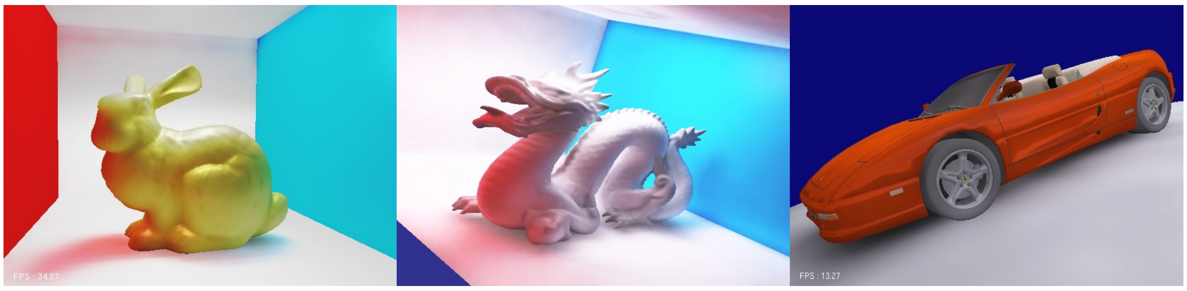
Light octree: global illumination fast reconstruction and realtime navigation

Vincent Vivanloc, Jean-Christophe Hoelt, Coong Binh Hong and Mathias Paulin
Institut de Recherche en Informatique de Toulouse - Université Paul Sabatier Toulouse III - France
{vivanloc,hoelt,paulin}@irit.fr



Abstract

We present a method to rapidly build an irradiance cache based on a local illumination environment approach. This cache is obtained by a stream simplification of a photon map. The photons are K-Means clustered per voxel into sets of virtual directional light. These lights are stored into an irradiance texture to provide a real-time rendering of a global illuminated scene. This method can be integrated into an existing GPU shader to obtain complex material rendering and can be accelerated by texture atlases.

Keywords global illumination, local illumination environment, irradiance cache, octree, real-time

1 INTRODUCTION

The overall quality of computer rendered scene can be greatly enhanced by taking into account the indirect illumination [TL04]. In one hand, the classical OpenGL direct lighting pipeline can be improved by multipass techniques for real time effects as shadows, specular reflections, refractions [SKALP05] and splat based caustics. On the other hand, the rendering of a lower frequency lighting is provided by precalculated light maps based on radiance transfers. These maps are often computationally expensive to generate and limited to diffuse radiosity.

We develop a method to generate these maps rapidly from photon maps by a simple and parallelizable algorithm. Our simplification relies on local illumination environment (LIE [FBG02]) approach. The scene space is divided into voxels filled with a compact representation (Virtual Directional Light) of the irradiance. Since the process is done locally, the simplification error is minimized compared to a global scene scheme.

Our representation of illumination could be used as an irradiance cache for ray tracing. However, such expensive rendering must be reprocessed at every viewpoint change. Since we would like a real-time rendering of global illumination solution, we develop a viewpoint independent rendering method, based on a GPU rendered octree. The sets of VDLs are directly stored in the GPU memory and can be integrated with advanced material shaders. Eventually, we propose a render-to-atlas procedure to improve the framerate and enable rendering on legacy OpenGL hardware.

Our contribution is :

- a fast and stream simplification of a photon map;
- a real-time rendering of indirect illumination on a GPU without scene remeshing
  - integrable in existing shader pipelines for advanced material rendering
  - running on legacy OpenGL hardware (requires a scene parametrization)

2 PREVIOUS WORKS

2.1 Creating an illumination cache

A global illumination solution can be provided in screen space, using path tracing [Kaj86, LW93] or better, in scene space, by radiosity [GTGB84] or photon mapping [Jen96, Jen01]. Contrary to the former screen space method, a scene space solution can be reused when the scene viewpoint change. We choose the photon map approach since it takes into account not only caustics but also directional diffuse and specular indirect irradiances.

2.1.1 *Cache sample representation*

A radiance cache is a scalar field of a reflectance function values. Such function links the incoming irradiances to the radiance. For both flexibility and efficiency, the irradiance is preferred to the radiance. The convolution between irradiance and the 5 dimension reflectance function can produce a high memory consuming vector field. Such field must then be compactly stored in a continuous or a discrete way.

A continuous representation consists in fitting a discrete set of photons into a set of coefficients bound to a function basis, such as spherical harmonics (SH) or wavelets. SHs were the first used for precomputed radiance transfer (PRT) [SKS02], for diffuse [RH01] and glossy [LSSS04, KGPB05] materials. Contrary to wavelets, SHs are not directly applicable to high frequency materials. Recently, wavelet coded BRDFs have been rendered on GPU [WTL06]. However, both of these continuous representations require heavy precomputations, for instance a clustered PCA analysis [SHHS03] for SHs.

In a discrete representation, the photon map can be considered as a set of discrete lights: point light (Virtual Point Light [Kel97]) or directional light (Light Vector) [ZSP98]. Such discrete representations of lights are grouped by clusters which similarity distances are based on density [CLSS97], visibility, power, position [SWZ96, PPD98] or perception values [FBG02, WFA$^+$05]. Even if discrete representations are more prone to aliasing than their continuous counterparts, their simplicity is well suited with a stream processing of the photon map.

2.1.2 *Global and local cache*

The transformation of a set of photons into an irradiance representation can be done on the whole scene, resulting in a *global* cache. The simplification bias can be reduced by working locally in limited regions of space, leading to a *local* cache. These space partitions can be constructed on a per scene object basis and more generally, with a regular grid, or a less memory consuming octree. Such structure has been used for renders with massive number of lights , assuming that such lights have only a local influence on the environment [FBG02]. The octree can be built on geometric criterion or subdivided using an irradiance threshold, to create an irradiance volume [GSHG98, PH04].

2.2 Rendering an illumination cache

A fast reconstruction of a global illumination solution is an open problem. Rendering directly the photon map needs a huge number of photons to be casted. Hybrid methods can reduce the number of casted photons: high frequency effects like shadows or specular reflections/refractions are processed by GPU rasterisation whereas photon mapping is reserved for lower frequency irradiances [LC04].

2.2.1 *Global cache rendering*

The reconstruction of a photon map can be done in screen, texture or object space.

The screen space reconstruction always offers the highest lighting quality since it is a per pixel computation. The *final gathering pass* is the most time consuming part of rendering [Jen96]. It has been parallelized on CPU [WKB$^+$02] and GPU [Hac05], but must be reprocessed on each camera move.

On the contrary, an object space reconstruction is viewpoint independent: the cache samples are bound to the mesh vertices. PRT renderings are often applied on high tessellated, static and single object scenes [SHHS03]. On the other hand, the scene could be dynamically re-meshed to match the illumination distribution with its geometry [WHSG97]. Such per vertex lighting could increase dramatically the geometric complexity of the scene.

A texture space reconstruction is also independent from the camera position [Arv86, Shi90]. For instance, a static or dynamic [Nie00, NC02] precomputed radiosity can be bound into a simple 2D texture. Otherwise, hybrid object/texture space reconstruction of the photon map has been done on a hybrid CPU/GPU hemicube final gathering but still require a well tessellated scene [LC04].

The photon splatting method can be affected to this texture space reconstruction category. This method is an application of point based rendering. The final gathering is replaced by its dual operation: instead of fetching the contribution of the neighbouring photons to estimate a photon density, this density is summarized by a radial function. Practically, a gaussian centered on photon is directly splatted on screen. It has been applied to diffuse [LP03] and glossy materials [GKBP05]. Since photon splatting adds geometry and fillrate overheads to the scene, real time framerates are only reached for low frequency indirect lighting [DS05] or space constrained caustics [SKALP05, KBW06, WD06].

2.2.2 *Local cache rendering*

The previous methods can also be applied to local irradiance caches, bound to localized regions. Like their global counterparts, such caches can be generally used by screen space rendering [CB04] to accelerate the final gathering stage. Otherwise, GPU rasterisation techniques are combined to render such caches in realtime. Precalculated radiosity can be linked to a sphere map [WTP00] or a cube map [NPG03]. These methods are limited by cube map constraints (convex objects and concave environment).

Eventually, the rendering can also be done per voxel. The first approach used a multi-pipe SGI GPU with well known clustered OpenGL lights [UH99]. The octree texture is a more affordable solution, running on

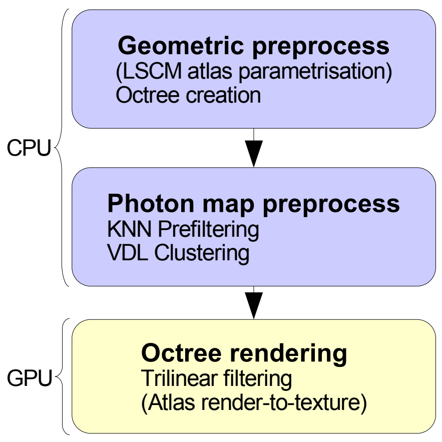


Figure 1: Rendering pipeline Atlas LSCM parametrisation and render-to-texture stages are only required for atlas rendering on OpenGL legacy hardware

commodity hardware. Such octree textures were proposed for a multiresolution and intuitive object volumetric painting. They do not require any texture surface parametrization [gDGPR02, BD02] and can be run on GPU [LHN05, KLS$^+$05]. Such methods can also be derivated for animated lights. The light source clouds are clustered and compressed into SH to be finally bound to voxels [KAMJ05]. However, such process requires hours of precomputations.

3 GENERAL ARCHITECTURE

The architecture of our rendering pipeline is based on a hybrid renderer, assuming that the illumination can be separated into direct and indirect components. Stochastic indirect illumination methods can hardly represent high frequency signals even with a large amount of photons. Practically, shadows, specular reflections or refractions are less detailed. Thus, these high frequency effects must be rendered by a different pipeline.

In this paper we focus on indirect illumination rendering pipeline. This pipeline is composed of 3 main stages (figure 1). The geometric preprocessing stage contains the octree creation. The photon map clustering stage, processed on the CPU, is a highly streamable process, described on the section 3.1. Finally, the GPU rendering provides a real-time rendering of indirect illumination, detailed in the rendering section 3.2.

3.1 Photon map clustering

3.1.1 *Prefiltering*

A photon map with less than one millon photons is often too much biased to be directly used. To reduce this bias, the photon map is clustered per voxel. However, such a histogram approach assumes estimation areas are equivalent [Jen01]. This is roughly true for a high resolution octree and a very tesselated geometry. In practice, this is false for coarse octree or for a geometry made of large planes. In such configuration, the photon density can be underestimated since the probability to hit the voxel/polygon section can be very low. Visually, it is translated into bands of different colors artefacts (see figure 16-a). Therefore, a KNN filtering is compulsory to smooth the photon density across the voxels.

KNN parametrisation

Such filtering requires two parameters, k, the number of nearest neighboors and r, the radius of region of gathering. In our case, these parameters are set arbitrary 5. The optimal (k, r) values are found experimentally to reach an acceptable visual aspect in a reduced preprocessing time. Concerning the gathering region, we have chosen an ellipsoid aligned on voxel normal. This could induce some artefacts explained in 5.3.1.

3.1.2 *Clustering*

In order to provide a GPU friendly representation of the photon map, the illumination cache sample must be compacted in order to fit in a tiny GPU memory. We conceived a simple method to cluster the photons into virtual directional lights.

Choice of clustering method

The principal component analysis (PCA) and its derivative (Clustered PCA [SHHS03]) can be very time consuming. Hierarchical clustering (HClust) produces a clustering at all levels and returns an unique solution, but is in $O(n^2)$, where n is the number of samples. KMeans [Mac67] cannot produce a unique solution but requires a lower memory footprint. Its theorical worst-case complexity is polynomial [HPS05, AV06]; however, in practice the number of Kmeans iterations required to obtain a solution, is low. In our case, for a given population of 50 to 500 photons per voxel, with the simplifications described below, the total time spent to create up to 8 clusters is below 1 second to 1 minutes depending on number of voxels.

KMeans parametrisation

As automatic a classification algorithm could be, we need to define at least two parameters: the sample to cluster-centroid distance function and the resulting entity after the fusion of cluster samples.

An accurate definition of a similarity between two photons would consider the 5 dimensions of the irradiance. This would be too computationally expensive, so we assume the voxel volume is little enough to discard the photons position. Consequently, the distance is defined as the cord length formed by the two photon incoming directions.

The first fusion entity considered was an accumulation of the irradiance. However, this simplification has a great visual impact on the rendering (see figure 2). That is why we introduce the virtual directional light (VDL), a (irradiance; incident direction) pair. The VDL can be compared with the light vector

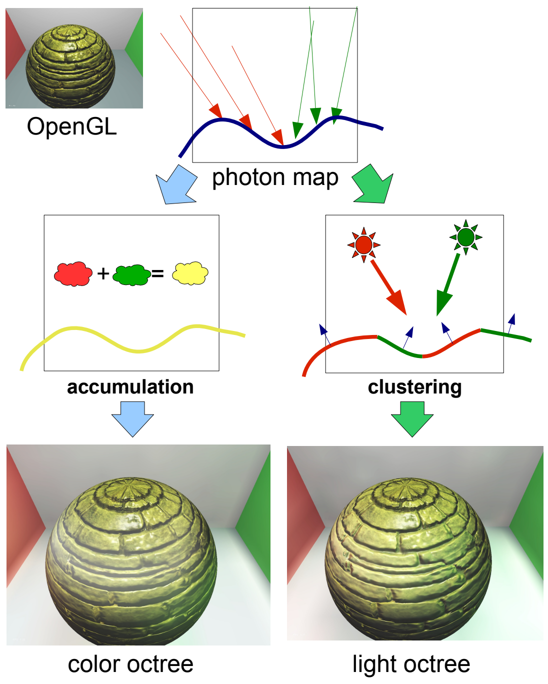


Figure 2: Color and light octree photon map simplifications. In the left picture, the color seems to be foggy, due to the trilinear filtering and to the loss of irradiance directional information. In the right picture, the ball is clearly illuminated by two distinct lights, thanks to the VDLs. OpenGL rendering is provided for comparison.

[ZSP98]. The light vector represents a directional radiance whereas the VDL contains only an irradiance and lets the GPU processes the radiance. Contrary to HClust, KMeans works on a given number of clusters. Since we are limited by the GPU memory and shader sizes, we fixed the maximum number of VDLs to 8.

3.2 Photon map rendering

Our clusterized photon cache could be used in a classical screen space rendering, but we applied it to GPU for real time rendering. The octree rendering with directional light rendering, contrary to photon splatting, is less sensitive to local geometric variation, since the irradiance is implicitly bound to the surfaces intersecting the voxel (figure 2).

The set of directional lights is bound to each voxel and can be integrated by shaders as classical OpenGL directional lights to process illumination locally. It takes into account the variation of the surface and thus, enables the rendering of surface dependant material. Therefore, our irradiance texture can be easily integrated to an existing shader to render surface-dependant complex material (figures 2,11).

The voxel volume is assumed to be little enough to ignore occlusions. Pratically, VDLs do not cast any shadows. However, parallax occlusion mapping algorithms [PO06, Tat06] could be integrated to our ren-

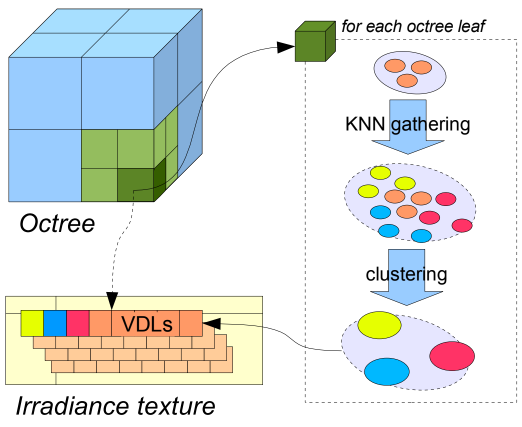


Figure 3: Octree clustering

```
octree.build(scene.geometry);
KDTree.build(scene.photonMap);

foreach octree.leaves {
    photons=gather(nbPhotons,radius,KDtree);
    VDLs=cluster(nbVDL, photons);
}
```

Figure 4: Photon map preprocessing on CPU

dering pipeline to improve the quality of surface dependant shaders (figure 12).

4 IMPLEMENTATION

Our pipeline is running on a dual bi-core 2Ghz AMD64 Opteron 270's with NVidia GeForce 4500 GPU. It is compiled under Linux with gcc 4.0.2 and shaders are implemented in Cg under a fp40 profile.

4.1 Photon casting and clustering

The photon map is loaded from a modified version of Yafray [WEdG$^+$06], a modular global illumination renderer. It has been chosen for its fast photon casting[1] and its tight integration with Blender, the well known powerful 3D modeler and renderer.

The octree is built on the whole scene, using a geometric subdivision criterion. An octree built on light irradiance density must be rebuilt at each new photon arrival and prevents any octree GPU filtering.

We attempt obtaining an efficient KNN filtering. We replaced the photon gathering by a scattering, implemented by screen and space octree splattings. For screen space splatting, we got a blurred rendering with low performances, induced by some fillrate overheads. For space splatting, we suffered from memory shortage due to a compulsory storage of the splatted photons. We also tried a GPGPU implementation but such *scattering* algorithm does not suit to existing GPU architectures. We fetched photons directly using the octree. Unfortunately, compared to a KD-tree, it almost doubled the photon gathering time. Finally, we came back to a classical KD-tree KNN gathering.

[1] few minutes for a 1 million photon map

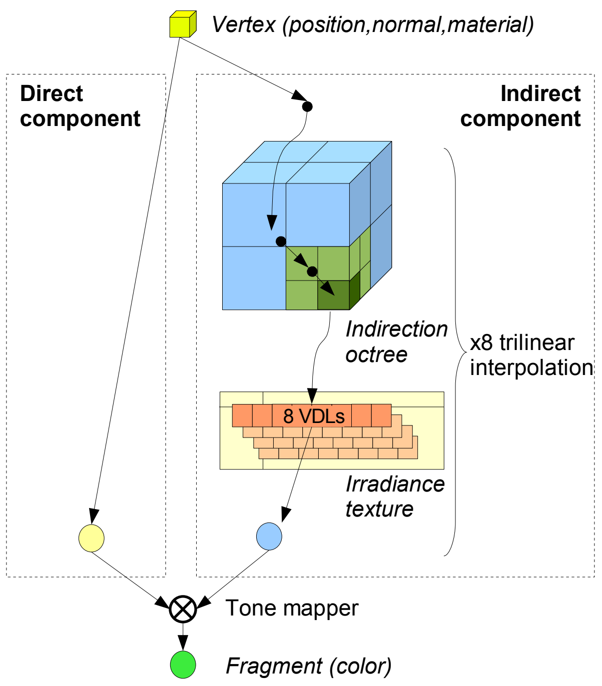


Figure 5: Octree rendering

```
// reach the leaf stored in the octree texture
@root=float3(0,0,0);
node=Tex3D(octreeTex,@root);
for (int depth=0;depth<MAX_DEPTH;depth++) {
  @child=getChildAddress(node.rgb,octreePos);
  node=Tex3D(octreeTex,@child);
}

if (node.a!=OCTREE_LEAF_TYPE)
   discard;
leaf=node;

// get VDL array location within the irradiance texture
@VDLarray=getIrradianceTexAddress(leaf.rgb);

// set leaf color using VDL array
leafColor=float4(0,0,0,0);
for (int iVDL=0;iVDL<MAX_VDL;iVDL++) {
    VDL=TexRect(irradianceTex,@VDLarray+iVDL);
    leafColor+=baseColor*brdf(VDL.direction,
                             VDL.intensity,
                             surfaceNormal);
}
```

Figure 6: Photon map rendering on GPU

Our photon map clustering is fully streamable. Photon octree compressors are run on 4 threads, reducing by preprocessing time by a factor 3 to 4 (figure 3).

4.2 Photon rendering

4.2.1 *Enhanced GPU octree*

The light octree is an extension of the GPU octree texture provided by [LHN05]. The octree is entirely coded in a texture stack implemented by a 3D texture. A flag in the texel alpha channel indicates if the RGB field is an address to the next octree sub-node or a leaf. For an octree of depth δ, a maximum of δ texel ac-

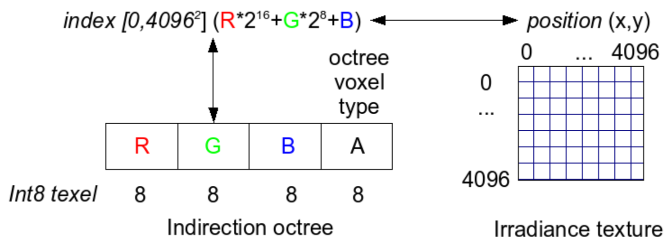


Figure 7: Irradiance address representation

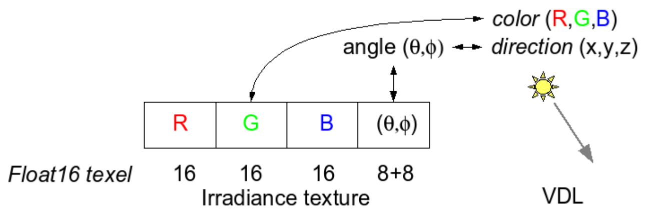


Figure 8: VDL representation

cesses is done. The octree texel are in 8 bit integer format because float 3D textures are not available on our hardware configuration. Since we do need a float storage, we introduce a secondary texture containing irradiance data (figure 5). The octree leaves contains the addresses to the data texels. Since actual textures are limited to a 4096x4096 resolution, a 24bit coding for these addresses is sufficient (see figure 7). A texture memory manager reduces the original memory footprint from 64MiB to 512KiB(octree)+4MiB(VDLs), for 100k leaves of a $\delta = 6$ octree with $k = 4$ VDLs.

Since the color is constant per voxel, the result has a blocky aspect. This can be alleviate with an interpolation. This per pixel smoothing is done by a brute force trilinear interpolation between 8 neighbour voxels[2].

4.2.2 *VDL rendering*

We consider a virtual directional light $VDL(c,d)$ where c is the irradiance converted in RGB color and $\mathbf{d}$, its direction. Storing the color and direction of the virtual lights in two texels wastes the two alpha channels and lowers the performances due to the additional texel access. The virtual light is packed into one unique texel by compressing the direction into a single float in the alpha channel. The direction is converted from a cartesian (x,y,z) to a spherical (θ,ϕ) frame[3] and the angles are discretized into 8+8 bits values (figure 8).

Once retrieved from the irradiance texture, the VDLs can be integrated easily within any illumination shader.

$$c_{VDLs} = \sum_{i=1}^{n_{VDLs}} brfd(\mathbf{d}_i, \mathbf{v}, \mathbf{n}).c_i$$

where $c_i, \mathbf{d_i}$ are resp. the irradiance and the direction of the ith VLD, n_{VDLs} is the number of VDLs, $\mathbf{v}$ the view direction and $\mathbf{n}$ is the surface normal.

[2] A bicubic filter would require 24 samples which is too expensive.
[3] Within a normalized frame, $r = 1$

```
// CPU : atlas parametrization
foreach meshes {
    mesh.buildCharts(charts);
    foreach charts
        chart.buildAtlas();
    charts.pack();
}
```

```
// GPU : atlas rendering
//render to atlas texture (if lighting is updated)
foreach meshes {
    octreeTexture.shader.bind();
    //draw atlas using (u,v) as atlas vertices
    foreach charts
       chart.drawAtlas();
    //capture framebuffer into atlas texture
    charts.captureFramebuffer();
}

//apply atlas texture as a simple texture
foreach meshes {
     charts.bind();
     //draw mesh using (u,v) as mesh texcoord
     mesh.draw();
}
```

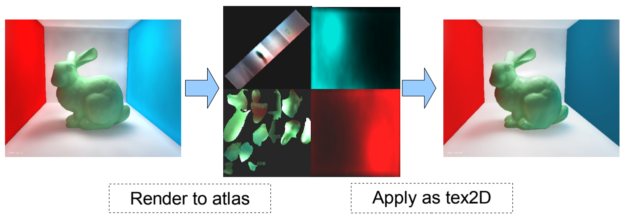


Figure 9: Render to atlas process

4.2.3 *Render to atlas*

For a light octree, a single fragment requires $n_{VDL} + \delta + 1$ texture accesses. The trilinear filtering can worsen the performances by multipling accesses by 8. The number of texture fetch can be drastically reduced to a single access using an atlas light map. Such process has been done for radiosity [RUCL03], we extended it to our light octree.

The scene objects must be parametrised into atlases before any rendering. This can be done manually with Blender or automatically using our geometric preprocessor. The charts are built using a Voronoï construction method [SWG$^+$03]. Each chart is then parametrised into atlas using the Least Square Conformance Map [LPRM02] algorithm and is finally tightly packed into a 2D texture (figure 9).

Once the scene is parametrized, the render to atlas is done on the fly by rendering the 3D octree in the atlas, capturing it and applying the resulting texture on the model (figure 10).

5 RESULTS

5.1 Visual aspect

The light octree can be integrated to surface bound shaders to simulate complex material. For instance, some specular materials have been simulated using

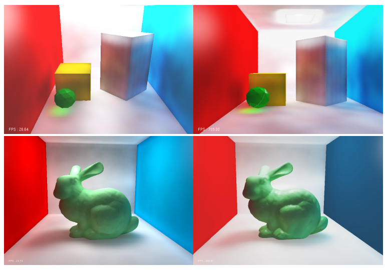


Figure 10: Scenes before & after render to atlas texture

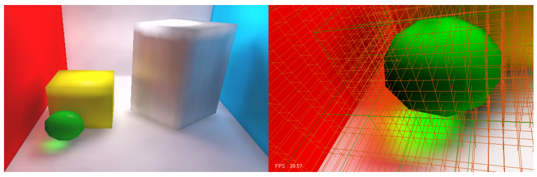


Figure 11: Caustics rendering

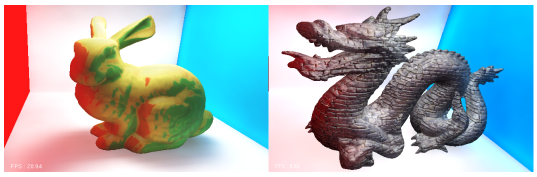


Figure 12: Integration with cube map and parallax bump map shaders

simple cube map and parallax bump shaders. Nothing but the shader complexity, would prevent to integrate the irradiance texture into more realistic shaders (figure 12).

The visual differences between texture atlas rendering and its light octree counterpart mainly come from the more of less visible atlas seams. An integer texture format could also clamp down the color dynamic (figure 10).

Contrary to PRT techniques based vertex sampling, the surface irradiance distribution is split per voxel. The octree is therefore more robust to the high variations of the irradiance, which can be seen on caustics close up of the figure 11).

5.2 Timing results

The CPU preprocessing time is linear in term of voxels (table 13). The geometry complexity has a lesser importance. The compression time spent for clustering is low (few seconds) regarding the KNN gathering (figure 17). The less photons are gathered, the faster the computation is processed (table 14). This stage is the most expensive part of the compression scheme but cannot be skipped, as explained in the subsection 3.1.1.

A progressive updating process allows the user to rapidly preview a globally illuminated scene. The octree is updated progressively from coarse to fine depth octree. A rough visualisation ($\delta = 4$) is available in less than 10 seconds and an acceptable result ($\delta = 5$) is achieved in less than 1 minute (on figure 13).

octree depth	Boxes	Bunny	Dragon	f355	Megane
4	5s	8s	6s	1s	15s
5	37s	40s	37s	10s	1m16s
6	2m45s	2m41s	3m47s	1m20s	5m37s

Figure 13: Compression preprocessing time: 600,000 photon map on boxes (50 polys), Bunny (70k), Dragon (200k), f355 (50k), Renault Megane (700k) scenes. Timing differences are caused by different photon map density distributions.

KNN/photons	150k	300k	600k	1,2M
50	4	5	6	6
100	6	7	10	14
250	16	21	20	29
500	33	44	44	47

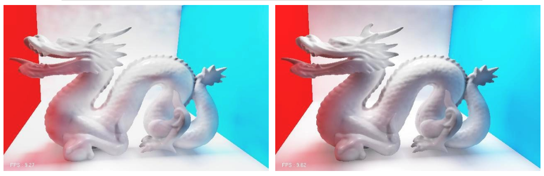


Figure 14: KNN gathering & compression times (in seconds on the Dragon scene with a $\delta = 5$ octree). Lowest and highest quality renders.

The GPU rendering framerate is more tied to the amount of fragments than to the vertices. Once preprocessed, the light octree is rendered by GPU at an average framerate from 40 fps for a 800x600 to 10fps for 1280x1024 resolution. In addition to the shader complexity, the framerate is limited by the great number of texture fetches needed per pixel. An atlas rendering could reach from 700 to 200 fps depending on the viewport size.

5.3 Discussion on accuracy

On one hand, we cannot compete qualitatively with Yafray offline global illumination renders. The screen space approach gives visually better results because the underlying coarse irradiance cache is compensated by a per pixel *final gathering*. On the other hand, our solution provides a real time navigation in a roughly equivalent scene (figure 15).

In our method, a compression error is directly translated into visual artifacts. The purpose of this subsection is to provide some solutions for such rendering errors.

5.3.1 *Octree aliasing*

Our method may suffer from three sampling artefacts that should be resolved by additional per voxel computations.

First, the density underestimation, visible on large planes (figure 16-a), can be alleviated using KNN filtering (subsection 3.1.1).

However, this filtering is also the cause a second artefact, the energy bleeding on object edges and corners (figure 16-b). In a screen based final gathering,

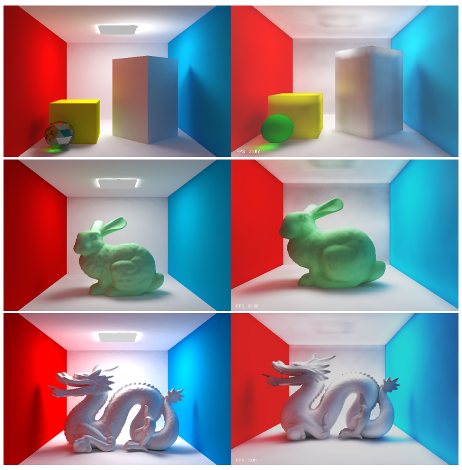
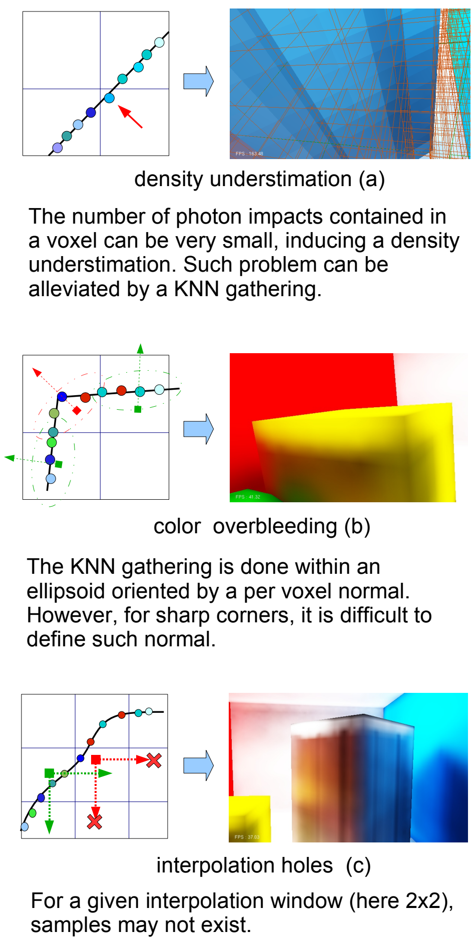


Figure 15: Offline global illumination reference renders and their light octree counterparts



density understimation (a)

The number of photon impacts contained in a voxel can be very small, inducing a density understimation. Such problem can be alleviated by a KNN gathering.

color overbleeding (b)

The KNN gathering is done within an ellipsoid oriented by a per voxel normal. However, for sharp corners, it is difficult to define such normal.

interpolation holes (c)

For a given interpolation window (here 2x2), samples may not exist.

Figure 16: Octree aliasing consequences

k	CPU preprocess	spread error (deg)	GPU (fps)
1	58s	30	25
2	1m25s	25	20
3	1m37s	22.5	17
4	1m48s	20	14
6	2m12s	15	12
8	2m30s	10	10

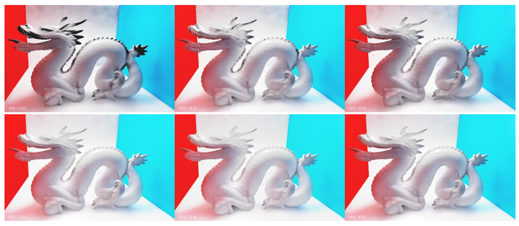


Figure 17: Number of light clusters & quality (KNN=300, $\delta = 6$, 600k photon map)

the gathering space is constrained by a cone built using the photon impact normal. In our gathering, we define an ellipsoid oriented along a voxel normal. Per definition, a voxel has no normal. In the current version, it is approximated as an average amongst the intersecting geometry.

Third, visual banding artifacts (figure 16-c) can be produced by the void voxels fetched by the trilinear interpolation. This may be prevented by virtually extending the voxel bounds. Unfortunately, a too large overlapping zone can produce visible defects.

Artificially expanding the voxel boundaries can also impede the photon density estimation: the photon gathering is more time consuming and less accurate. In summary, a careful balance must be set between a blurred picture and a better but slower result.

5.3.2 Number of VDLs

The second source of error is bound to the clustering itself. The more the clusters are, the better the results would be, even if the framerate would slightly decrease. The intra cluster error is measured in term of standard deviation between the direction of cluster centroid and the directions of the clustered photons, expressed in cord length unit. This angular error is easily explained by the low number of clusters: a hemisphere (180°) is partitioned between only k clusters.

For $k = 4$, the standard deviation around its cluster centroid (± 20 degrees) seems to be high but the overall quality of picture is preserved. For lowers k, the higher error rate induces visible artefacts (figure 17).

5.3.3 Voxel size

The voxel size often matters. If it is too large, the photon clustering is a low-pass filter and the irradiance is blurred. Otherwise, if it is too little, too few photons are intersected, leading to a lighting underestimation. Experimentally, the quality improves for $\delta = [4,6]$. A depth $\delta = 6$ brings the octree resolution to 64x64x64. For a given scene contained in $1m^3$, a voxel has a volume of $1/64 \simeq 1,5cm^3$,which could be low in term

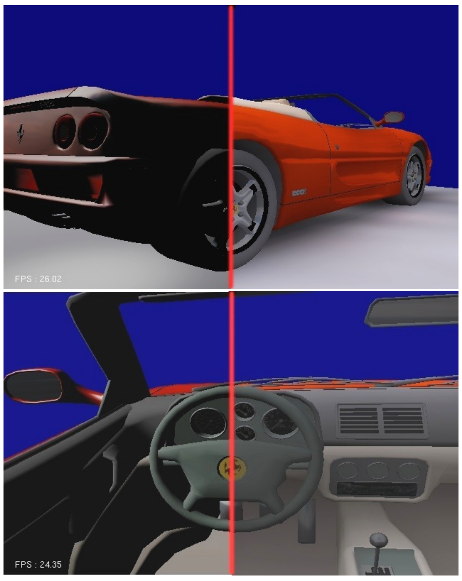


Figure 18: Ferrari 355:OpenGL & light octree

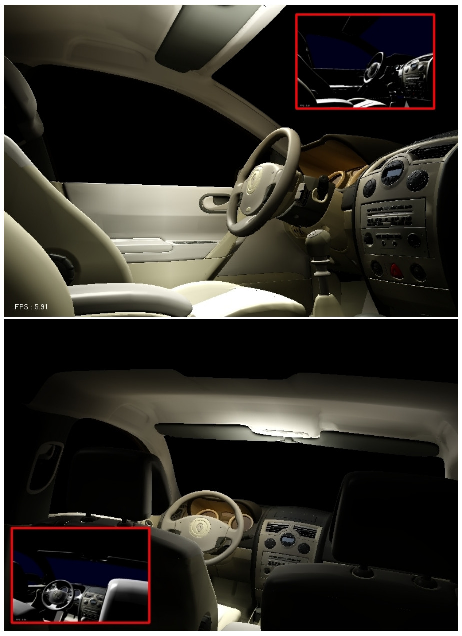


Figure 19: Renault Megane:OpenGL & light octree

of accuracy but rather sufficient to improve the visual aspect against a classic OpenGL rendering. For $\delta = [7,8]$, the effort spent to smooth the density does not provide significant improvements.

6 CONCLUSION AND FUTURE WORKS

The photon map clustering is an efficient technique to rapidly produce an irradiance cache. Compared to radiosity maps, the light octree provides more visually appealing scenes thanks to a photon map. Our sets of directional lights localized per voxel reduces the global compression error and is highly integrable to existing GPU shaders.

Techniques to incrementally update our octree should be examined. Many Kmeans algorithm incremental versions have been proposed. Eventually, a continuous representation, with SH or wavelets should be also compared with our discrete VDL representation.

ACKNOWLEDGMENTS

This research is performed in the frame of the RNTL Project Lumière, funded by the French Board of Research involving IRIT/SIRV, EADS/CCR laboratories, Renault, EADS companies and OKTAL IT service company. Models courtesy resp. by the Stanford 3D Scanning Repository, Fabrice HEILLOUIS (DMI cars) and Renault Centre Technique de Simulation.

More information on http://www.irit.fr/Lumiere

REFERENCES

[Arv86] J. Arvo. Backward ray tracing. In *Course Notes of the 1986 Conference on Computer Graphics and Interactive Techniques*, volume 12, pages 18–22, 8 1986. 2

[AV06] David Arthur and Sergei Vassilvitskii. How slow is the k-means method ? In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 144–153, New York, NY, USA, 2006. ACM Press. 3

[BD02] David Benson and Joel Davis. Octree textures. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 785–790, New York, NY, USA, 2002. ACM Press. 3

[CB04] Per H. Christensen and Dana Batali. An irradiance atlas for global illumination in complex production scenes. In *Rendering Techniques*, pages 133–142, 2004. 2

[CLSS97] Per H. Christensen, Dani Lischinski, Eric J. Stollnitz, and David H. Salesin. Clustering for glossy global illumination. *ACM Trans. Graph.*, 16(1):3–33, 1997. 2

[DS05] Carsten Dachsbacher and Marc Stamminger. Reflective shadow maps. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 203–231, New York, NY, USA, 2005. ACM Press. 2

[FBG02] Sebastian Fernandez, Kavita Bala, and Donald P. Greenberg. Local illumination environments for direct lighting acceleration. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 7–14, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association. 1, 2

[gDGPR02] David (grue) DeBry, Jonathan Gibbs, Devorah DeLeon Petty, and Nate Robins. Painting and rendering textures on unparameterized models. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 763–768, New York, NY, USA, 2002. ACM Press. 3

[GKBP05] Pascal Gautron, Jaroslav Křivánek, Kadi Bouatouch, and Sumanta Pattanaik. Radiance cache splatting: A GPU-friendly global illumination algorithm. In *Rendering Techniques*, pages 55–64, 2005. 2

[GSHG98] Gene Greger, Peter Shirley, Philip M. Hubbard, and Donald P. Greenberg. The irradiance volume. *IEEE Computer Graphics and Applications*, 18(2):32–43, 1998. 2

[GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 213–222, New York, NY, USA, 1984. ACM Press. 1

[Hac05] Toshiya Hachisuka. *High-quality global illumination rendering using rasterization, GPU Gems 2 - Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley, 2005. 2

[HPS05] Sariel Har-Peled and Bardia Sadri. How fast is the k-means method ? *Algorithmica*, 41(3):185–202, 2005. 3

[Jen96] Henrik Wann Jensen. Global Illumination Using Photon Maps. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 21–30, New York, NY, 1996. Springer-Verlag/Wien. 1, 2

[Jen01] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., 2001. 1, 3

[Kaj86] James T. Kajiya. The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, New York, NY, USA, 1986. ACM Press. 1

[KAMJ05] Anders Wang Kristensen, Tomas Akenine-Möller, and Henrik Wann Jensen. Precomputed local radiance transfer for real-time lighting design. *ACM Trans. Graph.*, 24(3):1208–1215, 2005. 3

[KBW06] Jens Krüger, Kai Bürger, and Rüdiger Westermann. Interactive screen-space accurate photon tracing on GPUs. In *Rendering Techniques (Eurographics Symposium on Rendering - EGSR)*, pages 319–329, June 2006. 2

[Kel97] Alexander Keller. Instant radiosity. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. 2

[KGPB05] Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance caching for efficient global illumination computation. In *IEEE Transactions on Visualization and Computer Graphics*, 2005. 2

[KLS$^+$05] Joe M. Kniss, Aaron Lefohn, Robert Strzodka, Shubhabrata Sengupta, and John D. Owens. Octree textures on graphics hardware. In *ACM SIGGRAPH 2005 Conference Abstracts and Applications*, August 2005. 3

[LC04] Bent Dalgaard Larsen and Niels Jørgen Christensen. Simulating photon mapping for real-time applications. In *Rendering Techniques*, pages 123–132, 2004. 2

[LHN05] Sylvain Lefebvre, Samuel Hornus, and Fabrice Neyret. *Octree Textures on the GPU in GPU Gems 2 - Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison Wesley, 2005. 3, 5

[LP03] Fabien Lavignotte and Mathias Paulin. Scalable photon splatting for global illumination. In *GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 203–ff, New York, NY, USA, 2003. ACM Press. 2

[LPRM02] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. In ACM, editor, *SIGGRAPH 02, San-Antonio, Texas, USA*, Jul 2002. 6

[LSSS04] Xinguo Liu, Peter-Pike Sloan, Heung-Yeung Shum, and John Snyder. All-frequency precomputed radiance transfer for glossy objects. In *Proceedings of the Eurographics Symposium on Rendering*, pages 337–344, 2004. 2

[LW93] Eric P. Lafortune and Yves D. Willems. Bidirectional Path Tracing. In H. P. Santo, editor, *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, Alvor, Portugal, 1993. 1

[Mac67] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. *5th Berkeley Symposium*, -:281–297, 1967. 3

[NC02] Kasper Høy Nielsen and Niels Jørgen Christensen. Real-time recursive specular reflections on planar and curved surfaces using graphics hardware. In *WSCG (Short Papers)*, pages 91–98, 2002. 2

[Nie00] Kasper Høy Nielsen. Real-time hardware-based photorealistic rendering. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 2000. 2

[NPG03] Mangesh Nijasure, Sumanta N. Pattanaik, and Vineet Goel. Interactive global illumination in dynamic environments using commodity graphics hardware. In *Pacific Conference on Computer Graphics and Applications*, pages 450–454, 2003. 2

[PH04] Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. 2

[PO06] Fabio Policarpo and Manuel M. Oliveira. Relief mapping of non-height-field surface details. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 55–62, New York, NY, USA, 2006. ACM Press. 4

[PPD98] Eric Paquette, Pierre Poulin, and George Drettakis. A light hierarchy for fast rendering of scenes with many lights. In N. Göbel and F. Nunes Ferreira (guest editor), editors, *Computer Graphics Forum (Eurographics '98 Conference Proceedings)*, pages 63–74. Eurographics, Sep 1998. held in Li. 2

[RH01] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500, New York, NY, USA, 2001. ACM Press. 2

[RUCL03] Nicolas Ray, Jean-Christophe Ulysse, Xavier Cavin, and Bruno Lévy. Generation of radiosity texture atlas for realistic real-time rendering. In *Eurographics 2003, Granada, Espagne*, Sep 2003. 6

[SHHS03] Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph.*, 22(3):382–391, 2003. 2, 3

[Shi90] Peter Shirley. *Physically Based Lighting Calculations for Computer Graphics*. PhD thesis, University of Illinois at Urbana-Champaign, 1990. 2

[SKALP05] Lázló Szirmay-Kalos, Barnabás Aszódi, István Lazányi, and Mátyás Premecz. Approximate ray-tracing on the GPU with distance impostors. *Rendering Techniques (Eurographics Symposium on Rendering - EGSR)*, 3(24):695–704, 2005. 1, 2

[SKS02] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 527–536, New York, NY, USA, 2002. ACM Press. 2

[SWG$^+$03] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 146–155, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. 6

[SWZ96] Peter Shirley, Changyaw Wang, and Kurt Zimmerman. Monte Carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, 1996. 2

[Tat06] Natalya Tatarchuk. Dynamic parallax occlusion mapping with approximate soft shadows. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 63–69, New York, NY, USA, 2006. ACM Press. 4

[TL04] Eric Tabellion and Arnauld Lamorlette. An approximate global illumination system for computer generated films. *ACM Trans. Graph.*, 23(3):469–476, 2004. 1

[UH99] Tushar Udeshi and Charles D. Hansen. Towards interactive photorealistic rendering of indoor scenes: A hybrid approach. In *Rendering Techniques*, pages 63–76, 1999. 2

[WD06] Chris Wyman and Scott Davis. Interactive image-space techniques for approximating caustics. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 153–160, New York, NY, USA, 2006. ACM Press. 2

[WEdG$^+$06] Mathias Wein, Alejandro Conty Estévez, Alfredo de Greef, Phillip Martin, and Juan David González Cobas. YafRay, Yet Another free raytracer, Free Rays for the masses, www.yafray.org, 2006. 4

[WFA$^+$05] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: a scalable approach to illumination. *ACM Trans. Graph.*, 24(3):1098–1107, 2005. 2

[WHSG97] Bruce Walter, Philip M. Hubbard, Peter Shirley, and Donald P. Greenberg. Global illumination using local linear density estimation. *ACM Trans. Graph.*, 16(3):217–259, 1997. 2

[WKB$^+$02] Ingo Wald, Thomas Kollig, Carsten Benthin, Alexander Keller, and Philipp Slusallek. Interactive global illumination using fast ray tracing. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 15–24, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association. 2

[WTL06] Rui Wang, John Tran, and David Luebke. All-frequency relighting of glossy objects. *ACM Transactions on Graphics*, 25(2):293–318, 2006. 2

[WTP00] Alexander Wilkie, Robert F. Tobler, and Werner Purgathofer. Orientation lightmaps for photon radiosity in complex environments. In *Proceedings of Computer Graphics International 2000 (CGI 2000)*, 2000. 2

[ZSP98] Jacques Zaninetti, Xavier Serpaggi, and Bernard Péroche. A vector approach for global illumination in ray tracing. *Computer Graphics Forum*, 17(3):149–158, 1998. 2, 4