Pacific Graphics 2008
T. Igarashi, N. Max, and F. Sillion
(Guest Editors)

Volume 27 (2008), Number 7

Photon-driven Irradiance Cache

J. Brouillat P. Gautron K. Bouatouch

INRIA Rennes University of Rennes 1

Abstract

We describe a global illumination method combining two well known techniques: photon mapping and irradiance caching. The photon mapping method has the advantage of being view independent but requires a costly additional rendering pass, called final gathering. As for irradiance caching, it is view-dependent, irradiance is only computed and cached on surfaces of the scene as viewed by a single camera. To compute records covering the entire scene, the irradiance caching method has to be run for many cameras, which takes a long time and is a tedious task since the user has to place the needed cameras manually. Our method exploits the advantages of these two methods and avoids any intervention of the user. It computes a refined, view-independent irradiance cache from a photon map. The global illumination solution is then rendered interactively using radiance cache splatting.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

High-quality global illumination computation becomes unavoidable in many fields, such as cinema and video games. Today, the most commonly used global illumination computation methods are photon mapping [Jen96] and irradiance caching [WRC88]. Based on Monte Carlo ray tracing, both methods have their own advantages and drawbacks. Photon mapping is view-independent but needs a costly second pass to achieve high quality rendering, called *final gathering*. This pass is computationally expensive, which is a serious problem when the aim is interactive rendering of globally illuminated scenes. As for irradiance caching, it is a faster method but the calculation it involves is view dependent. To make this method view-independent, one has to run it for a high number of cameras to get records (cached information) over the entire scene.

Radiosity [GTG84] can also be used to compute diffuse inter-reflection. This method exhibits discontinuities due to meshing and thereby needs postprocessing. It is also time consuming. Instant radiosity [Kel97] is also a Monte Carlo based method. It generates virtual point lights (VPL) that are used to compute the diffuse inter-reflection components within a scene. Indirect glossy reflection seems difficult to evaluate using the VPLs. Laine et al. [LSK*07] propose an incremental instant radiosity that consists in reusing the VPLs and incrementally maintaining their distribution over the scene. This method considers only one light bounce to achieve interactivity.

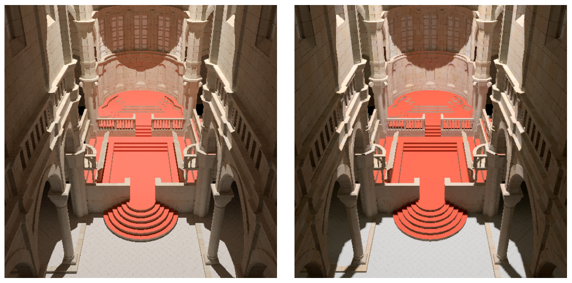
(a) (b)

Figure 1: *Sibenik Cathedral.* (a) *Pure Monte Carlo, 3 hours.* (b) *Our method, 163 seconds, including cache construction plus rendering.*

The goal of this paper is to propose a global illumination method that: (1) does not need any geometry meshing, (2) computes indirect diffuse illumination in densely occluded scenes by handling multiple light bounces, (3) allows fast acceptable preview of the computed global illumination solution, (4) renders this solution in real-time, (5) exploits spatial coherence, (6) can be part of a global illumination algorithm accounting for diffuse, specular and glossy materials, for which the indirect diffuse component (as a result of L(D|G)*DE paths, where G corresponds to moderate glossy reflections) is computed by our method while the glossy and specular inter-reflections are evaluated by using classical methods.

To achieve this goal, our method takes advantage of the photon mapping and irradiance caching techniques and combines them for fast and accurate global illumination computation. Starting with a photon map, an irradiance cache is created based on the photon positions. The irradiance assigned to each record having been computed coarsely using the nearby photons, we perform an additional refinement step for high quality rendering.

This paper is organized as follows. The next section briefly overviews photon mapping, irradiance caching and related methods. Section 3 details our approach of interactive global illumination. Section 4 presents some results.

2. Background

2.1. Photon mapping

Based on particle tracing, photon mapping [Jen96] consists of two main passes. In the first pass, photons carrying a part of the light flux are emitted from the light sources towards the scene. They can get either reflected or absorbed by the scene's objects. Impacts of photons are stored in a kd-tree, called the photon map. The indirect irradiance for a given visible point p is computed by estimating the density of photons surrounding p. However, photon mapping is a stochastic process which can introduce bias and variance in estimated irradiance: a rendered image exhibits then either noise or blur.

When high-quality rendering is needed, the second pass is performed differently, using *final gathering*. From each visible point p, rays are traced using distribution ray tracing. The contribution of each ray to the incoming radiance at p is computed by performing density estimation at the corresponding intersection point. Using several hundreds of rays, this process yields a high-quality estimate of the irradiance at the point of interest. Depending on the resolution of the image and the number of cast rays, final gathering can take up to several hours for a single image.

Several methods have been proposed to speed-up the final gathering step. In [Chr99], an approximate value of irradiance at each photon location is computed using density estimation and stored in its associated data structure. When performing final gathering, the contribution of each cast ray is computed using the irradiance value of the nearest photon to the intersection point, rather than using density estimation. In [HHS05], Havran et al. explain how to reverse the final gathering pass to exploit logarithmic behavior of kd-trees. However, accurately performing the final gathering for each pixel is still computationally expensive. The irradiance caching algorithm described hereafter performs the final gathering step only for a sparse set of visible points in the scene, hence reducing the rendering cost. Although final gathering remains very costly, practically it is almost always used as it provides artifact-free images.

2.2. Irradiance caching

The irradiance caching algorithm is based on the following observation: "the indirect irradiance tends to change slowly over a surface" [WRC88]. Therefore, this algorithm exploits spatial coherence by sparsely sampling and interpolating irradiance. The indirect irradiance is computed and cached at each sample point in an irradiance record.

Each record contains the following fields:

- p_k, position of the record,
- n_k, normal at p_k,
- E_k, irradiance value at p_k,
- R_k, harmonic mean of the distance to objects visible from p_k.

The weight of the contributions of a record k to indirect lighting at a given point p, with normal n, is given by:

$$w_k(p) = \frac{1}{\frac{\|p-p_k\|}{R_k} - \sqrt{1 - n \cdot n_k}} \tag{1}$$

The record locations are determined on-the-fly during the rendering step. Let S be the set of records surrounding a visible point p:

$$S = \left\{ k : w_k(p) > \frac{1}{a} \right\} \tag{2}$$

where a is a user defined constant.

If S is not empty and $\sum_{k \in S} w_k(p) > a$, then the irradiance estimate $E(p)$ is:

$$E(p) = \frac{\sum_{k \in S} w_k(p) E_k}{\sum_{k \in S} w_k(p)} \tag{3}$$

Otherwise a new irradiance record is computed at p and added to the cache. E and R are computed using distribution ray tracing. Note that a record k may be used for interpolation only for points within a sphere, centered at p_k with radius $a.R_k$, called *zone of influence* from now on.

In order to improve the quality of the irradiance interpolation, [WH92] and [KGBP05] introduced irradiance gradients. Note that irradiance cache can be rendered interactively on graphics hardware using radiance cache splatting [GKBP05].

2.3. Motivation

Photon mapping provides multiple-bounce global illumination. In addition, the lighting simulation pass is view-independent. However the rendering pass has to resort to computationally expensive final gathering to compute high quality images.

Radiance cache splatting [GKBP05] uses the GPU, to compute irradiance and radiance records, and considers only one-bounce reflection. Rendering is performed by splatting the records on the image plane and runs at interactive frame rates. Note that the splatting process can be used to display any cache, including those taking into account multiple bounces of light. When the camera moves, additional records have to be computed, which slows down rendering. As radiance cache splatting is view dependent, the user has to manually place virtual cameras so that the entire scene is covered by irradiance records. This is tedious and computationally expensive. However, once these records have been computed, rendering can be performed in real-time.

In this paper, we propose a method which exploits the advantages of photon mapping and irradiance caching while avoiding final gathering. Our algorithm computes an irradiance cache directly from the information contained in the photon map. The cache accounts for multiple-bounce reflections and covers most parts of the scene without any user intervention.

3. From photon map to irradiance cache

Our algorithm can be split into three parts. First, we create the photon map. Second, starting with an empty irradiance cache, record positions as well as a coarse approximation of irradiance at these records are computed from the photon map. The resulting cache can be displayed using radiance cache splatting (or converted into light maps). In the third pass, the irradiance assigned to each record is refined and irradiance gradients are computed.

Finally we discuss about density control for the photon map and its benefits for scenes with complex geometry and/or illumination.

3.1. First pass: photon map construction

We use standard photon mapping [Jen96]. Our photons store additional data that will be computed and used during the second and third passes of the algorithm:

- cumulative weight contribution of the surrounding records at the photon location: w_{sum},
- cumulative indirect irradiance contribution due to surrounding records,
- irradiance due to direct lighting (precomputed before refinement).

3.2. Second pass: irradiance cache from photon map

As photons cover all surfaces which are indirectly lit, their positions are used as potential record positions. Starting with an empty cache we add progressively new records by examining each photon in the map. This second pass is described by Algorithm 1.

To decide whether to create a new record at a given photon position p, we find the set S of the already computed records surrounding p (Equation 2). If S is empty or if the cumulative weight, $w_{sum} = \sum_{k \in S} w_k(p)$, is less than a, we

```
Input: the photon map PM
Output: an irradiance cache covering the scene
/* GenerateIrradianceCacheFrom(PM)  */
begin
    foreach photon $p \in PM$ with $w_{sum} < 1/a$ do
        find $S_n$, set of the $n$-nearest photons;
        estimate $R_k$ from $S_n$;
        find $S_{a.R_k}$, set of all photons within a sphere of
        radius $a.R_k$ around $p$;
        estimate $E_k$ from $S_{a.R_k}$;
        create record $k$ at photon location;
        foreach photon of $S_{a.R_k}$ do
            update $w_{sum}$;
            update cumulative indirect irradiance;
        end
    end
end
```

Algorithm 1: Generation of irradiance records

create a new record at position p. Because the number of photons in the photon map is usually high, building S for each photon then estimating the cumulative weight would be very expensive. Rather, for each photon we store the cumulative weight contribution of the already computed records surrounding the photon, w_{sum}. Since we start with an empty cache, w_{sum} is initialized to zero for every photon, then updated for each creation of a new record. In a similar way, each photon stores the irradiance at its position, derived from the existing irradiance cache. This value is also initialized to zero, then updated. When processing a photon, if $w_{sum} < a$, we create a new record k at the photon position and compute its associated E_k and R_k (Figure 2) from the nearby photons.

In photon mapping, irradiance at a given point is related to the density of photons around this point [Jen96]. Density estimation [Sil86] requires two parameters, kernel and bandwidth. The Epanechnikov kernel minimizes the error of density estimation, and is easy to compute. The choice of the bandwidth is more complex. A low bandwidth results in a noisy irradiance reconstruction, whereas large bandwidth in a smooth but biased estimation.

To obtain a more visually pleasant approximation, we rather choose a larger bandwidth than in standard photon mapping. Using the observations of [WRC88], we choose to set the bandwidth to the radius of the influence sphere, $a.R_k$. E_k is then estimated using only the photons lying in the influence sphere of record k.

The radius of the sphere of influence depending on R_k, we need to compute the harmonic mean distance to objects visible from p_k. This is usually performed by sampling the hemisphere around p_k and casting rays through the scene. To avoid ray casting during this pass, we estimate R_k using the neighbor photons. During the creation of the photon map, each photon is assigned a direction of incidence and the dis-

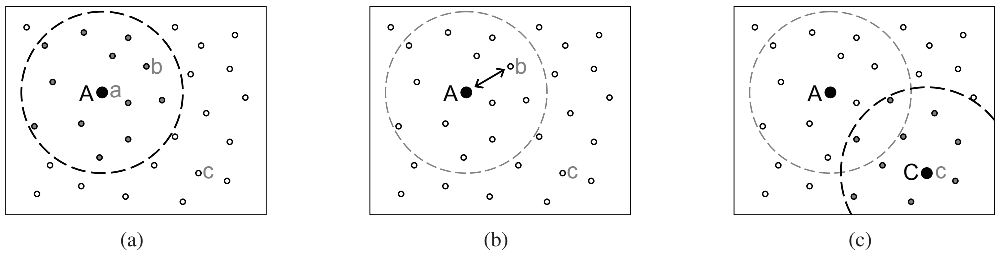

Figure 2: *We start with an empty irradiance cache, and process photons a, b and c.* (a) *As no records are present around photon* a*, a new record* A *is created at photon location.* (b) *As photon* b *is close to record* A*, $w_{sum} > a$, hence we do not create a new record.* (c) *Photon* c *is out of the sphere of A. A new record* C *is created at its location.*

tance from the last hit point. We then compute the distance between this last hit point and p_k. Using the k-nearest neighbors, we compute an estimate of R_k.

Depending on k and on the number of photons in the neighborhood of k, we may miss some objects when estimating R_k (Figure 3), due to scene undersampling. However, the impact of a bad estimation of R_k is dramatically reduced by using neighbor clamping method [KBPZ06].

When rendering from a photon map, irradiance is estimated for each pixel of the image. Hence, the noise is distributed over the pixels of the image. If E_k is computed the same way, the noise will be distributed over the irradiance records. As a record can have influence over hundreds of pixels of the final rendering, noisy records will result in unpleasant visual artifacts. As said before, during this pass we only compute an approximation of the irradiance which will be refined later.

Each record creation requires two queries to the photon map: a k-nearest neighbors search used to estimate R_k, and a query to find the set $S_{a.R_k}$ of all photons within the range of $a.R_k$ around record position. Once E_k and R_k have been computed, the new record k is added to the cache. Photons lying within the influence sphere of k get their cumulative weight updated with the new contribution of k. Then, the following photon of the photon map is processed. When all the photons have been processed, the irradiance cache is filled with records covering most of the scene. This coarse irradiance cache can already be used with radiance cache splatting for real-time rendering.

3.3. Third pass: Refining the cache

The irradiance cache computed during the second pass stores only a coarse approximation of irradiance. In this third pass we compute a more accurate value of E_k as well as irradiance gradients for each record. We perform this refinement using a slightly modified final gathering that exploits information given by the nearest photons. This information allows reducing the number of rays traced from the considered record. Note that this refinement can be performed while the user is walking through the scene interactively, the scene being rendered using the current irradiance cache. In our implementation, the visible records are refined in priority.

In the classical final gathering approach the hemisphere above the record location is sampled by tracing a high number of rays. In our approach we aim at reducing the number of rays by reusing the path of the photons lying within the zone of influence of the record. More precisely (Figure 4), for each photon lying in the zone of influence of a record k, we first retrieve the ray incident to this photon (each photon stores a direction of incidence and the distance to the last hit point). Assuming that the visibility at the location of the photon and the location p_k of the record are not significantly different, we reproject the last hit point p_{lh} of the photon to create a *virtual ray* between p_k and p_{lh}.

Let us now consider a hemisphere H_k divided into cells, located above p_k (Figure 4(a)). We determine the cell c_i intersected by the *virtual ray* (Figure 4(a)) and we assign the *virtual ray* to this cell as if an actual ray had been traced. Once all the photons within the zone of influence of record k have been processed, the hemisphere H_k is partially filled with *virtual rays* (Figure 4(b)). The remaining cells are filled by actually casting rays as in the classical final gathering approach (Figure 4(c)). Note that we assumed a negligible change of visibility across the zone of influence of a record, which may lead to errors in the computation. [BDT99] re-uses *primary* rays using radiance interpolants and error bounding, while our approach re-uses *secondary* rays during final gathering. It turns out that our naive reprojection does not entail visible artifacts. However, radiance interpolants [BDT99] could be used during the third pass.

Each cell c_i is assigned a ray and a hit point. As in [Chr99], the irradiance stored in the nearest photon data structure is assigned to this hit point. In our implementation, each photon stores a reference to the previous photon along its path. In the case of a *virtual ray*, the nearest photon to the hit point can then be retrieved immediately. The irradiance of the nearest photon is then multiplied by the diffuse reflectance at the hit point to get the radiance incoming

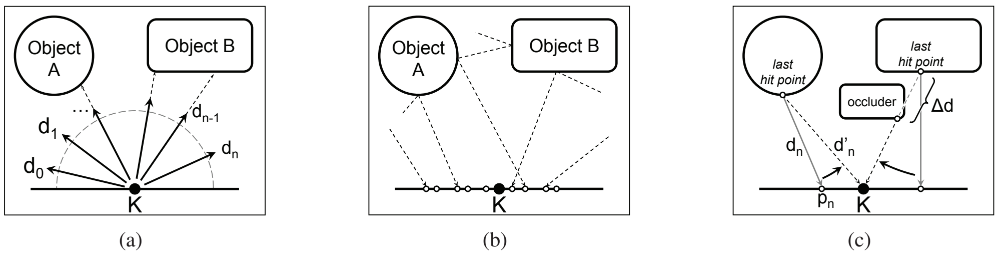

Figure 3: *Estimation of R_k.* (a) *Usually R_k is computed by casting N rays towards the scene: $R_k = \frac{1}{N}\sum_i \frac{1}{d_i}$.* (b) *In our method, we use the information contained in the photons surrounding record K.* (c) *Each photon p_n stores the distance d_n to the last hit and its incident direction. From these two data we deduce d'_n, distance between K and the last hit of the photons. Although we use photons that are close to K, we may miss some occlusions, and overestimate d_n.*

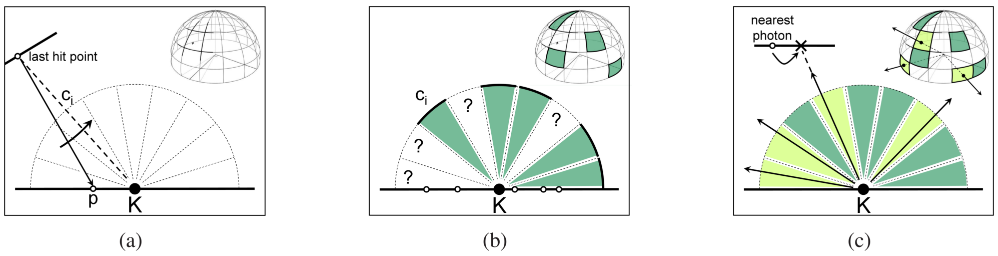

Figure 4: *Refinement of a record K.* (a) *p is a neighbor photon. Its associated hit point is reprojected onto the hemisphere above K. This reprojection yields a* virtual ray *which intersects the cell c_i.* (b) *After processing all the photons, the hemisphere above K is partially filled with* virtual rays. (c) *We shoot rays only for the cells which have not already been considered by any reprojection. The contribution of each ray is given by the photon closest to the hit point.*

through the cell. The radiances calculated for all the cells are used to compute the refined irradiance value as well as the irradiance gradients of the record using [KGBP05, WH92];

3.4. Discussion: density control

When computing a standard photon map in the first pass, two problems may arise in the second and third passes of our method: missing information in the reconstructed lighting and an insufficient number of photons for the refinement of some records. These two problems are related to the distribution of the photons in the scene. Let us recall that, in the second pass, records are placed at photon positions. If the photon density is low on a surface while R_k is not large enough, the zones of influence of the created records may not overlap. Consequently, this gives rise to holes corresponding to non covered zones for which new records have to be computed at the rendering step. One solution to this problem is to increase the number of photons in the photon map.

An ideal solution would lead to a situation in which the zone of influence of each record, whatever its size, contains approximatively the same number of photons. Suykens et al. [SW00] proposed a method to control the density of photons stored in a photon map. This can be done either during the construction of the photon map (by not storing some of the emitted photons) or during a second pass (by deleting a part of the already stored photons). The power carried by the discarded photons is then redistributed over the neighboring photons. We propose a new photon density function allowing to store more photons in the parts of the scene where R_k is low (such as corners or surfaces with complex visibility), and few photons in regions with large R_k.

A density control function is expressed in photons per unit surface. Let us recall that the zone of influence of a record k is bounded by the intersection between the surface supporting the record and a sphere of radius $a.R_k$ centered at the record position. The minimum area of this intersection is then $\pi.(a.R_k)^2$ (if the surface supporting k is planar). Then, the maximum value of the targeted density D_{kmax} around k is:

$$D_{kmax} = \frac{n}{a.R_k} \tag{4}$$

where n is the maximum desired number of photons within the zone of influence of each record. This density control function provides a photon map meeting the requirements of an accurate and efficient refinement. Note that for complex scenes, a high number of photons have to be cast and the resulting photon map may require an out-of-core memory management. To overcome this problem, we apply our den-

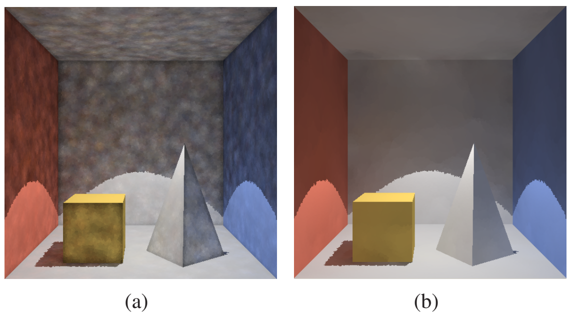

Figure 5: *Images rendered by tracing only primary rays and assigning the irradiance value of the nearest photon to each intersection point.* (a) *Using [Chr99], 150k photons, irradiance values are computed in 1.87 s.* (b) *Our method (end of the second pass), using 150k photons, the time of the second pass is 0.89 s.*

sity control function to obtain a fully reduced photon map which can now be stored in-core. Our algorithm can then efficiently generate a high quality irradiance cache.

We have implemented our density control function (using [HHS05]) and the obtained results confirmed the fact that it is efficient only for complex scenes and/or lighting, this is why it is not further discussed in our results.

4. Results

All results were gathered on a Pentium4 3.8Ghz computer with 2GB RAM and an nVidia GeForce 7800GTX 512MB.

4.1. General observations

As we use the photon locations as potential positions of irradiance records (Figure 6), we cast a high number of photons and account for many light bounces (up to 9 in our test scenes). However our method reuses the photon paths to refine irradiance records during the third pass, hence compensating for the overhead related to the high number of photons and light bounces. The results are given in Tables 1 and 2.

At the end of the second pass, our algorithm provides an irradiance cache covering most part of the scene. Though unrefined, this coarse cache can be previewed in real time using *radiance cache splatting*. In addition, for each created record, its indirect lighting contribution is distributed among the photons lying within its zone of influence. Note that the resulting photon map (augmented with irradiance values) is similar to that computed in [Chr99], but the irradiance values are computed differently. If we render the scene by tracing only primary rays and assigning the irradiance of the nearest photon to each intersection point, our method outperforms [Chr99] in terms of preview quality (Figure 5).

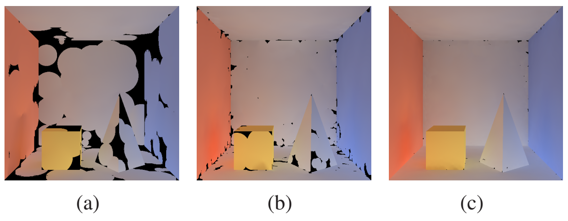

Figure 6: *Influence of the number of photons on the coverage of the scene by the computed irradiance cache.* (a) *500 photons.* (b) *5k photons.* (c) *50k photons.*

4.2. Test scenes

4.2.1. Cornell Box

This very simple scene contains one point light source and no textures to illustrate the main ideas behind our method. As shown in Figures 6 and 7, a high number of photons increases both the coverage of the cache and the quality of the unrefined records while speeding up the refinement process. Table 1 shows a breakdown of the rendering times for each step of the algorithm. The resulting cache is generated in 6.75 seconds and contains high quality irradiance values, which can be rendered at 40 fps.

We compared our refinement method used in our third pass with the classical irradiance computation using stratified sampling for the final gathering. As shown in Table 2, the reuse of the photon paths allows us to save 39.4% of rays without compromising the rendering quality. We also compared our method with classical per-pixel path tracing. Our method shows no significant error increase compared to the classical irradiance caching algorithm.

4.2.2. Sponza Atrium

This more complex scene features two point light sources and textures.

Our method saves 52% of rays during the third pass, hence speeding up the third pass by 47%. The obtained overall speedup is then 40.1% compared to the use of stratified sampling (final gathering) during the refinement process. The time for computing the irradiance cache is 163 seconds while the rendering frame rate is 5.6 fps.

4.2.3. Sibenik Cathedral

This scene contains many small features such as railings and stairs. Therefore, the constructed cache contains more records than the previous scene. The construction of the photon map takes 29.8 s.

In term of time and quality of preview, we compared two methods: (1) a classical photon mapping algorithm with only primary rays, the irradiance assigned to intersection points being computed using density estimation, (2) our approach

Scene (polygons)	First Pass: **Photon Map Generation**		Second Pass: **Irradiance Cache Generation**		Third Pass: **Refinement**	Total computation	FPS
	time (s)	photons (bounces)	time (s)	records	time (s)	time (s)	
Cornell (32)	0.81	250k (4)	1.20	2,972	4.64	6.75	40
Sponza (66k)	28.3	4M (9)	11.8	55,649	123	163	5.6
Sibenik (73k)	29.8	4M (9)	11.9	73,860	162	204	4.7

Table 1: *Timing of each pass of our method for different scenes. The fps value is the frame rate achieved using* radiance cache splatting *with the computed cache, before and after the refining pass.*

Scene (polygons)	Nb. photons (bounces)	Number of records	Standard Final Gathering		Our method		% Saved Rays	% Saved Time (s)
			time (s)	rays	time (s)	rays		
Cornell(32)	250k (4)	2,972	7.14	1,325,335	4.64	802,640	39.4	35
Sponza(66k)	4M (9)	55,649	232	24,205,162	123	11,628,159	52	47
Sibenik(73k)	4M (9)	77,360	285	31,583,736	162	17,006,361	46.2	43.2

Table 2: *Performance breakdown.*

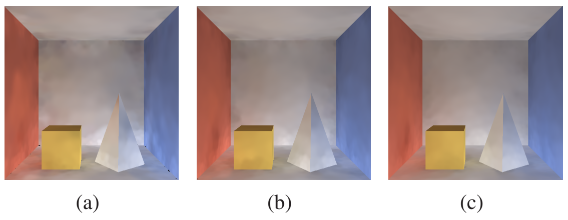

Figure 7: *Influence of the number of photons on the computed coarse irradiance cache.* (a) *50k photons.* (b) *150k photons.* (c) *250k photons: the computed irradiance cache can be used as a good preview of the illumination of the scene.*

(without the third pass) computing records from only the photons within the view frustum. The first approach takes 6.23 seconds while the second requires 2.87 seconds to compute the irradiance cache that can be rendered in real-time with better quality.

As irradiance caching methods are view dependent, we derived a view depend version of our approach to compare it to these methods. Let us call it VDPDIC (View Dependent Photon Driven Irradiance Cache). VDPDIC computes an irradiance cache by using only the visible photons. Let us now consider an irradiance cache method that we call CNIC (Close Neighbor Iradiance Cache). CNIC traces a ray through each pixel and computes (requiring an octree query), when needed, a new record whose irradiance is computed by performing final gathering at the intersection point. Each ray shot by the final gathering process intersects the scene at a point p for which we compute an irradiance value by picking up the one stored at the neighbor photon. For the two methods the image resolution is 512*512, and the number of shot rays for each final gathering is 384. The results obtained for these two methods are given by Table 3. Using reprojection in VDPDIC allows to get a saving of 40% for irradiance cache construction as well as for final gathering rays. Finally, VDPDIC is 2.5 faster than CNIC. Recall that this comparison concerns only a single view, while our method is capable of computing a single cache for all views.

5. Conclusion

We proposed a method for building an irradiance cache from a photon map. The two main advantages of our approach is that it makes possible good quality and fast preview of an existing photon map and, unlike [WRC88], it generates an irradiance cache that is view-independent, say it covers the entire scene. Our algorithm consists of three passes: (1) construction of the photon map, (2) creation of records, (3) refinement of the cached irradiance values. *Radiance cache splatting* [GKBP05] can be used for fast preview of the global illumination solution (end of the second pass) and for high quality interactive rendering (end of the third pass). We have also proposed and implemented a density control function for photon map construction. After many experiments we ascertained that it is efficient only for complex scenes needing high number of photons as it may avoid out-of-core memory management by drastically reducing the size of the generated photon map. To make our approach more efficient, one solution is to either improve the density control process (while keeping the same density function) or propose a more efficient method of guiding the photons emitted from the light sources.

[LZT*08] describes a global illumination method based on point sampling. Likewise, our approach make use of points (records) to cache the global illumination solution. For the two approaches, point placement is performed in a similar way. First, a set of candidate points is determined by particle tracing, then only a subset of these candidates is

Method	# Records	# Octree queries	# Density estimation	# Nearest photon queries	# Cast rays	Total time for creating the cache (s)
CNIC	15,783	262,144	2,002,300	6,043,835	8,323,857	145
VDPDIC (w/o reproj.)	15,905	0	207,517	6,071,735	8,036,377	96,9
VDPDIC (with reproj.)	15,905	0	181,360	2,598,815	4,320,403	57.8

Table 3: *Results for single view.*

kept, based on some heuristics. We use the reflection properties of the materials to guide the particle tracing as well as a criterion based on harmonic mean distance [WRC88], while [LZT*08] makes use of importance as well as another criterion relying on Poisson disk distribution with a new distance function. However, global illumination is determined differently with the two approaches. Also, the generalization of our method to dynamic scenes will be considered in future research.

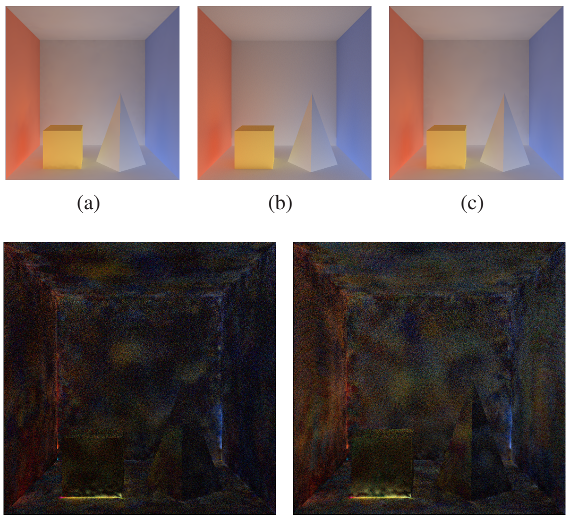

Figure 8: (a) *Standard irradiance caching algorithm.* (b) *Raytraced reference solution, 900 paths per pixel, 2500 s.* (c) *Our method, with reuse of rays during refinement.* (d) *Differences between (a) and (b).* (e) *Differences between (b) and (c). These differences are multiplied by 15.*

References

[BDT99] Bala K., Dorsey J., Teller S.: Radiance interpolants for accelerated bounded-error ray tracing. *ACM Trans. Graph. 18*, 3 (1999), 213–256.

[Chr99] Christensen P. H.: Faster photon map global illumination. *Journal of Graphics Tools 4*, 3 (1999), 1–10.

[GKBP05] Gautron P., Krivanek J., Bouatouch K., Pattanaik S.: Radiance cache splatting: a gpu-friendly global illumination algorithm. In *Proc. of EGSR* (2005), p. 36.

[GTG84] Goral C., Torrance K., Greenberg D.: Modelling the interaction of light between diffuse surfaces. In *Proc. of SIGGRAPH* (1984), pp. 212–222.

[HHS05] Havran V., Herzog R., Seidel H.-P.: Fast final gathering via reverse photon mapping. *Proc. of Eurographics) 24*, 3 (2005), 323–333.

[Jen96] Jensen H. W.: Global illumination using photon maps. In *Proc. of the EGWR* (1996), pp. 21–30.

[KBPZ06] Krivanek J., Bouatouch K., Pattanaik S. N., Zara J.: Making radiance and irradiance caching practical: Adaptive caching and neighbor clamping. In *Proc. of EGSR* (2006).

[Kel97] Keller A.: Instant radiosity. In *Proc. of SIGGRAPH* (1997), pp. 49–56.

[KGBP05] Krivanek J., Gautron P., Bouatouch K., Pattanaik S.: Improved radiance gradient computation. In *Proc. of SCCG* (2005), pp. 155–159.

[LSK*07] Laine S., Saransaari H., Kontkanen J., Lehtinen J., Aila T.: Incremental instant radiosity for real-time indirect illumination. In *Proc. of EGSR* (2007), pp. 227–286.

[LZT*08] Lehtinen J., Zwicker M., Turquin E., Kontkanen J., Durand F., Sillion F., Aila T.: A meshless hierarchical representation for light transport. In *Proc. of SIGGRAPH 2008)* (2008), vol. 27.

[Sil86] Silverman B. W.: *Density Estimation for Statistics and Data Analysis*. 1986.

[SW00] Suykens F., Willems Y. D.: Density control for photon maps. In *Proc. of EGSR)* (2000), pp. 23–34.

[WH92] Ward G., Heckbert P.: Irradiance gradients. In *In Proc. of EGWR* (1992), pp. 85–98.

[WRC88] Ward G. J., Rubinstein F. M., Clear R. D.: A ray tracing solution for diffuse interreflection. *Proc. of SIGGRAPH* (1988), 85–92.