Canned Lightsources

Wolfgang Heidrich, Jan Kautz, Philipp Slusallek, Hans-Peter Seidel
University of Erlangen, Computer Graphics Group
Am Weichselgarten 9, D-91058 Erlangen, Germany
Email: {heidrich,jnkautz,slusallek,seidel}@immd9.informatik.uni-erlangen.de

Abstract. Complex luminaries and lamp geometries can greatly increase the realism of synthetic images.
Unfortunately, the correct rendering of illumination from complex lamps requires costly global illumination algorithms to simulate the indirect illumination reflected or refracted by parts of the lamp. Currently, this simulation has to be repeated for every scene in which a lamp is to be used, and even for multiple instances of a lamp within a single scene.
In this paper, we separate the global illumination simulation of the interior lamp geometry from the actual scene rendering. The lightfield produced by a given lamp is computed using any of the known global illumination algorithms. Afterwards, a discretized version of this lightfield is stored away for later use as a lightsource. We describe how this data can be efficiently utilized to illuminate a given scene using a number of different rendering algorithms, such as ray-tracing and hardware-based rendering.

1 Introduction

Complex lightsources can greatly contribute to the realism of computer generated images, and are thus interesting for a variety of applications. However, in order to correctly simulate the indirect light bouncing off internal parts of a lamp, it is necessary to apply expensive global illumination algorithms.

As a consequence, it was so far not possible to use realistic light sources in interactive applications. Also, for offline techniques such as ray-tracing, complex light source geometry unnecessarily slows down the rendering, since reflections and refractions between internal parts of the lightsource have to be recomputed for every instance of a lightsource geometry in every frame.

In this paper we propose a new method for using realistic lightsources for image synthesis. For a given lamp geometry and luminary, the outgoing lightfield is computed using standard global illumination methods, and stored away in a Lumigraph [6, 9] data structure. Later the lightfield can be used to illuminate a given scene while abstracting from the original lamp geometry. We call a light source stored and used in this fashion a “Canned Lightsource”.

Instead of simulating the lightfield with global illumination methods, Canned Lightsources could also be measured [2], or even provided by lamp manufacturers in much the same way far field information is provided today. Thus, a database of luminaries and lamps stored as Canned Lightsources becomes possible.

Our method also speeds up the rendering process by factoring out the computation of the internal reflections and refractions of the lightsource into a separate preprocessing step. As a consequence, realistic lightsources can be efficiently used for interactive rendering, and for other applications where a complete global illumination solution

would be too expensive. The cost of computing the global illumination within the lamp can be amortized over a large number of lamp instances and frames.

The remainder of this paper is organized as follows: in Section 2, we review the relevant previous work in this area. Section 3 contains a brief discussion of the acquisition of lightfields using global illumination algorithms. Section 4, the main contribution of this paper, describes ways to use Canned Lightsources in combination with several rendering algorithms, including ray-tracing and hardware-based methods. We conclude with some results and a discussion in Sections 5 and 6.

2 Previous Work

Several researchers have used precomputed global illumination solutions to efficiently generate realistic walkthroughs. A common technique is to use Gouraud shading and texture mapping to render radiosity solutions of diffuse environments from any perspective [12, 3]. Other approaches for diffuse scenes include Instant Radiosity [8] and Irradiance Volumes [7]. Interactive radiosity methods such as [4] incrementally update the illumination in scenes with small numbers of moving objects. Recently, algorithms for walkthroughs of scenes containing specular surfaces have been developed [14, 15].

All of the above methods require a *full* global illumination solution for the complete scene. Thus, they cannot deal with large numbers of changing objects or moving lightsources. Furthermore, the methods are not capable of storing away partial solutions for future use in different environments.

Methods more closely related to our approach include virtual walls [1] for speeding up radiosity computations, and the Lucifer algorithm [5, 10], which computes the radiance distribution on a virtual surface in the environment. Radiance [16] allows the user to replace geometry seen through small openings such as windows by dense radiance samples. None of these methods reuses illumination information in different environments.

In this paper, we encapsulate the lightfield emitted by a lightsource and store it away for future use. To this end, the method presented here relies on a dense sampling of the radiance emitted by a lightsource. The Lumigraph data structure[1] used to store this information has been developed by Levoy and Hanrahan [9], and Gortler et al. [6].

3 Simulating and Sampling the Lightfield of a Lightsource

As mentioned above, we store the lightfield in a Lumigraph data structure. From this data structure the radiance values can be extracted efficiently. Moreover, Lumigraphs can be generated easily using existing rendering systems or real world images, and since they are merely a two dimensional array of projective images, they lend themselves to efficient algorithms using computer graphics hardware (see Section 4.3).

On the down side, the light slabs of a Lumigraph do not uniformly sample the lightfield, and multiple light slabs are required to cover all light directions. Despite these disadvantages, we think that the Lumigraph is well suited for our purposes.

The top of Figure 1 shows the geometry of a single light slab. Every grid point corresponds to a sample location. As in [6] the (u, v) plane is the plane close to the object (lamp), while the (s, t) plane is further away, potentially at infinity.

[1] In this paper, we use the term "lightfield" for the continuous 5D function that describes the radiance at every point in space in every direction. The "Lumigraph", which is composed of one or more "light slabs", refers to the 4D data structure used to store a finite number of samples from the lightfield in empty space.

Placing the (s,t) plane at infinity offers the advantage of a clear separation between spatial sampling on the (u,v) plane, and directional sampling on the (s,t) plane. In this situation, all rays through a single sample on the (u,v) plane, that is, the projective image through that point, represent the far field of the lightsource. The higher the resolution of the (u,v) plane, the more nearfield information is added to the Lumigraph. For most applications, the spatial resolution is much smaller than the directional resolution.

If, the (s,t) plane is *not* placed at infinity, it is not possible to clearly separate between spatial and directional resolution. Nonetheless, we can still view the light slab as an array of projective images with centers of projection on the (u,v) plane.

Because of this, a light slab representing a Canned Lightsource can be created by first generating a global illumination solution for the lamp geometry, and then rendering it from multiple points of view on the (u,v) plane. For the global illumination step, we can use any kind of algorithm, such as radiosity, Monte-Carlo ray-tracing, Photon Maps, or composite methods [13]. Alternatively, Canned Lightsources could be generated by resampling measured data [2] much in the same way described in [6]. Finally, it is also possible to generate non-physical lightsources for special effects.

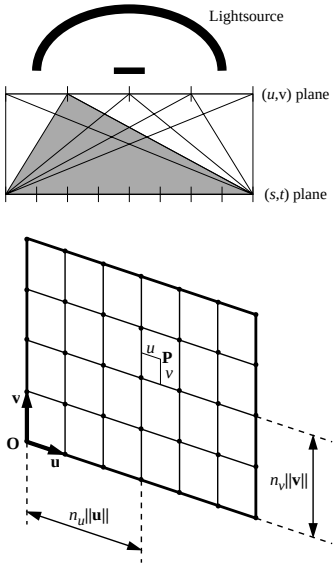

Fig. 1. Geometry of a single light slab within a Lumigraph.

One issue is the storage of Canned Lightsources. Due to the large dynamic range required to faithfully represent the lightfield of a lightsource, one of the commonly used 24 bit true color file formats is not sufficient. On the other hand, Lumigraphs can be quite large, and so a memory-efficient representation is mandatory. A good compromise is Larson's LogLuv extension to the TIFF file format [17], which uses 32 bit/sample to store high dynamic range images.

4 Reconstruction of Illumination from Lightfields

Given one or more Canned Lightsources, we can use them to illuminate a scene by reconstructing the radiance emitted by the lightsource at every point and in every direction. Like in [6] and [9], we use quadrilinear interpolation between the samples stored in the light slabs. For a ray-tracer, we could simply obtain a large number of radiance samples from the Lumigraph. However, the challenge is to obtain a good reconstruction with the smallest number of samples possible. It is important not to miss any narrow radiance peaks, because this would lead to artifacts in the reconstructed illumination.

In the following, we discuss three different approaches for reconstructing the illumination at every point in the scene. Section 4.1 describes a high-quality technique which can be used as a reference solution for the other methods. Section 4.2 deals with a generalization of this method for efficient ray-tracing. Finally, in Section 4.3, we describe a way to use the graphics library OpenGL for the reconstruction.

4.1 High Quality Reference Solutions

In order to describe the illumination in a scene caused by a Canned Lightsource, we view a light slab as a collection of $N_u \times N_v \times N_s \times N_t$ independent little area lightsources

("micro lights"), each corresponding to one of the light slab's 4D grid cells. Here, N_u, N_v, N_s, and N_t are the resolutions of the slab in each parametric direction. Each of the micro lights causes a radiance of

$$L^o_{n_u,n_v,n_s,n_t}(\mathbf{Q},\omega_o) = \int f(\mathbf{Q},\omega_i \rightarrow \omega_o) L^i_{n_u,n_v,n_s,n_t}(\mathbf{Q},\omega_i)\cos\theta d\omega_i \qquad (1)$$

to be reflected of any given surface point **Q**. The total reflected radiance caused by the Canned Lightsource is then the sum of the contributions of every micro light. $f(\mathbf{Q},\omega_i \rightarrow \omega_o)$ is the BRDF of the surface, θ is the angle between the surface normal and the direction ω_i, and $L^i_{n_u,n_v,n_s,n_t}(\mathbf{Q},\omega_i)$ is the radiance emitted from the micro light towards **Q** along direction ω_i.

Since $L^i_{n_u,n_v,n_s,n_t}(\mathbf{Q},\omega_i)$ is obtained by quadrilinear interpolation of 16 radiance values, Equation 1 can be rewritten in a more suitable form. Consider the geometry of a light slab as shown on the right side of Figure 1. Every point **P** on the (u,v) plane can be written as $\mathbf{P} = \mathbf{O} + (n_u+u)\vec{u} + (n_v+v)\vec{v}$, where the vectors $\vec{u}$ and $\vec{v}$ determine the dimensions of a 2D grid cell, and $u,v \in [0,1]$ describe the coordinates of **P** within the cell. In analogy we can describe a point $\mathbf{P}'$ on the (s,t) plane, where we assume that the vectors $\vec{s}$ and $\vec{t}$ are parallel to $\vec{u}$ and $\vec{v}$, respectively.

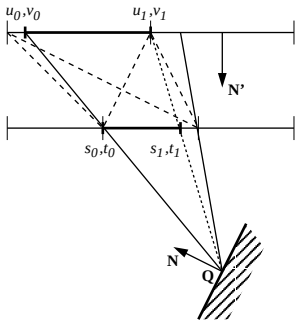

Fig. 2. Clipping a Lumigraph grid cell. $u_{\{0,1\}}$, $v_{\{0,1\}}$, $s_{\{0,1\}}$, and $t_{\{0,1\}}$ describe the portion of the cell that is visible from point **Q**.

For micro light (n_u,n_v,n_s,n_t) and point **Q**, we can now determine the region on the (u,v) plane that is *visible* from point **Q**. This is achieved by projecting the 2D cell (n_s,n_t) onto the (u,v) plane, using **Q** as the center of projection (see Figure 2). The resulting rectangle is clipped against the 2D cell (n_u,n_v).

This yields boundaries $u_{\{0,1\}}$, $v_{\{0,1\}}$, $s_{\{0,1\}}$, and $t_{\{0,1\}}$ relative to the 4D cell boundaries of the micro light. In this area, every ray $r(u,v,s,t)$ passing through **Q** is characterized by $u=(1-x)u_0+xu_1$, $v=(1-y)v_0+yv_1$, $s=(1-x)s_0+xs_1$, and $t=(1-y)t_0+yt_1$ for some $x,y \in [0,1]$. The values x and y can be used for the quadrilinear interpolation of the radiance emitted towards point **Q** by inserting the formulas for u, v, s, and t into the following equation:

$$\begin{aligned} L^i_{n_u,n_v,n_s,n_t}(x,y) =& (1-u)\cdot(1-v)\cdot(1-s)\cdot(1-t)\cdot L_{0000} + \\ & (1-u)\cdot(1-v)\cdot(1-s)\cdot t\cdot L_{0001} + \\ & + \cdots + u\cdot v\cdot s\cdot t\cdot L_{1111}. \end{aligned} \qquad (2)$$

$L_{0000}, L_{0001}, \ldots, L_{1111}$ are the 16 radiance values at the corners of the 4D grid cell. These are contained in the Lumigraph. With this result, we can now rewrite Equation 1 as follows:

$$\begin{aligned} & L^o_{n_u,n_v,n_s,n_t}(\mathbf{Q},\omega_o) \\ & \quad = \int_A f(\mathbf{Q},(\mathbf{P}-\mathbf{Q}) \rightarrow \omega_o) L^i_{n_u,n_v,n_s,n_t}(\mathbf{Q},\mathbf{P}\rightarrow\mathbf{Q}) \frac{\cos\theta\cos\theta'}{R^2} d\mathbf{P} \qquad (3) \\ & \quad = A\cdot\int_0^1\int_0^1 f(\mathbf{Q},(\mathbf{P}-\mathbf{Q}) \rightarrow \omega_o) L^i_{n_u,n_v,n_s,n_t}(x,y) \frac{\cos\theta\cos\theta'}{R^2} dxdy, \end{aligned}$$

where $A=(u_1-u_0)||\vec{u}|| \cdot (v_1-v_0)||\vec{v}||$ is the size of the visible region on the (u,v) plane, and $R=||\mathbf{P}-\mathbf{Q}||$ is the distance of the two points **P** and **Q**. Note that the vectors

$\vec{s}$ and $\vec{t}$ do not occur in this equation. That is, the exact position and parameterization of the (s,t) plane is only required for clipping. The use of an (s,t) plane at infinity is transparent in Equation 3.

While an analytical solution of the integral exists if there is no occlusion, it is too complicated for practical use. However, the integrand in Equation 3 is quite smooth, so that Monte Carlo integration performs well, and only a few samples are required to obtain good results.

In order to compute the complete illumination in **Q**, we have to sum up the contribution of all micro lights. Although there are $N_u \times N_v \times N_s \times N_t$ micro lights, only $O(\max(N_u, N_s) \cdot \max(N_v, N_t))$ have a non-zero visible area from any point **Q**. These visible areas are easily determined since 2D grids on the two Lumigraph planes are aligned. Thus, $u_{\{0,1\}}$ and $s_{\{0,1\}}$ only depend on the grid *column* (n_u, n_s), while $v_{\{0,1\}}$ and $t_{\{0,1\}}$ only depend on the grid *row* (n_v, n_t).

The result of this method is that we know how many samples are at least required for faithfully computing the illumination in a given point **Q**: at least one sample is necessary for each visible micro light. The left side of Figure 3 shows a simple scene illuminated with a Canned Lightsource at full resolution (64×64 on the (u,v) plane). The underlying geometry of the lightsource is described in Section 5.

Fig. 3. A simple test scene illuminated by a Canned Lightsource with full resolution of 64×64 (top) and half resolution in every direction (bottom). In the center you see a difference image between the two.

4.2 Efficient Ray-tracing

While the above method is capable of producing high-quality reference solutions, the number of visible grid cells is prohibitive for real world applications. Independent of the distance from the lightsource, we have to take at least one sample per visible grid cell, or risk missing important features such as thin radiance peaks. Ideally, the number of samples should depend on the solid angle covered by the lightsource.

To achieve this kind of adaptive sampling, we have developed a method based on a hierarchy of Lumigraphs at different resolutions, similar to the use of mipmaps. Let us first consider the case where the (s,t) plane is at infinity.

If we ignore occluding geometry, we can say that each point **Q** in front of the (u,v) plane sees all sample points there. The distances towards these samples will vary strongly for points **Q** close to the (u,v) plane. Thus, the number of visible micro lights is mostly dominated by the directional resolution, which is larger than the spatial resolution for our purposes (see Section 3). Therefore, $O(N_s \cdot N_t)$ micro lights are visible from **Q**. This number cannot be reduced without compromising quality, since the lightsource covers a large solid angle as seen from **Q**.

As the distance of **Q** from the (u,v) plane increases, the directions of the sample rays become almost parallel. In the end, all rays will pass through a single 2D cell

on the (s,t) plane. In this situation, there are exactly $N_u \cdot N_v$ visible micro lights. Although this number is smaller than the number of visible cells for close points $\mathbf{Q}$, we would like to further reduce it for points at very large distances. The number of samples should be reduced linearly with the solid angle covered by the lightsource.

This means that for far away points, a Lumigraph with a lower resolution on the (u,v) plane is sufficient. Similarly to mipmaping, we use a hierarchy of Lumigraphs, where the resolution in the u and v direction is reduced by a factor of two from one level to the next. Since we want to avoid storing the Lumigraph at multiple resolutions, we would like to do the downsampling on the fly during the rendering phase. We use simple averaging of the higher resolution samples. When switching from one level to the next, only the clipping has to be performed at a different resolution, and in Equation 2 averaged values are used for L_{0000}, L_{0001} and so on.

If the (s,t) plane is not located at infinity, a clear separation between spatial and directional resolution is not possible. Reducing the resolution of only one plane is therefore not adequate. Nonetheless, it makes sense to reduce the number of samples quadratically with the distance from the (u,v) plane, in order to maintain a certain number of samples per solid angle. This can be achieved by simultaneously reducing the resolution of both planes from level to level. Due to the lower directional resolution, the results will in general not be quite as good as for (s,t) planes at infinity.

The right side of Figure 3 shows the result of the same image as on the left side, but illuminated with a downsampled Canned Lightsource ($1/2$ the resolution in each direction). The performance improvement for this scene was roughly a factor of 7.4. Although a minor degradation of quality is noticeable in the difference image, the quality is still very good.

4.3 OpenGL Reconstruction

In addition to ray-tracing, it is also possible to use computer graphics hardware for reconstructing the illumination from a Canned Lightsource. As a hardware abstraction layer, we use the graphics library OpenGL, which is available on a variety of different platforms. Our method can be combined with one of the algorithms for generating shadows in OpenGL, such as shadow maps, or shadow volumes.

For hardware-based rendering, we assume that the BRDF $f(\mathbf{Q}, (\mathbf{P}-\mathbf{Q}) \rightarrow \omega_o)$ used in Equation 3 corresponds to the Phong model, which is the OpenGL lighting model. We numerically integrate Equation 3 by evaluating the integrand at the grid points on the (u,v) plane. For these points, the quadrilinear interpolation reduces to a bilinear interpolation on the (s,t) plane, which can be done in hardware.

[11] describes how projective textures can be used to simulate lightsources such as high-quality spotlights and slide projectors. Our method is an extension of this approach, which also accounts for the quadratic falloff and the cosine terms in Equation 3. To see how this works, we rewrite the integrand from this equation in the following form:

$$f(\mathbf{Q}, (\mathbf{P}-\mathbf{Q}) \rightarrow \omega_o) \cos\theta \cdot A \frac{\cos\theta'}{R^2} \cdot L^i_{n_u,n_v,n_s,n_t}(x,y) \tag{4}$$

The first factor of this formula is given by the Phong material and the surface normal N (see Figure 1). The second factor is the illumination from a spot light with intensity A, located at $\mathbf{P}$ and pointing towards N' with a cutoff angle of $90°$, a spot exponent of 1, and quadratic falloff. Finally, the third factor is the texture value looked up from the slice of the light slab corresponding to point $\mathbf{P}$ on the (u,v) plane.

In other words, the integrand of Equation 4 can be evaluated by combining projective textures as in [11] with the illumination from a spotlight using the standard

OpenGL lighting model. The results from texturing and lighting are combined using the "modulate" texture environment of OpenGL.

The integral from Equation 3 is approximated by the sum of the contributions from all grid points on the (u, v) plane. We compute this sum by adding multiple images, each rendered with a different spotlight position and projective texture. To this end we use either alpha blending or an accumulation buffer.

5 Results

For the images in Figure 3, we have used a Canned Lightsource that was created by ray-tracing a simple lamp geometry depicted in Figure 4. The lamp itself consists of a diffuse parabolic reflector and a grid of 3×3 long blocker polygons. The shadows cast by these blockers are quite visible in the nearfield of the lightsource, and can thus be used to evaluate the quality of the rendering. A small spherical luminary in the center of the paraboloid was used.

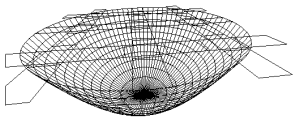

Fig. 4. Lamp geometry used for the images in this paper. A diffuse parabolic reflector with a small spherical luminary is covered with a grid of blockers.

In Figure 6 in the color section, you can see an office scene illuminated by variations of the same Canned Lightsource. The image was generated using ray-tracing. The Canned Lightsource had a resolution of $32 \times 32 \times 16 \times 16$, which occupied about 2MB of space in the LogLuv coding. On the right side the paraboloid is narrower, resulting in a more focussed emission of light. In both images the shadows created by the blockers are noticeable. The blurring of these shadows is a prominent feature of the nearfield of this lightsource.

Finally, we have used a hand-crafted Canned Lightsource that resembles a slide projector. In the top right image of Figure 6, we have pointed this lightsource at a certain angle onto a plane. This results in a focussed area in the center, where the "slide" is in focus, and in blurred outer regions, where the focus of the projector is either in front of or behind the plane. Also note the darkening towards the corners of the projection due to illumination from off-normal directions. Depending on the resolution of the (u, v) plane, the scene can be rendered interactively on current graphics systems. For example, we achieve around 10 frames per second on a Reality Engine 2 with a spatial resolution of 8×8, and roughly the same performance on an $O2$ with a resolution of 4×4. Figure 6 was rendered at a resolution of 8×8.

Fig. 5. Plane illuminated by a hand-crafted Canned Lightsource resembling a slide projector.

6 Conclusions

In this paper we have introduced the notion of a Canned Lightsource, which is a precomputed, discretely sampled version of the lightfield emitted by a complex lightsource or lamp. This information is stored away in a Lumigraph data structure. Later, the precomputed information can be used to illuminate a scene while abstracting from the lamp geometry. Thus, Canned Lightsources act as a blackbox for complex lightsources.

Canned Lightsources can be used to create databases of lightsources based on simulation or measurement. Ideally, this information could be directly distributed by the

lamp manufacturers in much the same way far field information is provided today.

The precomputed information can be used to speed up the lighting process in standard rendering algorithms such as ray-tracing and hardware-based rendering. Thus, Canned Lightsources are a valuable data structure for efficient image synthesis. Particularly interesting is the possibility to precompute or measure the global illumination of some parts of a scene (the lightsource), and reuse it at a different location or in a completely different scene.

7 Acknowledgments

We would like to thank Marc Stamminger for several interesting discussions regarding this topic. This work was in part supported by the German Research Council through the collaborative research center #603 (Model-based Analysis and Synthesis of Complex Scenes and Sensor Data).

References

1. B. Arnaldi, X. Pueyo, and J. Vilaplana. On the division of environments by virtual walls for radiosity computation. In *Photorealistic Rendering in Computer Graphics*, pages 198–205, May 1991.
2. I. Ashdown. Near-Field Photometry: A New Approach. *Journal of the Illuminating Engineering Society*, 22(1):163–180, Winter 1993.
3. R. Bastos. Efficient radiosity rendering using textures and bicubic reconstruction. In *Symposium on Interactive 3D Graphics*. ACM Siggraph, 1997.
4. G. Drettakis and F. Sillion. Intaractive update of global illumination using a line-space hierarchy. *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 57–64, August 1997.
5. A. Fournier. From local to global and back. In *Rendering Techniques '95*, pages 127–136, June 1995.
6. S. J. Gortler, R. Grzeszczuk, R. Szelinski, and M. F. Cohen. The Lumigraph. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 43–54, August 1996.
7. G. Greger, P. Shirley, P. Hubbard, and D. P. Greenberg. The irradiance volume. *IEEE Computer Graphics and Applications*, 18(2):32–43, March 1998.
8. A. Keller. Instant radiosity. *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 49–56, August 1997.
9. M. Levoy and P. Hanrahan. Light field rendering. *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 31–45, August 1996.
10. R. Lewis and A. Fournier. Light-driven global illumination with a wavelet representation of light transport. In *Rendering Techniques '96*, pages 11–20, June 1996.
11. M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli. Fast shadow and lighting effects using texture mapping. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):249–252, July 1992.
12. F. Sillion and C. Puech. *Radiosity & Global Illumination*. Morgan Kaufmann, 1994.
13. P. Slusallek, M. Stamminger, W. Heidrich, J.-C. Popp, and H.-P. Seidel. Composite lighting simulations with lighting networks. *IEEE Computer Graphics and Applications*, 18(2):22–31, March 1998.
14. W. Stürzlinger and R. Bastos. Interactive rendering of globally illuminated glossy scenes. In *Rendering Techniques '97*, pages 93–102, 1997.
15. B. Walter, G. Alppay, E. Lafortune, S. Fernandez, and D. P. Greenberg. Fitting virtual lights for non-diffuse walkthroughs. *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 45–48, August 1997.
16. G. Ward. The RADIANCE lighting simulation and rendering system. *Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 459–472, July 1994.
17. G. Ward Larson. LogLuv encoding for TIFF images. Technical report, Silicon Graphics, 1997. www.sgi.com/Technology/pixformat/tiffluv.html.

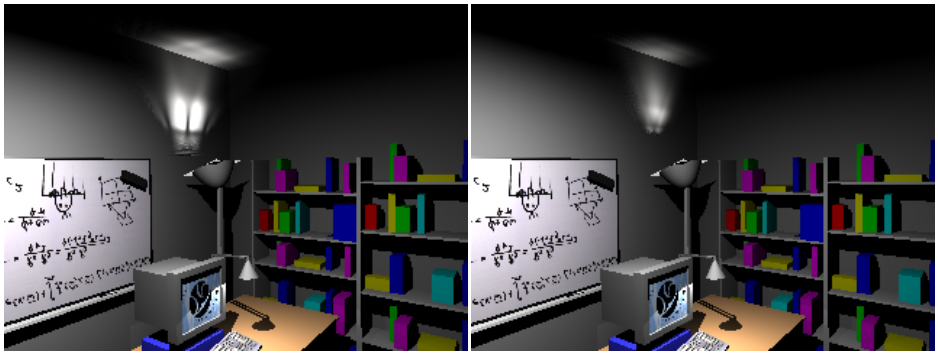

Fig. 6. Office scene rendered with two different Canned Lightsources based on different lamp sizes. The blurring due to nearfield effects is very prominent in both images.