# Efficient Search of Lightcuts by Spatial Clustering

Guangwei Wang[1,2] Guofu Xie[1,2] Wencheng Wang[1,2]
[1]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China
[2]Graduate University of Chinese Academy of Sciences, China

## Abstract

Lightcuts is an efficient illumination method for scenes with many complex lights, by hierarchical clustering of lights in a light tree. However, when the light tree is large, it is very time-consuming to traverse in the tree to get the suitable clusters of lights for illumination computation. For this, some methods proposed to use image coherence for neighboring pixels to share clusters of lights, and so save traversal cost in the light tree. However, with the image resolutions reduced, fewer and fewer coherences could be used and so their acceleration efficiency will be reduced dramatically, and they may even decrease the rendering efficiency.

This sketch proposes to exploit spatial coherences to enhance rendering by lightcuts. For the intersection points between rays and the scene, they are clustered on the fly and the points of a cluster search their respective suitable clusters of lights from a common set of clusters, called *a common cut*. In this way, the traversal cost from the tree root to the common cuts can be considerably saved for acceleration. Results show that our method can be faster than the methods using image coherence, and works stably with various image resolutions. And with the lights being more complex, our method can obtain more acceleration.

**CR Categories:** Computer Graphics [I.3.7]: Three-DimensionalGraphics and Realism—Color, shading, shadowing, and texture;

**Keywords:** illumination computation; lightcuts; spatial coherence

## 1 Introduction

Realistic rendering under complex light sources is an important topic of computer graphics. To reduce the number of lights for speeding up illumination computation, the popular method by lightcuts [Walter et al 2005] proposed to perform approximation computation with adaptive clustering of lights, where a binary tree is used to manage hierarchical clusters of lights, called *a light tree*. To compute the radiance $\boldsymbol{L}$ at a surface point $\boldsymbol{x}$ viewed from direction ω, it performs a top-down traversal of the global light tree to find the interior nodes whose representative lights of their corresponding clusters of lights can accurately approximate illumination from all source lights at this point. The direct illumination from a cluster, say $C$ with a representative light $j \in C$, is computed by using its representative light's material, geometric, and visibility terms $M_j(\boldsymbol{x}, \omega)$, $G_j(\boldsymbol{x})$ and $V_j(\boldsymbol{x})$ for all of its lights ($\sum_{i \in C} I_i$) in the following formula.

$$L_C(x) = \sum_{i \in C} M_i(x) G_i(x) V_i(x) I_i \approx M_j(x) G_j(x) V_j(x) \sum_{i \in C} I_i \qquad (1)$$

All the found interior nodes form a cut through the tree that every path from the root of the tree to a leaf contains exactly one such interior node. To get the cut, the root is taken as the initial cut, and progressively refined until the error criterion for illumination approximation is met. For each node in the cut, its cluster estimate (Equation 1) and upper bound on its error are computed. If its error bound is greater than required for approximation, it is removed and replaced by its children in the cut. Otherwise, it is sure in the cut and its subtree will not be investigated. The upper bound of a node is computed by Equation 1 with the supposition that the largest errors for its material, geometric, and visibility terms at the point $\boldsymbol{x}$ are $M_{upper}(\boldsymbol{x})$, $G_{upper}(\boldsymbol{x})$ and $V_{upper}(\boldsymbol{x})$. As the visibility of a point light to a surface point is typically zero and one, $V_{upper}(\boldsymbol{x})$ is always set to be 1.0 conservatively.

Clearly, the cost for searching the cuts is not cheap, and may seriously influence the rendering efficiency when the light tree is large. For this, some methods [Walter et al 2005] [Bodt 2008] proposed to use image coherence to reduce the cost on cut searching. Their strategy is to subdivide the image iteratively until the pixels of an image block can have very similar radiances. Then, the pixels of an image block get their radiances with a shared cut, or by interpolating the radiances of the corner pixels of the image block. However, when the image resolution is lower, the possibility for neighbouring pixels to have similar radiances will be smaller, and so the acceleration will be reduced.

This sketch proposes a method to enhance lightcuts by exploiting spatial coherence. It produces clusters of surface points by their materials and normals. For each cluster, a cut is first computed for its representative surface point, called *a common cut*, and then used to search the cuts for other surface points of the cluster. In this way, the traversal cost from the root of the light tree to the common cut can be shared by many surface points for acceleration, as illustrated in Figure1. Since our method is independent of the image resolution, it works stably. Moreover, benefited from our clustering scheme, we can get more acceleration when the lights are more complex, because we can better treat the occlusion relations between surface points and lights, which have more influence on the illumination computation efficiency. Results have shown the advantages of our method, and it can be faster than the methods using image coherences.

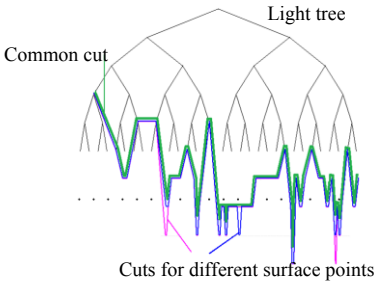



**Figure1:** *Cuts and a common cut. The blue and purple lines are cuts for some surface points, and their common cut is in green.*

## 2 Our Approach

As discussed in Section 1 for our method, it consists of two parts, with the one on clustering of surface points and the other on using the common cuts. They are discussed respectively in the following.

**Clustering**. To simplify clustering computation, we divide the bounding box of the scene in a grid, and perform clustering computation for the surface points in grid cells respectively. Our clustering is on the fly. When a surface point is obtained by a ray,


SIGGRAPH Asia 2011, Hong Kong, China, December 12 – 15, 2011.
ISBN 978-1-4503-0807-6/11/0012

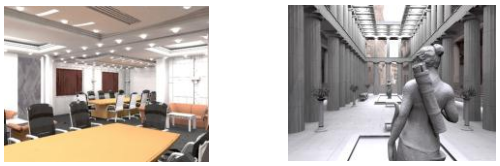
we get the grid cell that contains the point, and check whether it can join a cluster of surface points in the grid cell. If it is, the common cut of the cluster is used to find the cut for the point for its illumination computation. Otherwise, the point is taken as a representative to produce a new cluster in the grid cell, and its cut is found by searching the global light tree and taken as the initial common cut of this cluster.

**Using common cuts**. Common cuts are used for reducing the traversal cost in the light tree, aiming at speeding up rendering. When a node of the actual cut for a surface point is above some nodes of its common cut, called *over cut*, the lights of these nodes of its common cut will be used to compute the radiance at this point. This will cause no problem on the rendering quality, but may take more time than actually needed for illumination computation. To reduce the "over cut" events for fast rendering, we try to adjust the common cut to have its nodes always above the nodes of the actual cuts of the surface points in a cluster. The adjustment is executed when it is found "over cut" events. The corresponding techniques are in the following.

- **Visibility-based**. When a light is found visible to a surface point but not to the representative of its cluster, we suppose the light is visible to the representative to reduce its error bounds at related nodes, and so have the common cut move upwards in the light tree.
- **Geometry-based**. The $G_{upper}(\boldsymbol{x})$ is mainly dependent on the distance from the surface point to the lights. When it becomes smaller, the common cut can move upwards in the light tree. Thus, we compute the $G_{upper}$ values at the 8 corner points of a grid cell, and take the smallest one to adjust the common cut of the surface point clusters in this cell.

## 3 Results

We made tests on a PC installed with a 2.33GHz Intel Core E5650 CPU and 2GB RAM, to compare our method with the original lightcuts method [Walter et al 2005] and the improved methods using image coherences by cut reconstruction [Walter et al 2005] and cut sharing [Bodt 2008]. Two rendered images for the tested scenes are displayed in Figure 2, and the statistics data for the tests are listed in Table 1. In our tests, the bounding box of a scene is divided into $k*bk*kc$ grid cells with $k=0.143\sqrt{n}$ empirically, where $1:b:c$ represents the rates between the length, width and height of the bounding box, and $n$ is the number of the pixels covered by the scene.



(a) Room. (b) Hall.

**Figure2:** *The rendered images for the test scenes.*

| Scene | Polygons | Number of point lights | | | |
|---|---|---|---|---|---|
| | | Direct | Environment Map | Indirect | Total |
| Room | 514K | 3K | 0 | 4 K | 35 K |
| Hall | 3.01M | 16K | 4 K | 495 K | 515 K |

**Table1:** *Statistics for the tested scenes.*

To test the rendering efficiency of the compared methods under various image resolutions, we first set the image resolution be 1024*768 pixels and have it reduced gradually, where one viewing ray was shot per pixel. In Figure 3, it is given the acceleration ratios of our method and the methods using image coherences against the original lightcuts method. From the statistics in Figure 3, it is clear that our method can obtain acceleration stably, and faster than the methods using image coherences.

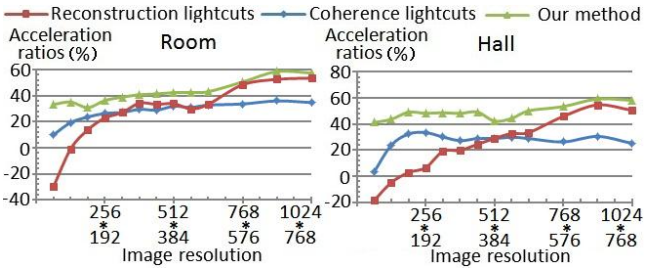



**Figure3:** *The speedups of our method and the methods using image coherences under different image resolutions.*

To test the efficiency of our method under various complex lights, we used three kinds of area lights, planar circles, cylinders and spheres, and divided them respectively into point lights. In these cases, it is listed in Figure 4 the acceleration ratios of our method against the original method by lightcuts. Clearly, our method tends to obtain more acceleration when the lights are more complex.

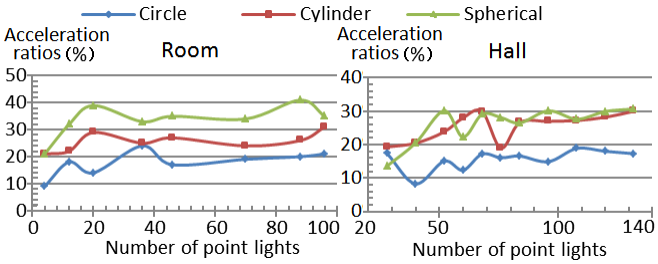



**Figure4:** *The speedups of our method under the lights with different complexities. Each area light is discretized into 256 point lights.*

## 4 Conclusions

To enhance the method by lightcuts in treating the scenes with very complex lights, we exploit spatial coherence to considerably reduce the cost on finding suitable cuts. Without preprocessing, our method sorts surface points on the fly, and have the points of a cluster search their respective cuts from a common cut, not from the tree root. As a result, our method can speed up stably, and always faster than the methods using image coherence for speedup. Especially, when the lights are more complex, we can obtain more acceleration. Therefore, our method is very helpful for massive complex illumination computation.

**Acknowledgements:** This work is partially supported by NSFC(Proj. No. 60773026, 60873182, 60833007).

## References

WALTER,B.,FERNANDEZ,S.,ARBREE,A.,BALA,K.,DONIKIAN,M., AND GREENBERG,D. 2005. Lightcuts: a scalable approach to illumination. *ACM Transactions on Graphics 24,* 3, 1098-1107.

BODT,T. 2008. Advanced global illumination using lightcuts. PhD thesis, Katholieke Universiteit Leuven.