Visibility based WSPD for Global Illumination

M. Maria[1], N. Mustafa[1], T. Bardoux[1], J. Defaye[2] and V. Biri[1]
[1]*University of Paris-Est, LIGM, France*
[2]*Ubisoft Motion Pictures, Montreuil, France*

Keywords: Ray Tracing, Many-light, VPL Clustering.

Abstract: In the context of rendering production, and especially for indirect diffuse global illumination, many-light techniques can be used to quickly render noise free images. In this paper, we propose a view-independent algorithm, based on Well Separated Pair Decomposition (WSPD), to handle efficiently clustering of virtual point lights. Our clustering, the Visibility based WSPD (V-WSPD) consider both geometric and visibility information of two sets of points, allowing an improved rendering time with a similar image quality.

1 INTRODUCTION

In computer graphics industry, computing accurately global illumination (GI) in complex scenes, like those illustrated in Figure 1, remains really challenging since the size of the assets and the number of lights are massive. In such productions, path tracing or point based GI methods are generally used to compute the final image (Křivánek et al., 2010). But their rendering engine also computes separated layers of this final image such as glossy reflection/refraction or indirect diffuse lighting. The latter is a particular GI problem that could be handled with Virtual Point Light (VPL) based techniques (Dachsbacher et al., 2014).

Visibility queries between pairs of points remain one of the main cost of all GI algorithms (Dutré et al., 2006). Thus, the knowledge of the overall scene structure is a key to compute indirect illumination more efficiently, especially for a full sequence of images of the same scene. Recent works, for instance Imperfect Shadow Maps (Ritschel et al., 2008), have followed this way. They approximate visibility for indirect illumination by rendering a low-resolution shadow map from a rough point-based representation of the scene geometry. This approximation is then used to achieve real-time GI on GPU.

Defined in the many-light framework, our work is also based on visibility approximation but handles a considerably larger number of VPL, for offline rendering. It focuses on the visibility of two different point sets, created along light paths vertices (such as VPL). We redefine the work of (Bus et al., 2015a) on the Well-Separated Pair Decomposition (WSPD) method. While they use a pure geometric criterion to build their structure, we drive the construction directly with visibility queries, along with the existing geometric factor, while remaining view-independent.

The result is a new data structure, the Visibility based WSPD (V-WSPD), that handles visibility to further reduce the number of shadow rays needed to compute indirect illumination. We show that this additional visibility criterion allows to significantly speed up rendering while keeping a similar image quality.

Therefore, our main contributions can be summarized as follows:

- an efficient data structure, the V-WSPD, optimized for fast rendering with a large number of VPL;
- a view-independent construction algorithm, allowing to reuse the V-WSPD for a full sequence of images in case of static scenes;
- an analysis of the performance scalability *w.r.t.* the visibility approximation accuracy.

This paper is organized as follows: Section 2 reviews the existing many-light rendering techniques related to our approach; Section 3 gives the theoretical background introducing our work; Section 4 details our new data structure, the V-WSPD; Section 5 presents and discusses experimental results; Section 6 concludes and gives several future work lines.

Maria, M., Mustafa, N., Bardoux, T., Defaye, J. and Biri, V.
Visibility based WSPD for Global Illumination.
In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 1: GRAPP*, pages 81-90
ISBN: 978-989-758-287-5

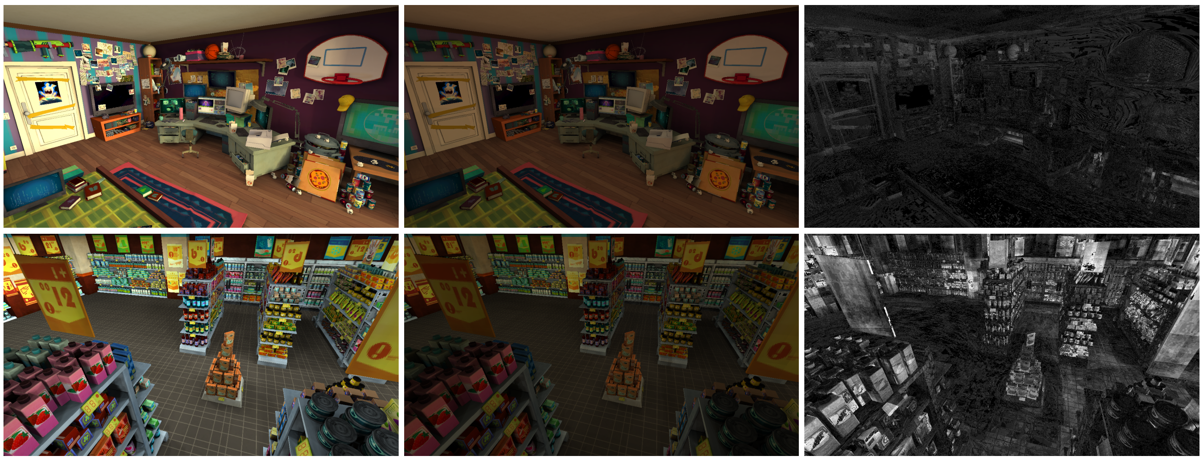


Figure 1: Examples of VPL rendering with the V-WSPD: rendered image, indirect illumination and error $\times$128. The scenes (GEEKROOM and MARKET) come from the TV Series *Rabbids Invasion* of Ubisoft.

2 RELATED WORK

During this last decade, many-light methods have become very attractive since they allow to render high-quality, almost artifact-free, but biased images in a fraction of the time taken by Monte-Carlo methods to converge to a noise-free image. Thus, a lot of recent researches focus on this topic (Simon et al., 2015; Hedman et al., 2016; O'Donnell and Chajdas, 2017)

In this section, we focus on global illumination in many-light methods, especially on techniques designed to compute (or approximate) efficiently the contribution of large amount of VPL. For real-time techniques, improvements on VPL distribution or lighting, and a detailed state-of-the-art report, the reader is referred to (Dachsbacher et al., 2014).

Generating a high-quality image with many-light algorithms usually requires from thousands to millions of VPL. A naive method, such as the original Instant Radiosity (Keller, 1997), which evaluates linearly the contribution of each VPL for each point to shade, would be prohibitively time-consuming to be used in practice. To overcome this problem, several methods, called *scalable* methods, have been proposed to accurately approximate the computation with a sub-linear complexity. We review the most relevant ones next.

2.1 VPL Clustering

Clustering methods aim at partitioning the set of VPL into clusters *w.r.t.* their contribution to the image. The goal is to store important VPL into small clusters and negligible ones into largest clusters. Then, each cluster is represented by a single representative light with adapted intensity. When rendering, for each point to shade, a subset of clusters (a *clustering*) is selected to approximate the radiance. Thus, the number of evaluations becomes drastically smaller than the total amount of VPL and the computation is much faster.

The idea of VPL clustering has been introduced in (Paquette et al., 1998). They build a hierarchical octree on the set of VPL and then extract a clustering of VPL for each point to be shaded. Their method significantly speeds up the process but is not robust, leading to badly chosen clustering. Since then, a lot of methods have been presented. We propose to classify them into two main families: tree-based methods and matrix-based methods.

Tree-based Methods. Lightcuts (Walter et al., 2005) is the first practical VPL clustering method. It is based on a binary tree, called *light tree*, from which a clustering is selected for each point to be shaded. This selection is performed by refining the tree from the root to minimize the error upper bound until it is below a given perceptual threshold. Many extensions of Lightcuts have followed this first work. Multidimensional Lightcuts (MDLC) (Walter et al., 2006) uses a shading point tree to handle distributed effects such as anti-aliasing or motion blur. Bidirectional Lightcuts (Walter et al., 2012) extends MDLC to handle more materials, such as glossy reflections or subsurface scattering. (Wang et al., 2011) aim at improving the clustering selection efficiency by exploiting cluster coherence. IlluminationCut (Bus et al., 2015b) uses two trees to cluster both pixels and VPL and to further minimize the radiance evaluation cost per pixel. (Rehfeld and Dachsbacher, 2016) propose to compute the lightcuts only for a sparse subsampled image and then to interpolate between these lightcuts for all others point to be shaded. As part of GPU computing, (Davidovič et al., 2012) address the problem of adapting Lightcuts to the GPU. Recently, (Bus et al., 2015a) introduced a view-independent VPL clustering method, built on a Well-Separated Pair Decomposition (WSPD) structure. As the structure we propose is based on this work, it is described with more details in Section 4.1.

Matrix-based Methods. Other methods directly study the light transport matrix: rows of the matrix represent the pixels of the image and columns correspond to the VPL so that each row/column combination encodes the contribution of a VPL to a pixel. (Hašan et al., 2007) first introduce the Matrix Row-Column Sampling (MRCS) to generate a single global clustering for the whole image. MRCS has been adapted for animation using tensors and exploiting temporal coherence (Hašan et al., 2008). The main drawback of these methods is that they fail to capture local lighting. (Davidovič et al., 2010) address this problem by capturing separately the global and local lighting and combining them to get the final image. With the same goal, Lightslice (Ou and Pellacini, 2011) partitions the image into slices and then refines the global clustering for each slice. This solution consumes prohibitively too much memory to be used with a large number of VPL. (Huo et al., 2015) use a matrix separation technique to reduce the number of shadow rays needed. These methods are inherently view-dependent (they have to be performed for each image to render) since they need the knowledge of (some) shading points. In this paper, we propose a structure that is fully view-independent. Thus, it could be computed once, and reused for a full sequence of images.

2.2 VPL Sampling and Caching

Another way to reduce the algorithm complexity is by choosing a relevant subset of VPL (or rejecting negligible ones) *w.r.t.* their contribution to the final image. For instance, Importance Sampling (Georgiev and Slusallek, 2010) rejects VPL while Importance Caching (Georgiev et al., 2012) builds a probability function by computing and recording the contribution of all VPL for a sparsely distributed shading points into caches, called *Importance Records*. Then, for a given point to shade, the closest Importance Records are used to sample the VPL. Later, (Yoshida et al., 2015) extend this work by proposing an adaptive cache insertion to further reduce variance.

Once again these methods are view-dependent and have to be performed for each image to render.

3 THEORETICAL BACKGROUND

Let $\mathcal{S}$ be the set of VPLs, and $\mathcal{P}$ be the set of points to be shaded. The radiance caused by the direct illumination of a single light source $s \in \mathcal{S}$, for a surface point $p \in \mathcal{P}$ observed from the direction ω is denoted $L_s(p,\omega)$ and defined as the product of its material (M), geometric (G), visibility (V) and intensity (I) terms (*cf.* (Walter et al., 2005)):

$$L_s(p,\omega) = M_s(p,\omega) \cdot G_s \cdot V_s(p) \cdot I_s \tag{1}$$

Then, for each point p to be shaded, the total radiance $L_{\mathcal{S}}(p,\omega)$ is computed by summing all light contributions:

$$L_{\mathcal{S}}(p,\omega) = \sum_{s\in\mathcal{S}} L_s(p,\omega) \tag{2}$$

For a single cluster $C \subseteq \mathcal{S}$, let $rep(C) \in C$ be the representative light of C. Then, the radiance can be approximated as:

$$L_C(p,\omega) = M_{rep(C)}(p,\omega) \cdot G_{rep(C)} \cdot V_{rep(C)}(p) \cdot I_{rep(C)} \tag{3}$$

The representative light's intensity is usually precomputed once, as $I_{rep(C)} = \sum_{s\in C} I_s$, and stored within the cluster. Thus, $rep(C)$ has to be chosen carefully to minimize the error caused by $V_{rep(C)}$, $G_{rep(C)}$ and $M_{rep(C)}$ computation.

Then, for a clustering $\zeta = \{C_1,\dots,C_n\}$ of $\mathcal{S}$ the radiance can be approximated as:

$$L_\zeta(p,\omega) = \sum_{C\in\zeta} L_C(p,\omega) \tag{4}$$

Rendering an image means solving Equation 2 for each $p \in \mathcal{P}$. This can be seen as a graph scan problem on the complete bipartite graph $G = (\mathcal{P},\mathcal{S},E_G)$, in which each edge $(p,s) \in E_G$ has a weight induced by the Equation 1. The naive rendering method to solve this problem would be to cast all the $|\mathcal{P}| \cdot |\mathcal{S}|$ rays, a prohibitive number.

To reduce the complexity of this problem, as seen in Section 2, several methods can be employed. For instance, Importance sampling (Georgiev and Slusallek, 2010) rejects some lights of $\mathcal{S}$ according to their contribution on the image while LightSlice (Ou and Pellacini, 2011) clusters both $\mathcal{P}$ and $\mathcal{S}$ to improve rendering time while keeping good image quality.

But, to approximate the problem in a view-independent way, and thus to get the same structure for all images to be rendered, only the $\mathcal{S}$ set can be considered. The goal is to get a clustering of $\mathcal{S}$ for each $s \in S$. Then, the problem can be rephrased as a scan of the complete Euclidean graph $K = (\mathcal{S},E_K)$ on $\mathcal{S}$ and can be trivially solved in $O(|\mathcal{S}|^2)$ since $|E_K| = \Theta(|\mathcal{S}|^2)$. Once again, given the usual huge number of lights in a scene, the graph is too dense to be used in this way.

The Well-Separated Pair Decomposition (WSPD) (Callahan and Kosaraju, 1995) is a fundamental structure in computational geometry, that provides a compact representation of K in $O(|\mathcal{S}|)$ space. In other words, it is a partition of the $\binom{n}{2}$ edges of K into $O(|\mathcal{S}|)$ subsets. Each subset of the partition is represented by two point sets (or clusters) $A, B \subseteq \mathcal{S}$ that are *well-separated*.

Let us denoted $b(C)$ the smallest enclosing ball containing all the points of the cluster C and $r(C)$ its radius. Two clusters A and B are *well-separated* if, for a *separation parameter* $0 < \varepsilon \leq 1$:

$$d(A,B) > \frac{1}{\varepsilon} \cdot max(r(A), r(B)),$$

where $d(A,B)$ is the distance between $b(A)$ and $b(B)$ (*cf.* Figure 2). We say that A and B are *ws*-clusters. From that, we can define a WSPD of a point set $\mathcal{S}$ with separation parameter ε as a set of pairs $W = \{\{A_1, B_1\}, \ldots, \{A_k, B_k\}\}$ such that, $\forall i$:

$A_i, B_i \subset \mathcal{S}; A_i \cap B_i = \emptyset$,

A_i and B_i are well-separated *w.r.t.* ε.

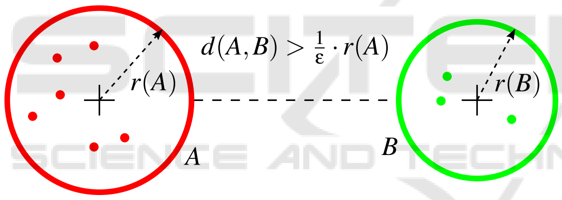


Figure 2: The clusters A and B are well-separated according to the separation parameter ε.

(Bus et al., 2015a) show that the WSPD structure can pre-compute *ws*-clusterings of $\mathcal{S}$ in a view-independent way and be used efficiently during rendering to extract *ws*-clustering for each point to be shaded. This *pure geometric WPSD* approximates only the geometric G and intensity I terms of the Equation 3. In their work, visibility is handled using additional structures. In this paper, we avoid using additional structures to approximate the visibility term V by driving the WSPD construction using visibility queries between *ws*-clusters candidates.

4 VISIBILITY BASED WSPD

In this section we describe our algorithm based on VPL rendering with WSPD. We first recall some basics about WSPD clustering, then we focus on the major improvement taking into account visibility between clusters of points.

4.1 WSPD Clustering Basics

Here we recall some basics about the use of WSPD for VPL rendering. More details can be found on (Bus et al., 2015a).

Construction. The structure construction is summarized in Algorithm 1. From a set of VPLs $\mathcal{S}$, we build a compressed octree in which a node represents an axis-aligned bounding box containing a non-empty subset of $\mathcal{S}$. Each leaf contains one and only one VPL and each node represents a cluster C defined by a single representative light with an adapted intensity. Then, the WSPD is constructed with a top down scan of the tree. In the end, each node A of the octree stores a list of nodes, denoted $pairs(A)$, with whom it forms a pair. In its pure geometric form, the function *checkPair* (*cf.* Alg. 1, line 8) works as follows (*cf.* Figure 3, left): if the two nodes are well-separated then they form a *ws*-pair; else, the process recursively goes on with the children of the biggest node. This function, *checkPair*, is the one we improve, as described in Section 4.2, to take into account visibility.

Algorithm 1: Build a WSPD from a set of point.

```
Require: $\mathcal{S}$: set of points;
 1: $O \leftarrow buildCompressOctree(\mathcal{S})$;
 2: $S$: stack of nodes of $\mathcal{S}$;
 3: $S.push(O.root)$;
 4: while $S.notEmpty()$ do
 5:   $C \leftarrow S.pop()$;
 6:   for $i \in \{1..8\}$ do
 7:     for $j \in \{1..(i-1)\}$ do
 8:       // ......................... cf. Figure 3
 9:       $checkPair(C.child(i), C.child(j))$;
10:     end for
11:     $S.push(C.child(i))$;
12:   end for
13: end while
```

Precomputation of Clusterings of $\mathcal{S}$. The goal is to precompute, for each $s \in \mathcal{S}$, a clustering $\zeta_s = \{C_1, \ldots, C_n\}$ of $\mathcal{S} \backslash \{s\}$. In fact, these clusterings are implicitly encoded in the WSPD structure. Given that each node contains the list of its well-separated clusters, ζ_s corresponds to the union of the pairs of all nodes from the leaf containing s to the root of the octree.

Extraction of Clusterings of $\mathcal{P}$ during Rendering. The final goal is to get, for any point $p \in \mathcal{P}$, a clustering ζ_p of $\mathcal{S}$. This can be efficiently extracted from the WSPD. First, we search for the (approximated) nearest light s to p, with distance d. Then, from the precomputed clustering ζ_s, ζ_p is constructed as follows:

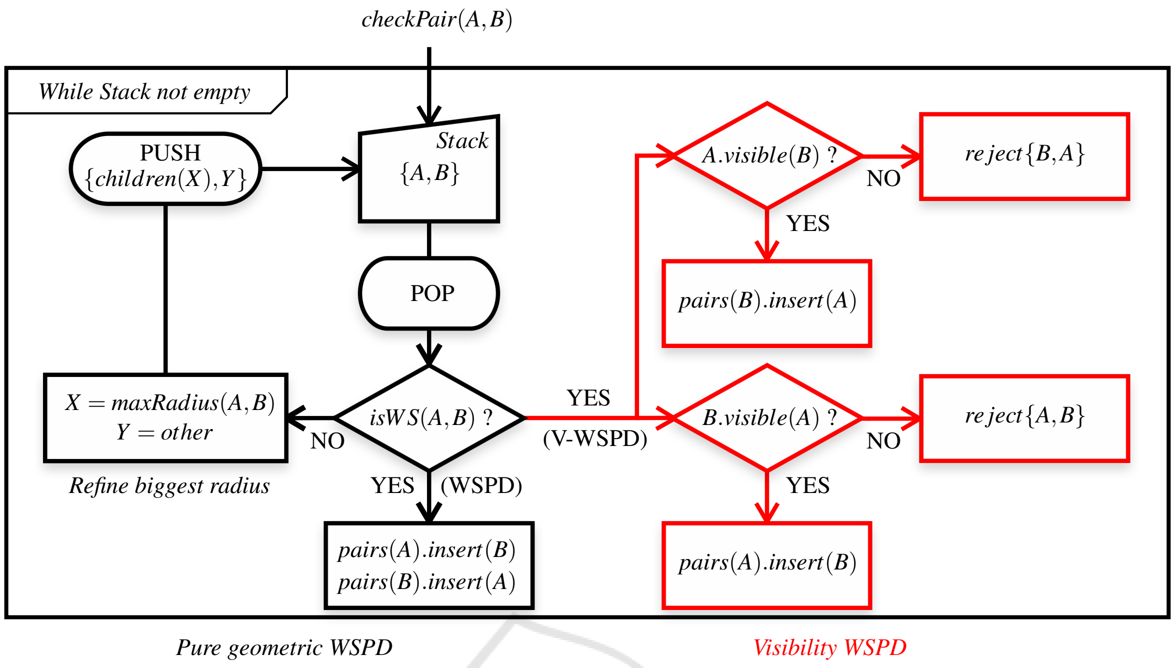


Figure 3: Schematic algorithm for function *checkPair*(A,B) (Algorithm 1, line 8): left, pure geometric WSPD; right, additional process for visibility based WSPD.

$\forall C_i \in \zeta_s$, if $d(s,C_i) \geq d$, add C_i to ζ_p, else repeat with $children(C_i)$.

This pure geometric WSPD approximates only the geometric term of the rendering equation but ignores the others. To answer this issue, (Bus et al., 2015a) use two additional structures. First, they compute sub-groups of lights in each clusters *w.r.t.* their normals. Second, they pre-compute, for each cluster, a small cube map to handle inner visibility inside this cluster. These cube maps are used directly during rendering, along with a single shadow test between p and the cluster C, to estimate the contribution. This approach leads to two drawbacks:

- its construction needs two passes, one for the pure geometric WSPD and another to compute visibility information and build the additional structures;
- the computation of the cube maps for each cluster is memory and time consuming;

Our purpose is to drive directly the WSPD construction using visibility queries in order to have a uniform data structure, easy to implement, and computed in one pass.

4.2 Driving Construction with Visibility

To make the WSPD approximate the visibility term of the rendering equation, we propose a new criterion to decide on pair creation. Our method is illustrated in Figure 3, the new part being in red, on the right. In this section, we explain how we check visibility between nodes and how this information can be used to optionally refine the WSPD. Benefits from this new method are presented and discussed in Section 5.

Visibility Checking. To reduce the number of shadow rays needed to compute indirect illumination, we use visibility queries to reject some pairs in the WSPD.

Let A and B be two nodes, respectively represented by their enclosing ball $b(A)$ and $b(B)$. In the pure geometric WSPD, two nodes form a pair if they are well-separated *i.e.* if, for a separation parameter $0 < \varepsilon \leq 1$, $d(A,B) > \frac{1}{\varepsilon} \cdot max(r(A), r(B))$, where $d(A,B)$ is the distance between $b(A)$ and $b(B)$ and r is the radius. Here, we first check for well-separateness and then pairs are created *w.r.t.* their visibility.

First, we consider A as a *shading* node and B as a *light* node. B will belong to $pairs(A)$ if its representative light $rep(B)$ is potentially visible from any point $p \in A$ (*cf.* Figure 4). We approximate the computation by checking the visibility between $rep(B)$ and some points H sampled on the hemisphere of $b(A)$ facing $rep(B)$. If each $h_i \in H$ is not visible from $rep(B)$, then the pair is rejected. Then, the same process is repeated, considering B as a shading node and A as a light node.

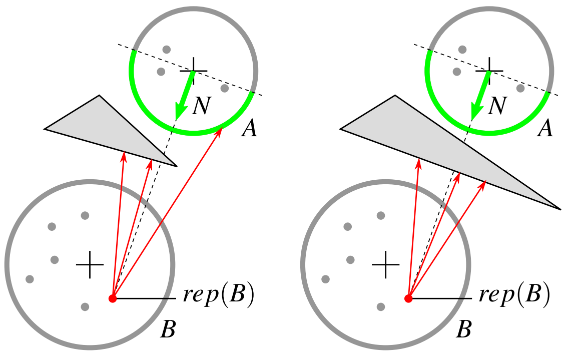


Figure 4: Visibility checking: A is the *shading* node and B the *light* node; left: $rep(B)$ is potentially visible from any point $p \in A$; right: $rep(B)$ is not visible.

We have to process differently the case where the shading node is a leaf. When building the pure geometric WSPD, leaves are considered with no radius. In our case, a shading node represents a subset of $\mathcal{P}$ so that if it is a leaf, we cannot ignore its radius. Thus, for a leaf shading node P and a light node S, the pair is automatically accepted if $rep(S)$ is inside $b(P)$.

The number of samples picked up on the shading node hemisphere (denoted *spv*) is set by the user. The higher *spv* is, the lower the error becomes, but with an increase in precomputation time. We discuss the impact of this parameter in Section 5. Now, we show how this visibility information could be used to refine the WSPD.

Visibility Refinement. The visibility check allows to discard pairs according to their visibility. One could think that these visibility queries could be used to improve the WSPD quality: if the test fails, then refine with shading node children. In practice, experiments reveal that this strategy is not relevant for two reasons.

First, it increases significantly the V-WSPD construction time: for a unique refinement, it requires to perform up to $8\times spv$ additional visibility queries. Moreover, in the worst case, when a shading node is not visible at all, this strategy will conduct to refine the full subtree, without any acceptation.

Second, the improvement on the overall image quality is negligible. Indeed, since the clusters are geometrically coherent due to the well-separation criterion, the probability for a shading node to be visible from a light while its parent has not been accepted is very low.

5 RESULTS AND DISCUSSION

In this section, we present our experimental results on four different scenes, illustrated in Table 2, with number of triangles given in Table 1. The first two, CONFERENCE and MUSEUM are commonly used scenes in computer graphics. The two others, GEEKROOM and MARKET are production scenes used for the TV series *Rabbids Invasion* by Ubisoft. All scenes contain roughly 600k VPL.

Timings are measured on an Intel Xeon E5-2670 running at 2,3Ghz with 32GB of RAM, for 1024×1024 pixels images without anti-aliasing. We use Embree for our ray tracing kernel (Áfra et al., 2016). For WSPD based methods, ε is fixed at 0.25. The influence of this parameter is detailed in (Bus et al., 2015a). We compare the V-WSPD with the pure geometric WSPD and with the state-of-the-art method IlluminationCut (Bus et al., 2015b) without adaptive sampling. We used the implementation provided on the IlluminationCut author's website[1]. We set the maximum error bound to have roughly similar rendering times, and we do not provide any limit for the maximal cut size. Unfortunately, we could not compare with Lightslice (Ou and Pellacini, 2011) since it uses too much memory to be run on our machine.

5.1 Performance

Table 1 shows statistics about preprocess time, indirect illumination rendering time and error averaged from several points of view. We measure the normalized RMSE in LAB color space to compute the numerical difference with the naive VPL reference image for indirect illumination. We choose to compare in LAB color space because it is designed to represent the human eye perception better than RGB. The error images shown in Table 2 are generated by computing the channel-wise Euclidean distance in LAB color space with the reference image and multiplying it by 128 (to be visible). We first evaluate the performance of the V-WSPD by comparing it with the pure geometric WSPD in terms of rendering and construction times. Then we compare it with the state-of-the-art *i.e.* with IlluminationCut.

Rendering. As part of rendering, the V-WSPD allows to reduce computation times by 45% (*cf.* Table 1) while keeping a similar RMSE. This improvement can be explained by analyzing the number of shadow rays shot to compute indirect illumination. Indeed, since the visibility checking rejects a lot of potentially non visible pairs, the number of shadow rays is

[1]`https://busnorbert.bitbucket.io/`

Table 1: Results and comparison for four scenes with 600k VPL: timing and error for naive method (VPL), pure geometric WSPD, visibility based WSPD (V-WSPD) with 5 *spv* and IlluminationCut (IC) (Bus et al., 2015b).

		CONFERENCE	MUSEUM	GEEKROOM	MARKET
		(262.3k *tr.*)	(1.53M *tr.*)	(39.04M *tr.*)	(46.06M *tr.*)
Naive VPL	Render (s)	2316.74	3579.62	3661.81	3845.64
WSPD	Preproc. (s)	2.85	3.65	4.63	6.51
	Render (s)	12.83	19.74	24.24	29.91
	RMSE	0.000450607	0.000695289	0.000602983	0.00189347
	# pairs ($\times 10^6$)	204.38	252.88	269.09	327.06
V-WSPD (5 spv)	Preproc. (s)	12.75	19.61	34.95	63.96
	Render (s)	8.64	11.12	13.68	13.87
	RMSE	0.000462769	0.000712769	0.000604522	0.001998
	# pairs ($\times 10^6$)	96.89	80.82	113.46	100.64
IC	Preproc. (s)	105.94	218.51	249.91	290.49
	Render (s)	8.53	10.86	13.88	13.71
	RMSE	0.000266304	0.000504664	0.00046284	0.00131467
	Err. bound (%)	1	4	1	1.2

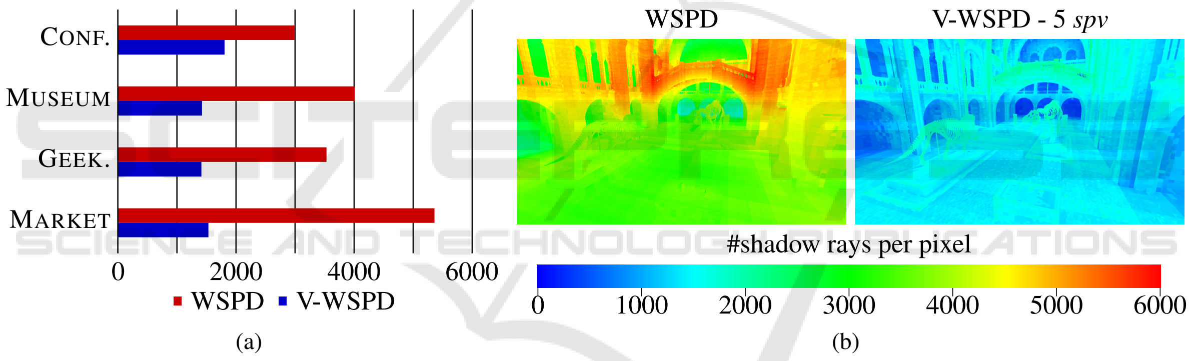


Figure 5: Number of shadow rays for WSPD and V-WSPD: (a) average number of shadow rays per pixel for each scene (same points of view as Table 2); (b) false-colored images of the MUSEUM scene.

highly reduced and rendering performance are higher (*cf.* Figure 5). We can notice that for the MARKET scene, the improvement on rendering times is much better, due to the high level of occlusion brought for instance by the shelves. For a shading node located near the shelves, a lot of pairs are rejected so a lot of shadow rays are avoided. For the same scene, we can see some blocking artifacts in the false-colored image in Table 2, due to the visibility approximation. These artifacts are visible because error is multiplied by 128 but are not perceptible in the final image.

Construction. As expected, V-WSPD construction times are higher than pure geometric WSPD ones. This is due to the visibility queries needed to drive the construction. In this way, the complexity does not only depend on the number of VPL but also on the number of polygons making up the scene: one visibility query requires to scan the geometry in order to determine if there is an intersection. Moreover, the higher the occlusion degree of a scene, the higher the construction time because more visibility queries have to be performed before rejecting a pair.

One should consider that, since the V-WSPD structure is completely view-independent, it has to be constructed only once for a full sequence of images. This is a concrete advantage in case of walk-through system or even to compute static layer in animation production. Indeed, the time lost when constructing the structure is negligible *w.r.t.* the time earned when rendering: for instance, with MARKET scene, construction time is amortized from the fourth image.

Comparison State-of-the-Art. For similar rendering time, we can notice that IlluminationCut generates images with a lower RMSE. In counterpart, preprocessing times are considerably higher (about 7 times). This shows that the way to build the WSPD basis structure is not necessarily the best and that some work has to be done to achieve better quality. Indeed, for now, the space partition used to build the WSPD is just a regular octree and does not rest on a specific metrics as the light tree of IlluminationCut.

5.2 Visibility Checking

As stated in Section 4.2, the number of samples (*spv*) used to approximate the potential visibility between a shading node and a light node can be tweaked to influence both performance and precision of our algorithm. In practice, for a shading cluster P and a light cluster S, the first visibility check corresponds to the direct visibility between $rep(S)$ and $b(P)$. Following points on $b(P)$ are determined with stratified hemisphere sampling. Figure 6 shows the impact of this parameter, *spv*, on construction time, rendering time and image quality, for the MUSEUM scene.

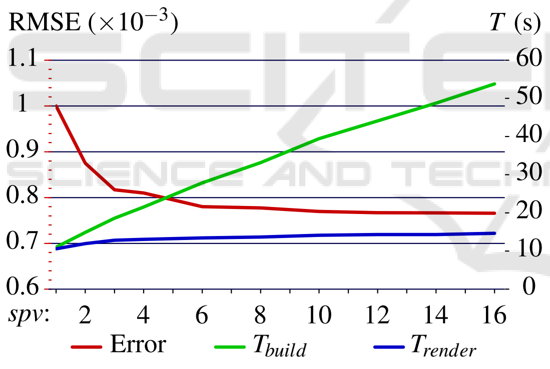


Figure 6: Error (RMSE), construction time (T_{build}) and rendering time (T_{render}) *w.r.t.* number of samples per visibility check (*spv*), for the MUSEUM scene.

As expected, construction time grows linearly with *spv* since more visibility queries are performed. But it does not influence rendering time. Of course the RMSE is improved by the growth of *spv*, but it tends to converge rapidly. Therefore, we noticed, in all our scenes, that a 4 to 6 value for *spv* is a good trade-off between speed and quality.

6 CONCLUSION AND FUTURE WORK

We have presented a new data structure for VPL clustering: the Visibility based WSPD (V-WSPD). It refines the work of (Bus et al., 2015a) on the WSPD by driving the construction with visibility queries (along with the existing geometric factor), while remaining view-independent.

We have shown that the V-WSPD allows to achieve far better rendering times than the pure geometric WSPD while keeping a similar image quality. This improvement implies an increase in construction times, since it requires to cast rays to determine visibility. But, given that the V-WSPD is view-independent, it could be reused for a full sequence of images, so that the loss in construction times is quickly amortized.

Compared to the state-of-the-art method, IlluminationCut (Bus et al., 2015b), our method generates images with higher error even if these errors are not visible to the naked eye (*cf.* table 2). In counterpart, IlluminationCut preprocessing time is far higher. To improve the accuracy of our method, we could change the basis structure of the V-WSPD, trying different partition structures such as a kd-tree (as it has been done in physics for N-body simulation (Lopes et al., 2014)) or a Multi-BVH (Ernst and Greiner, 2008). In that case, these partitions should be constructed cleverly *w.r.t.* VPL distribution and contribution.

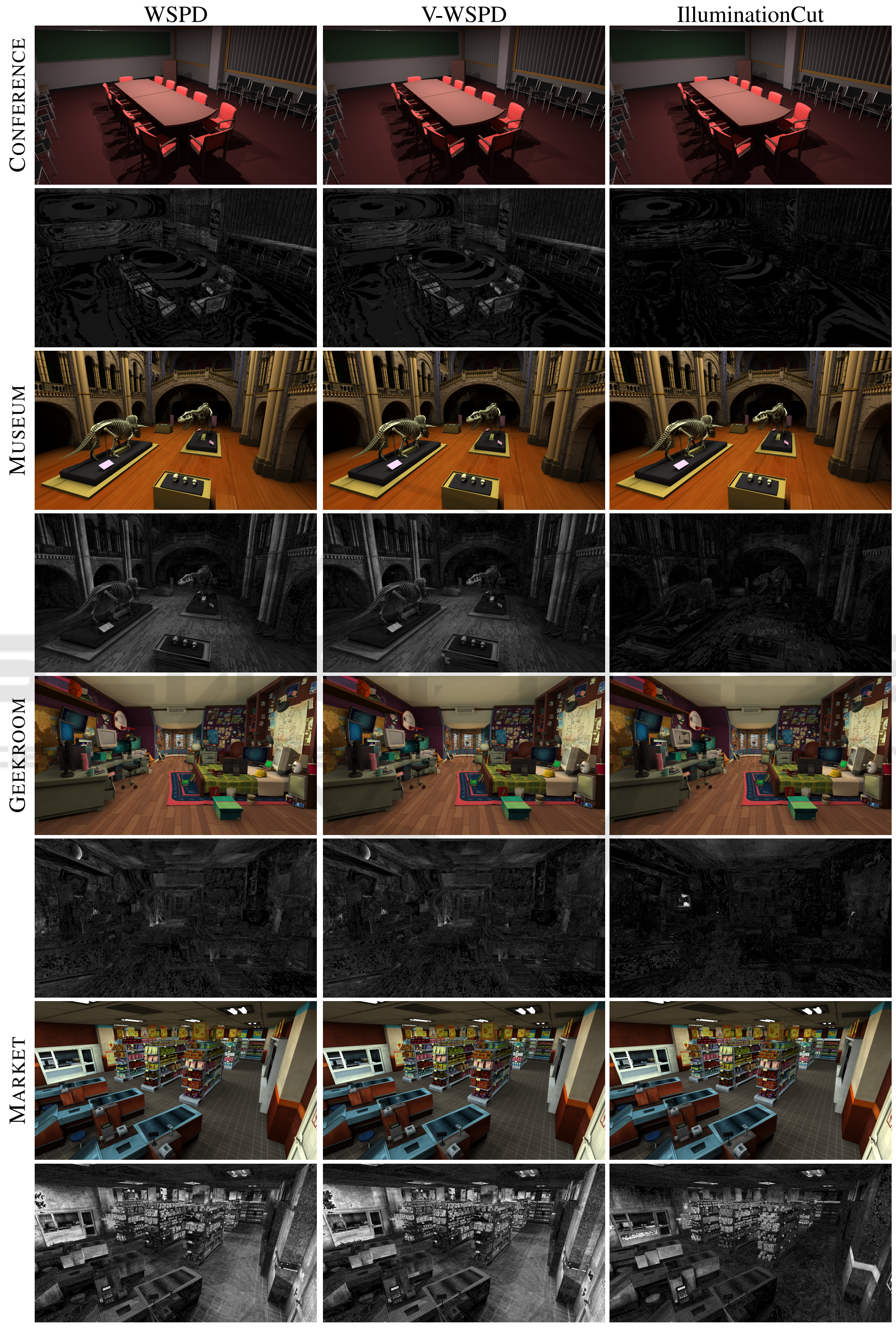
Another idea would be to compact the V-WSPD with the space partition used as acceleration structure for the scene geometry. Thus, rendering times could be further improved since only one structure scan would be necessary for both primary ray casting and indirect illumination.

Finally, we plan to use the concept of pair decomposition for different kinds of set of points: light source points, importance records (points viewed from the camera) and even pure geometrical points. The latter implies a good point sampling for the scene geometry. All these combinations of pair of point sets can be used for both points clustering, points sampling and points rejection. This drives us toward a framework for clustering, rejecting or selecting pair of points that can be useful in several situations in rendering (coarse evaluation of visibility, driving light transport in path tracing...).

ACKNOWLEDGEMENTS

This work has been supported by the BPI France PAPAYA grant, from the *Programme d'Investissement d'Avenir*. Authors would like to thank Ubisoft Motion Pictures Montreuil for giving us for the GEEKROOM and MARKET scenes, and K. Renaudeau for English language reviewing.

Table 2: Comparison for same rendering time: pure geometric WSPD, visibility based WSPD (V-WSPD), IlluminationCut (Bus et al., 2015b) and false-colored image for error (Euclidean distance $\times 128$ in LAB color space).



REFERENCES

Áfra, A. T., Wald, I., Benthin, C., and Woop, S. (2016). Embree ray tracing kernels: Overview and new features. In *ACM SIGGRAPH 2016 Talks*, SIGGRAPH '16, pages 52:1–52:2, New York, NY, USA. ACM.

Bus, N., Mustafa, N. H., and Biri, V. (2015a). Global illumination using well-separated pair decomposition. *Computer Graphics Forum*, 34(8):88–103.

Bus, N., Mustafa, N. H., and Biri, V. (2015b). IlluminationCut. *Computer Graphics Forum (Proceedings of Eurographics 2015)*, 34(2):561 – 573.

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

Dachsbacher, C., Křivánek, J., Hašan, M., Arbree, A., Walter, B., and Novák, J. (2014). Scalable realistic rendering with many-light methods. *Computer Graphics Forum*, 33(1):88–104.

Davidovič, T., Georgiev, I., and Slusallek, P. (2012). Progressive lightcuts for gpu. In *ACM SIGGRAPH 2012 Talks*, SIGGRAPH '12, New York, NY, USA. ACM.

Davidovič, T., Křivánek, J., Hašan, M., Slusallek, P., and Bala, K. (2010). Combining global and local virtual lights for detailed glossy illumination. *ACM Trans. Graph.*, 29(6):143:1–143:8.

Dutré, P., Bala, K., Bekaert, P., and Shirley, P. (2006). *Advanced Global Illumination*. AK Peters Ltd.

Ernst, M. and Greiner, G. (2008). Multi Bounding Volume Hierarchies. In *IEEE Symposium on Interactive Ray Tracing*, RT '08, pages 35–40.

Georgiev, I., Křivánek, J., Popov, S., and Slusallek, P. (2012). Importance caching for complex illumination. *Computer Graphics Forum*, 31(2pt3):701–710.

Georgiev, I. and Slusallek, P. (2010). Simple and Robust Iterative Importance Sampling of Virtual Point Lights. In Lensch, H. P. A. and Seipel, S., editors, *Eurographics 2010 - Short Papers*. The Eurographics Association.

Hašan, M., Pellacini, F., and Bala, K. (2007). Matrix row-column sampling for the many-light problem. *ACM Trans. Graph.*, 26(3).

Hašan, M., Velázquez-Armendariz, E., Pellacini, F., and Bala, K. (2008). Tensor clustering for rendering many-light animations. In *Proceedings of the Nineteenth Eurographics Conference on Rendering*, EGSR '08, pages 1105–1114, Aire-la-Ville, Switzerland. Eurographics Association.

Hedman, P., Karras, T., and Lehtinen, J. (2016). Sequential monte carlo instant radiosity. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '16, pages 121–128, New York, NY, USA. ACM.

Huo, Y., Wang, R., Jin, S., Liu, X., and Bao, H. (2015). A matrix sampling-and-recovery approach for many-lights rendering. *ACM Trans. Graph.*, 34(6):210:1–210:12.

Keller, A. (1997). Instant radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 49–56, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Křivánek, J., Fajardo, M., Christensen, P. H., Tabellion, E., Bunnell, M., Larsson, D., and Kaplanyan, A. (2010). Global illumination across industries. In *ACM SIGGRAPH 2010 Courses*, SIGGRAPH '10, New York, NY, USA. ACM.

Lopes, R. H. C., Reid, I. D., and Hobson, P. R. (2014). A well-separated pairs decomposition algorithm for k-d trees implemented on multi-core architectures. *Journal of Physics: Conference Series*, 513(5):052011.

O'Donnell, Y. and Chajdas, M. G. (2017). Tiled light trees. In *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '17, pages 1:1–1:7, New York, NY, USA. ACM.

Ou, J. and Pellacini, F. (2011). Lightslice: Matrix slice sampling for the many-lights problem. *ACM Trans. Graph.*, 30(6):179:1–179:8.

Paquette, E., Poulin, P., and Drettakis, G. (1998). A light hierarchy for fast rendering of scenes with many lights. *Computer Graphics Forum*, 17(3):63–74.

Rehfeld, H. and Dachsbacher, C. (2016). Lightcut interpolation. In *Proceedings of High Performance Graphics*, HPG '16, pages 99–108, Aire-la-Ville, Switzerland. Eurographics Association.

Ritschel, T., Grosch, T., Kim, M. H., Seidel, H.-P., Dachsbacher, C., and Kautz, J. (2008). Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. *ACM Trans. Graph. (Proc. of SIGGRAPH ASIA 2008)*, 27(5).

Simon, F., Hanika, J., and Dachsbacher, C. (2015). Richvpls for improving the versatility of many-light methods. *Comput. Graph. Forum*, 34(2):575–584.

Walter, B., Arbree, A., Bala, K., and Greenberg, D. P. (2006). Multidimensional lightcuts. *ACM Trans. Graph.*, 25(3):1081–1088.

Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M., and Greenberg, D. P. (2005). Lightcuts: A scalable approach to illumination. *ACM Trans. Graph.*, 24(3):1098–1107.

Walter, B., Khungurn, P., and Bala, K. (2012). Bidirectional lightcuts. *ACM Trans. Graph.*, 31(4):59:1–59:11.

Wang, G., Xie, G., and Wang, W. (2011). Efficient search of lightcuts by spatial clustering. In *SIGGRAPH Asia 2011 Sketches*, SA '11, pages 26:1–26:2, New York, NY, USA. ACM.

Yoshida, H., Nabata, K., Iwasaki, K., Y., D., and T., N. (2015). Adaptive importance caching for many-light rendering. *Journal of WSCG*, 23(1):65–72.