Real-time Image Based Lighting in Software Using HDR Panoramas

Jonas Unger, Magnus Wrenninge, and Mark Ollila
Norrköping Visualization and Interaction Studio
Linköping University, Sweden
marol@itn.liu.se

Abstract

We present a system allowing real-time image based lighting based on HDR panoramic images. The system performs time-consuming diffuse light calculations in a pre-processing step, which is key to attaining interactivity. The real-time subsystem processes an image based lighting model in software, which would be simple to implement in hardware. Rendering is handled by OpenGL, but could be substituted for another graphics API. Applications for the technique presented are discussed, and includes methods for realistic outdoor lighting. The system architecture is outlined, describing the algorithms used.

Introduction

Over recent years there has been much excitement about image based rendering, modelling and lighting [1,2,3,4,5]. The aim of this research is to create a system that allows dynamic objects to be lit accurately according to their environments, preferably using a global technique (i.e. everything in the surrounding scene can affect the given object), while still keeping the process fully real-time. The main benefit will is to obtain greatly enhanced realism, as well as better visual integration through the added object-scene interaction. This aim requires us to part with the ordinary way of doing real-time lighting; that is, with well-defined light sources. In real life, everything we are able to observe reflects light to a certain extent, and thus cannot be neglected in lighting calculations.

Disregarding the fact that other objects cast light onto each other is a great simplification, and the greatest limitation of real-time rendering has always been the necessary use of local illumination models. Local illumination accurately models what a given surface would look like considering one or a few point light sources, but does not take into account the interaction between illuminated surfaces. The downside of this is evident in the poor realism seen in most computer games and other real-time applications.

In the last few years the use of lightmaps has become popular, since they allow static objects to have a radiosity solution baked onto the model as a texture. This way, the time-consuming part of the lighting calculations is moved to a pre-processing step. Lightmaps work very well for buildings and such objects, but do not allow dynamic objects to be lit using global illumination. The fact that dynamic objects (for example player characters and opponents) are not lit as convincingly as their environments makes for poor and unrealistic integration.

Applications

Our technique of using images to light virtual scenes has many possible application areas. Here we will focus on lighting outdoor scenery. Lighting outdoor scenes has always been a difficult area in creating computer images and especially in real-time graphics. As mentioned earlier, this is a consequence of using local illumination models. Mainly, these fail to model the fact that in outdoor scenes a large contribution of incoming light is reflected light, whereas local illumination models only regard direct lighting, i.e. light sources that have a direct view of the object.

By capturing the light situation using a light probe [3], far more realistic lighting can be achieved. This kind of lighting has proven successful in, for example, motion pictures. All kinds of natural phenomena in outdoor scenes could potentially benefit from using image based methods for lighting.

The System

Figure 1 shows the basic dataflow of the system. As can be noted, it is divided into two main blocks – offline image pre-processing and online real-time rendering.

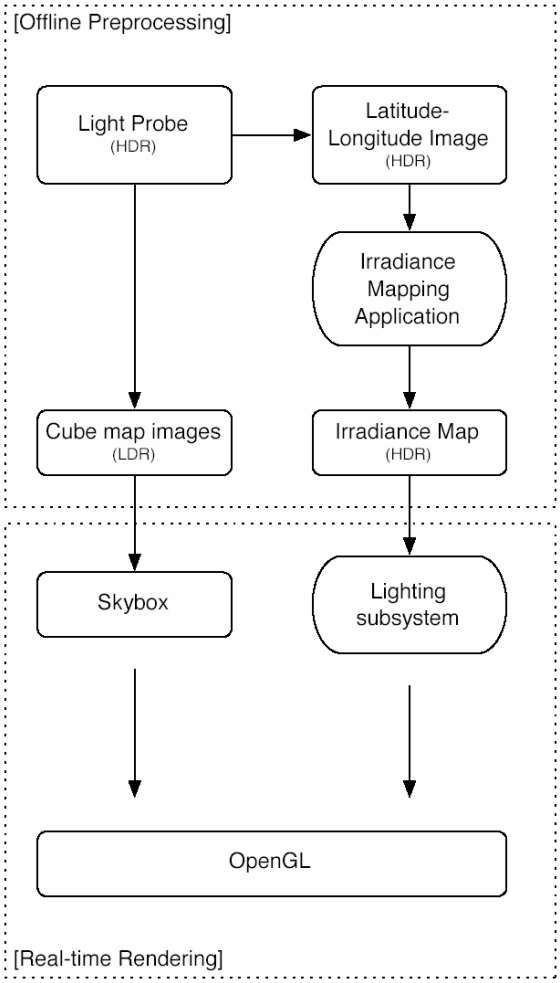

Figure 1: System architecture

Irradiance Mapping

In order to achieve real-time lighting, the time consuming integrations are performed in a pre-processing step. The HDR panorama image (the radiance map) is used to calculate a diffuse map which, during rendering, is used as a lookup table by mapping a normal direction $\mathbf{N}(\phi,\theta)$ into an irradiance value $I_N(\mathbf{N})$. Both maps are stored as a latitude-longitude map, since that gives a one-to-one correspondence with spherical coordinates.

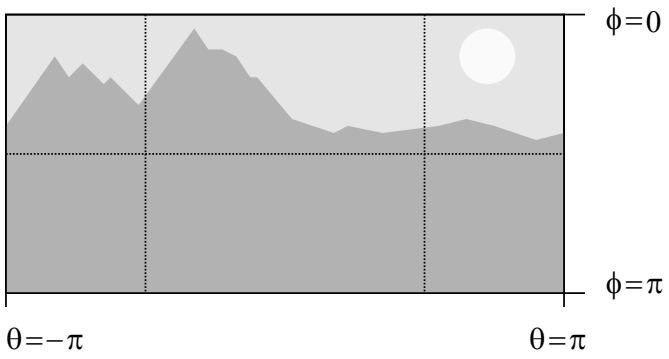

Figure 2: Radiance map.

The radiance map (see Figure 2) is a representation of the plenoptic function $I(x, y, z, \phi, \theta, \lambda, t)$ as $I_R(\phi,\theta)$. We assume that the interaction between an object at point $\mathbf{P} = (x, y, z)$ and the distant scene recorded in the irradiance map is one-way, i.e. we only consider light transport *from* the surrounding scene to the object and not vice versa. This is accurate as long as the distance between the object and the surrounding scene is sufficiently great.

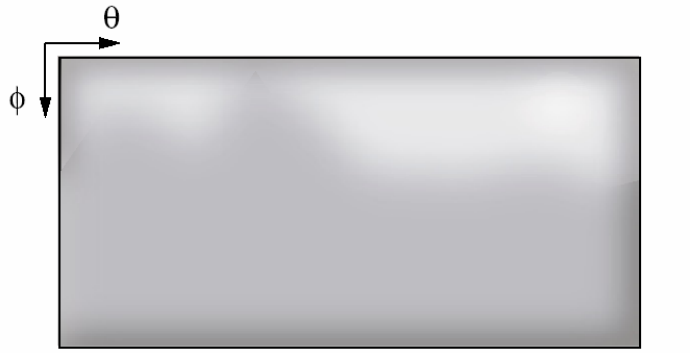

Figure 3: Diffuse map.

The diffuse map (see Figure 3) can be explained as follows: For every direction (ϕ, θ), given the point **P** where the radiance map was captured, there is an imaginary surface element with a normal $\mathbf{N}(\phi, \theta)$. The diffuse map then stores a measure of the incoming radiance onto the surface element for each direction **N**, which is calculated in as follows:

$$I_{\mathbf{N}} = \int_{-\pi}^{\pi} \int_{0}^{\pi/2} I_R(\mathbf{L}) \cdot (\mathbf{N} \bullet \mathbf{L}) d\phi d\theta$$

$$\mathbf{L} = (\phi, \theta), \ |\mathbf{L}| = 1$$

Where **L** is the unit vector in direction (ϕ, θ), $I_R(\square)$ is the intensity in the radiance map in direction **L** and **N** is the surface element normal. In Figure 4 we show how the integration limits comes from the hemisphere around the normal **N**. To produce an entire diffuse map, solve I_N for all **N** and store the result at the pixel corresponding to **N**.

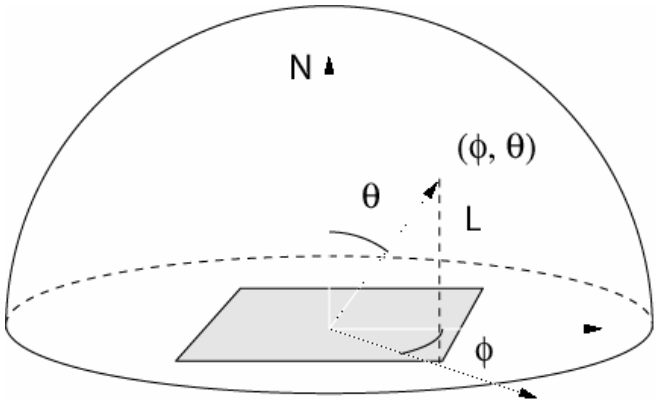

Figure 4: Hemisphere

Lighting

During rendering, the normal direction **N** of any given vertex in the object is used to look up the diffuse light intensity affecting it. Since the intensity values in the diffuse map are stored in spherical coordinates this is a straightforward process. The value is then used in combination with the vertex material properties to produce the final colour at the vertex. The algorithm is outlined below:

For each vertex
    Get the normal **N**
    Convert **N** to spherical coordinates **N** = (ϕ, θ)
    Look up the pixel value stored at (ϕ, θ) in the diffuse map
    Light vertex according to material properties and the light intensity
    Store result as vertex colour
End

Results

The system presented fulfils the goals set, being real-time and interactive. It implements a software image based lighting model, passing lit geometry to OpenGL for rendering. Inputs required are 3D-objects and HDR panoramas, making the process simple and straightforward. The panoramas are processed off-line in our custom image-processing

application to produce diffuse maps. This pre-processing removes the time-consuming integration involved in global illumination solutions allowing real-time performance during rendering.

[Image 1]

[Image 2]

Figure 5: We demonstrate the difference between using a single point light source and using the algorithm described in this paper. Most importantly, the second image shows realism in two areas: matching light intensity with that perceived as coming from the surroundings, as well as matching the overall colour balance of the object to the panorama.

[Image 3]

[Image 4]

Figure 6: We show two images which show the effect of colour blending. Great dynamism is evident in the second image where the light under the arcades is approximately 1000 times weaker than that coming from the somewhat cloudy sky. This gives rise to the truly dramatic, yet realistic, shading of the model.

Figure 7: The dynamic properties of the system are shown in images 5, 6 and 7. Here, the model is clearly lit differently as it is spun to face different parts of the surrounding scene. The effect is clearly evident in the forehead of the model, as well as on its coat.

Conclusions and Future Work

Further developments of this lighting technique can be divided into three main areas: improving the quality and speed of the rendering through hardware implementation, adding greater realism to the given system, and extending the system to allow completely dynamic scenes and views to be rendered.

In order to speed up the rendering process even further it could be implemented for consumer hardware with vertex and pixel programs. This would also allow the lighting to be calculated per pixel instead of per vertex, as is currently the case. The entire lighting process could then be performed using just a single texture unit, however this requires implementation of HDR texture support in hardware, as well as reprogramming the hardware's texture lookups.

Yet greater realism could be achieved by adding reflections and specular highlights. Both could be pre-processed by the same image-processing software as used to create diffuse maps. This way, rendering quality close to what ray-tracing software produces is possible for dynamic objects, not just static ones.

The system described is currently only able to display lighting situations in a single location. This would be quite sufficient to model outdoor lighting coming from the sun and clouds. A major advancement would be to extend the system into using many light situations. This would allow dynamic actors in a scene to be influenced by different light situations depending on where in a scene they are located.

Acknowledgements

Filip Wänström who was involved in the implementation of the software. Also to Paul Debevec for supplying the HDR panorama images that were used in this research.

References

[1] COHEN, J., TCHOU, C., HAWKINS, T., DEBEVEC, P. Real-Time High Dynamic Range Texture Mapping. Rendering Techniques 2001: 12th Eurographics Workshop on Rendering. Pp. 313-320, 2001.

[2] DEBEVEC, P. E. Rendering synthetic objects into real scenes. In SIGGRAPH 98 (July 1998).
[3] DEBEVEC, P. E., AND MALIK, J. Recovering high dynamic range radiance maps from photographs. In SIGGRAPH 97 (August 1997), pp. 369-378.
[4] DEBEVEC, P.E. Image-Based Lighting. IEEE Computer Graphics & Applications. 22 (2), pp. 26-34, 2002.
[5] DEBEVEC, P.E., TAYLOR, C.J., AND MALIK, J. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image Based Approach. Proceedings of SIGGRAPH 96. pp 11-20, 1996.