment of scenes with many light sources.

3.1. Octree Light Hierarchy

We have chosen to encode the light sources in an octree structure for the simplicity and compactness of its representation and its hierarchical nature.

The light hierarchy is stored separately from the ray-tracing acceleration structure used for the ray-object intersections (in our case also an octree). This choice has the advantage of creating a tighter bounding box around the light sources. The disadvantage resides in the fact that interaction between objects and light sources has to be treated separately. For the needs of our algorithm, we have found this solution to be satisfactory.

The leaf nodes of the octree store zero, one, or more point

light sources. Intermediate nodes are composed of eight children nodes. Every node keeps an approximate representation of the light sources contained at the current or at lower levels. In particular, *virtual light sources* stored at the nodes represent the set of light sources contained beneath this node. In Fig. 2, we show an example of such a point light hierarchy.

3.2. Light Hierarchy Construction

The creation of the light hierarchy begins by finding the axis-aligned bounding box of all the point light sources in the scene. This box is the root of our hierarchy. The hierarchy is subsequently subdivided in an octree fashion, until each unsubdivided octree cell contains less than a preset maximum number of light sources or a maximum subdivision depth is reached.

Once the octree has been subdivided, we proceed with the calculation of the representation of the virtual light sources. The virtual light source is simply a point light source, placed at a position corresponding to the weighted average of the light sources it represents. The weights are proportional to intensity of each real or virtual point light source over the sum of these intensities.

For tighter bounds on the approximation errors explained in the next section, we compute at each node of the octree the smallest axis-aligned bounding box of all the point light sources it represents. We call it the *minimal bounding box*. Table 1 summarizes this construction.

4. Shading with the Hierarchical Light Structure

Once the light hierarchy is built, the goal is to develop an algorithm allowing us to use the approximate representations (virtual light sources) where appropriate. We will thus avoid the cost of shading (and potentially shadowing) with each light source in the scene. To achieve this goal we need criteria allowing us to choose when the approximate shading gives a satisfactory result, thus permitting us to terminate our descent into the light hierarchy.

In what follows we first present some necessary preliminaries on shading for ray-tracing, and then proceed to describe the error bounds developed for the diffuse and specular cases. These error bounds are then used as the criteria to perform the actual shading. The algorithms for the shading are described for each case.

4.1. Preliminaries

A common shading formulation divides the reflection as a combination of diffuse (pure lambertian) reflection and specular (directional) reflection. One such popular model suggested by Phong[11] is shown in Fig. 3, and can be expressed as

$$I = \sum_{i=1}^{m} S_i f_{att_i} I_i \left(k_d(\vec{N} \cdot \vec{L_i}) + k_s(\vec{R_i} \cdot \vec{E})^n \right) \qquad (1)$$

where

- I = light radiance going from the light to the viewer and passing by p,
- i = i^{th} light source,
- m = number of light sources,
- S = visibility factor of the light source,
- f_{att} = attenuation of light from the light source,
- I_i = intensity of light source i,
- k_d = diffuse reflection coefficient of the surface,
- $\vec{N}$ = surface normal at p,
- $\vec{L}$ = vector from p to the light source,
- k_s = specular reflection coefficient of the surface,
- $\vec{R}$ = mirror reflection of the vector $\vec{L}$ at p with respect to $\vec{N}$,
- $\vec{E}$ = vector from p towards viewer,
- n = roughness coefficient.

All the vectors are normalized.

We first derive some bounds for the diffuse reflection, and then present the bounds for the specular reflection.

4.2. Diffuse Reflection

We slightly simplify the formulation for clarity, replacing the terms $S = 1$ (unoccluded case) and $f_{att} = 1/d^2$ where d is the distance from the light source to point p. We also replace the dot products using the identities :

$$\cos(\theta) = \vec{N} \cdot \vec{L}$$
$$\cos(\theta_i) = \vec{N} \cdot \vec{L}_i$$

The light reflected by a diffuse surface illuminated by m point light sources can be computed as:

$$I = \sum_{i=1}^{m} I_i \frac{1}{d_i^2} k_d \cos(\theta_i) \,. \qquad (2)$$

By taking a single virtual point light source approximating all m point light sources, we compute:

$$I_{approx} = \left(\sum_{i=1}^{m} I_i \right) \frac{1}{d^2} k_d \cos(\theta) \,. \qquad (3)$$

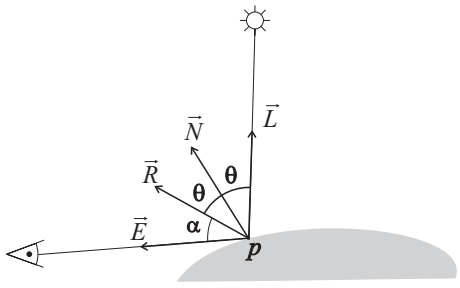


Figure 3: *Phong reflection model*

Intensity	$I_v = \sum_{i=1}^{m} I_i$
Position	$P_v = (\sum_{i=1}^{m} P_i \|\|I_i\|\|_1) / (\sum_{i=1}^{m} \|\|I_i\|\|_1)$
Dispersion	$B_{min} = \text{AxisAlignedBox}(\ (\min(Px_i), \min(Py_i), \min(Pz_i)),$ $(\max(Px_i), \max(Py_i), \max(Pz_i))\ )$

Table 1: *Cluster attributes*

4.2.1. Error Bound for the Diffuse Model

Given the expressions of the exact and the approximate illumination at a point, we need to develop a bound on the error committed by the approximation. This bound must be tight and efficient to evaluate so it can guide a hierarchical shading algorithm at a reasonable cost. In particular, we are looking for a function ΔI_{abs} such that for any point to shade

$$\Delta I_{abs} \geq |I - I_{approx}| .$$

To derive a suitable expression, we first define a few variables:

$$\Delta d_i = d_i - d$$
$$\Delta \theta_i = \theta_i - \theta$$
$$\Delta d_{max} = diag$$
$$\Delta \theta_{max} = \arctan\left(\frac{\Delta d_{max}}{d - \Delta d_{max}}\right)$$

where *diag* is the largest diagonal of the minimal bounding box at a node of the light octree. These quantities are illustrated in Fig. 4 and 5.

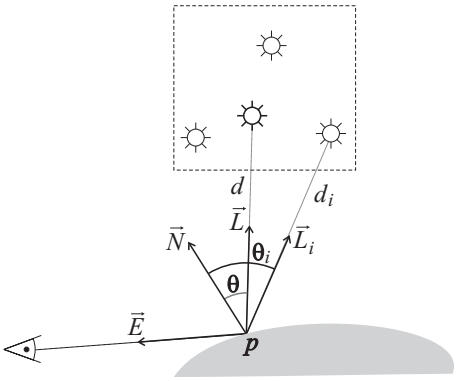


Figure 4: *Important quantities for the diffuse reflection model*

The following inequalities can then be applied to derive the desired bound:

$$0 \leq \Delta d_{max} < d$$
$$(0,0,0) \leq k_d, I_i$$
$$0 \leq \theta, \theta_i \leq \pi/2$$
$$0 \leq \cos(\theta), \cos(\theta_i) \leq 1.$$

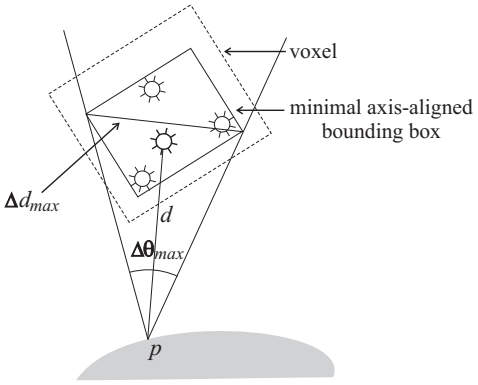


Figure 5: *Bounds on the quantities related to a cluster*

By bounding the approximate intensity between a minimal and a maximal intensity, I_{min} and I_{max}, ΔI_{abs} becomes:

$$\Delta I_{abs} = \max(I_{approx} - I_{min}, I_{max} - I_{approx}) .$$

The derivation of the bound from these equations and inequalities is given in Appendix A. The bound on the error produced by approximating the diffuse reflection at a point p by a single virtual point light source at a node of our octree structure is:

$$\Delta I_{abs} = k_d I_v \max\left(\frac{\cos(\theta)}{d^2} - \frac{\max(0, \cos(\theta) - \Delta\theta_{max})}{(d + \Delta d_{max})^2}, \frac{\min(1, \cos(\theta) + \Delta\theta_{max})}{(d - \Delta d_{max})^2} - \frac{\cos(\theta)}{d^2}\right) . \quad (4)$$

The cost of computing this error and the actual illumination using the virtual light source is about the same as that of computing the illumination due to three to four point light sources. The bound is therefore useful when the approximation replaces the evaluation of the illumination from at least three point light sources.

4.2.2. Diffuse Shading using the Light Hierarchy

Equation 4 tells us when a node of the light hierarchy is below the desired threshold. This means the virtual light source associated with this node could replace all the point light sources below this node. However if the traversal of the light hierarchy stops at several nodes, while each individual bound can be below the threshold, their sum might not be.

If the sum of the errors associated with the voxels stored in

the error list is greater than the threshold, we treat the nodes with the highest error at a lower level. We continue this process until the sum of the errors is below the threshold. This way we ensure that our absolute error bound is respected.

The algorithm in Fig. 6 illustrates how the light hierarchy is used to compute the shading.

```
HierarchicalIllumination( Point p, Voxel v )
{
  if v.Empty()
    return( 0 )

  if v.NumberOfSources() ≤ 3
    return( SimpleIllumination( p, v.sources ) )

  Evaluate I_v, k_d, d, Δd_max, θ, Δθ_max from p and v.

  if Δd_max > d
    // We cannot evaluate our bound if Δd_max > d.
    return( IlluminationAtLowerLevel( p, v ) )

  error = DiffuseBound( I_v, k_d, d, Δd_max, θ, Δθ_max)

  // Compare with diffuse error threshold
  if error < T_diffuse
    AddToErrorList( v, error )
    return( 0 )
  else
    return( IlluminationAtLowerLevel( p, v ) )
}

IlluminationAtLowerLevel( Point p, Voxel v )
{
  if v.Subdivided()
    ∀ v.voxelChild
      sum += HierarchicalIllumination( p, v.voxelChild )
    return( sum )
  else
    return( SimpleIllumination( p, v.sources ) )
}
```

Figure 6: *Illumination algorithm using light hierarchy for diffuse surfaces*

4.3. Specular Reflection

Specular reflection is mostly associated with highlights on surfaces. The light reflected by a specular surface illuminated by m point light sources can be computed as:

$$I = \sum_{i=1}^{m} I_i \frac{1}{d_i^2} k_s \cos^n(\alpha_i)\,. \tag{5}$$

Phong specular reflection has the form of a specular lobe $\cos^n(\alpha)$, where n controls the "eccentricity" of the lobe. The larger the value of n, the smaller the simulated roughness of the surface, and the sharper and narrower the highlights.

We consider here specular reflections off surfaces with a high roughness coefficient (values of $n > 20$). Specular reflections with a low roughness coefficient could be handled with an algorithm similar to the diffuse one.

For specular surfaces with a high roughness coefficient, typically very few light sources will contribute significantly to the illumination at a particular point. We can therefore use the light hierarchy to quickly identify the important light sources. Our approach is based on computing the maximal potential contribution of a voxel of light sources. Only those voxels with a potential contribution greater than a specified specular threshold T_{spec} will be treated at a finer level.

4.3.1. Error Bound for the Specular Model

We need to compute the maximal potential contribution of a voxel. It is derived from the Phong equation in a similar way to that of the diffuse error bound. Again, the complete derivation of this bound is given in Appendix A. The maximal specular contribution corresponds to

$$maxC = I_v k_s \frac{(\min(\cos(\alpha) + \Delta\alpha_{max}, 1))^n}{(d - \Delta d_{max})^2} \tag{6}$$

where $\Delta\alpha_{max}$ is computed in the same way as $\Delta\theta_{max}$ and $\cos(\alpha) = \vec{R} \cdot \vec{E}$.

4.3.2. Specular Shading using the Light Hierarchy

Starting at the root of the light hierarchy, if the maximal potential contribution of a voxel is greater than T_{spec}, we open that voxel and treat its children. If the potential contribution is below T_{spec}, we add the voxel to the list of ignored contributions. After the traversal of the light hierarchy is completed, we check the list of ignored contributions to ensure that its sum is lower than T_{spec}. As for the diffuse reflection, we then recompute the hierarchical illumination of the nodes with the larger maximal contribution at a lower level. This process is repeated until the sum of the ignored contributions is lower than T_{spec}. When the threshold is respected, the maximal error present in the image will be equal or smaller to it. Choosing a very small T_{spec} results in no visual artifacts.

Instead of ignoring the contributions of these voxels, we could adopt an approach similar to Ward[17], and add an approximation of the contribution of these voxels based on the illumination of their virtual light sources. This way, the resulting error would be smaller and we could use a larger threshold resulting in greater speedups. While reducing the error, this would not change the value of the bound. The average error would be smaller, but the maximal error would stay the same.

4.4. Combined Diffuse and Specular Reflections

To handle surfaces with both diffuse and specular reflections, the two algorithms are mainly executed independently. Some calculations as well as the traversal of the hierarchy

are shared, but the main parts of the algorithms are treated separately. The rendering times are a little less than the simple sum of both times, but are still proportional to it.

5. Results

5.1. Basic Test Scenes

We have developed a set of test scenes which allows us to evaluate our approach in several different configurations, while keeping a low intersection cost between rays and the scene made of only a few polygons. The three main scenes treated are shown in the following (the names subsequently used are in quotes): the first is an image of "*Light Strings*" lighting a street (Fig. 7(a)); the second is a scene lit by light sources forming the "*U de M*" logo (Fig. 7(b)); and the last is a simple scene illuminated by a "*Cluster*" of light sources (Fig. 7(c)). The rendering of this last scene is done with the camera on the left, just behind the light sources. In our tests, the rendering of this scene occurs with the light sources being invisible. All the light sources in one scene have equal intensities.

We present results on variations of these scenes containing a number of light sources ranging from 64 to 16384. This is done by increasing the density of the light sources in each case.

For all the specular tests, the roughness factor we use is $n = 200$.

In the result tables presented below, we show the cost of standard ray-tracing with octree acceleration (**RT**), the cost of Ward's (**Ward**) method[17], the cost of the light hierarchy method (**LH**), and the speedup achieved by our method over the ray-tracing method (**SU**). Our implementation of Ward's method is slightly different than the one presented in his paper[17] to ensure that, as with our algorithm, it has an absolute error bound.

For each test, the error thresholds (diffuse and specular) we use are 1% RGB or $(2.55, 2.55, 2.55)$ RGB for images quantized in $[0..255]$. Since the actual error is smaller than the threshold, the test runs result in images that are visually indistinguishable from those computed with the traditional ray-tracing. The maximal observed error at a pixel is $(1, 1, 1)$ RGB for all the images and the average pixel error is negligible since it is less than $(0.02, 0.02, 0.02)$ RGB.

5.2. Diffuse Case

In Table 2 we present the results of our algorithm for diffuse surfaces. Compared to standard ray-tracing, we see that the light hierarchy achieves important speedups (up to 90 times faster). For a high number of light sources (greater than 1024), we are consistently faster than Ward's method. It should be noted that with Ward's method, the increase in rendering time does not seem to be logarithmic. In comparison, our method shows a logarithmic increase in time for the *Light Strings* and *U de M* scenes, as can be seen in Fig. 8. The more linear behavior of the *Cluster* scene indicates that

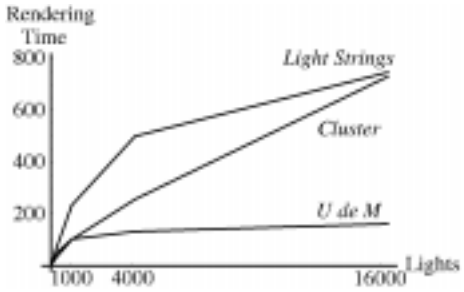


Figure 8: *Logarithmic behavior of the light hierarchy method*

for this scene, more light sources are required before reaching the "plateau" part of the logarithmic curve.

Scene	RT	Ward	LH	SU
Light Strings				
64 lights	11.4	70.2%	91.2%	1.1
256 lights	87.5	70.3%	73.5%	1.4
1024 lights	550.6	68.3%	42.4%	2.4
4096 lights	2716.3	70.8%	18.4%	5.4
16384 lights	13157.1	72.5%	5.7%	17.6
U de M				
64 lights	14.8	64.9%	85.1%	1.2
256 lights	108.6	60.0%	44.4%	2.3
1024 lights	632.6	57.8%	16.2%	6.2
4096 lights	3114.9	58.7%	4.2%	23.6
16384 lights	14609.3	59.5%	1.1%	90.1
Cluster				
64 lights	8.6	107.0%	95.3%	1.0
256 lights	33.9	112.4%	90.9%	1.1
1024 lights	136.1	119.6%	74.1%	1.3
4096 lights	545.5	127.6%	47.0%	2.1
16384 lights	2179.6	135.6%	33.5%	3.0

Table 2: *Results for the algorithm treating diffuse surfaces. All timings in seconds on a R10000 SGI computer. Notice how our method (LH) outperforms Ward's approach as the number of light sources increases.*

5.3. Specular Case

In Table 3 we present the results of our algorithm for the cases above using the specular algorithm. We see that the light hierarchy achieves an impressive speedup compared to simple ray-tracing (at least 6.4 times faster), and that it is always faster than both the ray-tracing and Ward's method. Compared to Ward's approach we also achieve significant speedup: up 32 times faster.

(a) “Light Strings” (b) “U de M” (c) “Cluster”

Figure 7: *The three different test scenes*

Scene	RT	Ward	LH	SU
Light Strings				
64 lights	12.8	26.6%	15.6%	6.4
256 lights	93.5	17.2%	8.3%	12.0
1024 lights	575.4	13.1%	6.3%	15.8
4096 lights	2815.3	12.3%	5.0%	20.1
16384 lights	13549.3	11.5%	3.9%	26.0
U de M				
64 lights	17.1	25.1%	7.0%	14.2
256 lights	117.9	16.1%	3.1%	32.8
1024 lights	670.2	12.5%	1.8%	55.4
4096 lights	3261.5	11.5%	1.3%	76.4
16384 lights	15195.3	11.0%	1.1%	91.6
Cluster				
64 lights	10.4	39.4%	5.8%	17.3
256 lights	41.0	43.7%	3.4%	29.3
1024 lights	164.1	49.7%	2.6%	39.1
4096 lights	656.7	56.0%	2.3%	43.8
16384 lights	2632.3	62.4%	1.9%	51.6

Table 3: *Results for the algorithm treating specular surfaces. All timings in seconds on a R10000 SGI computer. Notice how our method (LH) outperforms Ward’s approach everywhere.*

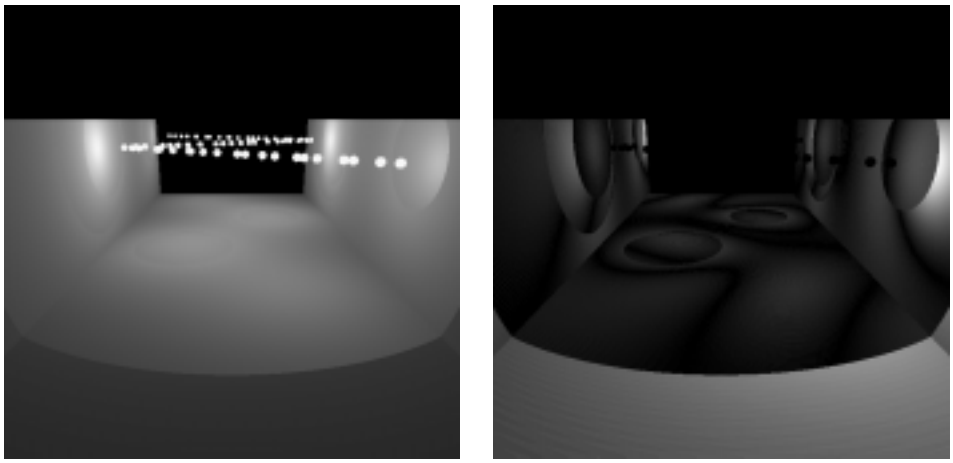
5.4. Discussion of Results

Overall, from tables 2 and 3 we see that the light hierarchy can provide a very important speedup in computation times compared to Ward’s approach and to ray-tracing.

The difference in speedup provided by the diffuse and specular algorithms is explained by the fact that, for highly specular surfaces, no shading computations need to be done for many of the pixels. For the diffuse case, most of the pixels have a large illumination value thus a large error to compensate by using lower levels. Using lower levels demands more work since we need to process more virtual or real light sources.

The *Cluster* scene shows less interesting results than the two other scenes for the diffuse case. As said before, more light sources are needed to reach the “plateau” part of the logarithmic curve. This result seems surprising for a scene with so well placed light sources. In fact, the light sources are not so well placed. They reside in a small cube, but in that cube, they are almost uniformly distributed. This results in a hierarchy that is extremely well balanced, with most of its nodes at a particular level containing the same number of light sources. This obviously results in a hierarchy with fewer levels. When using the hierarchical rendering, we will be often forced to completely calculate the illumination of a node since we do not have a finer representation for it. Uniform distribution of light sources also means that for a given level, we will have many nodes with few light sources compared to many light sources in few (non-empty) nodes if the distribution of light sources is not uniform (as in the *Light Strings* and *U de M* scenes). For the same number of light sources, we will have to sum the potential error of many voxels for the *Cluster* scene compared to the *Light Strings* and *U de M* scenes.

It is interesting to look at what happens when we increase the threshold. As it increases, voxels higher in the hierarchy are used to compute the shading. At a certain point, the resulting illumination shows discontinuities due to the choice of different levels of the hierarchy. Fig. 9 shows the resulting artifacts. To reduce these discontinuities, linear interpolation between the two levels could be used instead. The thresh-



(a) Rendering (b) Error

Figure 9: *Artifacts related to the increase of the diffuse threshold. (a) is the actual image while (b) is the error image. The intensity and contrast of the error image have been adjusted such that black represents no error and white the maximal error that occured in this image, (59, 59, 59) RGB.*

old we are using for our tests can still be increased without resulting in the artifacts present in Fig. 9. While keeping a maximal error of (1, 1, 1) RGB, we can increase the threshold to cut the rendering time of our tests by half. If fact, our threshold is approximately ten times greater than the actual (maximal) error. This is obviously conservative, but tolerable. Increasing the threshold increases the potential error in the image and the rendering speed. Fig. 9 for example, shows a speed up of ten compared to simple ray tracing with only 64 light sources. This is about 9 times faster than the result of Table 2.

6. Possible Extensions to Handle Occlusions

As mentioned in the introduction, the algorithms and data structures we present here treat the case of unoccluded illumination only. The goal of our paper is thus to present the principles of the light hierarchy and demonstrate, through experimental results, that the potential for speedup is very important. Nonetheless, it is clear that the utility of the light hierarchy will be much more widespread when occlusion is treated completely. For this reason, we present next our first ideas on the treatment of shadows.

We also list other improvements to the algorithms presented as well as interesting new research directions.

6.1. Treating Shadows

The work presented here shows very encouraging results for illumination without occlusion. In the presence of shadows, one can only hope to have more important gains from the use of the light hierarchy, since the cost of each light ray is multiplied by the cost of the actual ray intersection with the scene.

Even though our error bounds do not include shadow information, the maximum contribution (upper bound) is nonetheless valid, since shadowing can only reduce this contribution. We can thus use our hierarchical illumination algorithm as a basis of a solution for shadows. The problems of determining a lower bound, or at least an estimate of the minimum contribution, as well as the more delicate issue of preservation of shadow shapes have to be treated separately.

6.1.1. Volumetric Soft Shadows

An approach for using the light hierarchy for scenes with shadows could consist in using the hierarchy when a cluster of light sources is entirely visible from a given 3D point to shade. To do this, we need to make a decision on whether a voxel of the light hierarchy at a given level is completely unoccluded, completely occluded, or partially visible from the 3D point. In the first case, we use the algorithms presented above in Section 3; in the second case we do not need to shade at all; for the partially visible case, we must descend into the light hierarchy.

Algorithms for consistent visibility determination have been presented for other problems (e.g., shafts by Haines and Wallace[20], or conservative triage by Teller and Hanrahan[21]). What is unclear for these approaches is whether the cost of the visibility determination would make the gains of the light

hierarchy negligible or even useless. This research direction is nonetheless worthy of further investigation.

As an alternative, we can use an approximation to visibility determination by using the ideas presented in the work of Sillion[22] on volumetric visibility. Without going into details, we can represent the visibility characteristics of a cluster by a set of extinction coefficients, permitting us to avoid ray-intersections with the contents of the cluster of objects.

6.2. Hierarchy Construction and More General Models

The light hierarchy currently used is constructed very rapidly since we simply subdivide an octree. It is possible that more involved clustering approaches, such as that described in the work of Cazals *et al.*[23], which take into account certain geometric properties of the items being clustered (in our case the light sources), could result in improved performance.

The approach described here is not restricted to the sole use of ray-tracing direct light. The central ideas and concepts of our approach could be applied to improve on the clustering of light sources in radiosity-based algorithms.

7. Conclusions

The introduction of a new data structure in the form of a *light hierarchy* provides an efficient solution to the problem of ray-tracing scenes with many light sources. We have chosen to create an octree hierarchy of the light sources in a scene which is maintained independently of the rest of the geometry. Intermediate nodes of the light hierarchy approximate the illumination due to the light sources contained in the children octree voxels, by means of *virtual light sources.*

Error bounds on the error committed when using the virtual light sources were presented, both for the diffuse and the specular cases. Based on these bounds, the shading calculation at each visible point is performed by a hierarchical descent in the light hierarchy. When the descent ceases at a given hierarchical level, we avoid the cost of shading with all the light sources contained below that level, resulting in significant speedup.

To carefully evaluate the ideas and develop a deeper understanding of the issues involved, we have currently restricted our algorithm to the case of unoccluded scenes. The results for a set of such test scenes show very encouraging speedup, of up to 90 times for both diffuse and specular surfaces.

We believe that the introduction of the light hierarchy for ray-tracing, in conjunction with the error bounds and the hierarchical descent open a very promising research direction for efficient rendering of scenes with many light sources. The development of the complete solution including shadows could lead to significant acceleration of the rendering times when scenes with complex geometry would be used.

Acknowledgments

We acknowledge financial support from NSERC and FCAR. Thanks to the anonymous reviewers for their comments which improved the final version.

Appendix A: Derivation of the bounds

Diffuse bound

We first state few useful rules:

$$0 < y, 0 < \varepsilon \Rightarrow \frac{|x|}{y} > \frac{|x|}{y+\varepsilon} \tag{7}$$

$$\cos(\phi) - 1|\sigma| \leq \cos(\phi+\sigma) \leq \cos(\phi) + 1|\sigma| \tag{8}$$

$$\cos(\theta_i) = \min(1, \cos(\theta_i)) = \max(0, \cos(\theta_i)) \,. \tag{9}$$

We approximate the minimal diffuse illumination by:

$$\begin{aligned}
I &= \sum_{i=1}^{m} I_i \frac{1}{d_i^2} k_d \cos(\theta_i) \\
&= k_d \sum_{i=1}^{m} I_i \frac{\cos(\theta+\Delta\theta_i)}{(d+\Delta d_i)^2} \\
&\geq k_d \sum_{i=1}^{m} I_i \frac{\cos(\theta+\Delta\theta_i)}{(d+\Delta d_{max})^2} && \text{by 7} \\
&\geq k_d \sum_{i=1}^{m} I_i \frac{\max(0, \cos(\theta+\Delta\theta_i))}{(d+\Delta d_{max})^2} && \text{by 9} \\
&\geq k_d \sum_{i=1}^{m} I_i \frac{\max(0, \cos(\theta) - |\Delta\theta_i|)}{(d+\Delta d_{max})^2} && \text{by 8} \\
&\geq k_d \sum_{i=1}^{m} I_i \frac{\max(0, \cos(\theta) - \Delta\theta_{max})}{(d+\Delta d_{max})^2} \\
&\geq k_d I_v \frac{\max(0, \cos(\theta) - \Delta\theta_{max})}{(d+\Delta d_{max})^2} = I_{min} \,.
\end{aligned}$$

The same way, we approximate the maximal diffuse illumination by:

$$\begin{aligned}
I &= \sum_{i=1}^{m} I_i \frac{1}{d_i^2} k_d \cos(\theta_i) \\
&= k_d \sum_{i=1}^{m} I_i \frac{\cos(\theta+\Delta\theta_i)}{(d+\Delta d_i)^2} \\
&\leq k_d \sum_{i=1}^{m} I_i \frac{\cos(\theta+\Delta\theta_i)}{(d-\Delta d_{max})^2} && \text{by 7} \\
&\leq k_d \sum_{i=1}^{m} I_i \frac{\min(1, \cos(\theta+\Delta\theta_i))}{(d-\Delta d_{max})^2} && \text{by 9} \\
&\leq k_d \sum_{i=1}^{m} I_i \frac{\min(1, \cos(\theta) + |\Delta\theta_i|)}{(d-\Delta d_{max})^2} && \text{by 8} \\
&\leq k_d \sum_{i=1}^{m} I_i \frac{\min(1, \cos(\theta) + \Delta\theta_{max})}{(d-\Delta d_{max})^2} \\
&\leq k_d I_v \frac{\min(1, \cos(\theta) + \Delta\theta_{max})}{(d-\Delta d_{max})^2} = I_{max} \,.
\end{aligned}$$

From there, the bound on the error is the maximum between $I_{approx} - I_{min}$ and $I_{max} - I_{approx}$. Re-arranging few

terms gives:

$$\Delta I_{abs} = k_d I_v \max\left(\frac{\cos(\theta)}{d^2} - \frac{\max(0,\cos(\theta)-\Delta\theta_{max})}{(d+\Delta d_{max})^2}, \frac{\min(1,\cos(\theta)+\Delta\theta_{max})}{(d-\Delta d_{max})^2} - \frac{\cos(\theta)}{d^2}\right). \quad (10)$$

Specular bound

The specular bound is simply an approximation of the maximal specular illumination:

$$\begin{aligned}
I &= \sum_{i=1}^{m} I_i \frac{1}{d_i^2} k_s \cos^n(\alpha_i) \\
&= k_s \sum_{i=1}^{m} I_i \frac{\cos^n(\alpha+\Delta\alpha_i)}{(d+\Delta d_i)^2} \\
&\le k_s \sum_{i=1}^{m} I_i \frac{\cos^n(\alpha+\Delta\alpha_i)}{(d+\Delta d_{max})^2} && \text{by 7} \\
&\le k_s \sum_{i=1}^{m} I_i \frac{\min(1,\cos^n(\alpha+\Delta\alpha_i))}{(d+\Delta d_{max})^2} && \text{by 9} \\
&\le k_s \sum_{i=1}^{m} I_i \frac{\min(1,\cos^n(\alpha+|\Delta\alpha_i|))}{(d+\Delta d_{max})^2} && \text{by 8} \\
&\le k_s \sum_{i=1}^{m} I_i \frac{\min(1,\cos^n(\alpha+\Delta\alpha_{max}))}{(d+\Delta d_{max})^2} \\
&\le k_s I_v \frac{\min(1,\cos^n(\alpha+\Delta\alpha_{max}))}{(d+\Delta d_{max})^2} = maxC\,.
\end{aligned}$$

References

1. T. Whitted, "An improved illumination model for shaded display", *Communications of the ACM*, **23**(6), pp. 343–349 (1980).
2. M. F. Cohen and J. R. Wallace, *Radiosity and Realistic Image Synthesis*. Academic Press, (1993).
3. F. Sillion and C. Puech, *Radiosity and Global Illumination*. Morgan Kaufmann, (1994).
4. P. Shirley, C. Y. Wang, and K. Zimmerman, "Monte carlo techniques for direct lighting calculations", *ACM Transactions on Graphics*, **15**(1), pp. 1–36 (1996).
5. A. S. Glassner, "Space subdivision for fast ray tracing", *IEEE Computer Graphics and Applications*, **4**(10), pp. 15–22 (1984).
6. A. Fujimoto, T. Tanaka, and K. Iwata, "Arts: Accelerated ray-tracing system", *IEEE Computer Graphics and Applications*, pp. 16–26 (1986).
7. S. M. Rubin and T. Whitted, "A 3-dimensional representation for fast rendering of complex scenes", *Computer Graphics (SIGGRAPH '80 Proceedings)*, **14**(3), pp. 110–116, (1980).
8. P. Hanrahan, D. Salzman, and L. Aupperle, "A rapid hierarchical radiosity algorithm", in *Computer Graphics (SIGGRAPH '91 Proceedings)*, **25**(4), pp. 197–206, (July 1991).
9. B. Smits, J. Arvo, and D. Greenberg, "A clustering algorithm for radiosity in complex environments", in *Proceedings of SIGGRAPH '94*, Computer Graphics Proceedings, pp. 435–442, ACM SIGGRAPH, (July 1994).
10. F. Sillion, "Clustering and volume scattering for hierarchical radiosity calculations", in *Fifth Eurographics Workshop on Rendering*, (Darmstadt, Germany), pp. 105–117, (June 1994).
11. B.-T. Phong, "Illumination for computer generated pictures", *Communications of the ACM*, **18**(6), pp. 311–317 (1975).
12. C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile, "Modelling the interaction of light between diffuse surfaces", in *Computer Graphics (SIGGRAPH '84 Proceedings)*, **18**(3), pp. 213–222, (July 1984).
13. P. Hanrahan and D. Salzman, "A rapid hierarchical radiosity algorithm for unoccluded environments", in *Proceedings Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics*, (Rennes, France), pp. 151–169, (June 1990).
14. M. F. Cohen and D. P. Greenberg, "The Hemi-Cube: A radiosity solution for complex environments", in *Computer Graphics (SIGGRAPH '85 Proceedings)*, **19**(3), pp. 31–40, (Aug. 1985).
15. A. Woo, P. Poulin, and A. Fournier, "A survey of shadow algorithms", *IEEE Computer Graphics and Applications*, **10**(6), pp. 13–32 (1990).
16. P. Bergeron, "A general version of Crow's shadow volumes", *IEEE Computer Graphics and Applications*, **6**(9), pp. 17–28 (1986).
17. G. Ward, "Adaptive shadow testing for ray tracing", in *Eurographics Workshop on Rendering*, pp. 11–20, (1991).
18. C. Houle and E. Fiume, "Light-source modeling using pyramidal light maps", *CVGIP: Graphical Models and Image Processing*, **55**(5), pp. 346–358 (1993).
19. J. Stam and E. Fiume, "Depicting fire and other gaseous phenomena using diffusion processes", in *SIGGRAPH 95 Conference Proceedings*, pp. 129–136, ACM SIGGRAPH, (Aug. 1995).
20. E. Haines and J. Wallace, "Shaft culling for efficient ray-traced radiosity", in *Eurographics Workshop on Rendering*, pp. 122–138(1991).
21. S. Teller and P. Hanrahan, "Global visibility algorithms for illumination computations", in *Computer Graphics Proceedings (Proceedings of SIGGRAPH '93)*, pp. 239–246, (1993).
22. F. X. Sillion, "A unified hierarchical algorithm for

global illumination with scattering volumes and object clusters", *IEEE Transactions on Visualization and Computer Graphics*, **1**(3), pp. 240–254 (1995).

23. F. Cazals, G. Drettakis, and C. Puech, "Filtering, clustering and hierarchy construction: a new solution for ray tracing very complex environments", *Computer Graphics Forum (Proc. of Eurographics '95)*, **15**(3), pp. 371–382 (1995).