Lightcuts

Jeff Hui

Advanced Computer Graphics 2010

Rensselaer Polytechnic Institute

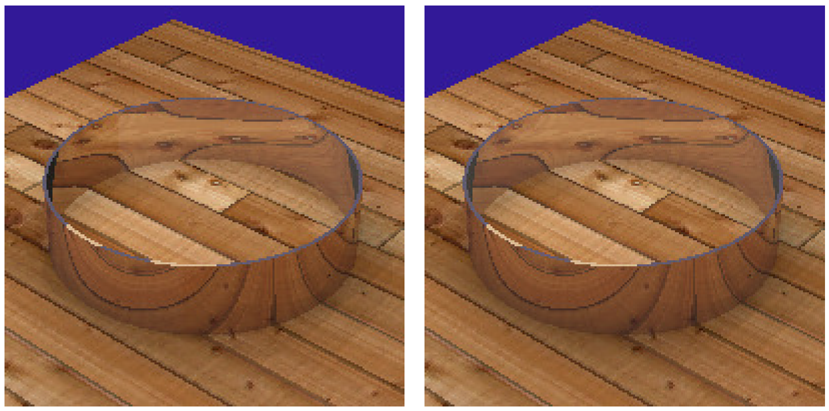


Fig 1. Lightcuts version on the left and naïve ray tracer on the right. The lightcuts took 433,580,000 clock ticks and the other took 988,800,000 clock ticks to render.

Abstract

Lightcuts is a technique to efficiently handle rendering scenes with many lights. The algorithm has sublinear performance in comparison to linear performance of traditional methods.

Keywords: illumination, ray tracing, Lightcuts rendering, many lights

1 Introduction

While much effort has been placed in improving rendering of scene geometry and shading, there is little effort in improving the linear performance time of lights in a scene. Authors of a rendering need to be concerned with the number and placement of lights.

Lightcuts[1] is one of the few techniques that tries to address this. The basic premise relies on utilizing a tree of lights to determine which lights to use at a particular query point.

2 Background

The original work[1] describes the original technique, although is void of some details, such as sublinear tree building, which is covered in a subsequent paper [2].

The technique was expanded upon to support advanced ray tracing features, such as depth-of-field, participating medium, and motion blur[3].

3 Lightcuts Technique

Given a set of lights, a subset of those lights can be used to approximate the illumination at a given point. The goal is to use the smallest subset of lights that incurs visual error below a threshold (usually 2%). The illumination of a group of lights to a particular point *x* being viewed at *w* is:

$$L_c = \sum_{i \in C} M_i(x,w)\, G_i(x)\, V_i(x)\, I_i \quad (Eq\ 1)$$

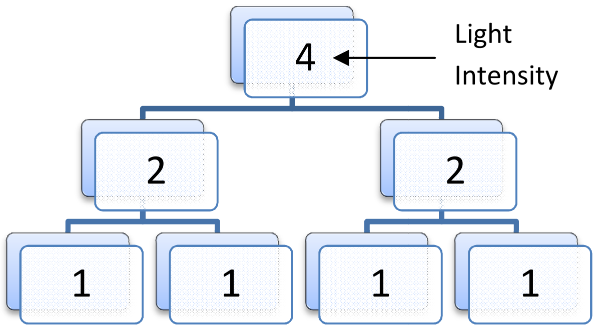


Fig 2. Light tree of four 1-intensity lights.

3.1 The Light Tree

To compute on a per ray cast efficiently, a light tree is built before rendering. This binary tree consists of light clusters. Each leaf of the tree represents an actual light in the scene. All the interior nodes are light clusters that contains two children light clusters, either real lights or other light clusters.

The clusters are organized with most similar lights sharing the same parents. For example, two lights with the similar position, color, and intensity would be grouped into a cluster. This is repeated until one cluster represents all the lights in the scene, the root node of the light tree.

The naïve method of building this tree is O(n3) where n is the number of lights in the scene. Alternatively, using a KDTree and heap can be used to achieve O(n log n), but Agglomerative Clustering provides empirical sublinear performance better than the heap method [2]. Agglomerative Clustering relies on the *non-decreasing property* of the dissimilarity of two clusters to create the same clusters consistently in any particular order.

Agglomerative Clustering can be multi-threaded to provide even faster bottom-up tree building times. This minimizes the impact of building the light tree.

3.2 Using the Light Tree

The light tree is used during rendering to determine which light clusters should be used. For each query point, we build a lightcut – a set of nodes that separates the root from its leaves. Leaf nodes are actual lights, while the interior nodes uses representative light: a random light from its children. The probability a given light is chosen as the representative light is dependent on the light's intensity. However, the cluster stores the intensity is of all the real lights it contains.

The cut first starts at the root. If the root's upper error bound (see 3.3) is greater than the acceptable error ratio (2%) times the total estimated illumination, the root is replaced with its children. This is repeated with the node with the highest upper error bound until the total estimate illumination times the ratio is greater than the upper error bound.

The illumination of a particular cluster to a point is estimated as:

$$L_c = M_j(x,w) G_j(x) V_j(x) \sum_{i \in C} I_i \quad (Eq\ 2)$$

Where j is the representative light for the cluster. This is used to calculate the total estimated illumination. The upper error bound uses the same equation, with different M, G, V functions.

3.3 Upper Error Bound

Ideally, we would take the difference from the exact and estimated illumination values, but that would require knowing the final illumination. Instead, an upper bound is used to estimate the difference.

Equation 2 is still used to calculate the upper error bound, with M, G, V functions changed to their upper bounds as mentioned in the original lightcuts.

Visibility. The visibility function is too difficult to calculate cheaply and accurately enough. For simplicity, it is always one. This means all lights are potentially visible.

Geometric. This is term is dependent on the type of light. Since all our senses assumed uniform point lights, the function is simply the closest distance from shade point *x* and the cluster's bounding box volume.

Where *y* is the position of the light and *x* is the point to shade.

Material. The phong material needs to be upper bounded. The equation was used is as follows: [4]

$$Fr(\Theta_i, \Theta_o) = k_d \frac{1}{\pi} + k_s \frac{n+2}{2\pi} \cos^n \alpha$$

Where Kd is the diffuse reflectance constant, Ks is the specular reflectance constant, n is specular and α is the angle from the ideal reflection direction and the eyepoint.

4 Difficulties

The primary difficulty is calculated acceptable error. The lightcuts paper is ambiguous and slightly disorganized when covering the upper error bound – also called 'bounding cluster error', 'upper bound on cluster error', and 'error bound'. The provided method to calculate the upper bound of the geometric term extends the number of lightcuts. This may be correct, but would require more than 400 lights in the scene (perhaps thousands). But render and tree-building times were too large in the limited time scope to complete.

Also due to time constraints, the single-threaded agglomerative clustering implementation was not implemented. The naïve O(n3) implementation was used.

Due to the poor error upper bound implementation, there are visual errors in the rendering examples.

5 Future Work

There plenty other avenues of improvement besides resolving the error calculation and adding agglomerative clustering. The current implementation assumes uniform point lights, which can be adjusted to support directional lights and oriented lights. Also, utilizing reconstruction cuts would aid in building faster lightcuts through the light tree, by interpolating lightcuts between similar shading points.

Furthermore, features mentioned in Multidimensional Lightcuts: use gather points to allow features that require multiple ray casts per pixel and smooth animations (motion blur, depth of field, participating medium).

We attempted to use ImageMagick's compare tool, to contrast the lightcuts version from the regular ray traced one. But compare finds exact pixel mismatches, not close to a perceived difference. A tool to do image subtraction would be useful, but time did not permit the creation of one. For this reason, most of the difference images are not included.

6 Conclusion

As mentioned, the final results are less than spectacular due to the weak error calculation method. The lightcuts method renders slower with few scene lights – due to the overhead of building the tree and performing cuts. However with many lights, the cost of iterating over every light can be reduced. Instead, we took the same distance calculate and took the square of the reciprocal. Scenes with few lights in unique positions performed worse, but since lightcuts is generally used in complex scenes with various kinds of lights (simulated with point lights), the technique is still useful.

6.1 Performance

Although the lightcut renderings produce artifacts, they were significantly faster when used on scenes with 100 lights. Scenes with few lights ran about 2 times slower than the naïve ray tracer. The lightcuts method rendered in half the time in scenes with 100 lights.

References

[1] Walter, Bruce, Sebastian Fernadez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. "Lightcuts: A Scalable Approach to Illumination." ACM SIGGRAPH 2005 Papers (2005): 1098-107. ACM. 7 Apr. 2010 <http://www.graphics.cornell.edu/~bjw/papers.html>.

[2] Walter, Bruce, Kavita Bala, Milind Kulkarni, and Keshav Pingali. "Fast Agglomerative Clustering." IRT (2008). 6 Apr. 2010 <http://www.graphics.cornell.edu/~bjw/papers.html>.

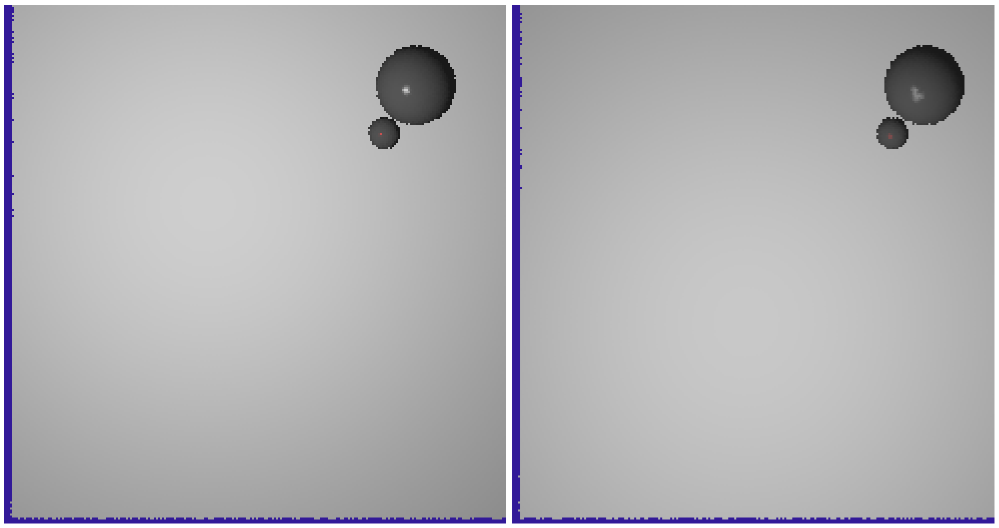
[3] Walter, Bruce, Adam Arbree, Kavita Bala, and Donald P. Greenberg. "Multidimensional Lightcuts." ACM SIGGRAPH 2006 Papers (2006): 1081-088. ACM. 6 Apr. 2010 <http://www.graphics.cornell.edu/~bjw/papers.html>.

[4] Miksik, Miroslav. "Implementing Lightcuts." CESCG (2007). CESCG. 6 Apr. 2010 <http://www.cescg.org/CESCG-2007/>.

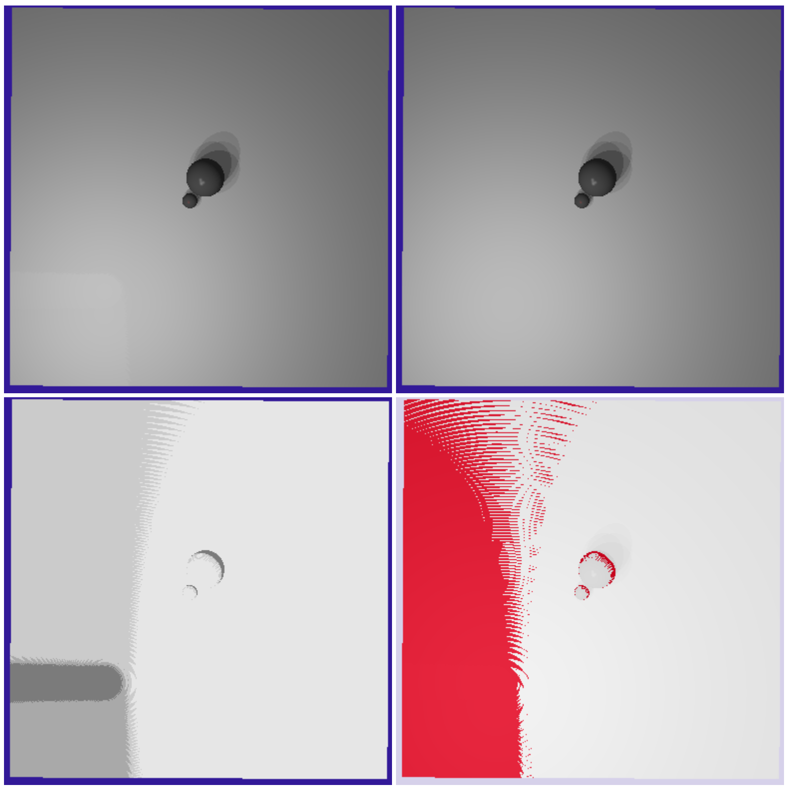
Timings for the scenes mentioned below. All timings are in ticks.

Scene #	Lightcuts	Tree Build Time	Total (Cuts + Tree Build)	Naïve Ray Tracer
2	4,370,000	0	4,370,000	1,870,000
3 (Top down)	190,300,000	150,000	190,450,000	719,260,000
3 (Side)	256,020,000	140,000	256,160,000	537,010,000
4 (w/o Reflection)	209,210,000	160,000	209,370,000	533,750,000
5 (w/ Reflection)	433,430,000	150,000	433,580,000	988,800,000

Scene 1: Five lights near the bottom left corner of the plane. Two lights are in the same location.

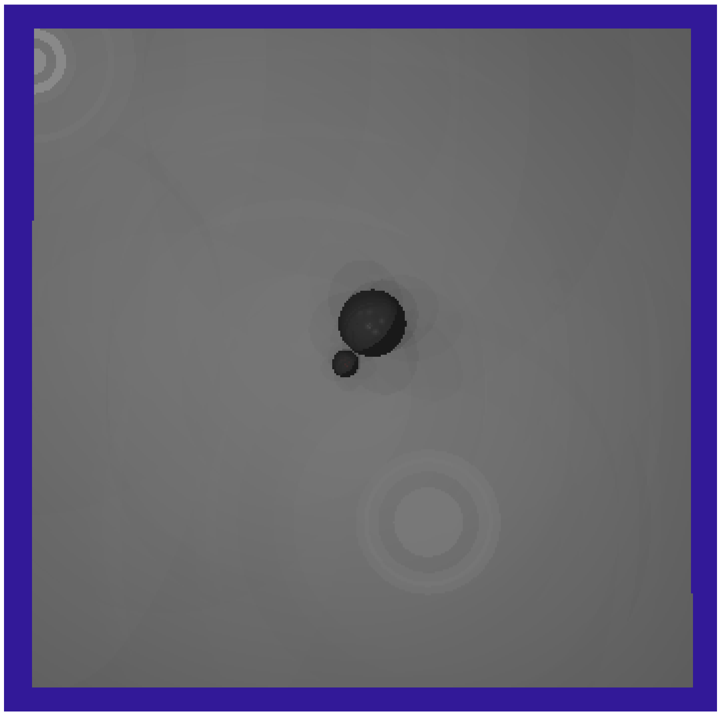


Left: Only the root light node is used. **Right**: All the lights are used for rendering. Notice the specular reflection for the spheres indicates the number of lights used in the scene.

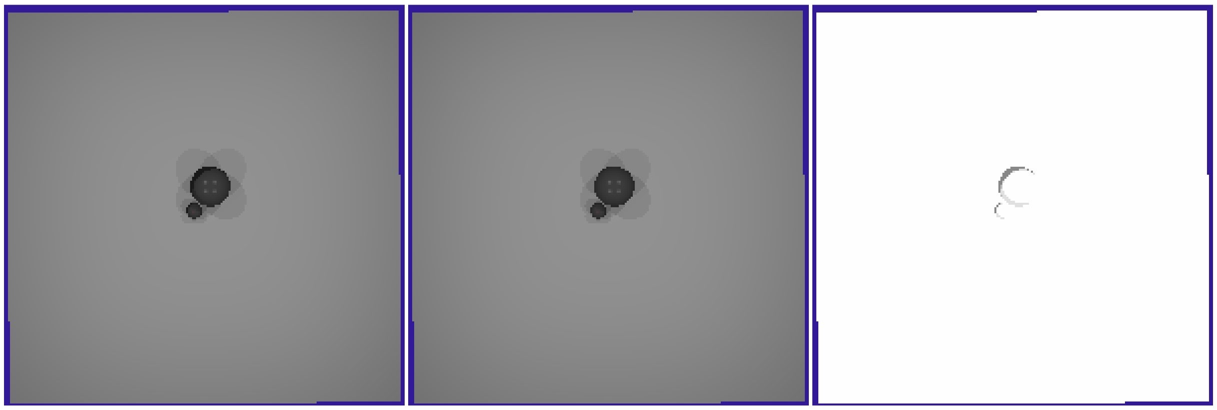


Top-Left: Lightcuts render, **Top-Right:** naïve ray tracer, **Bottom-Left:** Number of lights the lightcuts used per pixel (lighter = more lights in cut, darker = less lights in cut), **Bottom-Right:** Pixel error from true value (marked in red). The lightcuts version never used all 5 lights.

Blooper: Error calculation results in crop-circles when visualizing the number of lights used. (Scene 1)

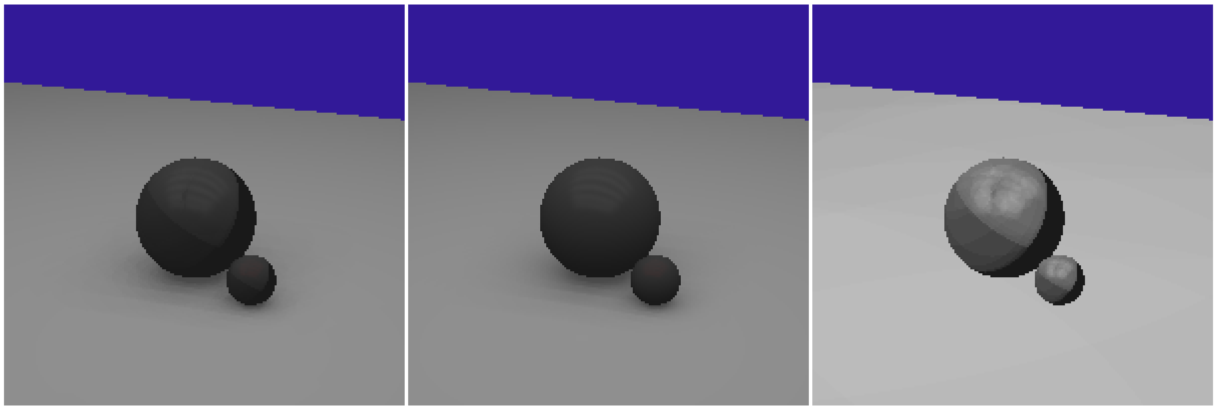


Scene 2: Four lights near the corners of the plane.

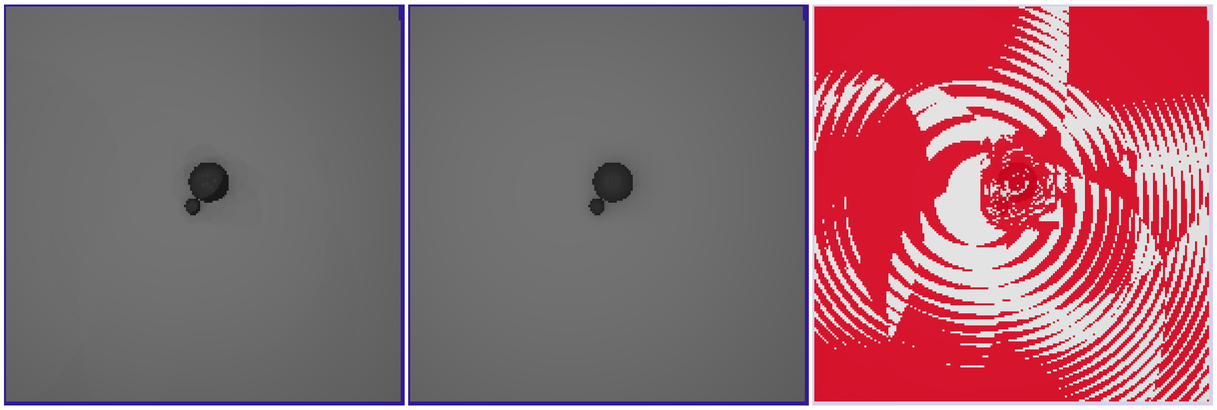


Left: Lightcuts Rendering, **Center**: is naïve ray tracer, **Right**: is the number of lights used where white is all four lights, light gray is three lights, and dark gray is two lights.

Scene 3: Scene with a 100 lights in a grid-like pattern over the ground plane.

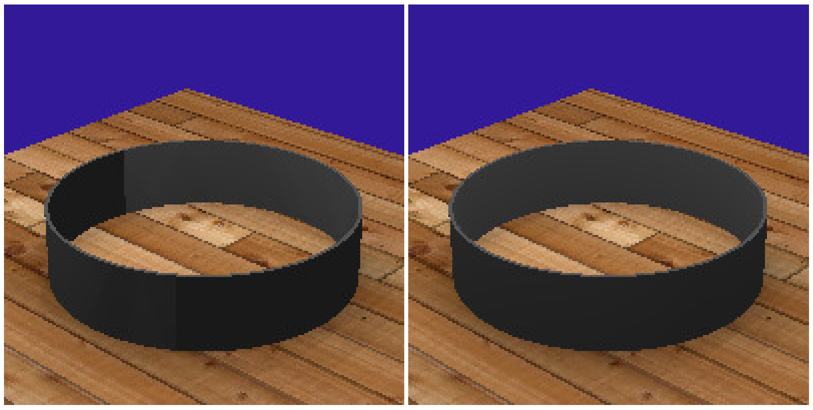


Left: Lightcuts version, **Center:** Naïve Ray Tracer, **Right:** Number of lights used for lightcuts per pixel.



Left: Lightcuts version (top-down), **Center:** Naïve Ray Tracer, **Right:** Difference - marked in red.

Scene 4: Ring with a wood texture floor with the same 100-light grid setup. Same scene as Fig 1.



Left: Lightcuts render, **Right**: Naïve ray tracer. The error is more noticeable without any reflection enabled.