Thirteenth Eurographics Workshop on Rendering (2002)
P. Debevec and S. Gibson (Editors)

Local Illumination Environments for Direct Lighting Acceleration

Sebastian Fernandez Kavita Bala Donald P. Greenberg

Program of Computer Graphics, Cornell University, Ithaca, NY, USA

Abstract

Computing high-quality direct illumination in scenes with many lights is an open area of research. This paper presents a world-space caching mechanism called ***local illumination environments*** *that enables interactive direct illumination in complex scenes on a cluster of off-the-shelf PCs.*
A local illumination environment (LIE) caches geometric and radiometric information related to direct illumination. A LIE is associated with every octree cell constructed over the scene. Each LIE stores a set of visible lights, with associated occluders (if they exist). LIEs are effective at accelerating direct illumination because they ***both*** *eliminate shadow rays for fully visible and fully occluded regions of the scene, and decrease the cost of shadow rays in other regions. Shadow ray computation for the partially occluded regions is accelerated using the cached potential occluders. One important implication of storing occluders is that rendering is accelerated while producing accurate hard and soft shadows. This paper also describes a simple perceptual metric based on Weber's law that further improves the effectiveness of LIEs in the fully visible and partially occluded regions.*
LIE construction is view-driven, continuously refined, and asynchronous with the shading process. In complex scenes of hundreds of thousands of polygons with up to a hundred lights, the LIEs improve rendering performance by $10\times$ *to* $30\times$ *over a traditional ray tracer.*

1. Introduction

Computing high-quality direct illumination at interactive rates in scenes with many lights is a hard problem. Existing hardware approaches do not scale with many lights. Software-based approaches, such as ray tracers, are more promising in their ability to scale with complex scenes and lighting [16]. However, such approaches do not scale well with the number of light sources.

This paper presents, *local illumination environments*, a world-space caching approach that accelerates direct illumination in scenes with many lights. LIEs are associated with the octree cells of the scene. Each LIE caches geometric and radiometric information: for visible lights, the light is stored, for partially visible lights, the occluders that might occlude shadow rays for the region are stored, and for fully occluded lights no information is stored.

LIEs are based on the observation that typically, the direct lighting observed in any particular region of a scene is not perceptually complex. This allows LIEs to perform the following optimizations:

- Lights that are not visible at any point on a surface within a cell are completely ignored.
- Lights that are visible at all points on surfaces within a cell do not have any shadow rays cast to them.
- Lights that are visible at some points on surfaces within a cell and not visible at others have shadow rays cast to them, but these shadow rays do occlusion testing only against a minimal set of occluders.

Thus, LIEs accelerate rendering by decreasing the number *and* cost of the expensive visibility computations for shadow rays; it is these visibility computations that make direct illumination so expensive in scenes with many lights. Additionally, a simple perceptual metric based on Weber's law is used to eliminate the contribution of fully and partially visible lights that are perceptually unimportant.

LIEs have three important properties. First, LIEs permit accurate computation of shadows because they include ge-

© The Eurographics Association 2002.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

ometry (the occluders) in the partially occluded regions. Thus, they accelerate visibility without introducing error.

Second, LIEs can be easily integrated into a ray-tracing system. Because they only cache the set of potentially visible lights and the geometry which might occlude them, they do not hinder the most valuable feature of a ray tracer, its flexibility. In particular, LIEs allow the use of any type of geometric primitive and any type of material that can be supported by a ray tracer. This also means that LIEs can be used in systems that require a ray tracer to perform direct lighting, such as global illumination systems.

Third, LIEs are designed to work interactively. They are constructed lazily in a view-driven manner as the user navigates the scene. Using 24 processors, we render scenes with hundreds of thousands of polygons with up to 100 lights at 1-2 frames per second, achieving performance improvements from 10$\times$ to 30$\times$ over a traditional ray tracer.

2. Previous Work

Several researchers have attempted to accelerate direct illumination. But most direct lighting algorithms do not scale well with the number of lights in the scene. The focus of this paper is on accurate direct illumination at interactive rates in scenes with many lights. Some researchers have started to address this problem; we discuss their approaches below.

Ward [17], accelerates the rendering of many lights using a user-specified threshold to eliminate lights that are less important. For each pixel in an image, the system sorts the lights according to their maximum possible contribution (assuming no occlusion). Occlusion for each of the largest possible contributors at the pixel is tested, measuring their actual contribution to the pixel, and stopping at a predetermined energy threshold. This approach can reduce the number of occlusion tests, however it does not reduce the cost of occlusion tests that do have to be performed and does not do very well when illumination is uniform.

Shirley et al. [15] devote a section to accelerating direct illumination from many lights. Their approach subdivides the scene into voxels and, for each voxel, partitions the set of lights into an important set and an unimportant set. Each light in the important set is sampled explicitly. One light is picked at random from the unimportant set as a representative of the set and sampled. The assumption is that the unimportant lights all contribute the same amount of energy.

To determine the set of important lights, they construct an “influence box" around each light. An influence box contains all points on which the light could contribute more than the threshold amount of energy. This box is intersected with voxels in the scene to determine which voxels the light is important for. This is an effective way to deal with many lights. However, the approach is geared towards producing still images since many samples per pixel are required to reduce the noise inherent in sampling the light set.

Paquette et al. [11] present a light hierarchy for rendering scenes with many lights quickly. This system builds an octree around the set of lights, subdividing until there are less than a predetermined number of lights in each cell. Each octree cell then has a “virtual light" constructed for it that represents the illumination caused by all the lights within it. They derive error bounds which can determine when it is appropriate to shade a point with a particular virtual light representation and when traversal of the hierarchy to finer levels is necessary. Their algorithm can deal with thousands of points lights.

One major limitation of this approach is that it does not take visibility (i.e., occlusion) into consideration. However, the approach of Paquette et al. is complementary to our approach and can be used to cluster the fully visible lights in our rendering system.

Kok et al. [9] present a method for accelerating the gathering phase of a radiosity algorithm. Their algorithm selects a set of light sources to explicitly sample and determines how well each of those light sources should be sampled based on several criteria. Light sources not chosen to be explicitly sampled are folded into the radiosity value for the patch and thus interpolated.

Scheel et al. [13] take a similar approach, expanding the criteria for selecting explicitly sampled light sources to include perceptual metrics. Their algorithm also allows the decision of interpolation vs. sampling to be made separately for the form factor and visibility terms.

While the techniques of both Kok et al. [9] and Scheel et al. [13] allow certain light sources to be interpolated instead of sampled, they do not provide an acceleration technique for the cases in which lights do have to be sampled. Although Scheel et al. [13] do permit interpolation of some partially occluded sources, this is possible only for broad penumbras, and not for sharper shadows. Additionally, having to store lighting information on every patch can limit the size of scenes that can be used with these approaches. Scheel et al. [14] address this latter problem in a paper published contemporanously with this one.

Haines et al. [5] substantially accelerated shadow rays by explicitly keeping track of occluding geometry and storing it in a cube around the light source. However, their technique does not work for area lights and also does no specific acceleration for many lights.

Priol et al. [12] performed a precomputation of the geometry that could possibly cast shadows upon a region in space. This allowed them to relieve network congestion in a parallel ray tracing environment. However, this computation is too conservative, particularly for regions that are completely occluded with respect to the light.

Wald et al. [16] have recently demonstrated the viability of interactive ray tracers on clusters of PCs. However, their work focuses on primary visibility and simple lighting and does not address the complex lighting issue.

Recently, there has been much research on shadow map techniques for interactive direct illumination [1, 4, 10]. However, these approaches do not scale well with large numbers of light sources in a scene.

The rest of this paper is organized as follows: Section 3 describes the overall system. Section 4 presents local illumination environments, and Section 5 describes a simple perceptual metric that further improves LIE performance. Section 6 presents results, and finally, we conclude with a discussion of future work in Section 7.

3. System

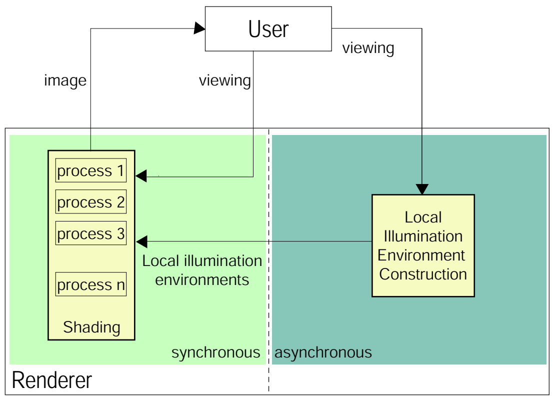

Figure 1: *System structure. Left: LIE Shaders compute subsets of the image. Right: LIE constructor generates LIEs to be used by the shaders.*

This section describes the overall structure of the system. A user sends navigation commands to the rendering system, which interactively displays images to that user.

The system is split into two modules. As the user navigates the scene, the viewpoint is sent to the LIE constructor. This module continuously computes new LIEs if needed or refines existing LIEs. The LIE constructor typically runs on a single system and communicates changes in the computed LIEs to the shaders which run on separate systems.

The LIE shaders consist of several parallel renderers that use LIEs to render the pixels assigned to them. The renderers are not synchronized with the LIE constructor, so the shaders do not have to wait on the results of LIE construction. Also, computing any one pixel does not depend on the computation of any other pixel, so there is no communication among the shaders. The computed pixels are sent over a local area network where they are assembled into an image.

At startup, the model is distributed to the LIE constructor and the LIE shaders. A regular grid acceleration structure is constructed and replicated on each system. This regular grid is used to accelerate both the ray casts needed to construct LIEs and the primary visibility ray computation used in both the constructor and the shaders.

4. Local Illumination Environments

A local illumination environment (LIE) consists of a bounding box (cell) along with a set of visible lights, each with a (possibly empty) accompanying set of blocking geometry. LIEs are used to faithfully reproduce the incident radiance upon every surface point within the cell. The set of lights and the set of occluders are taken directly from the scene geometry.

In the sections to follow, we will refer to lights as being *unoccluded*, *partially occluded*, or *fully occluded*. A light is *unoccluded* with respect to the cell associated with the LIE iff every point on a model surface contained within the cell can see every point on the light. A light is *fully occluded* with respect to the cell associated with the LIE iff no point on a model surface contained within the cell can see any point on the light. Otherwise, a light is *partially occluded* with respect to the cell associated with the LIE.

4.1. LIE Construction

For optimal performance, local illumination environments should be as simple as possible. Ideally, the local illumination environment for a cell should include only lights that affect illumination for surfaces within that cell. Similarly, for each visible light in a local illumination environment, the associated occluder set should only include the occluders that actually occlude the light.

Computing this minimal occluder set is potentially expensive. Shaft culling [6] could be used to construct a set of occluders; however, shaft culling is typically too conservative, including a lot of occluders that are not actually relevant for illumination. Therefore, in our LIE construction, we *sample* the visibility between a surface and light to determine occluder lists, as done in [7]. This technique is inexpensive and is guaranteed not to add any lights not seen from the cell nor any occluders not actually blocking a light from the cell. Sometimes a lot of samples are required to find all the required lights and occluders. Because of this, some artifacts are visible as LIEs are constructed. However, over time the LIEs converge to an accurate representation of visibility and the artifacts go away.

For interactive walkthroughs LIEs are computed lazily using a view-driven approach. The LIE constructor picks a point randomly on the image plane and traces a ray from the eye to the closest visible point on a surface. The LIE constructor then constructs an LIE or improves an existing LIE for the smallest enclosing octree cell for that point. A ray is cast from the point to random points on all the lights

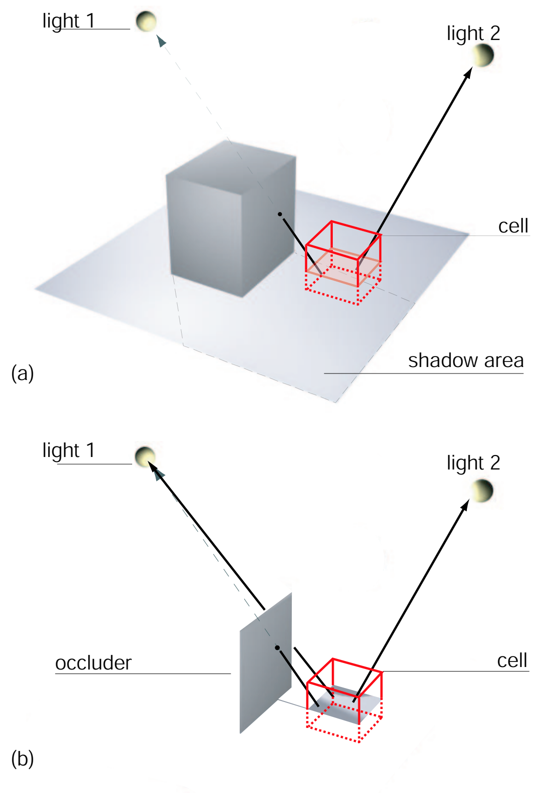

Figure 2: *LIE Construction. (a) Detection of occluders and lights, (b) the LIE for the cell.*

in the scene. If the ray is unobstructed for a particular light and the LIE does not contain that light, that light is added to the LIE (unoccluded case). If the ray is obstructed and the LIE does not contain that particular light, the LIE is not modified (occluded case). If the ray is obstructed but the LIE does contain that light, the obstructing geometry is added to the set of occluders for that light (partially occluded case). Figure 2 illustrates this process.

Different parts of the scene will have different illumination complexities. If it is determined that the LIE for a certain octree cell has become too complex, the cell is subdivided. We currently measure complexity as the sum of all the occluders in the LIE. The LIEs for the child cells are then generated from scratch. This allows us to maintain the invariant that only lights that can actually be seen from a cell are in an LIE and only occluders that occlude a cell's view of a light are in its occluder list. Currently we use a fixed maximum depth for the octree subdivision. We have found that the optimal maximum depth in terms of the LIE performance/LIE generation tradeoff varies as a function of the scene. While a maximum depth of 6 is appropriate for the Science Center scene, a value of 10 works better for the Mosque de Cordoba scene.

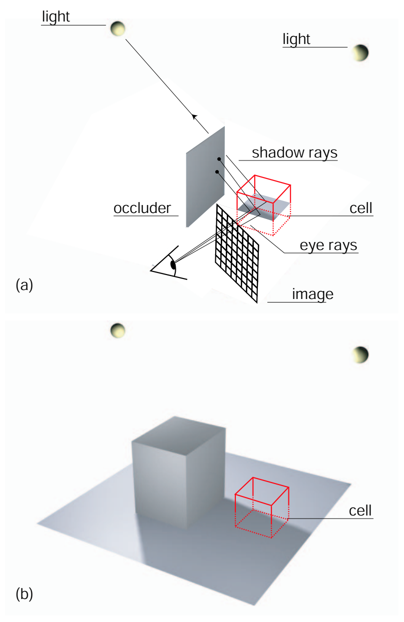

Figure 3: *LIE Shading. (a) Shaders using the LIEs, (b) Rendered image.*

4.2. Shading Using LIEs

When rendering a frame, the ray tracer is used to determine the closest visible object for each pixel. The appropriate LIE used for shading this point is found by descending the octree hierarchy. For each partially occluded light in the LIE a ray is cast from the point to be shaded to a sample point on the light, as shown in Figure 3. This ray is tested for intersection with the list of occluders for that light. If the ray is not blocked, the incident radiance from the light source is multiplied by the BRDF and added to the exitant radiance. Monte Carlo sampling is used to integrate the contribution from area light sources. For fully visible lights in the LIE no visibility computation is done.

LIEs tend to be small resulting in fewer intersection tests than with a traditional ray-tracing acceleration approaches. The simple structure of the occluder lists also incurs little overhead.

5. Masking

In scenes with large lighting complexity, parts of the scene could be illuminated with several lights that do not make important contributions to the perception of illumination. Ignoring these unimportant lights lets us further decrease the computational cost of rendering direct illumination. These lights must be picked carefully so as to avoid introducing noticeable artifacts. We use a simple perceptual metric based on Weber's law [8] that identifies lights (fully visible, or partially occluded) that are *masked*; i.e., these are lights that could be eliminated to accelerate rendering. Thus, these lights are effectively treated as being fully occluded.

Weber's law specifies that variations in brightness cannot be perceived if those variations are below a certain fraction of the average brightness. This fraction varies with intensity and spatial frequency, but has a minimum value of 2%.

To use Weber's law we have to obtain some estimate of the overall energy over an octree cell. As LIEs are constructed our system keeps track of the minimum and maximum exitant radiance over a region due to each light and sorts the lights based on their maximum exitant radiance. We then remove lights from the LIE starting with the light whose maximum contribution is smallest. We accumulate the error introduced, and stop removing lights when this error exceeds 2% of the minimum radiance for all the lights combined. A similar approach is used by Ward [17], on a per-pixel basis.

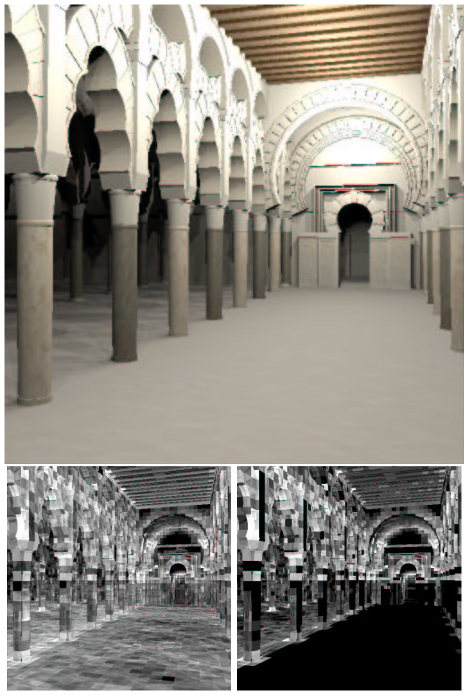
Figure 4: *Top: Image of Mosque de Cordoba scene. Bottom left: Cost to render each pixel without masking (whiter is more expensive). Bottom right: Cost to render each pixel with masking.*

Figure 4 shows an image of a particular section of the Mosque de Cordoba test scene. The scene happens to have a large number of relatively dim lights with a row of bright lights running down the main corridor seen in the picture. The columns surrounding the corridor cast shadows on the floor due to all the surrounding lights. However, the brighter lights above the corridor completely mask these shadows so that they are not visible in the image.

The bottom visualizations show the cost of rendering each pixel: on the left, masking is not used, on the right masking is enabled. The generated images are perceptually identical; in fact, the difference image (not shown here) appears black for this view. However, the cost to render the pixels in the corridor is significantly reduced when using masking as is particularly noticeable on the floor. The performance difference is shown in Table 1. Note that the difference image and results are for a converged set of LIEs. Non-converged LIEs would show visibility artifacts unrelated to masking.

6. Results

We tested our algorithm on several scenes. We present results for the following three scenes that differ in their lighting, materials, and complexity. The Science Center is a simple scene with 7 thousand polygons and one large area light source, with completely diffuse materials. Bar Carta Blanca is a scene with 240 thousand polygons and 70 spotlights with mostly diffuse materials and a glossy floor. Mosque de Cordoba is a scene with 980 thousand polygons, 100 omnidirectional point light sources, and glossy columns.

LIE construction was done on a single dual-processor Pentium-4 PC running at 1.7 Ghz. LIE shading was done on 10 dual-processor Pentium-4 PCs running at 1.7 Ghz. A 100 Mbit Ethernet network connects the machines. The LIE constructor communicates with the shaders on this network. Rendered pixels are also transmitted on this network. Both the standard ray tracer and the LIE system were written in Java and ran on the Sun Java Virtual Machine version 1.3.1.

Before obtaining timing results, the LIE data was precomputed by walking around the scenes until the LIEs were mostly converged. The Science Center required only seconds of precomputation, while Bar Carta Blanca required a couple of minutes and Mosque de Cordoba required about an hour. Timings are for particular views within the scenes and are representative of timings obtained for other views. Timings are in seconds per frame for a 512x512 image.

For comparison, Table 1 shows the performance of a standard ray tracer, the same ray tracer modified to use Ward's algorithm for many lights, and our LIE based ray tracer. Note that all these ray tracers have been parallelized. The timings shown are for representative viewpoints within each scene. The algorithm shows consistent speedups over traditional ray tracing. The standard ray tracer uses a two-level hierarchical regular grid acceleration structure whose resolution adapts to the size of the model. As model complexity increases and shadow-ray costs for the standard ray tracer go

Model	Science Center		Bar Carta Blanca		Mosque de Cordoba	
	Time	Speedup	Time	Speedup	Time	Speedup
Standard	15.0s	1.0x	10.1s	1.0x	35.0s	1.0x
Ward	15.0s	1.0x	10.1s	1.0x	11.5s	3.0x
LIE (no masking)	1.30s	**11.5x**	0.69s	**14.6x**	1.45s	**24.1x**
LIE (masking)	1.30s	**11.5x**	0.69s	**14.6x**	1.20s	**29.2x**

Table 1: *Performance comparison.*

up, our speedup increases, going from 11× for the Science Center to 29× for the more complex Mosque de Cordoba.

The Science Center scene (Figure 5) shows that we can achieve substantial speedups even in small environments where the cost of traditional shadow rays is not large. In this scene the performance improvement is mainly due to not having to cast shadow rays for fully occluded or fully visible lights. This cost savings can be substantial for scenes with large area lights because of the large number of shadow rays required.

The Bar Carta Blanca scene (Figure 6) demonstrates the algorithm's performance on complex scenes with many point lights. In this scene, accelerating shadow computation for the partially occluded regions is more important than in the previous scene due to the high cost of shadow rays. LIEs are effective at accelerating these rays. Although this scene contains many lights, masking is not very effective. In this environment, far away lights tend to be fully occluded and thus incur zero cost in rendering.

The Mosque de Cordoba scene (Figure 7) demonstrates the additional performance improvements that can be obtained by taking advantage of light masking. The mosque is lit by a row of 10 bright lights in the central aisle, and 90 dimmer lights to the sides. Light masking is particularly effective in reducing the cost of direct illumination in such a scene in the regions near the bright lights.

Table 2 breaks down the performance into three components. The visibility component is the cost of casting a ray from the eye to the first surface using a conventional ray tracer. This component is not accelerated by our algorithm. However, we present it as a point of comparison since the visibility computation is a lower bound on the time required to render a frame. The next component is the time spent rendering unoccluded lights. This involves a BRDF evaluation, a light emission evaluation and a few dot products for each light. The final component is time spent rendering partially occluded lights. This involves intersection testing with the set of occluders in the LIE for each partially occluded light as well as the emission and BRDF evaluation if the light is determined to be visible. For area lights, these computations have to be done once for each sample point on the light.

The table shows the performance breakdown for LIEs without masking enabled. A significant effect from masking can be seen in the Mosque de Cordoba scene where enabling masking further decreases the time for rendering partially occluded lights by about 0.25 seconds.

The breakdown shows that in several cases the cost of performing the direct lighting computations is on the same order as the cost of performing the visibility computation. Thus, the LIE is effective at accelerating direct illumination. The table also shows that neither fully visible nor partially occluded lights are bottlenecks in the performance. Instead, both have to be further optimized for better performance.

Memory usage for the LIE data structure is not very high. Although it does have to store a collection of lights and occluders for each octree cell, these are merely references to scene geometry and thus take up little space. For the 980 thousand triangle Mosque de Cordoba model, 50 megabytes of data were used by the LIE data structure. This is less than the memory used for the model.

7. Conclusions

We have introduced an approach for accelerating direct illumination calculations using local illumination environments. LIEs spatially cache visibility and radiometric information in a scene. This caching allows fast rendering by reducing the number of shadow rays cast and by decreasing the cost of the shadow rays that must be cast. LIEs are flexible and can be easily introduced into a ray-tracing system. We have also shown how perceptual masking can be used in conjunction with LIEs to reduce the number of shadow ray casts in regions where shadows are hard to perceive.

We have implemented a system that renders direct illumination at 1-2 fps using LIEs on a cluster of PCs. Our system demonstrates performance improvements over conventional ray tracing of 10× to 30×. We believe LIEs can be easily integrated in other systems to accelerate shadow ray computations for interactive walkthroughs.

In the future, we would like to improve the heuristics to guide sampling and octree subdivision during LIE construction. Additional perceptual metrics that capture texture

Model	Science Center	Bar Carta Blanca	Mosque de Cordoba
Visibility	0.35s	0.48s	0.59s
Unoccluded lights	0.05s	0.14s	0.34s
Partially occluded lights	0.90s	0.07s	0.52s
Total	1.30s	0.69s	1.45s

Table 2: *Computation breakdown.*

masking effects could also be included. A clustering approach for fully visible lights (similar to Paquette et al. [11]) would be useful to improve the performance of LIE construction and shading. Integrating LIEs with an interpolation technique [13] could further accelerate rendering. Additionally, a technique such as that described by Bala et al. [2] could be used to dynamically update generated LIEs for dynamic scenes. A preliminary implementation [3] on patch-based LIEs appears promising.

Acknowledgements

We would like to thank Guillermo M. Leal LLaguno for the use of his Bar Carta Blanca model. We would also like to thank Ivan Rossello and Yasemin Kologlu for building the Mosque de Cordoba model. Finally, we would like to thank Moreno Piccolotto for his assistance and unending supply of quality caffeine.

This work was supported by the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-8920219) and by the MRA parallel global illumination project (ASC-9523483), and performed using equipment generously donated by the Intel Corporation and the DeKorte family.

References

1. M. Agrawala, R. Ramamoorthi, A. Heirich, and L. Moll. Efficient image-based methods for rendering soft shadows. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 375–384. ACM SIGGRAPH, 2000. 3

2. K. Bala, J. Dorsey, and S. Teller. Interactive ray-traced scene editing using ray segment trees. In *Tenth Eurographics Workshop on Rendering*, pages 39–52, June 1999. 7

3. S. Fernandez, K. Bala, M. Piccolotto, and D. Greenberg. Interactive direct lighting in dynamic scenes. Technical Report PCG-00-2, Program of Computer Graphics, Cornell University, Ithaca, NY, January 2000. 7

4. R. Fernando, S. Fernandez, K. Bala, and Donald P. Greenberg. Adaptive shadow maps. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 387–390, August 2001. E. Fiume, editor. 3

5. E. Haines and D. Greenberg. The light buffer: A shadow-testing accelerator. *IEEE Computer Graphics & Applications*, 6(9):6–16, September 1986. 2

6. E. Haines and J. Wallace. Shaft culling for efficient ray-traced radiosity. In P. Brunet and F. W. Jansen, editors, *Photorealistic Rendering in Computer Graphics (Proceedings of the Second Eurographics Workshop on Rendering)*, New York, NY, 1994. Springer-Verlag. 3

7. D. Hart, P. Dutré, and D. Greenberg. Direct illumination with lazy visibility evaluation. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, Annual Conference Series, pages 147–154, August 1999. 3

8. D. C. Hood and Finkelstein. Volume 1: Sensory processes and perception. In *Handbook of perception and human performance*, New York, NY, 1986. John Wiley & Sons. 5

9. A. J. F. Kok and F. W. Jansen. Source selection for the direct lighting computation in global illumination. In P. Brunet and F. W. Jansen, editors, *Photorealistic Rendering in Computer Graphics*, pages 75–82, 1994. 2

10. T. Lokovic and E. Veach. Deep shadow maps. In *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 385–392, July 2000. K. Akeley, editor. 3

11. E. Paquette, P. Poulin, and G. Drettakis. A light hierarchy for fast rendering of scenes with many lights. *Eurographics '98*, 17(3), September 1998. 2, 7

12. T. Priol and K. Bouatouch. Static load balancing for a parallel ray tracing on a mimd hypercube. *The Visual Computer*, 12(5):109–119, 1989. 2

13. A. Scheel, M. Stamminger, and H.-P. Seidel. Thrifty final gather for radiosity. In *12th Eurographics Workshop on Rendering*, pages 1–12, June 2001. 2, 7

14. A. Scheel, M. Stamminger, and H.-P. Seidel. Grid based final gather for radiosity on complex clustered scenes. In *Eurographics '02*, 2002. To be Published. 2

15. P. Shirley, C. Wang, and K. Zimmermann. Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1), January 1996. 2

16. I. Wald and P. Slusallek. State of the art in interactive ray tracing. In *State of the Art Reports, Eurographics 2001*, pages 21–42. Eurographics, Manchester, United Kingdom, 2001. 1, 2

17. G. Ward. Adaptive shadow testing for ray tracing. In *Photorealistic Rendering in Computer Graphics (Proceedings of the*

Second Eurographics Workshop on Rendering), pages 11–20, New York, 1994. Springer-Verlag. 2, 5

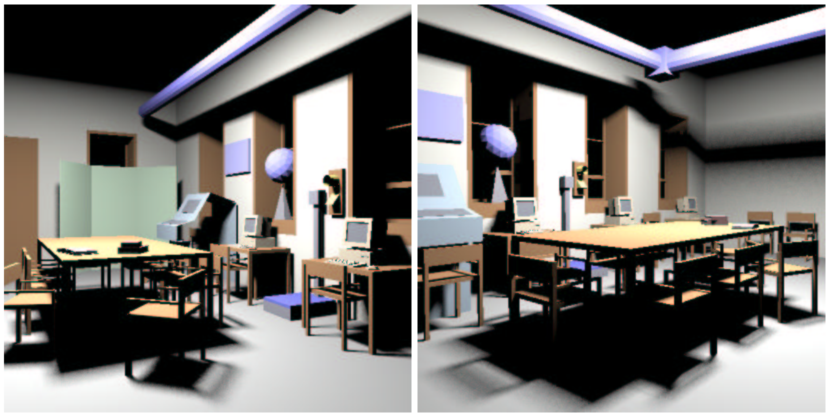

Figure 5: *Science Center.*

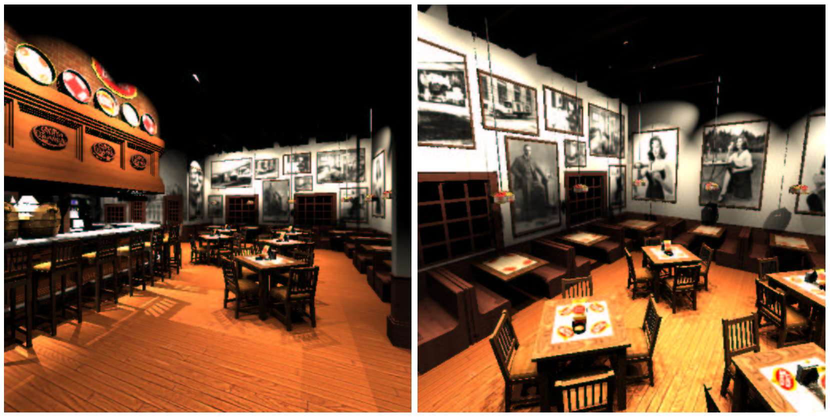

Figure 6: *Bar Carta Blanca.*

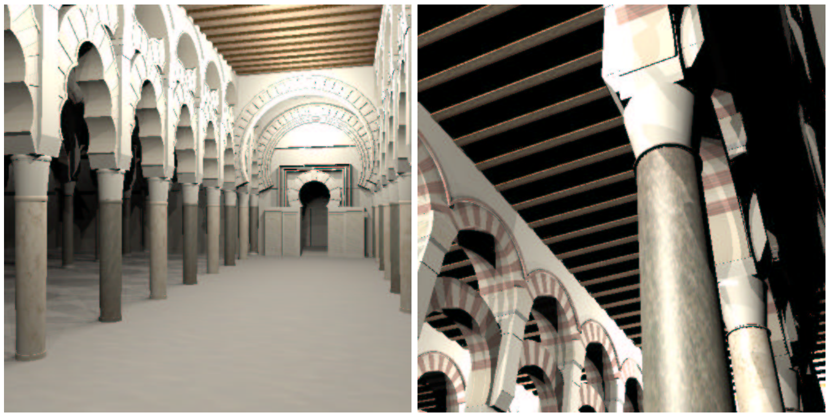

Figure 7: *Mosque de Cordoba.*