Implementing Lightcuts

Bruce Walter Sebastian Fernandez Adam Arbree Kavita Bala Michael Donikian Donald P. Greenberg
Program of Computer Graphics, Cornell University[*]

1 Introduction

Lightcuts is a new scalable framework for computing illumination in scenes with complex lighting. It handles different types of illumination, including HDR environment maps, sun/sky models, area lights, and indirect illumination and scales well to handle scenes with complex geometry and non-diffuse materials. The core component is a scalable method for accurate approximating the illumination from many point lights (e.g., thousands or millions).

First a large set of point lights is generated that approximate the exact lighting. Then these point lights are clustered together into a binary tree called the light tree. For each point to be illuminated, a cut through this tree is adaptively computed that satisfies a perceptually-based error threshold. Evaluating the lights and clusters on this cut produces a high quality approximation of the total illumination. A related technique, called reconstruction cuts, is used to exploit spatial coherence by interpolating illumination components in image regions where they are sufficiently smooth.

The novel components of the lightcuts framework are described in the companion paper, [Walter et al. 2005]. This sketch will concentrate on the software design and algorithms needed to create an efficient implementation of the lightcuts framework.

Figure 1: Bigscreen model: an office lit by two overhead area lights, two HDR flat-panel monitors, and indirect illumination. Our scalable framework quickly and accurate computed the illumination using 639528 point lights. The images on the monitors were also computed using our methods: lightcuts and reconstruction cuts.

2 Overview

The lightcuts framework divided into several stages. Important optimizations for each are listed below:

Tree Building We use a simply greedy approach to cluster the lights and build the tree. Naively implemented this would be $O(N^3)$ and be prohibitively expensive. Using a combination of a modified kd-tree, a hash set, and a max heap, we are able to quickly build light trees from large numbers of lights. The clustering is based both on spatial proximity and orientation similarity, so we use techniques to rapidly build bounding cones in direction space as well as axis-aligned bounding boxes in world space.

Because we want to scale to millions of point lights, we need to store our lights and clusters compactly. We also separate them into a read-only tree and separate per-cut scratch space to allow for parallel processing during shading.

Lightcut Selection and Refinement Tracing rays to check the visibility of the lights is the dominant cost in our rendering method. Whenever possible, we we reuse the results of previous rays to reduce cost. We cache previous visibility query results and force clusters to share representative light's with one of their children.

Computing bounds on the maximum BRDF (Bidirection Reflectance Distribution Function) value over a cluster's bounding box is also a significant cost. Since we compute many such bounds for each point, we split the bounds into components that can be computed once for each point and components that must be evaluated for each light or cluster. There are also tradeoffs between the bounds' tightness, computation cost, and storage requirements. We will discuss the particular bounds that we use and some additional optimizations such as cone pretests to reduce the average cost of bounding glossy BRDFs.

The refinement process requires a max heap which we implemented as a quadtree to reduce its depth. We will also describe the efficient handling of specular rays for materials with mirror or transmissive components.

Reconstruction Cuts Reconstruction cuts exploit spatial coherence by interpolating between samples in image space. These samples are processed versions of the cuts from above. We will describe how the samples can be processed and stored efficiently. The shading from reconstruction cuts is designed to use a simple tree traversal, which put several restrictions on their decision process. We will discuss the heuristics we use to appropriately prevent interpolation around high frequency shading features such as shadows and glossy highlights.

Image Decomposition Reconstruction cut cost is proportional to the coherence of the illumination in each image region. Thus we use several heuristics to divide the image into regions that are likely to have similar illumination. These include the material, cone, and surface normals tests. We will discuss the purpose and implementation of each of these tests and show the actual image decompositions used in our results.

References

WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. 2005. Lightcuts: A scalable approach to illumination. In *Proceedings of SIGGRAPH 2005*, Computer Graphics Proceedings, Annual Conference Series.

[*]email: {bjw,spf,arbree,kb,mike,dpg}@graphics.cornell.edu