Approximating Real-World Luminaires with OpenGL Lights

Georg Zotti Attila Neumann Werner Purgathofer
Institute of Computer Graphics and Algorithms
Vienna University of Technology, Austria
{gzotti,aneumann,wp}@cg.tuwien.ac.at

ABSTRACT

Dynamic illumination in real-time applications using OpenGL is still usually done with the classical light forms of point lights, directional lights and spot lights. For applications simulating real-world scenes, e.g. architectural planning, finding parameter sets for these simple lights to match real-world luminaires is required for realistic work. This paper describes a simple approach to process a luminaire data file in IESNA IES-LM63-95 format to create an approximation using at most 2 OpenGL lights to represent one luminaire.

Keywords
Real-world luminaires, OpenGL, interactive illumination planning

1. INTRODUCTION

Many rendering systems still use very simple light sources for illumination, typically point, directional and spot lights. In global illumination renderering systems, idealized area light sources are also commonly found. But, apart from the Sun, often modelled as directional light for terrestrial scenes, and special spotlights mostly used in theatres, most real-world luminaires as built by the illumination industry have a more complex radiation distribution, and in some global-illumination rendering systems photometric data files provided by the illumination industry can be used.

Still, for OpenGL-based real-time rendering, direct illumination is mostly performed with these simple lights (point, directional, spot), which are fast to evaluate and are built into the OpenGL standard.

In this paper, we want to demonstrate how to process data provided by the illumination industry to create a combination of at most two OpenGL lights which achieves an acceptable approximation of the luminaire for use in a real-time rendering context. The result however depends on the easiness with which a luminaire data set can be approximated with these simple lights.

WSCG 2005 SHORT papers proceedings, ISBN 80-903100-9-5
WSCG'2005, January 31-February 4, 2005
Plzen, Czech Republic.

2. DEFINITIONS AND RELATED WORK

For simulation of real-world light sources in computer graphics, their radiance distribution has to be known. Most real-world light sources are area light sources, so accurate simulation should take also the shape of the luminaire and variations of the light distribution along its surface into account. A method by GOESELE et al. [GGHS03] for acquisition of *near-field photometry* data of light sources and rendering of them with high accuracy in a global-illumination renderer yields good results, but at high computational expense.

However, in industrial practice still only *far-field photometry* data are available almost exclusively, and a common rule of thumb allows its application in cases where the luminaire's distance is at least five times greater than its greatest extension [Ash01]. Here the luminaire is modeled as point light source, and its radiation distribution is given as a set of candela measurements in a regular grid of angular values, either printable as polar 2D *goniometric diagram* along fixed planes or displayable as *photometric solid* [Ash99], where distance from the center represents the luminaire's brightness into a certain direction.

A good introduction into the complex field of illumination for industrial applications is presented in [Ash01]. A method of reconstruction of directionally dependent light sources from goniometric data for a global illumination rendering system can be found in [AP03], who also cite more papers on this topic.

Usage of luminaire description files is especially useful in fields like architectural modelling, where architects can simulate the placement and orientation of their light sources in buildings during the planning phase, e.g. in the well-known global illumina-

tion renderer Radiance [LS98]. Here, long rendering times usually make interactive display rates impossible. [WAL*97] presents a real-time simulation of *static* non-diffuse illumination by precomputing up to 8 directional "virtual lights" per specular object.

However, a simple, yet good approximation of a real-world luminaire with known goniometric data by combination of standard OpenGL lights for the purpose of *interactive illumination* in a real-time rendering context appears not to have been done yet.

3. APPROXIMATION OF GONIOMETRIC DATA

IES-LM63-95 [IES95] defines 3 types or variants of polar coordinates in which goniometric measurements can be given, depending on the luminaire application. In this paper, we will concentrate on Type C photometry data, which are typically used for luminaires for architectural and outdoor roadway lighting. The luminaire data are given as a series of candela measurements in a polar coordinate system. Vertical angles can be limited to $0^\circ \ldots 90^\circ$ (lower hemisphere only) or $90^\circ \ldots 180^\circ$ (upper) or cover all $0^\circ \ldots 180^\circ$. Similarly, only a subset of horizontal angles may be given in case of rotational or lateral symmetry.

Our task can be described as follows: For any IESNA Type C luminaire description, build a simple representation using at most 2 OpenGL lights [SWND04]. With only 8 light sources available in OpenGL implementations, this limitation had to be chosen, also because the number of light sources in a scene significantly influences rendering speed. Our work should be integrated into an existing scene graph system and run also on older, non-programmable graphics hardware, so exploiting programmable hardware was not required.

In this context, luminaires emitting light in one hemisphere can only be modeled with single spot lights. Except for highly collimated lights (e.g., searchlights), most real-world light sources show quadratic attenuation in far-field photometry context, so we can set linear and constant attenuation to zero. So, apart from a scaling factor w, only the cut-off angle θ_{cut} and the exponent p of the cosine can influence the difference between data and approximation.

If light data are given for the full sphere, we are trying a combination of a single spot directed upwards or downwards (in case the file just contains zeros for one hemisphere), a single point light, one point light and one spot directed up- or downwards, or a set of 2 spots pointing into arbitrary directions in opposite hemispheres. Here, also the orientation of the spots around vertical and horizontal axes may have to be taken into account.

Distance Metric

All fitting methods make use of some error function decribing a "distance" between the data set of the original (given) lightsource and the approximated lightsource, which in our case consist of one or more OpenGL lightsources. The goal of any search process is to find an approximation which has the smallest possible error value obtained with this error function. Thus the role of this error function is central, because it controls any optimisation, that is, it defines the meaning of the desired solution. So the sort of "distance" to be taken into consideration is crucial. In contrast to the Hausdorff metric used in [Ash99], we are interested only in pointwise radial differences, since the distance of two functions is conceptually different from the distance of their geometrical representations.

The Error Function

As a first step, and also for visualisation, the candela data given in the luminaire data file are scaled to relative "magnitudes" m_{lum} so that the largest candela value is scaled to $\max(m_{lum}) = 1$.

Then, for all directions given in the IESNA data file, a synthetic light which can be modeled directly by our OpenGL light combination is evaluated. This light consists of 3 parts, of which at most 2 may be active: a spotlight of relative brightness w_1 and cosine exponent p_1 pointing into the lower hemisphere with orientation angles θ_{ori1} from vertical down and φ_{ori1} around vertical axis, another spotlight of relative brightness w_2 pointing into the upper hemisphere with cosine exponent p_2, orientation angles θ_{ori2} from vertical up and φ_{ori2} around vertical axis, and a pointlight of relative brightness w_3. These partial lights are combined to a synthetic magnitude m_{approx} according to:

$$\begin{aligned} m_{approx}(\theta,\varphi) = {} & m_{down}(w_1,p_1,\theta_{cut1},\theta_{ori1},\varphi_{ori1},\theta,\varphi) \\ & + m_{up}(w_2,p_2,\theta_{cut2},\theta_{ori2},\varphi_{ori2},\theta,\varphi) \\ & + m_{point}(w_3) \end{aligned}$$

where e.g. $m_{down}(w_1,p_1,\theta_{cut1},\theta_{ori1},\varphi_{ori1},\theta,\varphi)$ describes the magnitude of the spotlight pointing into the lower hemisphere. To evaluate, we have to project the requested direction (θ,φ) along the orientation $(\theta_{ori1},\varphi_{ori1})$ of the OpenGL light, with $\theta_I = -90 + \theta,\ \varphi_I = \varphi,\ \theta_O = 90 - \theta_{ori1},\ \varphi_O = \varphi_{ori1}$:

$$\begin{aligned} x_I &= \cos\theta_I \cos\varphi_I & x_O &= \cos\theta_O \cos\varphi_O \\ y_I &= \sin\theta_I & y_O &= \sin\theta_O \\ z_I &= \cos\theta_I \sin -\varphi_I & z_O &= \cos\theta_O \sin -\varphi_O \end{aligned}$$

The projected cosine is $r = x_I \cdot x_O + y_I \cdot y_O + z_I \cdot z_O$, so

$$m_{down}(w_1,p_1,\theta_{cut1},\theta_{ori1},\varphi_{ori1},\theta,\varphi) = \begin{cases} 0 & \text{if} \quad r \le \cos\theta_{cut1} \\ w_1 \cdot r^{p_1} & \text{if} \quad r > \cos\theta_{cut1} \end{cases}$$

A similar evaluation can be done to compute $m_{up}(w_2, p_2, \theta_{cut2}, \theta_{ori2}, \varphi_{ori2}, \theta, \varphi)$.

m_{approx} represents a magnitude value just for one direction (θ, φ), where θ is counted from vertically downward ($\theta = 0°$) to vertically upward ($\theta = 180°$) and φ from the x-axis ($\varphi = 0°$) in a positive sense around the vertical y-axis. To estimate the total difference between this light combination and the measured data, we have to sum over all $n_\theta \cdot n_\varphi$ angular pairs (θ, φ) given in the IESNA data file:

$$err = \sqrt{\frac{\sum_{\substack{\text{vert. angles } \theta \\ \text{horiz. angles } \varphi}} \left(\frac{m_{approx}(\theta,\varphi) - m_{lum}(\theta,\varphi)}{m_{lum}(\theta,\varphi) + C} \right)^2}{n_\theta \cdot n_\varphi}}$$

Here, C is an arbitrary constant to avoid a division by 0 in case $m_{lum}(\varphi, \theta) = 0$.
Obviously, θ_{ori1}, φ_{ori1}, θ_{ori2}, φ_{ori2}, θ_{cut1}, θ_{cut2}, p_1 and p_2 are directly mapped to the orientation angles, cutoff angles and cosine exponents of the spotlights, if applicable. The weights w_i can be used to modify the lamp colours L_d and L_s. If we have more than one luminaire in the scene, we may have to adjust all lamps' colours (i.e. brightnesses) to match each other, so here the total lumen value given in the IESNA data file has to be taken into account. Also, w_i can be used to scale simulated photometric solids of the approximated lights in the interactive scene viewer application. In case a w_i is found which is very small in comparison with the others, so that its contribution in the scene is negligible, the light may be even omitted entirely to speed up the rendering.

4. OPTIMISATION

In order to optimise the overall error function we use a general function minimizer core algorithm applicable to different dimensionalities, in our cases 1–15, which correspond to the different light combinations. The situation is even more complicated because in some cases these parameters can be divided into independent subsets, e.g., when the two spotlights are oriented to directly opposite directions, having no relation between their weights on the overall errorfunction. Exploitation of this independence clearly decreases the computational cost in practice, and it is worth to exploit it in case of aiming for a fast optimisation for a case where the complexity of the problem is similar to ours.

Any commonsense error function must belong to the class C^2, meaning twice continuously derivable functions. More exactly, this property shall be fulfilled "almost everywhere" within the original domain. This means that discontinuities of the function or its derivatives can form only a very sparse subset in the entire domain, and also within commonsense less-dimensional subsets of it, like hyperplanes, lines or on its (hyper)spheres. We rely on these properties implicitly in the optimisation algorithm.

```
OverallOptimization(domain D, errorfunction f, point firstGuess,
clustersOfDirections (cl_i)_{i<I}):
optimalResultExternal := ∞
point p := firstGuess ∈ D
for attempts := 0 to attemptsnum do
  optimalResultInternal := ∞
  repeat
    for all i < I do
      dir_i := GoodCombinationOfDirections(p, f, cl_i)
    end for
    goodDir := GoodCombinationOfDirections(p, f, (dir_i)_{i<I})
    (result, p) := DirectionalOptimization(p, f, goodDir)
  until result ≤ optimalResultInternal
  if optimalResultInternal < optimalResultExternal then
    optimalResultExternal := optimalResultInternal
    optimalPoint := p
  end if
  select new random multidimensional point p ∈ D
end for
return optimalPoint

GoodCombinationOfDirections(point p, errorfunction f, direction set
(dir_j)_{j∈J}):
complete (dir_j)_{j∈J} by the generalized gradient, i.e. the linear combination of (dir_j)_{j∈J} with their corresponding directional derivatives (d_j)_{j∈J} :
Σ_{j∈J} d_j * dir_j
optimalResult := ∞
for attempts := 0 to attemptsnum do
  dir := Σ_{j∈J} c_j * dir_j {some linear combination of the (completed) set
  (dir_j) j ∈ J by some random coefficients within the unit sphere (c_j)_j ∈ J}
  result := directional derivative at p toward dir
  if result < optimalResult then
    optimalResult := result
    optimalDir := dir
  end if
end for
return optimalDir

DirectionalOptimization(point, errorfunction, direction):
reduce problem to a one variable problem, and use a combination of Newton and bisection methods.
Directional first and second derivatives are obtained by using finite difference forms.
return (point, value)
```

Figure 1: Pseudocode of the multidimensional solver

The optimisation algorithm consists of a structure of 3 nested loops, they are realized in the functions of Figure 1. The function **OverallOptimization** contains the outermost loop of independent attempts, each of them provides a quasi optimal result following from a random starting point. The core of its loop first defines a quasi optimal direction by calling the routine **GoodCombinationOfDirections** in a special way, then **DirectionalOptimization** is triggered by taking this direction.

Definition of the aforementioned quasi optimal direction uses information about classification of dimensions, in such a way that first defines a quasi optimal direction in each subspace belonging to the clustered dimensions' axis in question, and finally an overall quasi optimal direction is defined in the subspace defined by the obtained individual directions. This two pass definition with the appropriate **GoodCombina-**

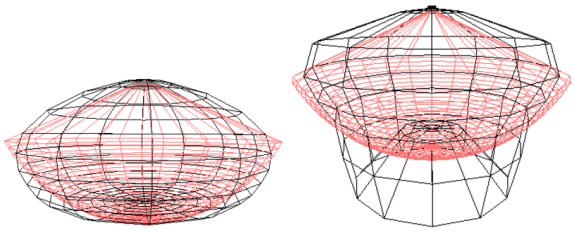

Figure 2: Photometric solids (black) and their OpenGL approximations for typical street lamps. The light bulbs sits on top of the lamps. *Left:* This light is modelled as a flattened spot with cutoff. Only some light emitted almost sideways is omitted by the approximated light. *Right:* The central obstruction of this luminaire causes the maximum brightness to fall along a ring. Such a distribution cannot be modelled properly with the simple OpenGL lights.

tionOfDirections function exploits the separation of dimensions properly.

The innermost **DirectionalOptimization** moves the point to an optimal position. This optimisation works fast on the different levels, exploiting the advantages of the regularities and conforms to the singularities as well. This flexibility characterises the innermost and also the middle loop. Consequently the algorithm provides a fast and reliable solution.

5. RESULTS AND CONCLUSIONS

We have implemented an IESNAlight class for use in a scene modeler and previewer application based on the OpenSG scene graph system [R*] to prepare scenes for a high quality rendering system. By showing wireframe models of the real and approximated photometric solids in addition to the approximative illumination given by the light, the scene modeler can easily see the quality of approximation for different kinds of luminaires (Fig. 2). The approximation computation takes a few seconds on a 2GHz class Athlon PC, so it is fast enough to include into an interactive application with IESNA file loader (Fig. 3).

Our method has shortcomings with lights that do not show spotlight characteristics (e.g. lights with central obstruction). An entirely different approach using the programmable shaders now available should be able to deal with the luminaire data directly and eventually bring even more realism into interactive applications.

Acknowledgements

This work was supported by the European Union within the scope of project IST-2001-34744, "Real-time Visualization of Complex Reflectance Behaviour in Virtual Prototyping" (RealReflect). Car model courtesy DaimlerChrysler.

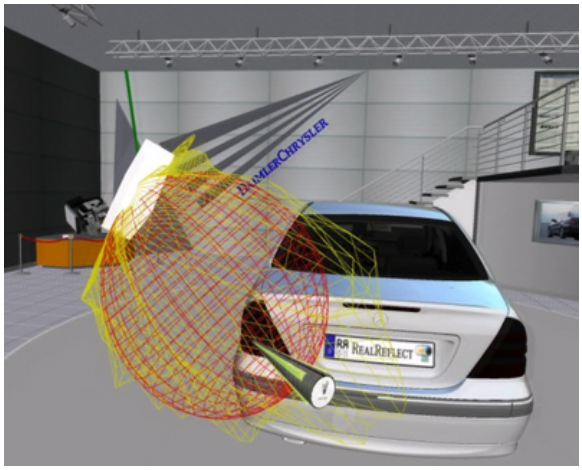

Figure 3: Usage of the visualization of a luminaire with asymmetric distribution in interactive illumination planning for an exhibition stage.

REFERENCES

[AP03] ALBIN S., PÉROCHE B.: Directionally dependent light sources. In *WSCG SHORT PAPERS proceedings* (Plzen, Czech Republic, 2003), UNION Agency – Science Press.

[Ash99] ASHDOWN I.: *Comparing Photometric Distributions*. Tech. rep., Department od Computer Science, University of British Columbia, 1999.

[Ash01] ASHDOWN I.: Thinking Photometrically Part II. LIGHTFAIR 2001 Pre-Conference Workshop, March 2001.

[GGHS03] GOESELE M., GRANIER X., HEIDRICH W., SEIDEL H.-P.: Accurate Light Source Acquisition and Rendering. In *Proceedings of ACM SIGGRAPH 2003* (2003), pp. 621–630.

[IES95] IESNA COMPUTER COMMITTEE: *IESNA Standard File format for Electronic Transfer of Photometric Data*. Illum. Eng. Soc. of N. America, New York, 1995. IESNA LM-63-95.

[LS98] LARSON G. W., SHAKESPEARE R.: *Rendering with Radiance*. Morgan Kaufmann Publishers, San Francisco, California, 1998.

[R*] REINERS D., ET AL.: OpenSG. Website. `http://www.opensg.org`.

[SWND04] SHREINER D., WOO M., NEIDER J., DAVIS T.: *OpenGL Programming Guide*, fourth ed. Addison-Wesley, 2004.

[WAL*97] WALTER B., ALPPAY G., LAFORTUNE E., FERNANDEZ S., GREENBERG D. P.: Fitting Virtual Lights for Non-Diffuse Walkthroughs. In *Computer Graphics Proceedings (SIGGRAPH)* (1997), pp. 45–48.