HAL
open science

Global Illumination Using Well-Separated Pair Decomposition

N. Bus, Nabil Mustafa, Venceslas Biri

To cite this version:

N. Bus, Nabil Mustafa, Venceslas Biri. Global Illumination Using Well-Separated Pair Decomposition. Computer Graphics Forum, 2015, 34 (8), pp.88 - 103. 10.1111/cgf.12610 . hal-01188993v2

HAL Id: hal-01188993

https://hal.science/hal-01188993v2

Submitted on 30 Oct 2017

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.

Global Illumination Using Well-Separated Pair Decomposition – PREPRINT –

N. Bus,[1] N. H. Mustafa[2] and V. Biri[1]
[1]Université Paris Est, LIGM, UMR CNRS, France
[2]Université Paris Est, LIGM, A3SI-ESIEE, France
{busn,mustafan}@esiee.fr, biri@u-pem.fr

Abstract

Instant radiosity methods rely on using a large number of virtual point lights (VPLs) to approximate global illumination. Efficiency considerations require grouping the VPLs into a small number of clusters that are treated as individual lights with respect to each point to be shaded. Two examples of clustering algorithms are Lightcuts [WFA*05] and LightSlice [OP11]. In this work we use the notion of geometric separatedness of point sets as a basis for a data structure for pre-computing and compactly storing a set of candidate VPL clusterings. Our data structure is created prior to rendering, is view-independent and relies only on geometric and radiometric information. For any point to be shaded, we show that a suitable clustering of the VPLs can be efficiently extracted from this data structure. We develop the above framework into an accurate and efficient clustering algorithm based on well-separated pair decompositions which outperforms earlier work in speed and/or quality for diffuse scenes.

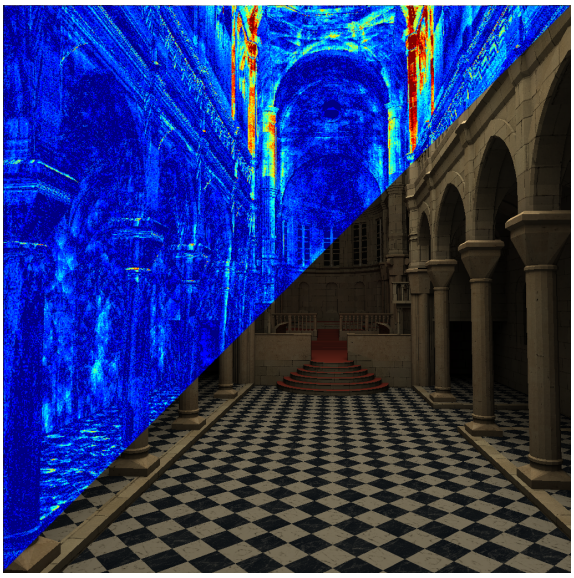


Figure 1: WSPD Clustering for Sibenik cathedral. The bottom half shows the rendered model and the top half the Euclidean error multiplied by 32 with color mapping.

1 Introduction

Several methods have been proposed to solve the global illumination problem, towards the ultimate goal of efficient and realistic rendering of large scenes in the presence of varied and complex lighting effects. Pure unbiased Monte Carlo algorithms such as Metropolis light transport [VG97] or simple bidirectional path tracing [VG95] are considered to be gold standards for reference solutions. However they produce noisy results that are slow to converge in the case of complex scenes. Biased algorithms like Photon Mapping [Jen01], point-based global illumination [Chr08] or the many-lights methods, such as Instant Radiosity [Kel97], provide good performance in many practical applications.

The last method and its improvements have proven to be very useful for approximating global illumination. By tracing light paths from light sources, they create virtual point lights (VPLs) at the intersections of the surface of the scene and the paths. Global illumination is estimated by computing the direct illumination from all of the VPLs (we refer the reader to the SIGGRAPH 2012 course notes on the many-lights problem [KHA*12] and to the EG state-of-the-art report [DKH*13]). Despite some limitations, they provide a unified and scalable approach to the problem of computing global illumination. However, as the number of VPLs needed for a good-fidelity approximation is large, computing the illumination for each point by summing up the contribution of each individual VPL can become prohibitively expensive.

Clustering in the many-lights method. Let $\mathcal{S}$ be the set of VPLs, henceforth considered the set of input lights and $\mathcal{P}$ be the set of points to be shaded, i.e., points in the scene hit by the rays traced for all the pixels. One key idea for speeding up computations is to cluster the VPLs. In other words, for each point $p \in \mathcal{P}$, partition $\mathcal{S}$ into a small number of *clusters* (each partition of $\mathcal{S}$ into clusters is called a *clustering* of $\mathcal{S}$), and then consider each cluster as a single VPL when calculating its contribution to the shading at p.

The goal is to compute, for each $p \in \mathcal{P}$, a clustering that minimizes the shading error at p. See Figure 1 for the image produced by our clustering algorithm as well as the differ-

ence from the reference image. This forces all algorithms to be *adaptive*, i.e., as one iterates over all the points to be shaded, the clustering has to be recomputed again with respect to the spatial and radiometric properties of each point (though various improvements are possible, at a loss of quality, by exploiting spatial coherence to re-use previous computation).

As the set $\mathcal{S}$ of VPLs is view-independent, an idea that improves efficiency is to compute the set of *all possible candidate* clusters of $\mathcal{S}$ *before* the rendering computation (and store in some hierarchical structure). Then during rendering, a clustering for each point is selected by choosing an appropriate subset of clusters from this pre-computed set. This is an expensive task which forces examining a large number of clusters (again, this can be somewhat ameliorated by exploiting the spatial coherence between neighboring pixels). This has been the basis of previous work; two well-known examples are Lightcuts [WFA*05] and LightSlice [OP11].

Our Contributions. We consider our main contributions to be two-fold. First, we make the next logical step in the many-lights clustering paradigm: rather than pre-computing a set of individual candidate clusters from which clusterings are computed during rendering, we pre-compute a number of clusterings of $\mathcal{S}$. Then, during the rendering phase, the clustering for a point p can simply taken to be one of these pre-computed clusterings, together with some minor modification. Our data structure stores the clusterings compactly, it is view-independent and computed prior to rendering. This results in a very efficient rendering phase. For a very natural criterion of clustering, we show that

- the total number of these pre-computed clusterings will be *independent* of the number of points in $\mathcal{P}$ to be shaded and will only depend on the size of $\mathcal{S}$.
- the modification required for each point will be provably small; in fact it will be independent of the size of $\mathcal{S}$ or $\mathcal{P}$.

Second, we develop the above framework into an accurate and efficient new many-lights clustering method. It computes a clustering relying on geometric and radiometric data for fast and accurate computation. We show that the complexity of our scheme is largely invariant on geometric scenes, and it is easily scalable with the number of VPLs.

We prove theoretical guarantees as well as experimentally validate the computational efficiency of our scheme, contrasting it with two of the most well-known earlier systems, Lightcuts [WFA*05] and LightSlice [OP11]. In particular, the advantages of our work include:

- Our pre-computation phase is view-independent, and so are the pre-computed clusterings. Unlike LightSlice and Lightcuts, this allows our algorithm to re-utilize computation with changing camera position.
- As all the clustering computations are moved to the pre-computation phase, the rendering phase takes constant time for each $p \in \mathcal{P}$, i.e., independent of the number of VPLs. On the other hand, Lightcuts has to maintain a heap and do clustering computations. Our method is able to produce similar output as Lightcuts with around 3 times average speedup.
- It outperforms LightSlice in speed and quality, achieving, e.g., 2 times speedup with consistently better quality. The errors of our algorithm are smooth, and visually difficult to detect, unlike for LightSlice which suffers from visible blocking effects.
- It also uses a significantly lower amount of memory than LightSlice which requires around 30 GB for scenes with around 0.5 million VPLs while our algorithm runs with 5 GB. This allows the usage of a considerably higher number of VPLs for rendering.

Broadly our work shows that the set of VPLs itself contain enough information such that with intensive preprocessing, a geometrically good clustering can be constructed for each point $p \in P$ with provably little effort.

Organization. In Section 2 we review previous work on global illumination with the many-lights method. After preliminaries in Section 3, we present the novel ideas involved in our approach in Section 4. Theoretical details on our clustering representation and computation follow in Section 5. Section 6 describes additional structures to further improve the clustering. Extensive experimental results and comparison with previous work are presented in Section 7. Finally, limitations and future work is discussed in Section 8. Appendix 10 contains all the formal proofs of the statements stated in the paper.

2 Previous Work

Many-lights rendering techniques estimate global illumination using VPLs. In Instant radiosity [Kel97], the original many-lights technique, each point is shaded using all VPLs. Since then, many improvements have been made to avoid the limitations such as clamping and diffuse-only global illumination. Further efforts to optimize the computation include carefully placing the VPLs, speeding up visibility computation or selecting a subset of the VPLs to use for each shaded point.

Real-time techniques. Approximate global illumination can be done in real-time with incremental selection of the VPLs [LSK*07], quick shadow computations and deferred shading [RGK*08] or using simple scene descriptors [HREB11]. Clustering VPLs into area lights for real-time GPU based rendering has been proposed in [PKD12]. [SIMP06] uses stochastic sampling of VPLs to achieve interactive frame rates. Our method has different scope from

all these approaches since it is designed to handle significantly larger number of VPLs.

Avoiding limitations. Several limitations of VPL-based algorithms can be addressed, e.g., clamping [KK06] or diffuse only surface BRDFs by using virtual spherical lights [HKWB09] (VSL) or Rich-VPLs [SHD15]. To cope better with highly glossy material [KFB10], Davidovič et al. [DKH*10] use row-column sampling combined with adaptive raycasting. Bidirectional lightcuts [WKB12] or specular gathering [DKL10] combine path tracing and VPL techniques, therefore extending the range of materials and effects. As those algorithms still rely on the classical clustering of VPLs, our technique is complementary to theirs.

VPL generation can also be optimized by sampling VPLs based on their overall image contribution [GS10]. This importance-driven VPL sampling can be further improved [GKPS12] using a sparse set of locations storing probability distributions derived from accurate lighting. Compared to light clustering implementations, sampling introduces unstructured noise. [SIP07] samples the VPLs using a modified Metropolis-Hastings algorithm [VG97].

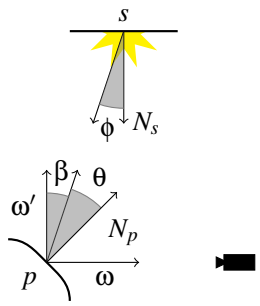
Optimization via clustering. Thousands of VPLs are needed for photo-realistic rendering and therefore, given a point to be shaded, a careful grouping of the VPLs can improve efficiency. This leads to clustering strategies such as Lightcuts [WFA*05] where VPLs are grouped together in light clusters and organized in a tree. This tree is then adaptively searched to extract pertinent VPL clusters for global illumination estimation for each point. This adaptivity works especially well for local lighting but tends to oversample occluded lights. This method has been further generalized by building a tree of the shaded points to further minimize the cost of shading [WABG06, BMB15].

Hašan et al. [HPB07] were the first to study the light transport matrix, where each row of the matrix represents a point to be shaded and each column represents a VPL. After sampling a small set of matrix rows to form a reduced matrix, they cluster the columns according to this matrix. Then each point is shaded using representatives of these clusters. It captures greatly the global lighting but fails to handle very local lighting and suffers from the inherent flaws of the shadow mapping algorithm as observed by Ou and Pellacini [OP11], who further studied the light transport matrix. They built LightSlice, a technique that creates matrix slices using a clustering of image pixels based on their geometric proximity. Then these slices are sampled to create a reduced matrix and compute a global clustering of VPLs. This clustering is then refined for each slice based on the sample of the given slice and its neighboring slices in an attempt to capture the local structure of the matrix. Their solution has a prohibitively big memory consumption and shows block artifacts.

3 Preliminaries

We use similar notation as in Lightcuts [WFA*05]. The radiance for a point $p \in \mathcal{P}$ in direction ω caused by the direct contribution of the lights in $\mathcal{S}$ is denoted by $L(p,\omega)$. It is a function that sums up over all lights the product of the material, geometry, visibility and intensity terms, where each product represents the radiance caused by a single light:

$$L(p,\omega) = \sum_{s \in \mathcal{S}} M_s(p,\omega) \cdot G_s(p) \cdot V_p(s) \cdot I_s \tag{1}$$



Here I_s is the intensity of the light s while the geometric term $G_s(p)$ captures the light attenuation. The VPLs are spotlights having a direction N_s, and therefore $G_s(p) = \cos(\phi)/d(p,s)^2$, where $d(p,s)$ is the Euclidean distance between p and s, and ϕ is the angle between N_s and the vector $p-s$. The material term $M_s(p,\omega)$ is the BRDF which depends on the local geometry and material at p. We will use Lambertian and Phong BRDFs, which only depend on the angles θ and β. The former is the angle between the surface normal N_p and the light direction $s-p$ and the latter is the angle between ω' (the view direction ω reflected on N_p) and the light direction. $V_p(s)$ denotes the visibility of p from the light s.

For a cluster $C \subseteq \mathcal{S}$, let $rep(C)$ denote a representative light $s \in C$. Then the radiance at p from lights in C with representative $rep(C)$ can be approximated as:

$$L_C(p,\omega) = M_{rep(C)}(p,\omega) \cdot G_{rep(C)}(p) \cdot V_p(rep(C)) \cdot \sum_{s \in C} I_s \tag{2}$$

Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ denote a clustering of $\mathcal{S}$ into k clusters. The radiance at P from all the lights in $\mathcal{S}$ can be approximated as:

$$L_{\mathcal{C}}(p,\omega) = \sum_{C \in \mathcal{C}} L_C(p,\omega) \tag{3}$$

4 Our Approach

For a point $p \in \mathcal{P}$ to be shaded, consider a clustering of the set of VPLs $\mathcal{S}$ into k clusters $C_1, \ldots, C_k$ such that the radius of the smallest-ball containing each cluster is much smaller than the distance of that ball to p (this will be formulated precisely in Section 5). Call such a cluster *well-separated*, or a *ws*-cluster for brevity, from p, and the clustering $\{C_1, \ldots, C_k\}$ a *well-separated clustering*, or a *ws*-clustering, for p. Intuitively, from the point of view of

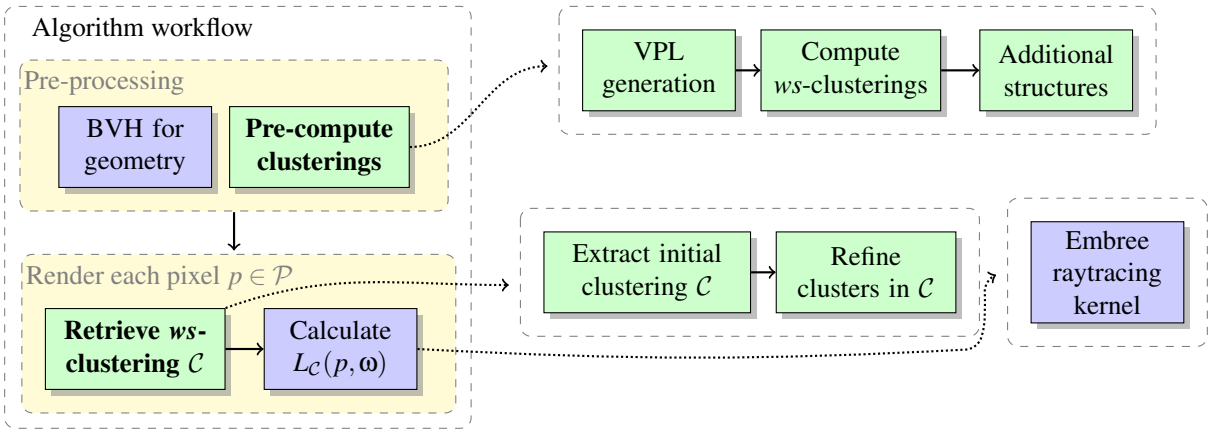


Figure 2: Overview of the system building blocks. The Embree framework (in purple) has been augmented by our algorithms (in green). Details of each block follow in Sections 4, 5 and 6.

p, all the points in each cluster behave roughly like a single point. Figure 3 illustrates this for an arbitrary point p, shaded bright green, a few of the *ws*-clusters of $\mathcal{S}$ around it. Note that as the *ws*-cluster's distance to p decreases, the well-separated criterion automatically ensures that the radius of the cluster decreases as well. Our goal is to compute a *ws*-clustering of $\mathcal{S}$ for each point p to be shaded during the rendering phase.

Our approach has two main components: first we propose a method to pre-compute and compactly store several *ws*-clusterings of $\mathcal{S}$. Then during rendering, we show how to quickly extract a *ws*-clustering for each point to be shaded using this pre-computed structure. This pre-computed clustering will then be slightly modified to fit the spatial properties of the shaded point. We sketch the main components of the system in Figure 2, and outline them below.

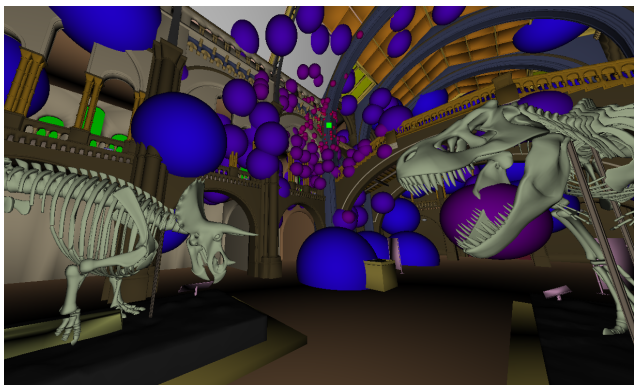


Figure 3: The *Museum* scene with the well-separated clusters (represented by their enclosing spheres) around a shaded point, shown as a bright green square in the middle of the image. For visualization purposes we only included a quarter of the clusters.

Pre-compute clusterings

The computation of *ws*-clustering for each point p is dependent on spatial properties of p, and requires individual computation during rendering for each point, an expensive task. Instead, we will do the following in a view-independent pre-rendering phase: compute a *ws*-clustering of $\mathcal{S}$ with respect to each light $s \in \mathcal{S}$. In other words, for each light s in $\mathcal{S}$, compute the partition of the remaining lights into clusters satisfying the *well-separatedness* criterion. The key to this construction will be the use of a partitioning data-structure, the *well-separated pair decomposition* [CK95] (henceforth denoted as WSPD). These *ws*-clusterings will be stored *implicitly* in a compact structure from which the *ws*-clustering for any light $s \in \mathcal{S}$ can be extracted quickly.

Well-separated clusters approximate the geometric terms of the rendering equation, but ignore visibility and material properties. To adjust for this, we will further compute two additional structures in this pre-processing phase. First, we further group the lights in each *ws*-cluster C into a small number of subgroups by similar light normals. This additional grouping will be used to evaluate the illumination from the cluster more precisely. Second, we introduce representative lights that approximate local visibility for each *ws*-cluster C as follows: sample a number of directions and compute the illumination of the lights in C reaching the boundary of the ball $b(C)$ in the sampled directions, where $b(C)$ is the smallest-ball containing C. This will be used to estimate the visibility of the lights in C to $b(C)$; the visibility test from $b(C)$ to p will be performed during the view-dependent rendering phase.

Retrieve ws-clustering

During the rendering computation, for an arbitrary point p, find the closest point in $\mathcal{S}$ to p (an approximate nearest-neighbor is sufficient and will be used), and start with its (pre-computed) clusters as the clustering for p. Furthermore, refine each cluster by subdividing it into new clusters

until they are well-separated from p. The number of clusters required to achieve this *ws*-clustering criteria could potentially be quite high, for two reasons: *i*) the clustering for the closest point in $\mathcal{S}$ to p could have many clusters, and *ii*) refinement could add many more new clusters to this initial clustering. It will be shown that refinement can only add a *constant* number of new clusters for any point p. This constant is provably *independent* of the number of lights in $\mathcal{S}$ or points in $\mathcal{P}$. Also, under some basic assumptions on the geometry of scenes, we will show that the average size of a *ws*-clustering for points of p will be logarithmic in the size of $\mathcal{S}$. Experimental evidence will confirm this behavior.

Calculate $L(p,\omega)$

Let $\mathcal{C}_p = \{C_1,\ldots,C_k\}$ be the final constructed *ws*-clustering for $p \in P$ during rendering. Furthermore let $\{C_i^1,\ldots,C_i^{k_i}\}$ be the k_i subgroups of each C_i by similar light normals. From a single subgroup C_i^j of $C_i \in \mathcal{C}_p$, we compute:

$$L_{C_i^j}(p,\omega) = M_{rep(C_i^j)}(p,\omega) \cdot G_{rep(C_i^j)}(p) \cdot \sum_{s \in C_i^j} I_s \quad (4)$$

Then we approximate the illumination at p from the cluster $C_i \in \mathcal{C}_p$ as:

$$L_{C_i}(p,\omega) = \left(\sum_{j=1}^{k_i} L_{C_i^j}(p,\omega) \right) \cdot V_p(\partial(b(C_i))) \cdot R(C_i,p) \quad (5)$$

where $V_p(\partial(b(C_i)))$ denotes the visibility from p to the boundary of the sphere $b(C_i)$ enclosing cluster C_i; this is computed by a shadow ray with the Embree raytracing kernel. $R(C_i,p)$ is the proportion of the summed intensity of the lights in C_i reaching the boundary of the ball $b(C_i)$ in the direction from the center of $b(C_i)$ to p; see Section 6 for its precise definition and computation.

5 Constructing *ws*-clusterings

For a cluster $C \subseteq \mathcal{S}$, define $b(C)$ to be the smallest-enclosing ball of the points in C. Let $r(C)$ be the radius of the ball $b(C)$. For any point p, $d(p,C)$ denotes the Euclidean distance of p to $b(C)$.

A well-separated cluster. We introduce a necessary condition that each cluster must satisfy when constructing a clustering of $\mathcal{S}$ w.r.t. a shaded point p – namely that it is *well-separated* from p, for a given parameter $0 \le \varepsilon \le 1$ (ε will be called the *separation parameter*).

Definition 5.1. *A cluster of lights C is well-separated from a point p if $r(C) < \varepsilon \cdot d(p,C)$, where ε is the separation parameter.*

The lights in a *ws*-cluster w.r.t. p are 'far enough' from p, and concentrated in a small ball (see Figure 4). This condition implies that from the point of view of p, all the lights

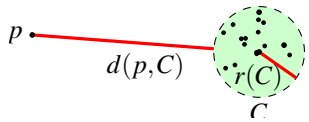


Figure 4: For $\varepsilon = 0.5$, C is *ws* from p. In other words, we have $d(p,C) > 2 \cdot r(C)$.

in a *ws*-cluster are in a similar direction and the distances of p to the lights in C are approximately the same. Since the luminosity reaching p depends on the angle and the distance of lights in $\mathcal{S}$ from p (differences regarding visibility and light normals will be accounted for later in Section 6) it can be argued that treating all the lights in a *ws*-cluster as one point does not introduce significant error. This intuition is captured in the following theorem:

Theorem 5.1. *For a point p and a ws-cluster $C \subseteq \mathcal{S}$, assume that all lights in C face in the same direction and they have the same visibility from p. Then the error from representing C with any one light in C (which has the cumulative intensity summed over all the lights in C) is bounded by a function depending only on ε. In case the point to be shaded has Lambertian BRDF, it is*

$$|L(p,\omega) - L_C(p,\omega)| = O(\varepsilon) \sum_{s \in C} \frac{I_s}{d(p,s)^2} \quad (6)$$

where $L(p,\omega)$ denotes the exact illumination from lights in C.

Proof is in the Appendix.

To compute *ws*-clusterings efficiently, we will need to use a basic structure in the theory of geometric computing, the well-separated pair decomposition.

Well-separated pair decompositions. We first need to extend the notion of well-separatedness between a point and a cluster to that of between two clusters. Two point sets R and Q are *well-separated* from each other if, for a given separation parameter $\varepsilon > 0$, the radius of both the balls $b(R)$ and $b(Q)$ is smaller than $\varepsilon \cdot d(R,Q)$, i.e., $\max(r(R),r(Q)) < \varepsilon \cdot d(R,Q)$ where $d(R,Q)$ is the distance between $b(R)$ and $b(Q)$.

Definition 5.2. *A well-separated pair decomposition of $\mathcal{S}$ for a given separator parameter ε is a list of pairs of clusters $\{\{R_1,Q_1\},\ldots,\{R_s,Q_s\}\}$, where each $R_i,Q_i \subseteq \mathcal{S}$, and*

i) for every pair of points $p,q \in \mathcal{S}$, there is a unique index i such that $p \in R_i$ and $q \in Q_i$, and

ii) for all $i = 1\ldots s$, the clusters R_i and Q_i are well-separated from each other, with separation parameter ε.

Here s is called the size of the WSPD. See Figure 5 for some example pairs $\{R_i,Q_i\}$ for a point set in two dimensions. A remarkable fact about WSPDs is that there

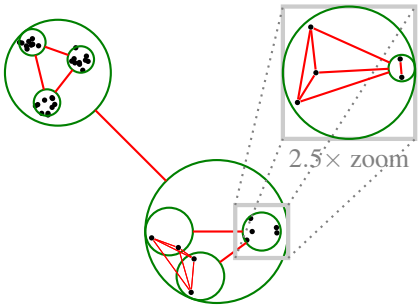


Figure 5: Each red edge represents a pair $\{R_i, Q_i\}$, where the sets R_i, Q_i are enclosed in green circles.

always exist WSPDs of size *linear* in $|\mathcal{S}|$. In particular, for any $\varepsilon > 0$, and any set of points $\mathcal{S} \subset \mathbb{R}^d$, there exists a WSPD of size $O(|\mathcal{S}|\varepsilon^{-d})$ that can be computed in time $O(|\mathcal{S}|\log|\mathcal{S}| + |\mathcal{S}|\varepsilon^{-d})$ [CK95] .

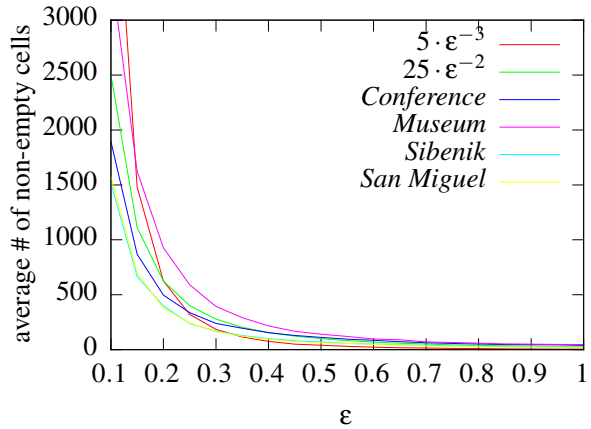


Figure 6: A geometric condition.

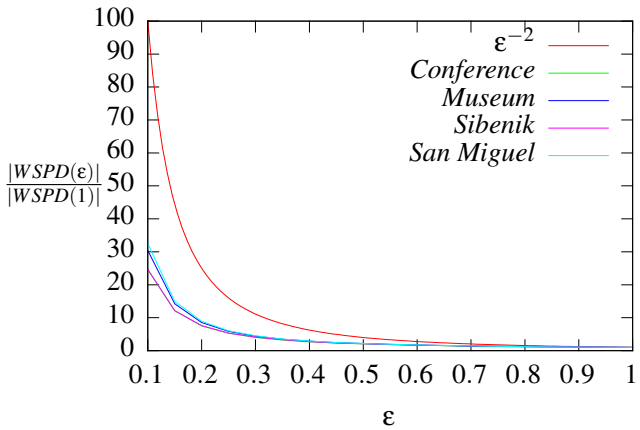


Figure 7: WSPD size ratios for some graphical scenes.

Recall that the set of VPLs $\mathcal{S}$ was placed on the surface of objects by tracing random light particles from the light sources. The geometry of graphics scenes is usually such that it satisfies a technical geometric condition. Namely, that if the three-dimensional space was partitioned into equal-sized cubes of size δ, the boundary of scene objects would intersect on average $O(\varepsilon^{-2})$ cubes in a ball of radius δ/ε (note that there are $\Theta(\frac{(\delta/\varepsilon)^3}{\delta^3}) = \Theta(\varepsilon^{-3})$ cubes in a ball of radius δ/ε). Figure 6 shows that, for a variety of scenes, the average number of cells containing VPLs within a ball of radius δ/ε in our octree behaves more like $O(\varepsilon^{-2})$ than $O(\varepsilon^{-3})$. This implies better than cubic dependency on $1/\varepsilon$ of WSPD sizes in three dimensions, since the proof [CK95] uses a trivial cubic upper bound for non-empty cubes of size δ within a ball of radius δ/ε. Experimental results confirm this: Figure 7 plots, for several scenes, the ratio of the WSPD size for varying separation parameter ε to WSPD size for separation parameter 1. Observe that the behavior of the WSPD is relatively unchanged from one scene to the next.

Constructing well-separated pair decomposition of $\mathcal{S}$. We use a compressed octree of $\mathcal{S}$ as an underlying data structure to compute the initial *ws*-clusters for $\mathcal{S}$, to find approximate nearest neighbors, and for the local refinement for each point p during the rendering computation. The compressed octree is an octree where the non-branching paths are contracted into one edge. The compressed octree can be directly computed in linear time [Sam95]. Each node corresponds to an axis-aligned bounding box. We associate with each node the set of VPLs of $\mathcal{S}$ contained in its bounding box. Note that each leaf of the octree contains exactly one unique point of $\mathcal{S}$. For a node w, denote by R_w the corresponding set of VPLs. Note that the height of the octree is linear in the worst-case, though in practice it is logarithmic. In Table 1 we show the depth of the tree for 300K VPLs, which is logarithmic for a variety of scenes.

	Tree depth	
Scene	**Octree**	**Compressed**
Conference	19	14
Sibenik	20	14
Museum	18	15
San Miguel	22	14

Table 1: Octree depths with $320K$ VPLs, both with and without compression.

After constructing the compressed octree, Algorithm 1 computes the WSPD of $\mathcal{S}$ by utilizing a top down search on the tree for *ws*-pairs. Two nodes w, v of the octree will form a *ws*-pair if the corresponding sets R_w, R_v are well-separated. Note that instead of simply storing all the pairs of the WSPD as a list, we directly store the WSPD structure in the octree by storing, for each node w of the octree, a list of pointers to all the other nodes with which w forms *ws*-pairs in the WSPD. We denote this list by $pairs(w)$. In Figure 8 we show a simple example for 2 dimensional data with the red links denoting the pairs of sets. The construction ensures a hierarchical structure on the clusters which

Algorithm 1 Create WSPD for the set of points $\mathcal{S}$

```
function CreateWSPD(S)
    root ← create compressed octree on S
    S: stack of pairs, S.pushback({root, root})
    while notEmpty(S) do
        {w, v} = S.pop()
        if isWellSeparated(R_w, R_v) then
            Insert v into pairs(w)
            Insert w into pairs(v)
        else
            if r(R_w) < r(R_v) then
                Swap(w, v)
            end if
            for all i ∈ children(w) do
                S.pushback({i, v})
            end for
        end if
    end while
end function
```

enables us to easily refine a cluster if needed by simply descending in the octree.

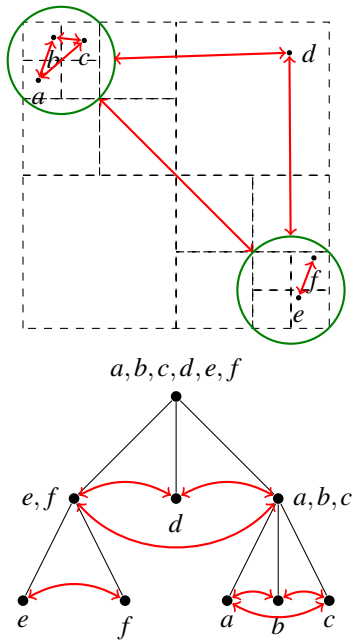


Figure 8: The compressed octree and the WSPD for a set of points.

Pre-computing *ws*-clusterings of $\mathcal{S}$. From the WSPD of $\mathcal{S}$ one can compute a set of clusterings $\{\mathcal{C}_s, s \in \mathcal{S}\}$, where $\mathcal{C}_s$ will be a *ws*-clustering of $\mathcal{S}$ for the point $s \in \mathcal{S}$. Namely, for each $s \in \mathcal{S}$, $\mathcal{C}_s$ is a partition of the lights in $\mathcal{S} \setminus \{s\}$ into a number of clusters, each of which is well-separated from s. Note that if the two sets $\{R, Q\}$ are well-separated, then for every point $p \in R$, Q is a *ws*-cluster with respect to p. The definition of WSPD ensures that for every pair of points p_1 and p_2, there is a unique $\{R, Q\}$ such that $p_1 \in R$ and $p_2 \in Q$. Therefore $\mathcal{C}_s = \{Q \mid \{R, Q\} \text{ is } ws, s \in R\}$ is a *ws*-clustering of $\mathcal{S}$ for s. If the WSPD has been computed, then a *ws*-clustering for each point $s \in \mathcal{S}$ can be extracted from it efficiently, as follows. Consider the leaf node of the compressed octree corresponding to s. Any node w of the octree on the path from this leaf to the root has $s \in R_w$, and so the *ws*-clustering of s is simply the union of $pairs(w)$ for all such nodes w, and can be computed by traversing the octree from this leaf to the root.

Algorithm 2 Constructing *ws*-clusters $\mathcal{C}_p$ for $p \in \mathcal{P}$.

```
function ConstructWSPDClustering(p)
    C_p ← ∅
    s ← closest point in S to p
    C_s ← ws-clustering of s
    for all Q ∈ C_s do
        if d(s, Q) ≥ d(p,s)/ε then
            add Q to C_p
        else
            refine Q into subclusters that are ws from p
            add the resulting clusters to C_p
        end if
    end for
    return C_p
end function
```

Computing *ws*-clusterings of P during rendering. We now show how to use the clusterings $\mathcal{C}_s$, pre-computed for each $s \in \mathcal{S}$ before the rendering phase, to quickly compute a *ws*-clustering of $\mathcal{S}$ with respect to any point $p \in \mathcal{P}$.

Consider the case for an arbitrary point $p \in \mathcal{P}$. Compute the closest light, say $s \in \mathcal{S}$, to p. One could use a variety of known optimal algorithms, but for us an approximation will suffice. We find the smallest node of the compressed octree containing p and return an arbitrary light contained in it. A calculation shows that for randomly shifted octrees, the expected distance from the true nearest neighbor is bounded. Point location in a compressed octree takes $O(\log n)$ time with some additional data structures, but for us the naive implementation suffices as the tree has logarithmic depth (see Table 1).

Say s is at distance d from p. Take the *ws*-clustering $\mathcal{C}_s$ of s. These were pre-computed, and can be efficiently retrieved. The key idea now is to consider two types of clusters in $\mathcal{C}_s$ separately: *far* clusters in $\mathcal{C}_s$ are at distance further than d/ε from s and *close* clusters are those closer than d/ε from s. We show that each far cluster in $\mathcal{C}_s$ is an approximately *ws*-cluster from p. For the remaining close clusters in $\mathcal{C}_s$, we recursively subdivide them until they are *ws*-clusters from p. The subdivision uses the same octree that was used for the construction of the WSPD. See Algorithm 2. Note that the above algorithm is *adaptive* to the local geometry of the scene: for a point p closely surrounded by VPLs, it will refine at a smaller radius.

We now prove that the clusters far from s are approximately *ws*-clusters from p.

Lemma 5.1. *Let p be an arbitrary point and s be its nearest-neighbor with $d(p,s) = d$. Any ws-cluster $C_o \in \mathcal{C}_s$ disjoint with the ball of radius $\frac{d}{\varepsilon}$ around s is approximately well-separated from p, namely it is well-separated with separation parameter $\varepsilon' = \frac{\varepsilon}{1-\varepsilon}$.*

Proof is in the Appendix.

We next prove that the additional number of clusters added in the refinement of a cluster close to s is low.

Theorem 5.2. *Let p be an arbitrary point and s be its nearest neighbor with $d(p,s) = d$. After refining the clusters in the set $\mathcal{C}^* \subseteq \mathcal{C}_s$ which intersect the ball with radius $\frac{d}{\varepsilon}$ around s, there are at most $O(\frac{1}{\varepsilon^6})$ new ws-clusters $\mathcal{C}'$ created. The resulting set $\mathcal{C}_s \backslash \mathcal{C}^* \cup \mathcal{C}'$ is a partition of $\mathcal{S}$ into ws-clusters around p, with separation parameter $\varepsilon/(1-\varepsilon)$.*

Proof is in the Appendix.

While the above theorem may be surprising at first glance, we hope the following intuition sheds some light: consider the distance of the closest point $s \in \mathcal{S}$ to p. If this distance is small, then it is not hard to argue that the clusters for the closest light provide a good approximation of the clustering for p, and so little refinement is necessary. On the other hand, if this closest distance is large, then all points of $\mathcal{S}$ are 'far' from p, and so any *ws*-cluster from s is far from p and thus approximately well-separated from p; again little refinement is needed to approximate the separation (and thus illumination). This intuition is formalized in the proof of the theorem. For empirical validation, see Table 2 for the maximum number of new clusters added per point for a number of scenes with varying values of ε.

	Refined clusters for varying ε				
Scene	0.9	0.7	0.5	0.3	0.1
Conference	58	75	114	227	1401
Sibenik	57	74	109	219	1505
Museum	81	103	159	326	2350
San Miguel	80	108	167	370	3411

Table 2: Maximum number, over all $p \in P$, of added clusters during refinement with $320K$ VPLs.

Cluster sizes. We have proved that the clustering stored in the WSPD can be used to retrieve clustering for arbitrary points without increasing the number of clusters more than an additive constant. It remains to argue that the initial clustering for every light $s \in \mathcal{S}$ is compact. While one can construct examples of arbitrary points where the average number of *ws*-clusters for a point p is linear (as a function of $|\mathcal{S}|$), those are never realized in practice for the set of lights arising in geometric scenes. The spatial partitioning structures (octrees) turn out to be roughly balanced, and so the number of *ws*-clusters is logarithmic.

6 Additional structures for illumination computation

We enhance the purely geometric WSPD based clustering with the following additional structures that improve the efficiency and accuracy in calculating illumination. Recall that given a *ws*-clustering $\mathcal{C}_p$ for p, for a *ws*-cluster $C_i \in \mathcal{C}_p$, the approximation of the radiance with a single representative has the form:

$$M_{rep(C_i)}(p,\omega) \cdot G_{rep(C_i)}(p) \cdot V_p(rep(C_i)) \cdot \sum_{s \in C_i} I_s. \quad (7)$$

Clustering refinement by direction. The WSPD data structure is able to efficiently bound angles and distances between points. However, the normals of the lights could vary widely in directions. To overcome this difficulty, the lights in each *ws*-cluster are grouped into a few subgroups with similar normals. For a *ws*-cluster C_i, construct the subgroups $C_i^1, \ldots, C_i^{k_i}$, where all the lights in each C_i^j, $j = 1 \ldots k_i$, will have similar normal directions. The approximation then becomes

$$L_{C_i}(p,\omega) = \left(\sum_{j=1}^{k_i} L_{C_i^j}(p,\omega) \right) \cdot V_p(rep(C_i)) \quad (8)$$

where $L_{C_i^j}(p,\omega)$ is defined in Equation 4. The subgroups are constructed by first picking a center $c^j \in C_i$ for each subgroup C_i^j, and then assigning each VPL $s \in C_j$ to the subgroups with most similar center. see Algorithm 3. During the shading of a point $p \in \mathcal{P}$, as before, visibility test will still be performed once for each *ws*-cluster of p. However, the radiance $L_{C_i}(p,\omega)$ for a *ws*-cluster C_i will be calculated by summing up the radiance contributions separately for each subgroup C_i^j of C_i, using the normal of the center light for each subgroup. The distance $d(s,c^j)$ used in the algorithm is the Euclidean distance with *threshold* set to 0.01, which did not result in too many subgroups on average (see Table 3). The number of subgroups is slightly higher for more complex scenes and decreases with ε since the clustering becomes more fine.

Algorithm 3 Computing subgroups of a given cluster C_i

```
function ClusterNormals(C_i)
    j ← 1; c^1 ← random light in C_i
    largestDistance ← ∞
    while largestDistance ≥ threshold do
        For each s ∈ C_i, d_s ← min_{k≤j} d(s, c^k)
        c^{j+1} ← q, where q has largest d_q value
        j ← j + 1
        largestDistance ← d_q
    end while
    For each l, C_i^l ← {s ∈ C_i | l = arg min_{k<j} d(s, c^k)}
end function
```

Scene	Average number of subgroups			
ε	0.9	0.7	0.5	0.3
Conference	8.3	8.1	6.7	5.3
Sibenik	9.5	9.3	8.0	5.9
Museum	23.3	21.2	16.8	11.4
San Miguel	44.72	39.6	31.2	18.5

Table 3: Average number of subgroups per cluster.

Visibility testing. Visibility differences within a cluster can cause errors with a fixed representative light. Consider for example a flat object with VPLs on both sides. Since the representative light is either on one side or the other, visibility queries falsely return occlusion if the pixel is on the other side of the object. To overcome this problem we propose to augment our *ws*-clusters with additional visibility information.

As stated in Theorem 5.1, without taking into account visibility differences, a *ws*-clustering with a single representative and clustered normals gives a good approximation to L (as a function of ε). The visibility computation for a *ws*-cluster C_i will be divided into two parts: a simple shadow test from p to the boundary of the ball of C_i for the outside visibility and then shadow testing from the boundary to each light for visibility inside the ball of C_i. Here again the geometric well-separated property of the *ws*-clustering comes in useful, as the angles from p to C_i are bounded (as a function of ε). We use a new approximation for $L(p,\omega)$, to better handle visibility:

$$L_{C_i}(p,\omega) = \left(\sum_{j=1}^{k_i} L_{C_i^j}(p,\omega)\right) \cdot V_p(\partial(b(C_i))) \cdot R(C_i,p) \quad (9)$$

$R(C_i,p)$ is the proportion of the summed intensity of the lights in C_i reaching the boundary of the ball $b(C_i)$ in the direction of p from the center of $b(C_i)$.

$$R(C_i,p) = \frac{\sum_{s\in C_i} V_p(s,\partial(b(C_i))) \cdot I_s \cdot \cos\phi_s}{\sum_{s\in C_i} I_s \cdot \cos\phi_s} \quad (10)$$

where $V_p(s,\partial(b(C_i)))$ denotes the visibility from s to the boundary of the sphere in the direction from the center of $b(C_i)$ to the shaded point p and ϕ_s is the angle between the same direction and the normal of the light. Note that as $\varepsilon \to 0$, equation 9 converges to $L(p,\omega)$. Since computing $R(C_i,p)$ during rendering would be expensive, we do the following in the pre-processing phase. For each cluster C_i and for a small uniform set of directions on $\partial b(C_i)$, $R(C_i,p)$ is pre-computed and stored in a cubemap (with a resolution of 6×6 on each side). This enables quick lookup of $R(C_i,p)$ for p. During the rendering of a point p, nearest-neighbor interpolation on the cubemap yields $R(C_i,p)$. A shadow test to $\partial b(C_i)$ gives the V_p term.

High intensity clusters. To further minimize the error coming from a badly chosen representative for high intensity clusters we limit the radiance from each cluster (by further refining the cluster if necessary) to be less than 1% of the radiance received by a pixel. This refinement happens at the pre-processing phase only using approximate intensities between the cluster pairs.

7 Results and discussion

In this section we present the experimental results on several scenes of varying complexity. Timings are for a workstation equipped with two Xeon X5570 processors each with 4 cores running at 2.93GHz and with 32 GB of memory. We compare our algorithm with two well-known methods: Lightcuts [WFA*05] and LightSlice [OP11]. The authors of LightSlice have made their code publicly available, which also includes an implementation of Lightcuts. In order to do fair comparisons between all three methods, we have ported their implementation of LightSlice (and Lightcuts) into the ray-tracing system INTEL EMBREE [WFWB13] without modifying the core of the algorithms. Our code is also written to use Embree as its ray-tracing engine.

Unless otherwise stated, we run LightSlice and Lightcuts with similar parameters as used in [OP11]: Lightcuts error bound is set to 1% and unbounded the maximum cut size. In order to compare our method to Lightcuts with the parameters set as in [WFA*05], we include results with 2% error and maximum cut size set to 2000. We use the version with 1% error threshold as the reference for equal quality comparisons. LightSlice is run with approximately 1400 slices and 400 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 per point). For our method, the user is free to set the separation parameter ε. This parameter closely tracks both theoretically and practically the quality of the resulting image. In general, setting ε to 0.5 gives a good compromise between quality and speed.

One inherent disadvantage of the many-lights technique is the presence of certain artifacts due to the overly-high contribution of some VPLs to their neighboring points. The standard way to avoid these problems is by *clamping*: limiting the contribution of any one VPL within some small Euclidean distance by applying a clamping threshold. The bias introduced by this technique requires the image rendered with all VPLs to be used as our reference image and not the path traced one. Our experiments use 1 and 4 samples per pixel to compare the quality of clustering obtained by the different methods.

Scenes. We test the algorithms on standard collection of scenes with only moderately glossy materials. The *Museum* has specular materials like the bones of the dinosaur and the canvas on its stand. Most of the primary lights are facing the ceiling to ensure that the scene is mainly lit by indirect illumination. In *Sibenik*, the light sources are facing upwards

		Museum	Sibenik	Conference	San Miguel
Scenes	Triangles	1.5M	0.07M	0.33M	10.5M
	Resolution	1024x1024@1	1024x1024@1	1024x1024@1	1024x1024@1
	VPLs	472K	540K	516K	400K
Lightcuts(1%) max cut: ∞	Preproc. time (s)	0.38	0.47	0.44	0.29
	Render time (s)	**515.06**	**216.29**	**219.43**	**1583.63**
	Avg # of rays	1872	1399	957	3226
	RMSE	**0.004674**	**0.003094**	**0.002220**	**0.005828**
	LMSE	0.000735	0.001124	0.003228	0.003134
	Rel. Error(%)	1.123480	1.094558	1.273182	2.617376
	Speedup	1.0	1.0	1.0	1.0
Lightcuts (2%) max cut: 2000	Preproc. time (s)	0.38	0.47	0.43	0.29
	Render time (s)	218.81	103.31	125.98	306.39
	Avg # of rays	884	724	556	714
	RMSE	0.006223	0.004401	0.003030	0.021054
	LMSE	0.001155	0.001599	0.002794	0.019569
	Rel. Error(%)	1.727831	1.860882	1.689145	8.528274
	Speedup	2.3	2.1	1.8	5.2
Lightslice	Preproc. time (s)	0.02	0.02	0.02	0.01
	Render time (s)	208.17	225.12	202.44	106.94
	Avg # of rays	665	795	750	463
	RMSE	0.009606	0.006820	0.005657	0.034085
	LMSE	0.009220	0.005171	0.009392	0.236342
	Rel. Error(%)	3.273127	3.342225	4.164747	13.855263
	Speedup	2.5	1.0	1.1	14.8
WSPD 0.7	Preproc. time (s)	52.00	50.01	40.52	35.10
	Render time (s)	62.93	31.10	35.24	86.91
	Avg # of rays	711	532	642	715
	RMSE	0.007049	0.004366	0.003515	0.014771
	LMSE	0.002025	0.002145	0.003384	0.028564
	Rel. Error(%)	2.416694	2.148956	2.592963	7.539747
	Speedup	8.2	7.0	6.2	18.2
WSPD 0.5	Preproc. time (s)	55.88	52.86	44.47	38.73
	Render time (s)	86.72	37.85	42.33	115.87
	Avg # of rays	901	635	665	915
	RMSE	0.005750	0.004254	0.002858	0.013161
	LMSE	0.001642	0.002147	0.003961	0.023084
	Rel. Error(%)	1.923812	1.826188	2.180026	6.760511
	Speedup	5.9	5.7	5.2	13.7
WSPD ε (equal RMSE compared to Lightcuts(1%))	Preproc. time (s)	76.86	64.31	60.41	232.91
	Render time (s)	**190.27**	**68.00**	**80.28**	**950.88**
	Avg # of rays	1913	1183	1318	6972
	RMSE	**0.004652**	**0.002833**	**0.002219**	**0.005864**
	LMSE	0.001040	0.001822	0.002624	0.004080
	Rel. Error(%)	1.439894	1.261353	1.597909	2.886980
	Speedup	2.7	3.2	2.7	1.6
	ε	0.25	0.25	0.25	0.09

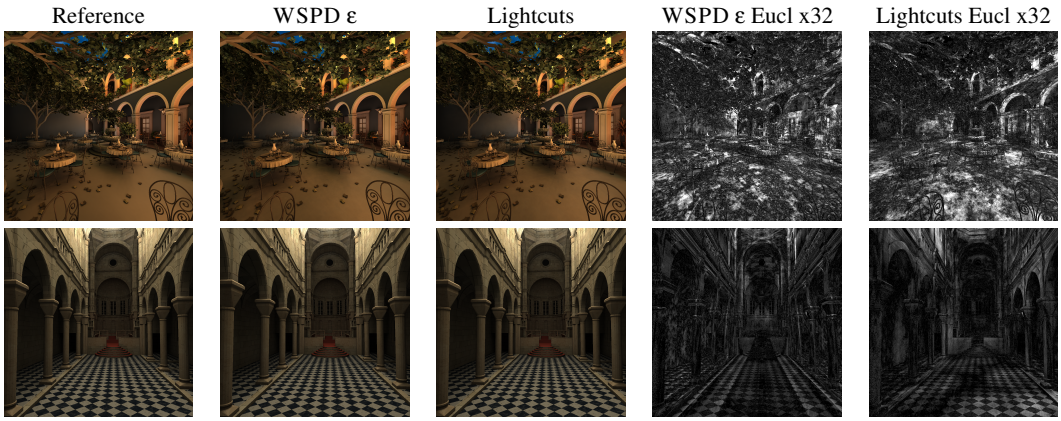


Table 4: Rendering statistics and images for the three different methods, Lightcuts, LightSlice and the WSPD algorithm.

in the dome such that the scene is again lit by indirect illumination, and is made up of purely diffuse materials. The big uniformly colored surfaces in the *Conference* are challenging since the clustering methods have to be spatially consistent, with moderately shiny materials. The outdoor scene *San Miguel* is our largest scene consisting of 10M triangles lit by an environment map of sunset. This is our most challenging scene since the area under the tree and in the corridors is mostly lit by indirect illumination with lots of smooth shadows.

Performance. The images are rendered at a 1024×1024 resolution with 1 sample per pixel (*spp*) and with approximately 500K VPLs. We provide the running times for the pre-processing and the rendering phase along with the average number of shadow rays per pixel.

We provide three different error metrics. Denote by $F(x,y,c)$ the value of a color channel c in the image at coordinate (x,y) and by $\hat{F}(x,y,c)$ the same value in the reference image. The number of pixels multiplied by the number of color channels is $m = 3 \cdot 1024 \cdot 1024$ and each value of F and $\hat{F}$ is between 0 and 1.

- The normalized *RMSE* provides numerical difference against the VPL reference image:

$$\text{RMSE} = \sqrt{\sum \frac{(F(x,y,c) - \hat{F}(x,y,c))^2}{m}}$$

  where the summation is over all pixels and color channels of the images.

- The *LMSE* represents the average squared difference of the gradients between the rendered image and the reference [SPA07]:

$$\text{LMSE} = \frac{\sum(\nabla F(x,y,c) - \nabla \hat{F}(x,y,c))^2}{\sum \nabla F(x,y,c)}$$

  where $\nabla F(x,y,c) = F(x+1,y,c) + F(x-1,y,c) + F(x,y+1,c) + F(x,y-1,c) - 4F(x,y,c)$. A high *LMSE* error implies sharp discontinuities (e.g., sharp error edges), identifying more noticeable errors.

- Average relative error is given by:

$$\text{Rel. Error} = \frac{100}{m} \cdot \sum \frac{|F(x,y,c) - \hat{F}(x,y,c)|}{\hat{F}(x,y,c)}.$$

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 32.

Table 4 shows the results with all these statistics. In general, we find that with $\varepsilon = 0.25$, the quality of our results is similar to Lightcuts with around $3\times$ speedup (the last row in Table 4). The average number of shadow rays for WSPD can be larger than that of Lightcuts; however, as proved earlier, almost no other computation except visibility testing is done by the WSPD algorithm. The WSPD method solely relies on the pre-computed pairs which results in a shorter rendering time since there is no additional work done. Lightcuts, on the other hand, has to descend the tree, maintaining an expensive heap data structure during this traversal. The cost of calculating upper bounds during rendering is also significant. Bounding the maximum cut size can result in significant loss of quality unless the ideal cut size is known a priori (e.g., as in *San Miguel*). Considering this run as a reference for equal quality comparison, our method with $\varepsilon = 0.5$ still shows similar or even better quality with around 3 times speed-up (e.g., as in *Museum*).

LightSlice is able to explore the structure of VPLs and adapt to it more efficiently than Lightcuts, but dividing the image into slices results in visually disturbing blocking artifacts if the error is not low enough. This is captured by the high errors (especially the LMSE) of LightSlice compared with WSPD ε in all the scenes. The WSPD method locally adjusts the cluster radius based on the well-separated criteria. Thus the errors in the resulting image are smoothly distributed, with visually minimal artifacts. The value of the parameter ε closely tracks the quality; for scenes with complex shadows like *San Miguel*, ε has to be set lower (0.1) for comparable quality to Lightcuts. LightSlice relies also on a fixed parameter (number of columns, set to 400). For *San Miguel* it is unable to adapt to the complexity of the shadows with such a low number of columns, resulting in faster running times with high errors.

Scalability. See Table 5 for the total memory (GB) used by the three methods. Lightcuts is the most efficient on memory consumption, followed by WSPD. LightSlice, due to the light transport matrix storage, has prohibitively high memory consumption.

# VPLs:	75K	115K	200K	375K	776K
Lightcuts	0.59	0.6	0.63	0.74	0.84
LightSlice	4.53	6.31	9.85	18.81	36.58
WSPD 0.5	1.14	1.40	1.98	3.38	6.17

Table 5: Memory requirements for the *Museum* scene (GB).

In Figure 9 we plot the rendering times of the three methods with varying number of VPLs in the Museum scene. Note how our algorithm consistently outperforms Lightcuts even with 3M VPLs and both of them scale sub-linearly in the number of lights. LightSlice is only usable as long as it does not allocate more memory than the system has.

Trade-off: In Figure 10 we plot the relative error and the rendering time against ε in the *Museum* scene. The curve is not strictly monotonic since our algorithm is not deterministic (e.g., approximate nearest neighbor).

Blocking artifacts. In contrast to LightSlice, our method and Lightcuts currently do not take advantage of using different representative lights for each sample per pixel. We have found that increasing the number of samples increases

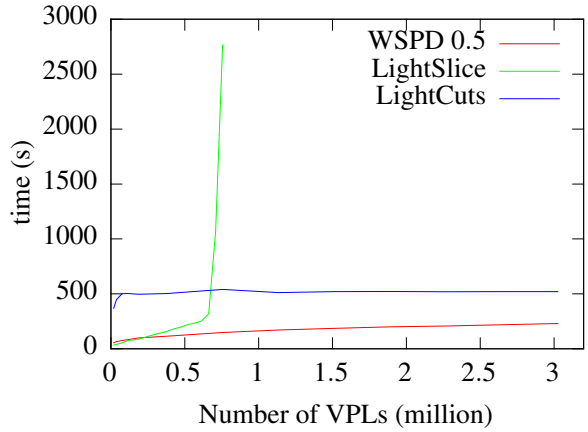


Figure 9: Varying number of VPLs for the *Museum* scene.

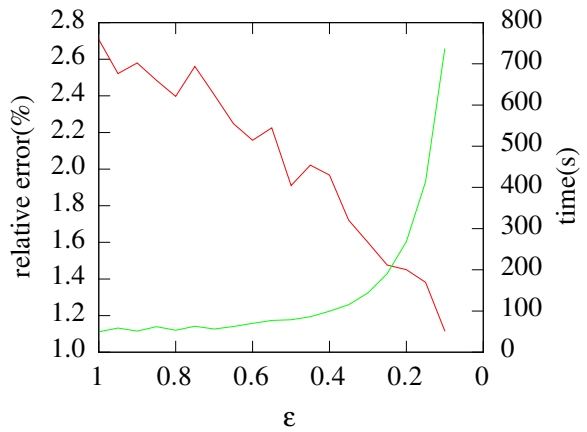


Figure 10: Relative error (red) and render time (green) with varying ε for the *Museum* scene.

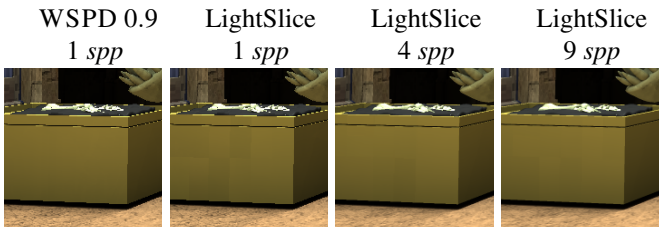


Figure 11: Part of the *Museum* scene rendered with different values of samples per pixel.

the quality of the images rendered with LightSlice. Table 6 shows some results with 4 samples per pixel. Note that our algorithm still matches LightSlice in performance. See Figure 11 for the effect of multiple samples per pixel on the *Museum* scene. Our method has no visible artifacts (nor does Lightcuts) even with $\varepsilon = 0.9$ and 1 sample per pixel. On the other hand LightSlice has visible blocks on the image due to the clustering of the shaded points. These errors can be reduced by increasing the number of columns for LightSlice but this results in a higher running time. If one increases the number of samples then LightSlice becomes competitive, although some artifacts remain even with 9 samples per pixel.

Refinement. This method ensures that the final clustering used for a point satisfies our theoretical criteria. However, with a high number of VPLs the approximate nearest neighbor search is very accurate therefore our refinement method has less importance. If the density of the VPLs is low in an area its usage becomes more important in order to avoid blocking artifacts. We demonstrate this in Figure 12, by rendering the scene with and without refinement.

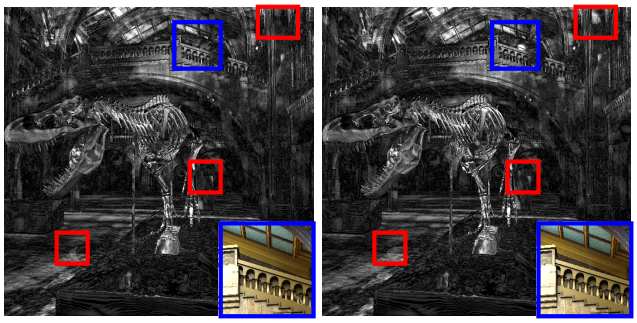


Figure 12: Error images for the Museum with 10K VPLS, rendered with refinement (left) and without refinement (right), for $\varepsilon = 0.25$.

8 Limitation and future work

We have showed that the WSPD structure is suitable for compactly storing clustering information and for providing fast extraction of clusters during render time. We showed theoretical bounds on the error for diffuse materials. The data structure proposed in this paper gives a new perspective on how to efficiently store and retrieve a view-independent clustering for scenes. This framework is very flexible and leaves several possibilities for improving and enhancing the current solution.

One of the main limitations of our method is that it is suited towards diffuse surfaces, and the quality decreases with highly glossy surfaces. The upper-bound proved in Theorem 5.1 increases as a function of glossiness, and so higher glossiness requires smaller values of ε in the WSPD construction for similar error upper-bounds. One can compensate for it with decreasing ε, at the loss of efficiency. In order to demonstrate this, we have replaced the Phong BRDFs in the *Conference* scene with Blinn microfacet BRDFs. This BRDF results in a very significant contribution from the direction of the reflected view ray. See Figure 13 for the results with 1 sample per pixel. The most prominent error is around the shadows of the chairs on the highly glossy floor where the exponent for the Blinn microfacet BRDF is set to 100. Lightcuts can more efficiently adapt to highly glossy materials than our method.

A similar problem is the need to use a small ε for scenes with highly varying visibility properties since in this case the number of clusters increases globally without only refining the clustering where it is necessary. To overcome these

		Museum	Conference
Scenes	Triangles	1.5M	0.33M
	Resolution	1024x1024@4	1024x1024@4
	VPLs	474K	516K
Lightcuts	Preproc. time (s)	0.48	0.44
	Render time (s)	2080.97	876.96
	Avg # of rays	7544	3828
	RMSE	0.003374	0.002119
	LMSE	0.001008	0.003789
	Rel. Error(%)	0.905316	1.208336
	Speedup	1.0	1.0
Lightslice	Preproc. time (s)	0.01	0.02
	Render time (s)	**344.73**	**262.89**
	Avg # of rays	1696	1792
	RMSE	**0.005662**	**0.003025**
	LMSE	0.005431	0.005524
	Rel. Error(%)	1.836033	2.328994
	Speedup	6.0	3.3
WSPD 0.7	Preproc. time (s)	63.47	40.27
	Render time (s)	265.25	158.72
	Avg # of rays	2660	2528
	RMSE	0.006984	0.003378
	LMSE	0.002270	0.004733
	Rel. Error(%)	2.318444	2.425038
	Speedup	7.8	5.5
WSPD 0.5	Preproc. time (s)	68.38	43.25
	Render time (s)	**356.20**	**165.58**
	Avg # of rays	3728	2588
	RMSE	**0.005661**	**0.003029**
	LMSE	0.001876	0.005271
	Rel. Error(%)	1.809967	2.099148
	Speedup	5.8	5.3
WSPD 0.3	Preproc. time (s)	84.68	56.42
	Render time (s)	646.19	260.38
	Avg # of rays	6228	4192
	RMSE	0.005458	0.002664
	LMSE	0.001481	0.004486
	Rel. Error(%)	1.563184	1.717990
	Speedup	3.2	3.4

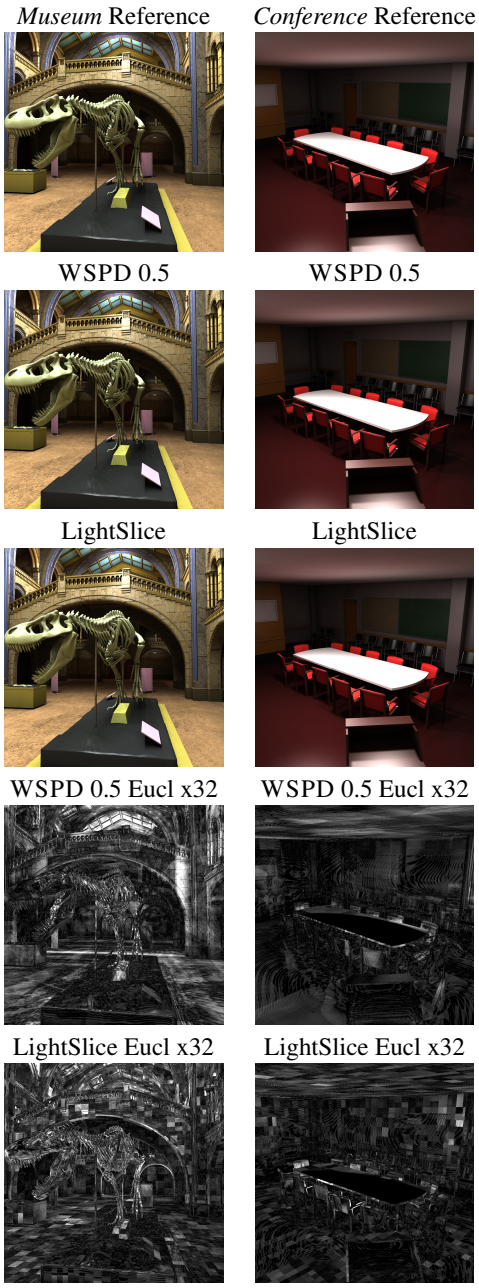


Table 6: Rendering statistics and images for 4 samples per pixel.

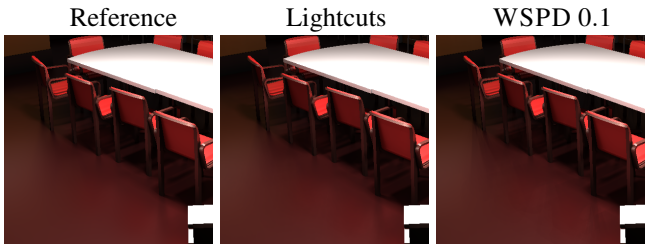


Figure 13: Part of the Conference scene with highly glossy floor using 1 sample per pixel.

limitations one might consider the following improvements:

- Currently our definition of *well-separatedness* is purely geometric, without taking into account the visibility properties of the clusters. It would be interesting if the pre-processing phase can use visibility queries to adaptatively guide the construction of cluster pairs.
- One of the strengths of our approach is that the pre-processing phase already computes pairs of clusters, which are then modified in a limited manner during the rendering phase. Furthermore this computation is view-independent. This opens up the possibility of re-using this computation with changing camera position,

towards animations, interactive rendering, or even the final goal of real-time global illumination.

9 Acknowledgments

We thank the following people for the models. Greg Ward for the Conference scene, Alvaro Luna Bautista and Joel Andersdon for the Museum scene, Guillermo M. Leal Llaguno for the San Miguel scene and Marko Dabrovic for the Sibenik Cathedral scene.

References

[BMB15] BUS N., MUSTAFA N. H., BIRI V.: IlluminationCut. *Computer Graphics Forum (Proceedings of Eurographics 2015) 34*, 2 (2015), 561–573.

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

[CK95] CALLAHAN P. B., KOSARAJU S. R.: A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM 42*, 1 (Jan. 1995), 67–90.

[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.

[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.

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

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

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

[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.

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

[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.

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

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

[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.

[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).

[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.

[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.

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

[PKD12] PRUTKIN R., KAPLANYAN A., DACHSBACHER C.: Reflective shadow map clustering for real-time global illumination. In *Eurographics (Short Papers)* (2012), Andújar C., Puppo E., (Eds.), Eurographics Association, pp. 9–12.

[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).

[Sam95] SAMET H.: Spatial data structures. In *Modern Database Systems*. 1995, pp. 361–385.

[SHD15] SIMON F., HANIKA J., DACHSBACHER C.: Rich-VPLs for improving the versatility of many-light methods. *Computer Graphics Forum (Proceedings of Eurographics 2015) 34*, 2 (2015), 575–584.

[SIMP06] SEGOVIA B., IEHL J. C., MITANCHEY R., PÉROCHE B.: Bidirectional instant radiosity. pp. 389–398.

[SIP07] SEGOVIA B., IEHL J.-C., PÉROCHE B.: Metropolis Instant Radiosity. *Computer Graphics Forum 26*, 3 (Sept. 2007), 425–434.

[SPA07] SILVA E. A., PANETTA K., AGAIAN S. S.: Quantifying image similarity using measure of enhancement by entropy. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* (May 2007), vol. 6579.

[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.

[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.

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

[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.

[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.

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

10 Appendix

10.1 Proofs

In this section we give the proofs of Theorem 5.1, 5.2 and Lemma 5.1.

Theorem 5.1. First we show ε-dependent bounds of the changes of angles and distances between p and the lights in a *ws*-cluster C. See Figure 14 for the interaction of p with two lights s, r in C where r denotes the light chosen as the representative of the cluster.

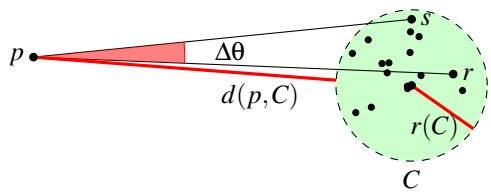


Figure 14: Changes in the angles and distances are bounded by a function of ε.

Estimating the change in the distances follows immediately from the *ws* property:

$$d(p,s) \leq d(p,r) + 2r(C) \leq d(p,r) + 2\varepsilon d(p,s) \tag{11}$$

For bounding the change of angles $\Delta\theta$, it is clear that the angle between the two tangents of the circle from p is the upper bound. The center of the circle, p and the point on tangency form a right angle triangle:

$$\sin\frac{\Delta\theta}{2} = \frac{r(C)}{d(p,C)+r(C)} \leq \frac{\varepsilon d(p,C)}{d(p,C)} \leq \varepsilon \tag{12}$$

Hence using the approximation for the *sine* function with Taylor series we have $\Delta\theta = O(\varepsilon)$.

In the following we approximate the error for one cluster using the previous results. We assume that the visibility is always 1 (the case of 0 is trivial). Therefore the error is:

$$|L(p,\omega) - L_C(p,\omega)| = \tag{13}$$

$$= |\sum_{s\in C} V_p(s) I_s M_s(p,\omega) G_s(p) - V_p(r) M_r(p,\omega) G_r(p) \sum_{s\in C} I_s| \tag{14}$$

$$= \sum_{s\in C} I_s \, |M_s(p,\omega)G_s(p) - M_r(p,\omega)G_r(p)| \tag{15}$$

The above formula has the important property that the M_s and G_s functions are dependent on the cosines of angles and distances which are closely bounded because of the *ws* property. This result intuitively means that the error cannot be too big for a BRDF that relies on distances and angles. In the case of the diffuse BRDF, $M_s(p,\omega) = k_d(p)\cos\theta_s$ where $k_d(p)$ is the diffuse reflection coefficient and θ_s is the angle between the surface normal at p and $s-p$. Denote by ϕ_s

the angle between the light normal and $p-s$ and for brevity denote $d(p,s)$ by r_s, then the formula becomes:

$$\sum_{s\in C} I_s \left| \frac{k_d(p)\cos\theta_s\cos\phi_s}{r_s^2} - \frac{k_d(p)\cos\theta_r\cos\phi_r}{r_r^2} \right| = \quad (16)$$

$$k_d(p)\sum_{s\in C} I_s \left| \frac{\cos\theta_s\cos\phi_s}{r_s^2} - \frac{\cos(\theta_s+\Delta\theta_s)\cos(\phi_s+\Delta\phi_s)}{(r_s+\Delta r_s)^2} \right| \quad (17)$$

We omit a complete analysis and just minimize the subtrahend assuming that $\Delta\theta, \Delta\phi, \Delta r \geq 0$ (the other cases are similar). Using the *ws* property and that $\cos(\theta+\Delta\theta) \geq \cos\theta - \Delta\theta$,

$$\leq k_d(p)\sum_{s\in C} I_s \left(\frac{\cos\theta_s\cos\phi_s}{r_s^2} - \frac{(\cos\theta_s-\Delta\theta_s)(\cos\phi_s-\Delta\phi_s)}{(1+2\varepsilon)^2 r_s^2} \right) \quad (18)$$

Using that $\cos x \leq 1$ and the bound on $\Delta\theta$

$$\leq k_d(p)\sum_{s\in C} I_s \left(\frac{(4\varepsilon+4\varepsilon^2+\Delta\theta_s+\Delta\phi_s-\Delta\theta_s\Delta\phi_s)}{(1+2\varepsilon)^2 r_s^2} \right) \quad (19)$$

$$= O(\varepsilon)k_d(p)\sum_{s\in C}\frac{I_s}{r_s^2} \quad (20)$$

In other words, the error is proportional to the complete intensity received by a pixel. Note that this does not account for visibility difference of course, which we address with other methods. □

Remark: In the case of Phong BRDF the $M_s(p,\omega)$ function becomes $k_s(p)cos^n\beta_s\cos\theta_s$ where β_s is the angle between ω and the direction of $s-p$ reflected on the surface normal (the subscript of k now refers to the word specular). In this case the above calculations are as follows:

$$\sum_{s\in C} I_s \left| \frac{k_s(p)\cos^n\beta_s\cos\theta_s\cos\phi_s}{r_s^2} - \frac{k_s(p)\cos^n\beta_r\cos\theta_r\cos\phi_r}{r_r^2} \right| \quad (21)$$

$$= k_s(p)\sum_{s\in C} I_s \left| \frac{\cos^n\beta_s\cos\theta_s\cos\phi_s}{r_s^2} - \frac{\cos^n(\beta_s+\Delta\beta_s)\cos(\theta_s+\Delta\theta_s)\cos(\phi_s+\Delta\phi_s)}{(r_s+\Delta r_s)^2} \right| \quad (22)$$

$$\leq k_s(p)\sum_{s\in C} I_s \left(\frac{\cos^n\beta_s\cos\theta_s\cos\phi_s}{r_s^2} - \frac{(\cos\beta_s-\Delta\beta_s)^n(\cos\theta_s-\Delta\theta_s)(\cos\phi_s-\Delta\phi_s)}{(1+2\varepsilon)^2 r_s^2} \right) \quad (23)$$

One could upper bound this error but the subtrahend converges to 0 as $n\to\infty$ hence in this case one could only give a weak bound which depends on n as well. If we were to set ε according to such a bound one would have to build a too fine WSPD which would result in a prohibitively big runtime. We remark that this could be possibly overcome by refining the WSPD with a smaller epsilon during the rendering phase but only for those clusters that have a high value for $M_r(p,\omega)$. However we have not experimented with this approach.

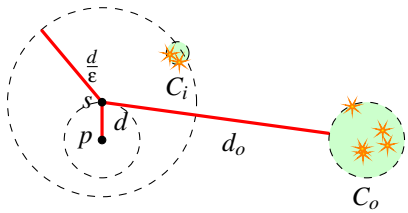


Figure 15: A close *ws*-cluster C_i and a far *ws*-cluster C_o from s.

Lemma 5.1. See Figure 15. As C_o is *ws* from s, it follows that $\varepsilon d_0 \geq r(C_o)$, where $r(C_o)$ is the radius of the cluster C_o. Also $d \leq \varepsilon d_o$ since the cluster is disjoint. Triangle inequality implies $d(s,C_o) = d_o \leq d + d(p,C_o)$, and so $d(p,C_o) \geq d_o - d \geq d_o(1-\varepsilon) \geq \frac{r(C_o)(1-\varepsilon)}{\varepsilon}$. Thus for the slightly larger value of the separation parameter $\varepsilon' = \frac{\varepsilon}{1-\varepsilon}$, C_o is *ws* from p. □

Theorem 5.2. The second statement of the proof comes from Lemma 5.1. Take a cluster $C_i \in \mathcal{C}_s$ lying inside the ball of radius d/ε around s. If it is not *ws*-separated from p, partition the bounding-box of C_i into 8 equal-volume bounding boxes (the children of the node in the octree), and recursively check the *ws*-separated property of these new refined clusters with p. Eventually when a newly refined cluster is finally a *ws*-cluster from p, add it to $\mathcal{C}'$. To count the total number of new *ws*-clusters added to $\mathcal{C}'$, consider a cluster $C \in \mathcal{C}'$. It exists because its parent, say cluster D, was not a *ws*-cluster with p. i.e., $r(D) > \varepsilon d_p$, where d_p is the distance of the ball of D to p. Because s is the nearest neighbor of p we know that no other point is in the small ball of radius d around p; in particular it cannot completely contain the ball of D and hence $r(D)/\varepsilon \geq d_p \geq d - 2r(D)$, which implies that $r(D) \geq \frac{d}{2+\varepsilon^{-1}}$. So each cluster added to $\mathcal{C}'$ has a parent with radius at least the above value. Grouping the parents by size (higher level parents are the same size but multiplied by some power of two) we can give an upper bound on their number by a simple packing argument since parents with the same size are disjoint (since octree nodes can either contain each other or be disjoint):

$$\sum_{i=0}^{\infty}\left(\frac{d/\varepsilon}{2^i(\frac{d}{2+\varepsilon^{-1}})}\right)^3 = O(\frac{1}{\varepsilon^6}) \quad (24)$$

Since we have bounded the number of parents and each of them can have at most 8 children, this finishes the proof. □