Interactive Visualization of Complex Real-World Light Sources

Xavier Granier, M. Goesele, W. Heidrich, H.-P. Seidel

Interactive Visualization of Complex Real-World Light Sources

Xavier Granier* Michael Goesele+ Wolfgang Heidrich* Hans-Peter Seidel+

*The University of British Columbia
Vancouver, Canada
{xgranier,heidrich}@cs.ubc.ca

+MPI Informatik
Saarbrücken, Germany
{goesele,hpseidel}@mpi-sb.mpg.de

Abstract

Interactive visualization of complex, real-world light sources has so far not been feasible. In this paper, we present an hardware accelerated direct lighting algorithm based on a recent high quality light source acquisition technique. By introducing an approximate reconstruction of the exact model, a multi-pass rendering approach, and a compact data representation, we are able to achieve interactive frame rates. The method is part of the processing pipeline from light source acquisition to high quality lighting of a virtual world.

Keywords: Local Illumination, Light Source Modeling, Hardware Rendering, Image-based Rendering, Physically-based Modeling and Acquisition

1 Introduction

A major contributing factor to the realism of computer generated images is the complexity of lighting effects. With some recent image-based techniques, it is possible to acquire real light sources and to use them for global illumination algorithms [2, 18, 6]. In recent work [6], we introduce a new optical filtering approach, which projects real light sources into a predefined basis. With this technique, a high-quality representation of a real light source can be acquired with a simple camera system.

Unfortunately, the visualization of direct lighting from such data is still time-consuming for an exact reconstruction (see Section 2). Interactivity, on the other hand, would allow a convenient processing pipeline by presenting a preview even during an acquisition. A fast visualization of the light source including shadow computations not only facilitates the design of a scene for high-quality rendering but is also essential for the acquisition process, giving a rapid feedback of the quality of the currently acquired light source.

1.1 Previous Work

We can divide the light representations used in the literature into two categories. The first one is an approximation of a light viewed from a far distance compared to its size. In this case, knowledge of the directional distribution (2D), for example in the form of a goniometric diagram [20], is sufficient. Unfortunately, this far field approximation is only valid for a distance greater than about 5 times the largest dimension of the luminaire [2] and does not allow for the simulation of near field effects.

On the other hand, light field [7, 12] based approaches [1, 2] can capture both the far field and the near field of a light source (4D). We recently introduced a new approach [6] based on prefiltering with an optical system that allows an accurate acquisition. Although this approach allows high-quality global illumination rendering, the storage complexity of the data makes an interactive visualization difficult.

On the rendering side, a lot of interactive techniques have been developed to introduce more complex light sources. The currently existing solutions are mainly based on projective textures (for slide projector like light sources [17]), light maps (storing precomputed irradiance [8, 14]) and environment maps (for glossy direct reflection of the environment [3]). Combined with a depth test [16, 19], these techniques show that the creation of realistic direct lighting is possible using graphics hardware acceleration.

Heidrich et al. [11] have described an efficient method to interactively render a representation similar to ours, which they called a *canned light source*. However, this approach cannot be directly applied to visualize the direct lighting from our new representation, which requires specialized reconstruction filters. Some approximations to the exact reconstruction [6] and a new rendering path have to be developed to achieve this goal.

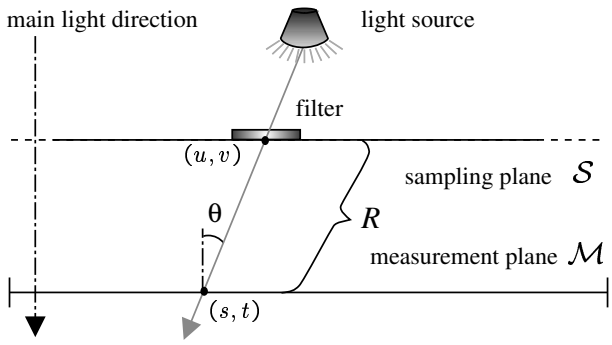

Figure 1. Configuration of the acquisition setup. Light is emitted from the light source, prefiltered on the sampling plane, and recorded on the measurement plane.

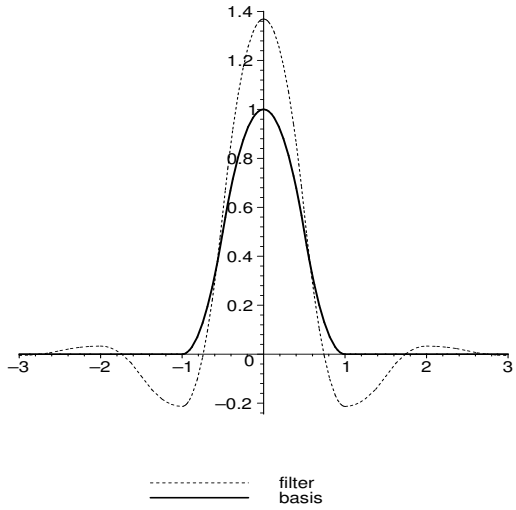

Figure 2. A quadratic reconstruction basis and one dual used as a filter.

1.2 Overview and Contributions

The main contributions of this paper are the detailed presentation and justification of the shift-invariant approximation to our original representation, a hardware accelerated implementation of direct lighting including shadows, and a compact representation of a 3 signed-float vector.

The remainder of the paper is structured as follows: first, we briefly re-introduce the light source representation presented in [6]. In Section 3, we then describe in more detail the shift-invariant approximation that allows for interactive rendering and our solutions for an hardware implementation in Section 4. Finally, we present our results and conclude with some ideas for future work.

2 Light Source Representation

We assume that a light source can be well represented by the projection of its exitant light field into a 4D basis $\{\Psi_{ijkl}(u,v,s,t)\}_{ijkl\in\mathbf{Z}}$

$$L(u,v,s,t) \approx \tilde{L}(u,v,s,t) = \sum_{ijkl} \Psi_{ijkl}(u,v,s,t)\cdot L_{ijkl}, \quad (1)$$

where (u,v) (resp. (s,t)) are the 2D coordinate on the sampling (resp. measurement) plane as depicted in Figure 1.

If we use a filter Φ'_{mn} on the sampling plane, the measured irradiance on the measurement plane $\mathcal{M}$ is:

$$\begin{aligned} E_{mn}(s,t) &\approx \int_{\mathcal{S}} \frac{\cos^2(\theta)}{R^2}\cdot\Phi'_{mn}(u,v)\cdot\tilde{L}(u,v,s,t)\,du\,dv \\ &= \sum_{ijkl}\int_{\mathcal{S}} \frac{\cos^2(\theta)}{R^2}\cdot\Phi'_{mn}(u,v) \\ &\qquad \cdot\Psi_{ijkl}(u,v,s,t)\cdot L_{ijkl}\,du\,dv \end{aligned} \quad (2)$$

We can then define the basis Ψ_{ijkl} as the product of two 2D bases – Φ_{ij} on the sampling plane and Φ_{kl} on the measurement plane – with a geometric term, where $\{\Phi'_{mn}\}_{mn\in\mathbf{Z}}$ is the dual basis of $\{\Phi_{ij}\}_{ij\in\mathbf{Z}}$:

$$\Psi_{ijkl}(u,v,s,t) := \frac{R^2(u,v,s,t)}{\cos^2(\theta(u,v,s,t))}\cdot\Phi_{ij}(u,v)\cdot\Phi_{kl}(s,t) \quad (3)$$

With this basis, the measured irradiance is now expressed as

$$E_{mn}(s,t) = \sum_{kl}\Phi_{kl}(s,t)\cdot L_{mnkl}, \quad (4)$$

and Equation 1 can be rewritten as

$$\begin{aligned} \tilde{L}(u,v,s,t) &= \sum_{mnkl}\Psi_{mnkl}(u,v,s,t)\cdot L_{mnkl} \\ &= \sum_{mnkl}\frac{R^2}{\cos^2(\theta)}\cdot\Phi_{mn}(u,v)\cdot\Phi_{kl}(s,t)\cdot L_{mnkl} \\ &= \sum_{mn}\frac{R^2}{\cos^2(\theta)}\cdot\Phi_{mn}(u,v)\cdot E_{mn}(s,t) \end{aligned} \quad (5)$$

This result shows that we can *exactly* reconstruct the projection $\tilde{L}$ of the light field L into the function space spanned by our basis. In our measurements, as described in [6], we use a piecewise quadratic basis with compact support on the sampling plane (cf Figure 2):

$$\Phi_{i,\text{bellq}} = \begin{cases} 1-2x^2 & |x| \le \frac{1}{2} \\ 2(|x|-1)^2 & \frac{1}{2} < |x| \le 1 \\ 0 & \text{else} \end{cases}. \quad (6)$$

The basis funtion on the sampling plane is a piecewise constant function.

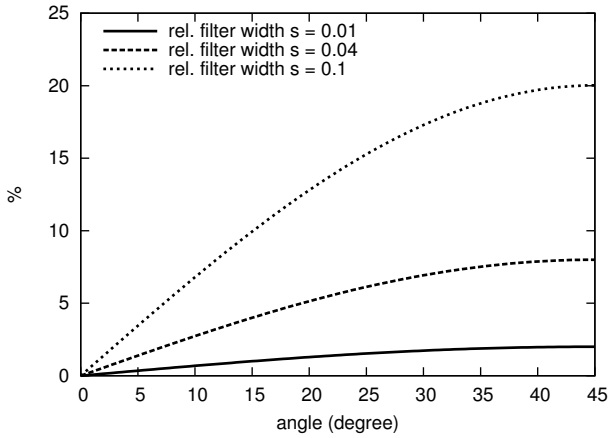

Figure 3. Error analysis for the geometric term. The plots show the relative error $\mathbf{E_r}(\theta_0)$ for angles θ_0 and varying relative filter width s.

3 Introducing Approximations

In order to provide a fast evaluation for the direct lighting, we introduce two approximations, one for the reconstruction of the continuous light field, and one for the estimation of the illumination. These approximations allow for a texture-based rendering approach, that is described in more details in Section 4.

3.1 Shift-Invariant Representation

The exact reconstruction presented before contains a shift-variant geometric term (i.e. $R^2/\cos^2$), which makes this approach not well suited for interactive rendering. We therefore introduce a shift-invariant reconstruction based on the same measured irradiance as defined in the previous sections, but with a different relationship between Ψ_{ijkl}, the measurement and the reconstruction bases. We define $\Psi_{ijkl}(u,v,s,t) := \Phi_{ij}(u,v) \cdot \Phi_{kl}(s,t)$. From this, we get

$$E_{mn}(s,t) = \sum_{ijkl} \int_{\mathcal{S}} \frac{\cos^2(\theta)}{R^2} \cdot \Phi'_{mn}(u,v) \cdot \Phi_{ij}(u,v) \cdot \Phi_{kl}(s,t) \cdot L_{ijkl}\, du\, dv \quad (7)$$

To provide a real shift-invariant approximation, we use the following assumption: if the distance d between the (u,v)-plane and the (s,t)-plane is large compared to the support of $\Phi_{ij}(u,v)$, and if θ is small, then the geometric term $\cos^2\theta/R^2$ is well approximated by one constant for each point on the (s,t)-plane:

$$g(u,v,s,t) := \frac{\cos^2(\theta(u,v,s,t))}{R^2(u,v,s,t)} \approx \frac{\cos^2(\theta_{kl}(s,t))}{R^2_{kl}(s,t)}. \quad (8)$$

This yields an approximation of the measured irradiance:

$$E_{mn}(s,t) \approx \sum_{kl} \frac{\cos^2(\theta_{kl}(s,t))}{R^2_{kl}(s,t)} \cdot \Phi_{kl}(s,t) \cdot L_{mnkl}. \quad (9)$$

Since both the geometric term and $\Phi_{kl}(s,t)$ are known, it is in principle possible to compute the approximate incoming radiance

$$E'_{mn}(s,t) \approx \sum_{kl} \Phi_{kl}(s,t) \cdot L_{mnkl} \quad (10)$$

by de-convolution. In practice, this is only feasible for basis functions $\Phi_{kl}(s,t)$ with a small support. This is not a major problem, however, since the practical measurement setups presented in the previous section have a very high resolution on the (s,t)-plane, so that a bilinear or even a box filter can be used. Like in the previous section, we apply the definition of Ψ_{ijkl} to determine the appropriate reconstruction filter:

$$\begin{aligned} \tilde{L}(u,v,s,t) &= \sum_{mnkl} \Psi_{mnkl}(u,v,s,t) \cdot L_{mnkl} \\ &= \sum_{mnkl} \Phi_{mn}(u,v) \cdot \Phi_{kl}(s,t) \cdot L_{mnkl} \\ &\approx \sum_{mn} \Phi_{mn}(u,v) \cdot E'_{mn}(s,t). \end{aligned} \quad (11)$$

The quality of this approximation depends on the error introduced by assuming the geometric term constant over the support of the basis function $\Phi_{ij}(u,v)$ in Equation 8. To evaluate the validity of this approximation, we define the following relative error

$$E_r(\theta_0) := \frac{\max_{\theta \in \mathcal{F}} cos^4(\theta) - \min_{\theta \in \mathcal{F}} cos^4(\theta)}{cos^4(\theta_0)}$$

where $\mathcal{F}$ is the support of the basis Φ_{ij}, θ_0 is the angle at the center of this support, and $cos^4(\theta)$ corresponds to the geometric term g of Equation 8 for a distance of 1 between the planes. Its evaluation shows (cf. Figure 3 and [10]) that the error is below 8% if the ratio s between filter width and the distance of the two planes $\mathcal{S}$ and $\mathcal{M}$ equals 0.04.

3.2 Estimation of Direct Illumination

To compute the direct illumination from this shift-invariant approximation, we have to evaluate the following equation for a visible point $\mathbf{x}$ and viewing direction $\vec{e}$ (see Figure 4 for the notation):

$$\begin{aligned} L(\vec{e},\mathbf{x}) &= \sum_{mn} \int_{\mathcal{S}} \rho(\mathbf{x},\vec{e},u,v) V(\mathbf{x},u,v) \frac{\cos(\theta)\cos(\theta')}{\delta^2(\mathbf{x},u,v)} \cdot \\ &\qquad \tilde{L}(u,v,s(\mathbf{x},u),t(\mathbf{x},v))dudv \\ &= \sum_{mn} \int_{\mathcal{S}} \rho(\mathbf{x},\vec{e},u,v) V(\mathbf{x},u,v) \frac{\cos(\theta)\cos(\theta')}{\delta^2(\mathbf{x},u,v)} \cdot \\ &\qquad E'_{mn}(s(\mathbf{x},u),t(\mathbf{x},v))\Phi_{mn}(u,v)dudv, \end{aligned} \quad (12)$$

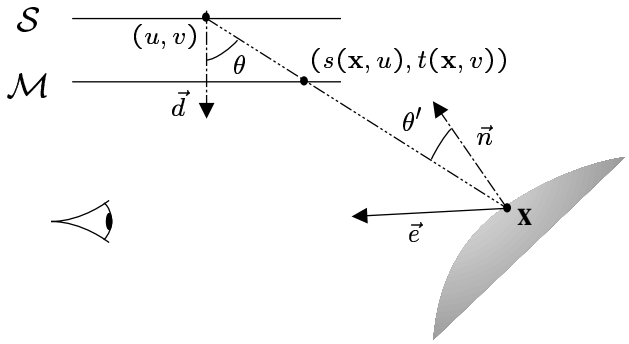

Figure 4. Configuration of a direct reflection.

where $\rho(\mathbf{x},\vec{e},u,v)$ describes the object's reflection properties. $V(\mathbf{x},u,v)$ and $\delta(\mathbf{x},u,v)$ encode the visibility and the distance between the light sample (u,v) and the position $\mathbf{x}$.

To simplify this expression, we make the common assumption that the reflection properties, visibility, and the geometric term are constant on the support of Φ_{mn} and are estimated at its center:

$$\rho_{mn}(\mathbf{x},\vec{e}) \approx \rho(\mathbf{x},\vec{e},u,v),\ V_{mn}(\mathbf{x}) \approx V(\mathbf{x},u,v),$$
$$f_{mn}(\mathbf{x}) \approx (\cos(\theta)\cos(\theta^{'}))/\delta^2(\mathbf{x},u,v).$$

Given these assumptions, Equation 12 can be approximated by:

$$L(\vec{e},\mathbf{x}) \approx \sum_{mn} V_{mn}(\mathbf{x})\rho_{mn}(\mathbf{x},\vec{e})f_{mn}(\mathbf{x})\cdot \int_{\mathcal{S}} E^{'}_{mn}(s(\mathbf{x},u),t(\mathbf{x},v))\Phi_{mn}(u,v)dudv \quad (13)$$

For a position $\mathbf{x}$, we then compute $\overline{E^{'}}_{mn}(\mathbf{x})$, the average value of $E^{'}_{mn}(s(\mathbf{x},u),t(\mathbf{x},v))$ on the support of Φ_{mn} in order to make our final approximation of Equation 12:

$$L(\vec{e},\mathbf{x}) \approx A\sum_{mn} V_{mn}(\mathbf{x})\rho_{mn}(\mathbf{x},\vec{e})f_{mn}(\mathbf{x})\overline{E^{'}}_{mn}(\mathbf{x}), \quad (14)$$

where $A = \int_{\mathcal{S}} \Phi_{mn}(u,v)dudv$.

Each term of this sum corresponds now to the reflection of a textured spot-light [17], located at the center of a filter support, and pointing toward $\mathbf{x}$ (similar to the canned light source approach [11]).

4 Hardware Implementation

A straightforward implementation of the approximation described in the previous section, requires two hardware rendering passes for every term in the sum, one for the depth map creation to determine V_{mn} and one for the lighting computation, leading to a total of $2N$ passes (where N is the number of measurements). In this section we show how the rendering can be accelerated by combining some of the passes and by speeding up each of the individual passes.

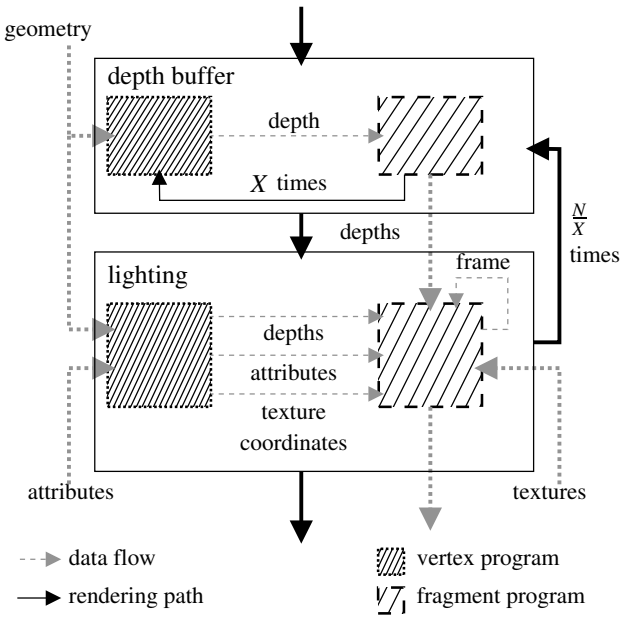

Figure 5. Hardware implementation diagram: data flow and rendering path

4.1 Combining Multiple Rendering Passes

Using the features of current graphics hardware, the number of rendering passes can be reduced to speed up the computation. While the total number of passes for the depth map computation is fixed, we can combine 3 lighting evaluations into a single pass. The rendering is then organized in $N/3$ iterations of the following steps (cf Figure 5):

1. creation of depth maps for 3 positions on the sampling plane, results are stored in a RGB texture

2. illumination computation for these 3 positions, and addition of the current result to the previous solution

With this approach, the total number of passes is now reduced to $4N/3$.

Generally speaking, if we manage to evaluate the lighting from X measurements at a time, we need $(X+1)N/X$ passes. Currently, it is possible to store up to 4 depth maps in a RGBA texture. But due to the limited number of attributes available for a fragment in graphics hardware, and given the fact that we still need to provide either diffuse color and/or texture coordinates, we are limited to $X = 3$. Thus, considering that the gain for combining a new lighting (i.e. $X = 4$) is 6.25%, the limit of 3 is not so severe.

4.2 Floating-Point Representation

All operations described in the previous section can (and should) be performed using the floating-point buffers available on recent graphics hardware. But as there are no blending operations available for floating-point precision buffers [15] on current hardware, we need to save each intermediate frame into a texture in order to make it available for the next rendering pass.

The large amount of on-board memory transfer needed to exchange both the depth buffer and the frame buffer between rendering passes, is currently one of the main bottlenecks of our approach, even with current available bandwidth. To reduce this transfer cost, we save both buffers as classical 8 bit per component textures. The frame-buffer is converted to an extended version of the RGBE [21] format, that we call *signed RGBE* (or sRGBE), and transfered as a RGBA texture. The depth buffer values are hereby scaled from $[z_{near}, z_{far}]$ to $[0,1]$ and transfered as a RGB texture. This linear scaling insure a uniform discretization of the depth range, reducing the impact of a lower resolution on the depth test (as described in [9]).

The sRGBE representation allows to encode positive as well as negative values and is adapted to the color representation in the frame buffer, where each color component is clamped to $[0,1]$ as an 8 bit fixed point value. A floating-point RGB value (R_f, G_f, B_f) can be converted to an (R,G,B,E) sRGBE value by the following operations:

1. Computing the exponent e:

$$e = \lfloor \log_2(max(|R_f|, |G_f|, |B_f|)) \rfloor + 2$$

   where $\lfloor . \rfloor$ is the floor operation. (Note that the classical RGBE exponent is $e-1$.)

2. Scaling the RGB value to $[0,1]$:

$$[R,G,B] = [R_f, G_f, B_f] \cdot 2^{-e} + 0.5$$

3. Scaling the exponent to $[0,1]$:

$$E = (e + 126)/255$$

The decompression is done by:

$$[R_f, G_f, B_f] = ([R,G,B] - 0.5) \cdot 2^{255 \cdot E - 126}$$

This representation saves 67% of on-board memory and requires only about 15 fragment program instructions. The Cg [13] code for compressing and decompressing in sRGBE format is available in Appendix A.

	Size	Crypt (500 polygons)	Cloister (8000 polygons)
one iteration	—	40 (35) ms	85 (70) ms
Mag-Lite 1	5×5	2.7 (3.2) fps	1.3 (1.6) fps
Mag-Lite 2	7×7	1.5 (1.7) fps	0.7 (0.85) fps
bike light	9×7	1.2 (1.4) fps	0.55 (0.7) fps

Table 1. Rendering speeds for a single pass and the complete light source(cf Figure 2): models. Numbers in brackets correspond to renderings without shadow computation.

5 Results

We implemented this approach on a Linux workstation with an Intel Xeon 1.7 GHz processor, 512 MB memory and an NVIDIA GeForce FX 5800 Ultra, using the `NV_vertex_program2`, `NV_fragment_program`, `NV_texture_rectangle` and `NV_float_buffer` OpenGL extensions [15]. Using several measured light datasets and the two test scenes depicted in Figure 7, we evaluated the influence of the light source model and scene complexity on the frame-rate.

The light source datasets were acquired with the method introduced in [6] which corresponds directly to the schematic drawing in Figure 1. The light source is projected onto a screen through the filter kernel (implemented as a printed slide) and the projected pattern is recorded with a digital camera. A complete dataset is captured by moving the light source to all positions on the sampling grid (determined by the filter size). All measurements use as filter kernel the dual of a piecewise biquadratic basis (cf Section 2) with a filter spacing of 5 mm or 7.5 mm corresponding to a dual filter of 20 mm (resp. 30 mm) width. The acquired images are down-sampled to a size of 300×300 pixels.

5.1 Analysis

The real light sources and some of the rendered images are depicted in Figures 8, 6, 9, and 10. The global shape of the generated lighting patterns as well as the near field effects are faithfully reproduced. The changes in the projected pattern in Figures 6 and 9 – apart from simple scaling operations – and the presence of soft shadows (cf Figure 6-right) are due to near field effects. The blocky appearance in this image is due to the fact that the current floating point textures do not support linear interpolation combined with the traditional problem of limited depth map resolution for shadow generation.

The current implementation allows to render all of our light source models interactively with a frame rate from

3.2 fps down to 0.55 fps at a window size of 400×400 pixels. This corresponds to a range of 40 to 85 ms for each iteration (one depth and lighting computation), depending on the geometric complexity of the scene. A detailed list of rendering times is given in Table 1. There we also show that disabling the shadow computation leads to a speedup of only about 15% caused by the fact that the corresponding rendering passes are relatively cheap (only geometry is rendered, with no attributes on the vertices, and the corresponding vertex/fragment programs are minimal).

There are two main user-controllable factors which influence the frame rate. The first one is, of course, the size of the light source datasets. As the number of passes is proportional to the number of measurements, rendering a larger dataset takes also more time. The upcoming extensions `GLX_ARB_render_texture` can improve the full rendering with a lower latency for transferring the buffers to textures.

The geometric complexity directly influences the rendering time of a single pass, as shown by the decrease of the frame rate between the two scenes. The twenty fold increase in complexity leads to 2 times lower frame rate corresponding to fact that the rendering time is changing from 40 ms to 85 ms. For very large scenes, this factor will be the major bottleneck of our approach.

This shows also that in the current implementation, most of the time is spent in the fragment program for the lighting computation. This time is highly dependent on the instruction order and on the parameter access. The rendering time can definitively be improved by further optimizations.

6 Conclusion and Future Work

In this paper we presented a new approach for an interactive visualization of direct lighting from complex light sources. The acquisition technique includes an optical prefiltering that allows an accurate projection into a predefined function basis. The shift-invariant approximation presented here is suitable for hardware accelerated rendering techniques. By additionally combining multiple rendering passes into a single pass and with an compact data representation, we reach an interactive frame rate of up to 3.2 frames per second depending on the complexity of the light model and of the illuminated scene.

In the future, we plan to investigate in different directions in order to increase both the quality of the reconstruction and the frame rate. Introducing techniques similar to mip-mapping (like summed-area tables [5]) for the light source models would allow us to improve the rendering quality, as unfortunately, mip-mapping of floating point textures is currently not supported by graphics hardware. The quality of the shadows can be improved by adding more advanced shadow mapping algorithms [19, 4]. We expect furthermore that better support of floating point buffers in the next generation of graphics cards combined with additional research on more compact representations for floating point data, will improve both the frame rates and the quality of the results.

References

[1] I. Ashdown. Near-Field Photometry: A New Approach. *Journal of the Illuminating Engineering Society*, 22(1):163–180, Winter 1993.

[2] I. Ashdown. Near-Field Photometry: Measuring and Modeling Complex 3-D Light Sources. In *ACM SIGGRAPH '95 Course Notes - Realistic Input for Realistic Images*, pages 1–15. ACM, 1995.

[3] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, 1976.

[4] S. Brabec, T. Annen, and H.-P. Seidel. Practical shadow mapping. *Journal of Graphics Tools*, 2003. to be published.

[5] F. C. Crow. Summed-area tables for texture mapping. In *Proceedings of the SIGGRAPH 84 annual conference*, pages 207–212. ACM Press, 1984.

[6] M. Goesele, X. Granier, W. Heidrich, and H.-P. Seidel. Accurate light source acquisition and rendering. In *Proceedings of the SIGGRAPH 2003 annual conference*. ACM Press, July 2003. to be published.

[7] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Proceedings of the SIGGRAPH 96 annual conference*, pages 43–54. ACM Press, 1996.

[8] P. S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *Proceedings of the SIGGRAPH 90 annual conference*, pages 145–154. ACM Press, 1990.

[9] W. Heidrich. *High-quality Shading and Lighting for Hardware-accelerated Rendering*. PhD thesis, University of Erlangen, Computer Graphics Group, Erlangen - Germany, 1999. http://www.cs.ubc.ca/~heidrich/Papers/phd.pdf.

[10] W. Heidrich and M. Goesele. Image-based measurement of light sources with correct filtering. Technical Report TR-2001-08, Department of Computer Science, The University of British Columbia, 2001. http://www.cs.ubc.ca/~heidrich/Papers/TR-2001-08.pdf.

[11] W. Heidrich, J. Kautz, P. Slusallek, and H.-P. Seidel. Canned lightsources. In *Rendering Techniques '98*, pages 293–300. Eurographics, June 1998.

[12] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of the SIGGRAPH 96 annual conference*, pages 31–42. ACM Press, 1996.

[13] W. R. Mark, S. Glanville, and K. Akeley. Cg: A system for programming graphics hardware in a c-like language. In *Proceedings of the SIGGRAPH 2003 annual conference*. ACM Press, July 2003. to be published.

[14] K. Myszkowski and T. Kunii. Texture mapping as an alternative for meshing during walkthrough animation. In G. Sakas, P. Shirley, and S. Mueller, editors, *Photorealistic Rendering Techniques*, pages 375–388. Springer, 1994.

[15] NVIDIA OpenGL extensions specification, Jan. 2003. http://developer.nvidia.com.

[16] W. T. Reeves, D. H. Salesin, and R. L. Cook. Rendering antialiased shadows with depth maps. In *Proceedings of the SIGGRAPH 87 annual conference*, pages 283–291. ACM Press, 1987.

[17] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli. Fast shadows and lighting effects using texture mapping. In *Proceedings of the SIGGRAPH 92 annual conference*, pages 249–252. ACM Press, 1992.

[18] M. W. Siegel and R. D. Stock. A General Near-Zone Light Source Model and its Application to Computer Automated Reflector Design. *SPIE Optical Engineering*, 35(9):2661–2679, September 1996.

[19] M. Stamminger and G. Drettakis. Perspective shadow maps. In *Proceedings of the SIGGRAPH 2002 annual conference*, pages 557–562. ACM Press, 2002.

[20] C. P. Verbeck and D. P. Greenberg. A comprehensive light source description for computer graphics. *IEEE Computer Graphics & Applications*, 4(7):66–75, July 1984.

[21] G. Ward. Real pixels. In J. Arvo, editor, *Graphics Gems II*, pages 80–83. Morgan Kauffman Publishers Inc., San Francisco, CA, USA, 1991.

A Cg code for sRGBE representation

The compression procedure, from high-dynamic range RGB color values to sRGBE coefficients between $[0, 1]$, is expressed as follow:

```
float4 RGB2sRGBE(float3 rgb)
{
   float e = max(abs(rgb.r),
              abs(rgb.g));
   e = max(e,abs(rgb.b));
   e = floor(log2(e))+2;
   return float4((rgb*exp2(-e))+0.5,
               (e+126)/255);
}
```

The decompression procedure is expressed as follow:

```
float3 sRGBE2RGB(float4 srgbe)
{
   return (srgbe.rgb-0.5)*
       exp2(srgbe.a*255-126);
}
```

Figure 6. Rendering of the 5×5 Mag-Lite data set in the low polygon count environment (Crypt). Left: original position - Center: closer position - Right: zoom on a detail with soft shadow. Rendering speed: 2.7 to 3.2 frames per second.

Figure 7. Test scenes. Left: "Crypt" (500 polygons). Right: "Cloister" (8000 polygons).

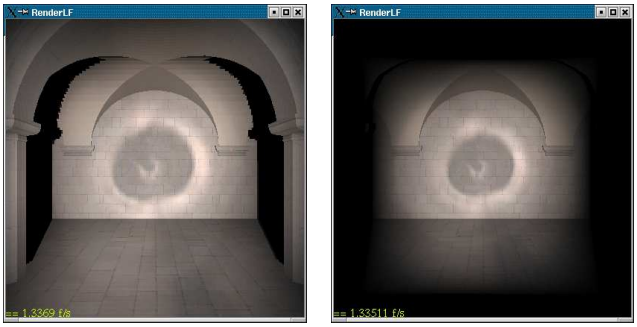

Figure 9. Rendering of the 5×5 Mag-Lite data set the complex environment. Rendering speed: 1.3 frames per second.

Figure 8. The Mag-Lite and a rendering of the 7×7 dataset in the "Crypt" environment at 1.5 frames per second.

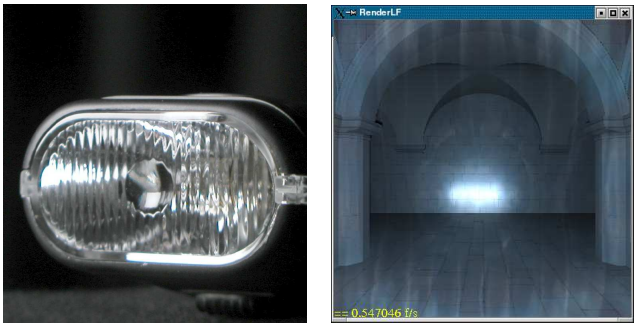

Figure 10. The bike light and the bike light dataset (9×7 measurements) rendered at 0.55 frames per second in the "Cloister".