HAL
open science

IlluminationCut

Norbert Bus, Nabil Mustafa, Venceslas Biri

To cite this version:

Norbert Bus, Nabil Mustafa, Venceslas Biri. IlluminationCut. Eurographics, 2015, Zurich, Switzerland. 10.1111/cgf.12584 . hal-01188989

HAL Id: hal-01188989
https://hal.science/hal-01188989v1

Submitted on 11 Dec 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

EUROGRAPHICS 2015 / O. Sorkine-Hornung and M. Wimmer
(Guest Editors)

Volume 34 (2015), Number 2

IlluminationCut

N. Bus[1], N. H. Mustafa[2] and V. Biri[1]

[1]Université Paris Est, LIGM, CNRS(UMR 8049)
[2]Université Paris Est, LIGM, A3SI-ESIEE
{busn, mustafan}@esiee.fr
biri@u-pem.fr

Abstract

We present a novel algorithm, IlluminationCut, for rendering images using the many-lights framework. It handles any light source that can be approximated with virtual point lights (VPLs) as well as highly glossy materials. The algorithm extends the Multidimensional Lightcuts technique by effectively creating an illumination-aware clustering of the product-space of the set of points to be shaded and the set of VPLs. Additionally, the number of visibility queries for each product-space cluster is reduced by using an adaptive sampling technique. Our framework is flexible and achieves around $3-6$ *times speedup over previous state-of-the-art methods.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Color, shading, shadowing, and texture—

1. Introduction

Rendering photo-realistic images efficiently is a challenging task in computer graphics. As the complexity of scenes, materials and lighting increases, so does the need for fast and accurate rendering methods. Unbiased algorithms such as Metropolis light transport [VG97] or bidirectional path tracing [VG95] result in the best quality images and handle the widest range of illumination types but take long time to converge due to their stochastic nature. Several solutions have been proposed to speed up rendering and to alleviate noise quickly but most of them do not retain the unbiased property of pure path tracing algorithms. Such solutions include Photon Mapping [Jen01], point-based illumination [Chr08] and many-lights methods such as Instant Radiosity [Kel97].

This paper proposes a new algorithm belonging to the family of instant radiosity methods. These methods have been successfully used for both real-time and high quality off-line rendering (see [DKH*13]). By tracing light paths from the original light sources, they place virtual point lights (VPLs) on the surfaces at every reflection point of the path (i.e., where the light path hits an object). These VPLs are then used to approximate global illumination, where the radiance of each point to be shaded (points in the scene hit by the rays traced from the camera) is calculated by summing up the illumination from each individual VPL. In high quality off-line rendering, typically hundreds of thousands of VPLs are needed to approximate an image. With such a vast number of light sources, calculating the radiance in brute-force manner is prohibitively expensive.

One efficient solution is to cluster all the VPLs into a small number of groups, which are then treated as individual VPLs. Current state-of-the-art clustering algorithms are, e.g., Lightcuts [WFA*05] and LightSlice [OP11]. Lightcuts builds a tree on the VPLs where each node of this tree represents a cluster of VPLs in the subtree of that node. For each point to be shaded, it descends in the tree to select a set of nodes (a 'cut') which is taken to be the VPL clustering for that point. While the method is robust and able to bound the error resulting from treating each cluster as a single VPL, the cut is recomputed for every point and descending in the tree is expensive for complex lighting situations.

LightSlice first groups all points to be shaded into a small number of roughly equal-sized clusters, called point-clusters, based on their geometric proximity. Then it uses visibility and shading information to obtain a clustering of the VPLs for each of these point-clusters. Thus all the points in the same point-cluster have the same clustering of the VPLs. The main advantage of the method is its speed, since it is able to detect occluded clusters and amortize the cost of creating VPL clusterings across points in a point-cluster. However, it has no error bound and as the construction of the point-clusters is not adapted to the illumination of the scene,

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

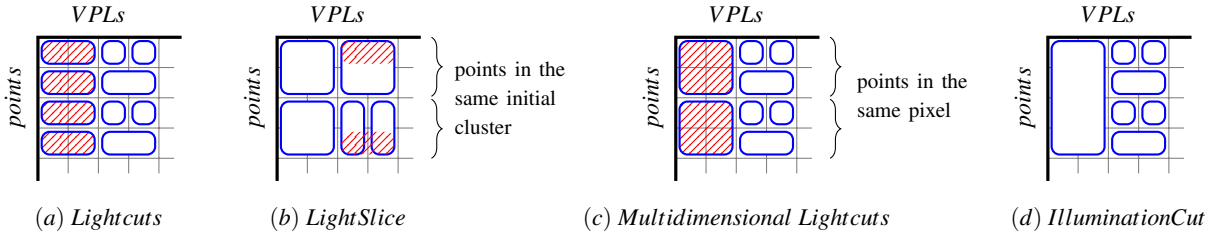


Figure 1: *Partial light transport matrices, with rectangles denoting product-space clusters. Red stripes denote parts that could be improved. (a) Lightcuts creates clusters that could be merged; (b) LightSlice creates clusters that should be merged or refined; (c) Multidimensional Lightcuts only merges and refines clusters limited to points originating from the same pixel; (d) IlluminationCut merges and refines clusters for any set of points and VPLs.*

it is prone to failure if the radiance of points within a group is highly varying.

Lightcuts constructs a different clustering of VPLs for each point; LightSlice clusters all the points to be shaded into a number of point-clusters for which the same clustering of VPLs is computed. Our idea is based on the following observation: instead of clustering points or VPLs independently, what is required is *clustering their product-space*, namely to cluster all point-VPL pairs. Each cluster in this product-space consists of a subset of points (to be shaded) paired with a subset of VPLs.

In fact, both Lightcuts and LightSlice can be seen as constructing constrained product-space clusterings. Each cluster created by Lightcuts consists of a single point paired with a set of VPLs. This constraint is wasteful as two points which are very similar could have been grouped together in many product-space clusters. See Figure 1 (a).

LightSlice, on the other hand, constructs a product-space clustering where the same set of points are grouped together in any cluster. For efficiency reasons each point-cluster is large, which severely limits how well the VPL-clusters paired to them can be adapted to each individual point in the point-cluster. Furthermore, as the initial clustering of points used only geometric information, these clusters cannot be completely adapted to illumination, and are likely to introduce artifacts on the cluster boundary. See Figure 1 (b).

Our contribution. Our proposed method, *IlluminationCut*, targets high fidelity off-line rendering by constructing an illumination-aware clustering of the product-space of all point-VPL pairs. We create illumination-aware product-space clusters without any a priori constraints on either the points or the VPLs that can appear in product-space clusters. These clusters capture similar point-VPL pairs such that shading every point in a cluster by using a single representative VPL instead of all VPLs in the cluster causes error that remains under a threshold. Treating cluster pairs enables us to amortize calculations that were previously carried out separately for each point in Lightcuts; and to construct non-uniform clusters with different subsets of points with different subsets of VPLs, which is more adaptive clustering than that of LightSlice. Our method is further extended by adaptive visibility sampling, reducing the number of rays traced for each product-space cluster without introducing high error. See Figure 1 (d).

IlluminationCut builds on the Multidimensional Lightcuts approach [WABG06], in that they both utilize two hierarchies (trees), one on points and one on VPLs to construct product-space clusters with bounded error by simultaneously descending on the trees. The difference is that the latter has to maintain a heap and repeatedly builds a tree only for points that originate from the same pixel (e.g., for use in spatial anti-aliasing). It does not exploit possible similarity among points originating from different pixels and so does not improve upon Lightcuts if there is only one point per pixel. See Figure 1 (c).

Our results improve on both the quality and the efficiency of previous methods. We achieve $3-6$ times speed-up by reducing the number of visibility queries, dramatically decreasing the computations needed to construct clusters, as well as eliminating the need for maintaining a heap during rendering.

Organization. In Section 2 we review previous work on global illumination with VPLs. A detailed description of our algorithm is given in Section 3. Experimental results and comparison with state-of-the-art methods are presented in Section 4. Finally, limitations and future work are discussed in Section 5.

2. Previous Work

Many-light methods have gained much attention recently due to the fact that they can produce high quality images in a fraction of the time taken by Monte Carlo methods. The family of many-light methods is derived from the original technique [Kel97] where each point is shaded using the direct contribution of a set of VPLs. See the SIGGRAPH 2012

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

course notes on the many-lights problem [KHA*12] and the EG state-of-the-art report [DKH*13] for a detailed description and recent advances.

Real-time techniques. The many-lights framework can be used for rendering images in real time with incremental updating of VPLs, though this is limited to few hundreds of VPLs [LSK*07]. Other methods achieving interactive frame rates include calculating level of details structures efficiently [HREB11] and imperfect shadow maps [RGK*08]. For a summary see [Rad08].

Extensions and limitations. To render participating media, virtual ray and virtual beam lights have been introduced in [NNDJ12a, NNDJ12b], respectively. Limitations of VPL-based algorithms, like clamping [KK06] of VPL contributions due to the singularities or the limitation of only representing diffuse global illumination, can be solved using virtual spherical lights [HKWB09] instead of virtual point lights. To handle more efficiently highly glossy material [KFB10], Davidovič et al. [DKH*10] use row-column sampling with an adaptive ray-casting strategy. Other techniques such as specular gathering [DKL10] combine path tracing techniques and VPL global illumination. These methods greatly increase the rendering time compared to pure many-lights algorithms.

Good fidelity off-line rendering requires a large number of VPLs (typically hundreds of thousands of VPLs). There are two main approaches that avoid computing the radiance for all point-VPL pairs.

Sampling. To decrease the complexity of the problem, one can sample the VPLs according to their contribution to the image [GS10]. The method proposed in [GKPS12] calculates the exact illumination at a sparse set of locations and builds the probability distribution of the incoming light at each sample. These distributions are then used to importance sample the VPLs. [SIMP06] uses stochastic sampling of the VPLs to achieve real-time global illumination.

Clustering. The second set of methods clusters the VPLs and/or the points that are shaded. Hašan et al. [HPB07] studied the light transport matrix, where each row represents a point to be shaded and each column represents a VPL. The clustering of the VPLs is done according to a reduced matrix (sampled from the full matrix) and finally every sample is shaded using this clustering. This technique captures successfully the global lighting and it is very efficient but fails to cluster local lighting properly. Ou and Pellacini [OP11] have further studied the light transport matrix to propose LightSlice. The family of methods based on Lightcuts [WFA*05] build a tree on the VPLs, and construct a clustering by selecting a set of nodes in this tree with an upper bound on the error caused by each cluster. Several modifications have been proposed to reduce the computations performed by Lightcuts in different scenarios. For the case of multiple point samples per pixels (e.g., used in anti-aliasing) Multidimensional Lightcuts [WABG06] extends Lightcuts by building a tree on the samples for each pixel (see Section 1). Bidirectional Lightcuts [WKB12] aims to handle a broader variety of materials and to reduce the bias present in instant radiosity methods. None of the methods exploit the similarity of points originating from different pixels. The methods in [BD08, WXW11] reduce the calculations for constructing clusters by maintaining a common cut for a group of spatially clustered pixels. These methods have to balance the trade-off between the number of points a common cut can represent and the amount of refinement performed for each point when calculating a clustering from the common cut. Reconstruction cuts [WFA*05] aims to decrease the number of visibility queries by interpolating radiosity for pixels in small image patches. In contrast, IlluminationCut is able to reduce the number of visibility queries without introducing interpolation errors. Pixelcuts [KSW11] only clusters the points to be shaded and try to minimize the calculations for each cluster by only calculating shading for one representative point. This method is primarily suited for low intensity global illumination. IlluminationCut improves over these methods by creating product-space clusters that are more general and so can exploit similarity among points in a more efficient way.

Visibility estimation. To further speed up rendering [PGSD13] caches visibility queries. The authors in [BELD13] propose a novel framework for stochastic evaluation of visibility. VisibilityCluster [WC13] first clusters the pixels and the VPLs separately and creates pairs of these clusters. Then it calculates approximate visibility for these pairs and uses this information to improve the importance sampling of VPLs.

3. Algorithm

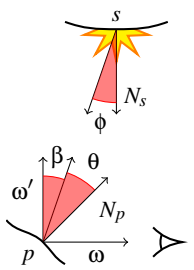
In this section we introduce notations, data structures and the algorithms used in our method. Denote by $\mathbb{S}$ the set of VPLs and by $\mathbb{P}$ the set of points to be shaded (i.e., points in the scene hit by the rays traced from the camera).

Preliminaries. Given a set of Lambertian VPLs $\mathbb{S}$, the radiance $L(p,\omega)$ at point p in direction ω can be computed as a discretization of the rendering equation:

$$L(p,\omega) = \sum_{s\in\mathbb{S}} M_s(p,\omega)\cdot V_s(p)\cdot I_s\cdot G_s(p) \qquad (1)$$

I_s is the intensity of the light s and $V_s(p)$ denotes the visibility between s and p. $M_s(p,\omega)$ is the BRDF which depends on the material at p. We use Lambertian and Blinn micro-facet BRDFs. They have the form $\frac{1}{\pi}k_{\text{diff}}\cos\theta$ and $\frac{1}{2\pi}k_{\text{spec}}(n+2)\cos(\beta)^n\cos\theta$ respectively, where each component of k_{diff} and k_{spec} has values between 0 and 1, and n is the specular coefficient. The angles β,ϕ and θ are denoted in the figure below where ω' is the view direction ω reflected with the surface normal N_p, and N_s is the normal of the light s.

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.



With these notations β is the angle between ω' and $s-p$, ϕ is the angle between N_s and $p-s$ while θ denotes the angle between N_p and $s-p$. The geometric term $G_s(p)$ captures the light attenuation $G_s(p) = \cos(\phi)/d(p,s)^2$, where $d(p,s)$ is the Euclidean distance between p and s.

Overview. The method first constructs a cluster hierarchy on $\mathbb{P}$, called the *point tree* and on $\mathbb{S}$, called the *light tree*. Then using these trees, Phase I computes a coarse but fast approximation for every point $p \in \mathbb{P}$. In Phase II, this approximate image is used to guide a top-down search of both trees simultaneously to construct the list of desired product-space clusters $(R_1,Q_1),\ldots,(R_k,Q_k)$. Here each (R_i,Q_i) is a product-space cluster composed of the set of points $R_i \subseteq \mathbb{P}$ and the set of VPLs $Q_i \subseteq \mathbb{S}$. Finally, for all product-space clusters (R_i,Q_i), illumination contribution from Q_i is added to the radiance of each point $p \in R_i$.

Light and point trees. For the light tree we use the same structure as Lightcuts [WFA*05]. For the point tree, we use a compressed octree, which differs from a simple octree by the fact that paths without branching are contracted into a single edge. Then each node in the tree represents a unique cluster of the points in its bounding box. To ensure that the points located in the same node face approximately in the same direction, we make a slight modification: the subdivision of the first 3 levels into octants correspond to the subdivision of the space of normals of the points (these are unit vectors in $\mathbb{R}^3$) and the remaining levels follow the standard octree subdivision rule. The points in $\mathbb{P}$ are stored in an array. In our implementation the octree is constructed by repeatedly subdividing the bounding box of the scene along planes perpendicular to the three axes. Thus recursively, at each subdivision, the points corresponding to a node are partitioned in-place into two contiguous subarrays. As a result the points in the array are in z-order (also called Morton-order [Gar82]). The construction ensures that each node in the tree contains points located in a contiguous part of the array. Therefore retrieving points associated to a node is efficient since one has to only iterate over a subarray.

Our algorithm stores additional auxiliary data with the nodes of both trees. These are the bounding box of the points inside the node and representative lights/points. The latter are sampled in the same way as Multidimensional Lightcuts (the sampling ensures that the algorithm remains unbiased in the Monte Carlo sense). For each light tree node we also store the bounding cone of the light directions of VPLs associated with that node. Each node of the point tree stores the maximum/minimum BRDF components in the subtree of the node $(k_{\text{spec_max}}, k_{\text{diff_max}}, n_{\text{min}}, n_{\text{max}})$. We also need to associate color data (*color*) with the nodes of the point tree.

Clusters and representatives. We will identify the nodes of the tree with the points they contain, e.g., the root of the light tree is simply denoted by $v(\mathbb{S})$. Let us denote by Q a cluster of lights $Q \subseteq \mathbb{S}$ and its corresponding octree node as $v(Q)$. Denote the radiance at p caused by lights in Q as $L_Q(p,\omega)$.

$$L_Q(p,\omega) = \sum_{s\in Q} M_s(p,\omega)\cdot V_s(p)\cdot I_s \cdot G_s(p) \qquad (2)$$

For a node $v(Q)$ let $rep(Q) \in Q$ denote its representative light, and then compute the approximate radiance at p from lights in Q with representative $rep(Q)$ as:

$$\widetilde{L}_Q(p,\omega) = M_{rep(Q)}(p,\omega)\cdot G_{rep(Q)}(p)\cdot V_{rep(Q)}(p)\cdot \sum_{s\in Q} I_s \qquad (3)$$

For a cluster R denote the radius of the enclosing ball of its bounding box by $r(R)$, and by $d(R,Q)$ the distance between the enclosing balls of clusters R and Q.

Phase I: Computing approximate shading. Our algorithm needs an estimate of the radiance of each point $p \in \mathbb{P}$. It is computed by descending in both trees until for a pair of point and VPL nodes $(v(R),v(Q))$, the condition $max(r(R),r(Q)) < 0.1\cdot d(R,Q)$ is satisfied and the aperture of the light node's cone is less than 20°; we then add the contribution of the VPL cluster Q to each point in R. This criteria attempts to ensure, though without any guaranteed bound on the error, that this estimated radiance roughly matches the value that would result from exhaustively evaluating the radiance for every point-VPL pair in (R,Q). Notice that for shading a pair (R,Q) we only take into account the represen-

Figure 2: *Approximate images for various scenes. These images are used to guide the search for product-space clusters in Phase II.*

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

Algorithm 1: IlluminationCut

```
Data: W: stack of pairs; light and point trees for 𝕊, ℙ
Function IlluminationCut()
    W ← ∅
    W.pushback(v(ℙ), v(𝕊))
    while notEmpty(W) do
        (v(R), v(Q)) = W.pop()
        if IsIllumAwarePair(v(R), v(Q)) then
            foreach p ∈ R do
                L(p, ω) += L̃_Q(p, ω)
        else
            if r(Q) > r(R) then
                if v(Q) has no children then
                    foreach p ∈ R do
                        L(p, ω) += L̃_Q(p, ω)
                else
                    foreach u ∈ children(v(Q)) do
                        W.pushback(u, v(R))
            else
                if v(R) has no children then
                    foreach p ∈ R do
                        L(p, ω) += L̃_Q(p, ω)
                else
                    foreach u ∈ children(v(R)) do
                        W.pushback(v(Q), u)

Function IsIllumAwarePair(v(R), v(Q))
    return δ · v(R).color > L_UB(v(R), v(Q))
```

tative point's BRDF and do not shade every point individually. This necessarily introduces error to our approximation image (e.g., the texture can vary within clusters). However, since this image is only used as an error upper bound it does not have a significant effect on the final image (Section 5 contains a more detailed discussion). Furthermore, instead of adding the calculated contribution of Q to all the points in $v(R)$, we can simply accumulate it in the node and later with a tree traversal, distribute it to the leaves (each containing a point of $\mathbb{P}$). We also store the minimum of the approximate radiances of the points in a node $v(R)$ as $v(R).color$. This is used in Phase II.

In Figure 2 we show the images rendered with this approximation to illustrate how they capture illumination.

Illumination-aware pairs. A set of points $R \subseteq \mathbb{P}$ and a set of lights $Q \subseteq \mathbb{S}$ form an illumination-aware pair if

$$\max_{p \in R} |\widetilde{L}_Q(p,\omega) - L_Q(p,\omega)| < \delta \cdot \min_{p \in R} L(p,\omega) \qquad (4)$$

where δ is the error threshold, e.g., 1%.

The definition ensures that for any point $p \in R$, using Q as a light cluster would result in a small error. This is the most conservative error bound but other variations, e.g., average of $L(p,\omega)$ could be used as well. In order to evaluate this condition one requires the knowledge of the true radiance a priori. This is estimated using the approximate image calculated in Phase I; i.e., use the minimum radiance in the approximate image of points belonging to R. This minimum was stored in each node as $v(R).color$ in Phase I.

The left-hand side of Equation (4), denoted $L_{UB}(v(R), v(Q))$, can be upper bounded in a similar manner as Multidimensional Lightcuts. Let M_{max} and G_{max} be the upper bounds on the material and geometric terms over all point-VPL pairs in (R,Q). As the visibility term can be upper bounded by 1, $L_{UB}(v(R), v(Q))$ can be written as:

$$L_{UB}(v(R), v(Q)) = M_{max} \cdot G_{max} \sum_{s \in Q} I_s \qquad (5)$$

We show how to calculate $G_{max} = \frac{\cos \phi_{max}}{r_{min}^2}$ where $r_{min} = \min_{p \in R, s \in Q} d(p,s)$ and ϕ_{max} is the angle between the light normals and light directions for which the cosine function attains its maximum value. r_{min} is set to $d(R,Q)$. For ϕ_{max}, we use the same technique as described in Multidimensional Lightcuts. First, we simplify the problem by calculating the bounding box of all possible light-point vectors between R and Q then apply a linear transform to this bounding box such that the direction of the light node's cone is aligned with the z axis. This transformation enables the direct evaluation of ϕ_{max}. Calculating M_{max} can be done in a similar fashion by using the surface normal and the reflected view ray ω' of $rep(R)$ in the role of the cone direction (the surface normals and reflected view rays for points in a node are located in a small cone due to our octree construction). See the details in [WABG06].

Phase II: Rendering with illumination-aware pairs. Once we have an approximate radiance for each point $p \in \mathbb{P}$ as well as the minimum radiance of all the points in a subtree rooted at v (stored as $v.color$), we again traverse the two trees simultaneously top-down to construct the illumination-aware pairs. For each illumination-aware pair (R,Q), add the illumination contribution of the VPL cluster Q to each point in R. See Algorithm 1. The straightforward way of adding the illumination contribution of Q to each point in R is by computing $\tilde{L}_Q(p,\omega)$ separately via a shadow test from each p to $rep(Q)$ and then using Equation (3). We refer to this method as IlluminationCut.

Adaptive sampling. Algorithm 1 computes, for each illumination-aware pair (R,Q), the contribution $\tilde{L}_Q(p,\omega)$ of Q to each point $p \in R$ by performing a visibility query from p to $rep(Q)$. Instead, one can use an adaptive sampling technique that reduces the number of visibility queries to be considerably less than the number of points in R, denoted by $|R|$. Given an illumination-aware pair (R,Q), we have access to

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

Algorithm 2: Adaptive sampling for a pair (R,Q)

```
Function AdaptiveSampling(R,Q)
    if Q is a leaf or |R| < 8 then
        foreach p ∈ R do
            L(p,ω) += L̃_Q(p,ω)
    else
        vs: the number of samples
        if |R| > 32 then
            vs ← 16
        else
            vs ← 8
        compute R', with |R'| = vs
        if V_rep(Q)(q) = 1 for all q ∈ R' then
            foreach p ∈ R do
                L(p,ω) += L̃_Q(p,ω) with
                V_rep(Q)(p) set to 1
        else if V_rep(Q)(q) = 0 for all q ∈ R' then
            return
        else
            foreach R_i defined by R' do
                AdaptiveSampling(R_i,Q)
```

the points in R as a subarray in z-order; we refer to this subarray as R, this ambiguity shall not cause a problem. Take a subset R' of R dividing it into at most 16 equal length subarrays and calculate the visibility between points of R' and the representative light $rep(Q)$. If all points in R' are visible or all are occluded, use this visibility for shading R; if not, then we recurse on the subarrays R_i defined by R' . We note that choosing R' in this manner makes the algorithm biased. Though errors are unlikely, as the subarrays consist of spatially proximate points. See Algorithm 2. We refer to this method as IlluminationCut-Sampling.

4. Discussion

In this section we present experimental results for several scenes with complex lighting, highly glossy materials and varying geometric complexity. Timings are for a server equipped with Intel(R) Xeon(R) E5-2680 CPUs utilizing in total 20 cores running at 2.8 GHz, with 74 GB of memory.

Implementation. We compare our algorithm with two well-known methods: Lightcuts [WFA*05] and LightSlice [OP11]. Since the authors of LightSlice published their code, we have ported their implementation into the ray-tracing system *Intel Embree 2.3* [WFWB13, WWB*14]. We have improved their Lightcuts implementation with the agglomerative clustering method presented in [Mik10]. The code is written in C++ with a very efficient ray-tracing engine. Due to recent advantages in packeted ray-tracing algorithms and their implementations, shading and other calculations account for a significant portion of the overall rendering time in our system; e.g., approximately 20% of the total time for Lightcuts is used by ray tracing (see [WPS*03]).

High resolution images are available in supplemental material. The code can be downloaded at the website of the authors.

Scenes. We test the algorithms on a collection of scenes, all of them having highly glossy materials except for *Sponza*, which is completely diffuse. The outdoor scene *San Miguel* is our largest scene, consisting of 10M triangles lit by an environment map. Many of the VPLs are placed on walls facing outward; therefore this scene is exploiting the weakness of our algorithm and Lightcuts, namely that occluded clusters are not quickly discarded. *Banquet* has a grid of point light sources directed towards the ceiling and lights inside the lamps, both creating a challenging global illumination setup as there is significant indirect lighting. It also contains a strip of area lights running around the ceiling. *Sponza* has all its original point light sources facing the ceilings on the side corridors and a moderately dark environment map. The light filtering through the gaps around the curtains are challenging to capture properly. *Kitchen* is lit by upward facing spotlights located inside the lamps and an area light under the shelf.

Parameters. For Lightcuts the error bound is set to 1% (as in earlier work [OP11]). We give the results of both IlluminationCut and IlluminationCut-Sampling, setting the error to 1.5% in the former case and to 1% in the latter. LightSlice is run with approximately 1500 slices and with varying columns. The number of slices determines the size of the reduced light transport matrix while the number of columns determines the number of clusters used for rendering a point. For the sake of compactness we refer to these algorithms in the figures as LC(1%), LS(3200), IC(1.5%) and IC-S(1%). The images have 1600×1200 resolution and use 1 sample per pixel for a clear comparison of the quality of clusterings obtained by the different methods. We give a second table with comparison for anti-aliased images with 9 samples per pixel to show how the methods behave in this case. For each scene, around 650K VPLs are generated by tracing light paths from the original light sources up to depth 10. Our implementation uses clamping by setting the point-VPL distance not smaller than 5% of the scene radius. Due to the consequent energy loss, we use the image rendered with all VPLs as our reference image.

Performance. Table 1 shows the results with various statistics for 1 sample per pixel. We set the error bound of Lightcuts to 1% and adjusted the other methods (error bound or column number) to provide similar *RMSE* quality. We note that all algorithms exhibit small variations in quality due to

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

		Banquet	San Miguel	Sponza	Kitchen
Scenes	Triangles	0.74M	10.5M	0.28M	0.17M
	VPLs	638K	677K	641K	551K
LC(1%)	Preproc. time (s)	56.05	109.22	80.57	89.80
	Render time (s)	116.71	749.41	233.31	183.07
	Avg # of rays	947	5161	2294	1597
	RMSE	0.010468	0.011369	0.005913	0.010336
	Rel. Error	2.740256	2.770336	4.316007	8.957002
	Upper bound	2878	14202	6183	4313
LS	Render time (s)	365.74(230)	227.88(41)	263.35(129)	116.61(67)
	Avg # of rays	2879	2722	2849	1362
	RMSE	0.016666	0.028958	0.011842	0.013217
	Rel. Error	3.469874	7.718524	4.048875	10.411767
	Columns	3200	3200	3200	1600
	Speedup	0.3	3.3	0.9	1.6
IC(1.5%)	Preproc. time (s)	58.28	112.48	82.07	93.02
	Render time (s)	36.94	186.12	71.83	43.04
	Avg # of rays	1301	5561	3064	1258
	RMSE	0.010401	0.012534	0.005747	0.012562
	Rel. Error	2.442368	3.137196	4.113609	9.543214
	Upper bound	90	236	69	216
	Speedup	3.1	4.0	3.3	4.2
IC-S(1%)	Preproc. time (s)	58.32	112.40	82.31	92.31
	Render time (s)	23.30	78.58	23.89	30.26
	Avg # of rays	463	1849	720	602
	RMSE	0.010097	0.012956	0.006997	0.011005
	Rel. Error	2.639261	3.807332	4.382032	9.087991
	Upper bound	155	362	102	350
	Speedup	5.0	9.6	9.7	6.1

Table 1: *Rendering statistics for* 1600×1200 *resolution images with* 1 *sample per pixel. The parameters are set to achieve approximately equal RMSE error, except for LightSlice which fails to resolve certain artifacts.*

randomness. We provide the running times for the preprocessing and the rendering phase. For our method and Lightcuts, the preprocessing consists of building the light tree while for LightSlice there is no view independent preprocessing phase. The normalized *RMSE* and average relative error provides numerical difference against the VPL reference image. Both our method and Lightcuts are using similar error upper bound calculations. In order to compare the reduction of such calculations we have included in the table the number of upper bound calculations averaged over the number of pixels. We also give average number of shadow rays per pixel. Note that these are not identical, since lights facing away are not tested and additional shadow rays are traced in other parts of the algorithms. The latter happens for Phase I and for building the reduced light transport matrix in LightSlice. The error images are calculated by taking the channel-wise Euclidean distance between the image, and the VPL reference image, and multiplying it by a factor of 16. For LightSlice we report in parenthesis the time to cluster the reduced matrix because that is single threaded and it is a significant part of the rendering causing the main bottleneck of the algorithm. We also include highlights of typical errors (see Figure 3).

Comparison to Lightcuts. In general, the quality of our results is similar to Lightcuts with $3-6$ times speedup. Both methods adapt well to the scenes, keeping the error low with the upper bounding methods and both methods oversample shadowed areas. The only visible artifacts are the non smooth shading of uniform surfaces, e.g., in *San Miguel*. Note that as Embree is a high performance ray tracing kernel we gain speed-up by significantly reducing the cost of clustering. In Figure 4 we illustrate the number of upper bound calculations for each point $p \in \mathbb{P}$. The images are using false coloring with a logarithmic scale. In Lightcuts, for each p we add 1 at each upper bound calculated while shading the point. In IlluminationCut, when an upper bounding calculation is carried out for an illumination-aware cluster we add to each p in the cluster the inverse of the number of points in the cluster. This shows the amortized cost of clustering and it is consistent since an upper bounding calculation still increases the total value by 1, just as in Lightcuts.

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

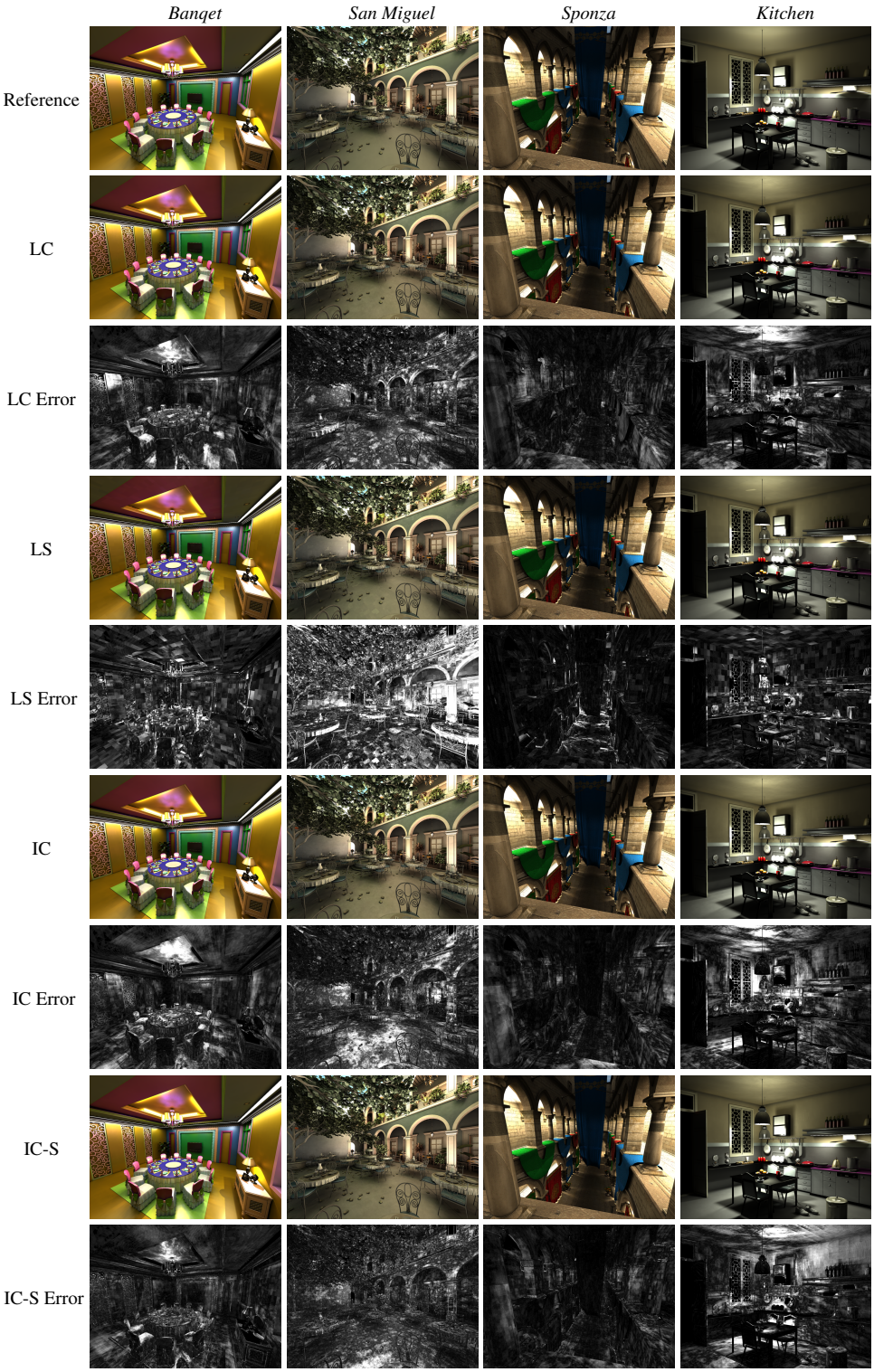


Table 2: *The images rendered with the* 4 *methods (LC,LS,IC,IC-S) with error images for* 1600×1200 *resolution with* 1 *samples per pixel.*

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

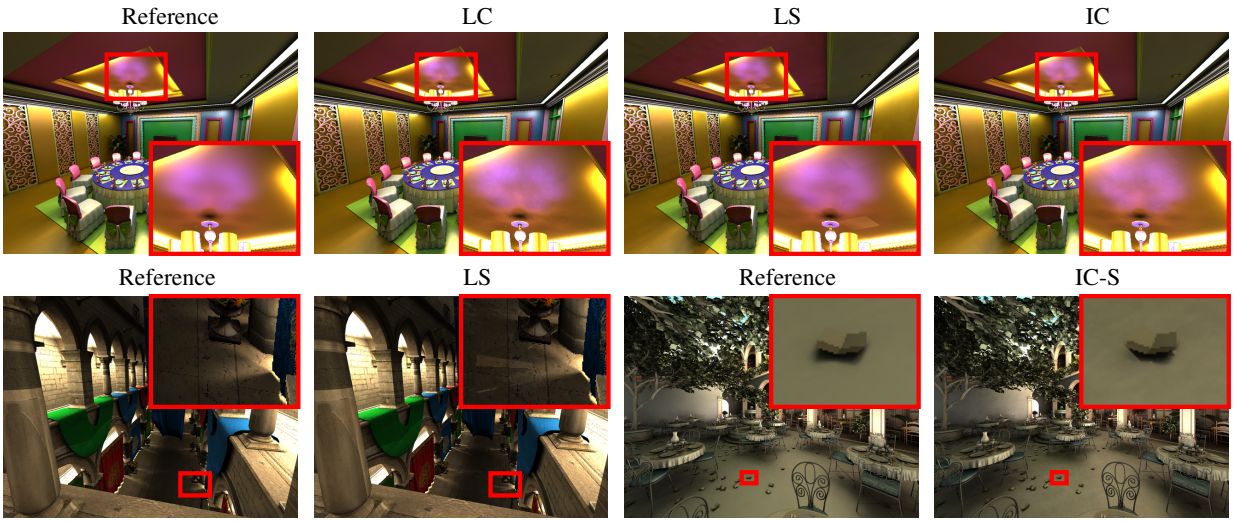


Figure 3: *Typical errors of the four methods (LC, LS, IC, IC-S). The first row shows that LC and IC both fail to reproduce smooth color gradients in complex illumination (the pink color is the result of light reflected on the lamps). LS performs better but it introduces blocking artifacts. Sponza shows that if within a point-cluster the radiance of points changes drastically LS has no means to adapt to it. San Miguel demonstrates that IC-S might fail to detect small shadows.*

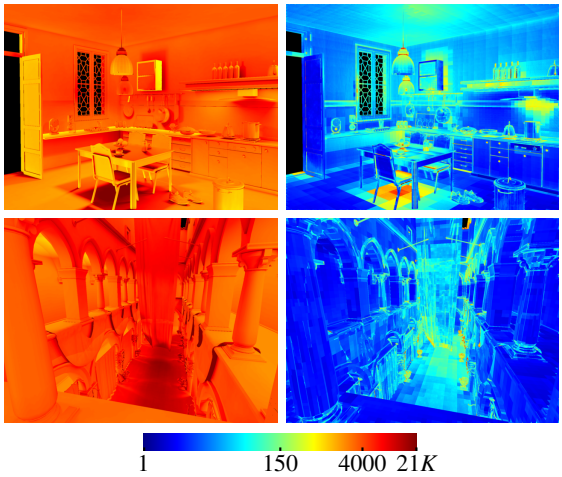


Figure 4: *The logarithm of the number of upper bound calculations per point for Kitchen and Sponza for LC (left) and IC (right). IC amortizes the cost of upper bounding calculations very efficiently but suffers from descending too deep in the tree for dark areas, just as Lightcuts.*

Comparison to LightSlice. LightSlice performs less efficiently in our highly glossy environments. Its ability to explore the structure of VPLs and to adapt to occluded lights reduces the number of rays traced. But this comes with a cost. Clustering the reduced matrix becomes the bottleneck and clustering the points into slices causes blocking artifacts, especially on glossy surfaces. The method fails to handle complex lighting situations, since using randomly sampled representatives easily miss important details. Consider, e.g., *Sponza*, where it is unable to calculate a good shading for the floor in a reasonable time while the other parts of the image are very close to the reference.

High fidelity images. We have set our error threshold for Lightcuts to 1% which closely matches the parameters used in previous work [WFA*05, OP11]. Despite that, the quality of our images are not matching the reference. In order to show that our methods are capable of producing nearly indistinguishable results we have set more strict error thresholds. See the result for *Banquet* in Table 3. Note that we still maintain our speed up over Lightcuts while the quality is the same. LightSlice converges very slowly to the correct solution if one only varies the number of columns. On the other hand, using more slices requires prohibitively large memory.

	LC(0.1%)	LS(6400)	IC(0.15%)	IC-S(0.1%)
R. time (s)	602.98	720.46	180.57	178.83
RMSE	0.00315	0.01490	0.00319	0.00425
Rel. Err.	1.12375	2.37685	1.08088	1.18171

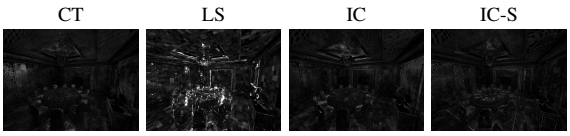


Table 3: *Results for Banquet where the images are visually indistinguishable from the reference. For concision, we only include times for rendering and error images.*

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

Maximum cut size. The original Lightcuts method sets the maximum cut size to 2000. We now present the results of Lightcuts with this additional constraint. For *Banquet* the results remained basically unchanged. For *San Miguel* the rendering time becomes similar to our method but the quality is significantly worse. For *Sponza* this results in degradation of quality while for *Kitchen* the quality remains unchanged. We believe that maximum cut size is effective mainly if one can properly set it prior to rendering a scene. In Table 4 we included results with setting the maximum cut size to 2000.

	Banquet	San Mig.	Sponza	Kitchen
R. time (s)	103.88	175.54	99.34	82.47
RMSE	0.01066	0.02548	0.00632	0.01581
Rel. Err.	3.04835	8.95983	5.16506	10.10555

Table 4: *Lightcuts with maximum cut size set to* 2000.

Speed up. The ray tracing kernel used in our implementation is more highly optimized than the renderer (see [WWB*14] for details on the ray tracing kernel). Therefore the speed up achieved by the method is less if these two components are similarly efficient. In order to have a more objective comparison we have measured the proportion of different components in our implementation of Lightcuts. Approximately 20% is spent on ray tracing, 60% on upper bounding computations and the remaining 20% is spent on shading and heap maintenance. The upper bounding calculations in our method are only a fraction of Lightcuts', therefore we achieve on average 3 times speedup over Lightcuts.

Memory. See Table 5 for the peak memory consumption (in GB) of the four algorithms LC(1%), LS(800), IC(1.5%), IC-S(1%) run on *Banquet* with 1 sample per pixel and 1600×1200 resolution. Lightcuts is the most efficient on memory consumption, followed by our method. For the latter the point tree consumes most memory, scaling linearly with the number of points (e.g., 9 samples per pixels would require extra 8 GB of memory). We note that in order to alleviate the memory consumption one could partition the tree into a few subtrees and process them independently. LightSlice, due to the light transport matrix storage (the size of it is the number of slices times the number of VPLs), has a very high memory consumption.

# VPLs:	50K	300K	600K	1.2M
LC(1%)	0.31	0.62	1.02	1.74
LS(800)	2.95	14.18	29.53	55.10
IC(1%)	1.35	1.65	2.24	2.70

Table 5: *Peak memory requirements for Banquet (in GB).*

Scalability. In Figure 5 we plot the rendering times of four methods (LC(1%), LS(1600), IC(1.5%), IC-S(1%)) with varying number of VPLs for *Banquet* (1 sample per pixel, 1600×1200 resolution). Our method consistently outperforms Lightcuts and LightSlice. The rendering times of both LC and IC are sub-linear in the number of VPLs. For LS, above a certain number of VPLs, the construction and clustering of the reduced light transport matrix becomes the dominant cost, therefore scaling approximately linearly.

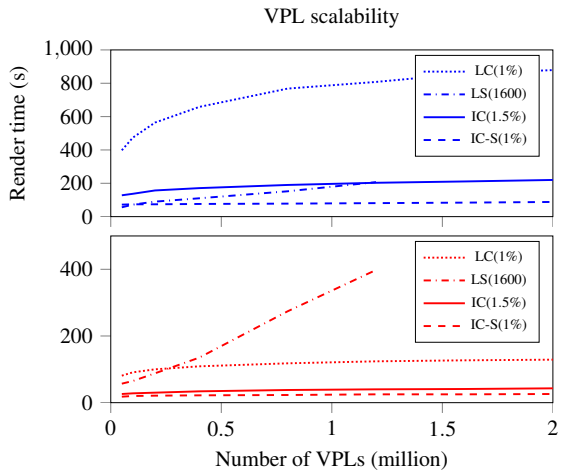


Figure 5: *Render times with varying number of VPLs for San Miguel (top) and for Banquet (bottom).*

Anti-aliasing. We have found that increasing the number of samples increases the quality of the images rendered with LightSlice. Given a VPL-cluster, the method uses different representative lights of the light cluster for shading the different samples in a pixel. See Table 6 for the results of the methods with 9 samples per pixel. This technique smooths the errors, thus enables LightSlice to achieve better quality with only 800 columns, improving its rendering time significantly. For *Kitchen*, LightSlice now outperforms our algorithm but it is still not well-suited for highly glossy environments (e.g., *Banquet*) and challenging illumination (e.g., *Sponza*). In these cases IlluminationCut-Sampling performs better. Our method can be further enhanced for the case of multiple samples per pixel, in a similar way as Multidimensional Lightcuts, by limiting the number of traced rays and shadings. Namely, in each illumination-aware pair use only a single representative for the points originating from the same pixel. This would result in fewer visibility queries but likely increase the error on surfaces with non-uniform textures.

5. Limitations and future work

We have presented an implementation of a flexible and efficient framework handling highly glossy materials. It is several times faster than Lightcuts and has similar speedup as LightSlice while guaranteeing low perceptual error which can be set a priori to rendering.

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

The usage of an octree for the point tree can cause blocking artifacts, however with low error threshold these disappear. The approximate image used in our algorithm has errors and it is not progressively refined (contrary to Lightcuts), thus it might introduce errors in Phase II. This effect, however, is unnoticeable since even a 100% error in the approximate image only results in at most 1% additional error (per cluster) in the final image. IlluminationCut reduces calculations by exploiting the similarity of points, therefore it is less efficient for scenes where the shaded points have highly varying properties (e.g., heterogeneous BRDFs or spatial incoherence). Our current implementation requires the BRDFs to belong to a family with a low number of parameters since otherwise bounding these parameters for nodes of the octree would become prohibitively expensive.

The construction method of the point tree is not important (to some extent) for Algorithm 1 to extract pairs of clusters. Thus one step further would be to utilize a complex metric for the point tree, e.g., incorporating material properties as well. Such a strategy could be useful in a scene with highly varying materials since it would enable tighter error bounds for the individual clusters.

Acknowledgments

We thank the anonymous reviewers for their insightful feedback that helped improve the content and presentation of this paper. We also thank the following people for the models: Guillermo M. Leal Llaguno for the *San Miguel* scene, Marko Dabrovic for the *Sponza* scene, and the authors of the *Kitchen* and *Banquet* scenes.

References

[BD08] BODT T., DUTRÉ P.: Coherent lightcuts. *PhD thesis, Katholieke Universiteit Leuven* (2008). 3

[BELD13] BILLEN N., ENGELEN B., LAGAE A., DUTRÉ P.: Probabilistic visibility evaluation for direct illumination. *Computer Graphics Forum 32(4) (Proceedings of Eurographics Symposium on Rendering 2013) 32*, 4 (2013), 39–47. 3

[Chr08] CHRISTENSEN P. H.: *Point-Based Approximate Color Bleeding*. Tech. rep., Pixar, 2008. 1

[DKH*10] DAVIDOVIČ T., KŘIVÁNEK J., HAŠAN M., SLUSALLEK P., BALA K.: Combining global and local virtual lights for detailed glossy illumination. *ACM Trans. Graph. 29* (December 2010), 143:1–143:8. 3

[DKH*13] DACHSBACHER C., KŘIVÁNEK J., HAŠAN M., ARBREE A., WALTER B., NOVÁK J.: Scalable Realistic Rendering with Many-Light Methods. In *Eurographics 2013 – State of the Art Reports* (Girona, Spain, 2013), Eurographics Association, pp. 23–38. 1, 3

[DKL10] DAMMERTZ H., KELLER A., LENSCH H.: Progressive point-light-based global illumination. *Computer Graphics Forum 29*, 8 (2010), 2504–2515. 3

[Gar82] GARGANTINI I.: An effective way to represent quadtrees. *Communications of the ACM 25*, 12 (Dec. 1982), 905–910. 4

[GKPS12] GEORGIEV I., KŘIVÁNEK J., POPOV S., SLUSALLEK P.: Importance caching for complex illumination. *Computer Graphics Forum 31*, 2 (2012). EUROGRAPHICS 2012. 3

[GS10] GEORGIEV I., SLUSALLEK P.: Simple and Robust Iterative Importance Sampling of Virtual Point Lights. *Proceedings of Eurographics (short papers)* (2010). 3

[HKWB09] HAŠAN M., KŘIVÁNEK J., WALTER B., BALA K.: Virtual spherical lights for many-light rendering of glossy scenes. *ACM Trans. Graph. 28*, 5 (2009), 143:1–143:6. 3

[HPB07] HAŠAN M., PELLACINI F., BALA K.: Matrix row-column sampling for the many-light problem. *ACM Trans. Graph. 26*, 3 (2007). 3

[HREB11] HOLLÄNDER M., RITSCHEL T., EISEMANN E., BOUBEKEUR T.: Manylods: Parallel many-view level-of-detail selection for real-time global illumination. In *Proceedings of the Twenty-second Eurographics Conference on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2011), Eurographics Association, pp. 1233–1240. 3

[Jen01] JENSEN H. W.: *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001. 1

[Kel97] KELLER A.: Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 49–56. 1, 2

[KFB10] KŘIVÁNEK J., FERWERDA J. A., BALA K.: Effects of global illumination approximations on material appearance. *ACM Trans. Graph. 29*, 4 (2010), 112:1–112:10. SIGGRAPH '10. 3

[KHA*12] KŘIVÁNEK J., HAŠAN M., ARBREE A., KELLER C. D. A., WALTER B.: Optimizing realistic rendering with many-light methods. In *SIGGRAPH 2012 Course* (2012). 3

[KK06] KOLLIG T., KELLER A.: Illumination in the presence of weak singularities. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*, Niederreiter H., Talay D., (Eds.). Springer Berlin Heidelberg, 2006, pp. 245–257. 3

[KSW11] KHUNGURN P., SARANURAK T., WATCHAROPAS C.: Pixelcuts: Scalable approximate illumination from many point lights. *Chiang Mai Journal of Science (Special Issue) 38*, 1 (2011), 8–16. 3

[LSK*07] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering* (2007), Eurographics Association, pp. 277–286. 3

[Mik10] MIKSIK M.: Implementing lightcuts. In *CESCG* (2010). 6

[NNDJ12a] NOVÁK J., NOWROUZEZAHRAI D., DACHSBACHER C., JAROSZ W.: Virtual ray lights for rendering scenes with participating media. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 31*, 4 (July 2012). 3

[NNDJ12b] NOVÁK J., NOWROUZEZAHRAI D., DACHSBACHER C., JAROSZ W.: Progressive Virtual Beam Lights. *Computer Graphics Forum 31*, 4 (2012), 1407–1413. 3

[OP11] OU J., PELLACINI F.: Lightslice: matrix slice sampling for the many-lights problem. *ACM Trans. Graph. 30*, 6 (2011), 179. 1, 3, 6, 9

[PGSD13] POPOV S., GEORGIEV I., SLUSALLEK P., DACHSBACHER C.: Adaptive quantization visibility caching. *Computer Graphics Forum 32*, 2 (2013). EUROGRAPHICS 2013. 3

[Rad08] RADAX I.: *Instant Radiosity for Real-Time Global Illumination*. Tech. rep., Institute of Computer Graphics and Algorithms, Vienna University of Technology, 2008. 3

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H.-P., DACHSBACHER C., KAUTZ J.: Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. *ACM Trans. Graph. (Proc. of SIGGRAPH ASIA 2008) 27*, 5 (2008). 3

[SIMP06] SEGOVIA B., IEHL J. C., MITANCHEY R., PÉROCHE B.: Bidirectional instant radiosity. In *EGSR* (2006), pp. 389–398. 3

[VG95] VEACH E., GUIBAS L.: Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*, Sakas G., Müller S., Shirley P., (Eds.), Focus on Computer Graphics. Springer Berlin Heidelberg, 1995, pp. 145–167. 1

[VG97] VEACH E., GUIBAS L. J.: Metropolis light transport. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 65–76. 1

[WABG06] WALTER B., ARBREE A., BALA K., GREENBERG D. P.: Multidimensional lightcuts. In *ACM SIGGRAPH 2006 Papers* (2006), SIGGRAPH '06, pp. 1081–1088. 2, 3, 5

[WC13] WU Y.-T., CHUANG Y.-Y.: Visibilitycluster: Average directional visibility for many-light rendering. *Visualization and Computer Graphics, IEEE Transactions on 19*, 9 (Sept 2013), 1566–1578. 3

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: a scalable approach to illumination. *ACM Trans. Graph. 24*, 3 (July 2005), 1098–1107. 1, 3, 4, 6, 9

[WFWB13] WOOP S., FENG L., WALD I., BENTHIN C.: Embree ray tracing kernels for cpus and the xeon phi architecture. In *ACM SIGGRAPH 2013 Talks* (2013), SIGGRAPH '13, pp. 44:1–44:1. 6

[WKB12] WALTER B., KHUNGURN P., BALA K.: Bidirectional lightcuts. *ACM Trans. Graph. 31*, 4 (July 2012), 59:1–59:11. 3

[WPS*03] WALD I., PURCELL T. J., SCHMITTLER J., BENTHIN C., SLUSALLEK P.: Realtime Ray Tracing and its use for Interactive Global Illumination. In *Eurographics State of the Art Reports* (2003). 6

[WWB*14] WALD I., WOOP S., BENTHIN C., JOHNSON G. S., ERNST M.: Embree: A kernel framework for efficient cpu ray tracing. *ACM Trans. Graph. 33*, 4 (July 2014), 143:1–143:8. 6, 10

[WXW11] WANG G., XIE G., WANG W.: Efficient search of lightcuts by spatial clustering. In *SIGGRAPH Asia 2011 Sketches* (New York, NY, USA, 2011), SA '11, ACM, pp. 26:1–26:2. 3

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

		Banquet	San Miguel	Sponza	Kitchen
LC(1%)	Preproc. time (s)	53.70	92.53	82.36	68.16
	Render time (s)	1031.02	6665.94	2007.74	1522.25
	RMSE	0.010991	0.012007	0.006238	0.012434
	Rel. Error	2.543368	2.681210	3.279278	9.397545
LS(800)	Render time (s)	346.61	399.48	332.62	221.34
	RMSE	0.016470	0.026163	0.005793	0.008920
	Rel. Error	4.010837	6.959285	2.938075	8.923159
	Speedup	3.0	16.7	6.0	6.9
IC(1.5%)	Preproc. time (s)	55.14	95.35	84.90	70.48
	Render time (s)	281.11	1567.65	665.01	360.33
	RMSE	0.008770	0.011267	0.005986	0.011585
	Rel. Error	2.280708	2.772934	2.815512	9.152297
	Speedup	3.7	4.3	3.0	4.2
IC-S(1%)	Preproc. time (s)	55.20	95.21	83.57	70.10
	Render time (s)	117.58	444.25	143.87	170.65
	RMSE	0.008096	0.010349	0.005480	0.012178
	Rel. Error	2.257898	2.686910	2.791644	9.766247
	Speedup	8.8	15.0	14.0	8.9

Reference images

Table 6: *Statistics for images of* 1600×1200 *resolution with* 9 *samples per pixel.*

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.