Incremental Updates for Rapid Glossy Global Illumination

Xavier Granier, George Drettakis

EUROGRAPHICS 2001 / A. Chalmers and T.-M. Rhyne
(Guest Editors)

Volume 20 (*2001*), *Number 3*

Incremental Updates for Rapid Glossy Global Illumination

Xavier Granier and George Drettakis

iMAGIS–GRAVIR/IMAG-INRIA
2004 route de Lucioles, Sophia-Antipolis, F-06902, France†

Abstract

We present an integrated global illumination algorithm including non-diffuse light transport which can handle complex scenes and enables rapid incremental updates. We build on a unified algorithm which uses hierarchical radiosity with clustering and particle tracing for diffuse and non-diffuse transport respectively. We present a new algorithm which chooses between reconstructing specular effects such as caustics on the diffuse radiosity mesh, or special purpose caustic textures, when high frequencies are present. Algorithms are presented to choose the resolution of these textures and to reconstruct the high-frequency non-diffuse lighting effects. We use a dynamic spatial data structure to restrict the number of particles re-emitted during the local modifications of the scene. By combining this incremental particle trace with a line-space hierarchy for incremental update of diffuse illumination, we can locally modify complex scenes rapidly. We also develop an algorithm which, by permitting slight quality degradation during motion, achieves quasi-interactive updates. We present an implementation of our new method and its application to indoors and outdoors scenes.

Keywords: Global Illumination, Rapid/Interactive Updates, Radiosity, Particle Tracing, Textures, Reconstruction

1. Introduction and Motivation

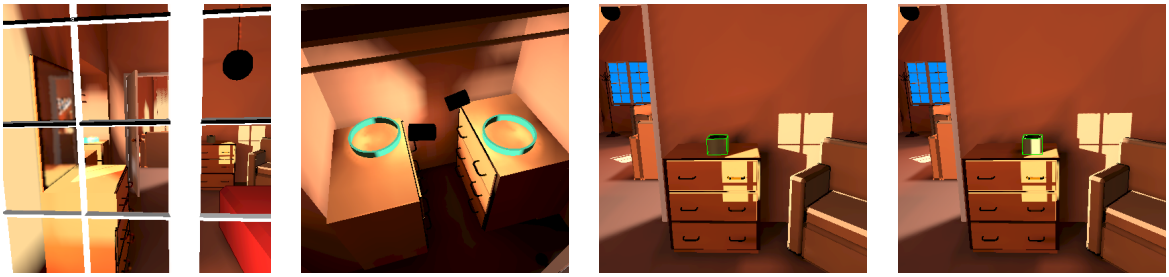
Global illumination algorithms which include specular effects are currently very computationally expensive. Despite many recent advances, it is still very tedious to run such methods in a trial-and-error manner to correctly position lights and objects or try out different lighting designs. Ideally, we want to provide a way for a user to modify a scene with such global and non-diffuse effects and rapidly or even interactively see the resulting changes in lighting (shadows, reflections, caustics etc.).

The efficient treatment of global specular effects, such as caustics, is still problematic for scenes which are moderately or very complex. Most existing approaches are either ray-based[15], view-dependent solutions, and thus cumbersome for walkthrough and interactive use when modifying the scene, or depend on particle tracing for all light transport[29] including diffuse light, also making interactivity hard to achieve. For diffuse-only global illumination, previously developed solutions[2, 7, 6, 20], allow the user to move objects at interactive rates. Also, recent methods based on ray-tracing allow a limited amount of interactivity and scene modification[17, 28]. These approaches are however typically limited in the lighting effects they can reproduce, in the resulting image resolution, and can suffer from visual artifacts. In summary, it is still currently almost impossible to move around a globally illuminated scene, containing both global diffuse lighting and global specular effects (such as caustics) and rapidly move objects in the scene to see the resulting effects.

We present an initial solution to these shortcomings. We build on previous work, which combined the advantages of hierarchical radiosity for diffuse transport and limited particle tracing for non-diffuse interactions[10], which we will refer to as the *unified algorithm* from now on. We also extend the line-space hierarchy approach[6] (LSH), and use it for incremental diffuse transport.

The main problem of the unified algorithm as presented previously[10], was its limitation to moderately complex scenes. This was due to the fact that caustics were reconstructed on the radiosity-based mesh, resulting in the

† iMAGIS is a joint research project of CNRS/INRIA/UJF/INPG. E-mail:{Xavier.Granier|George.Drettakis@sophia.inria.fr
http://www-imagis.imag.fr/



(a) (b) (c) (d)

Figure 1: *(a) Global illumination solution computed in 35 min. There are lights on in the house, as well as sunlight which is coming in through the window, hits the mirror on the left and reflects on the wall dresser and armchair on the right. (b) details of caustics in the alcove on the left. (c)-(d) The container on the dresser can be moved at 20 sec/frame. We can also achieve 3 sec/frame updates with slightly lower quality (see Fig. 5).*

need for excessive subdivision leading in turn to unmanageable memory and computational requirements.

We present a solution to this problem by reconstructing non-diffuse effects separately on caustic textures, when needed. Our algorithm can thus either choose from the "spread-out" reconstruction on a relatively coarse radiosity mesh, or fine caustics on a texture, which we call caustic textures. Caustic textures are not part of the global illumination solution, since they are part of the final image reconstruction; as a result the unified algorithm remains valid for all light transfers. We present methods to choose the appropriate representation (mesh or texture), to choose the resolution of the texture and to perform appropriate reconstruction.

Next, we present a novel solution which restricts the number of particles emitted due to a local modification in the scene. A dynamically modified octree is used to compactly store the information of particles transiting in the scene. Fast inserts and deletes are achieved with appropriate data structures and update algorithms. We then introduce a progressive update algorithm, which allows quasi-interactive updates including non-diffuse effects by degrading quality during object motion. When motion stops, the image quality is restored.

Finally, we have reduced the memory requirements of the original line-space hierarchy[6], and improved the performance in the case of clusters.

We will present results of our implementation which show that for moderately complex scenes containing many caustic effects we can achieve a good quality solution with reasonable computation time. Fig. 1 shows an example of solution computed in approximately 35 min†. Note the detailed specular reflections of morning sunlight off the mirror on the left onto the wall and dresser on the right as well as on the dressers due to the spotlights (see Fig. 1(b)). We can then rapidly modify the scene locally, achieving rapid updates. The container on the dresser can be moved at 20 seconds per frame (Fig. 1(c)-(d)). If we degrade quality slightly during motion (see Fig. 5) we can achieve quasi-interactive updates at around 3 seconds per frame.

† All images were generated with standard OpenGL hardware rendering using raytracing with the Render Cache to fill in the $(L|D)S^+E$ paths (such as the mirror). All timings are on a Pentium III 733MHz PC (Sgi 330 with a VR3 (Nvidia Quadro) graphics card.)

2. Related work

For diffuse-only transfer, radiosity methods (e.g., [8, 4]) can produce high-quality images. Combined with clustering[24, 21] (HRC), hierarchical radiosity[12] can simulate radiosity for large environments rapidly. We use such an approach for diffuse light transfer.

Much work has been done on global illumination including non-diffuse light transport. In particular, several extensions to the radiosity algorithm have been developed (e.g., [14, 27, 23, 22, 3]) which either use explicit storage for directional information and/or a second pass to incorporate non-diffuse effects. Two methods are particularly related to our approach: The Photon Map approach[15] and density estimation[29] which both use particles or photons for light transport. The photon map relies on a relatively expensive Monte-Carlo ray-cast for diffuse light gathering and final image generation. Density estimation uses sophisticated reconstruction techniques, similar to those we will use, and involved mesh decimation techniques for final representation. However, the computational and memory requirements of all these approaches are quite high, precluding interactive or near-interactive use.

Hardware acceleration and multi-pass methods have also been investigated (e.g., Diefenbach et al.[5] and Stürzlinger and Bastos[26]). These approaches tend to be limited in the phenomena they can handle however, lacking generality.

Recently, researchers have turned to ray-tracing based approaches to achieve interactivity. Examples include the parallel ray-tracing system of Parker et al.[17] and the Render Cache[28]. As mentioned in the introduction, these approaches

suffer somewhat from limitations in image resolution, visual artifacts and in the type of lighting simulation they can achieve.

As we will discuss later, we will use an spatial subdivision structure for dynamic updates. Such structures have been recently used to further accelerate interactive ray-tracing for dynamic scenes[18]. We will also be using textures to represent caustic or specular effects. The origins of this idea are actually quite old, since similar ideas have been developed by Heckbert in his radiosity textures approach[13], Myszkowski and Kunii[16] and Bastos et al.[1] among others.

In what follows we use Heckbert's[13] regular expression notation for light paths and light transport. L is used for the light, E the eye, S a non-diffuse transfer, D a diffuse transfer, while "*" represents zero or more bounces, "+" at least one bounce and "|" is the "or" operator. Thus for example LD^*E are light paths leaving the light, bouncing off zero or more diffuse objects and arriving at the eye. These paths are well treated by radiosity. All possible light paths can be described by $L(D|S)^*E$.

2.1. HRC, LSH and the unified method

As mentioned above, we build on the line-space hierarchy[6] and the unified algorithm we developed previously[10], which integrates hierarchical radiosity for diffuse transfer and particle tracing for non-diffuse transfers. To facilitate comprehension we briefly overview the basic structures and algorithms of HRC, as well as the salient points of the two methods we build on.

For diffuse transport, HRC starts by building a complete hierarchy of *elements*, from input objects, to *clusters* of objects hierarchically grouped into a root cluster containing the entire scene. Light transfers between such elements are represented by *links*, which are established according to a refinement criterion, thus representing light transfer at an appropriate level. After this refinement and consequent subdivision of surfaces as needed, light is *gathered* across links at different levels of the hierarchy, and a *push-pull* pass ensures that radiosity values remain consistent over the entire hierarchy.

The links of a radiosity system induce a subdivision of the line-space of the scene, following the flow of light between scene elements. Drettakis and Sillion[6] have augmented links with an explicit representation of all lines passing between two elements via a *shaft* [11]. When an object moves, they efficiently identify the links affected, and the parts of the radiosity hierarchy which are modified, by hierarchically descending in this line-space. This traversal permits efficient identification of the paths in the hierarchy which are modified. Fast resolution of the modified part of the system is achieved, by restricting push-pull to the parts of the hierarchy which have been marked as changed. We will use the line-space hierarchy to accelerate diffuse light updates.

In the unified lighting algorithm[10], the cluster hierarchy is built as in HRC. Each link is examined, and the refinement process decides whether to refine based on the chosen criterion. A "gather" step follows, where each link is checked: for links leaving diffuse objects and arriving on specular objects, diffuse to specular transfers are performed by particle tracing. The use of the link structure restricts the number of particles emitted since only a limited part of space is treated. This also substantially accelerates particle tracing itself by accurate visibility classification. Particles are subsequently propagated into the environment both by reflection and refraction, affecting other regions of the scene.

A full iteration of this process is completed by updating the hierarchy. Particles are first placed at an appropriate level in the hierarchy (on objects or at subdivided elements). The contribution of particles is then added to the diffuse light at the leaves of the hierarchy; all transfers to diffuse surfaces are thus recorded. This can be also considered as a reconstruction step for particles. In this manner all $L(D|S)^*DE$ light transfer is accounted for.

Rendering is performed by reconstructing the radiosity values on a fine grid or using the mesh subdivision to generate triangular elements. Specular paths $(L|D)S^+E$ to the eye must then be added by ray-tracing.

3. Dynamic Caustic Textures

The reconstruction of caustics according to the subdivision of hierarchical radiosity as done in the original unified method[10] leads to problems for complex scenes. The representation of fine caustics containing significant high-frequency detail, leads to meshes which are highly subdivided. This implies an unacceptable memory cost, as well as a higher computational cost for global illumination solutions for all steps involving hierarchy traversal (refinement and push-pull).

To solve this problem we decided to use caustic textures where needed, similar in spirit to Heckbert[13]. This idea is closely related to that of caustic maps of Wann Jensen[15]; nonetheless, we choose a representation linked to the object as opposed to an independent spatial data structure. This allows direct use of graphics texture hardware for display. We also provide automatic solutions for a number of important parameters as described below. It is important to also note that our goal is very different from that of Wann Jensen, since we create solutions which can subsequently be viewed interactively in walkthroughs, by integrating the strengths of HRC and particle tracing. In addition, our solution is dynamic, since we will be modifying the textures during user interaction with the scene.

The following three problems need to be solved for caustic textures. The first is the choice of which representation to use: for low-frequency, "spread out" illumination, the representation of the DS^*D transfers can be achieved satisfacto-

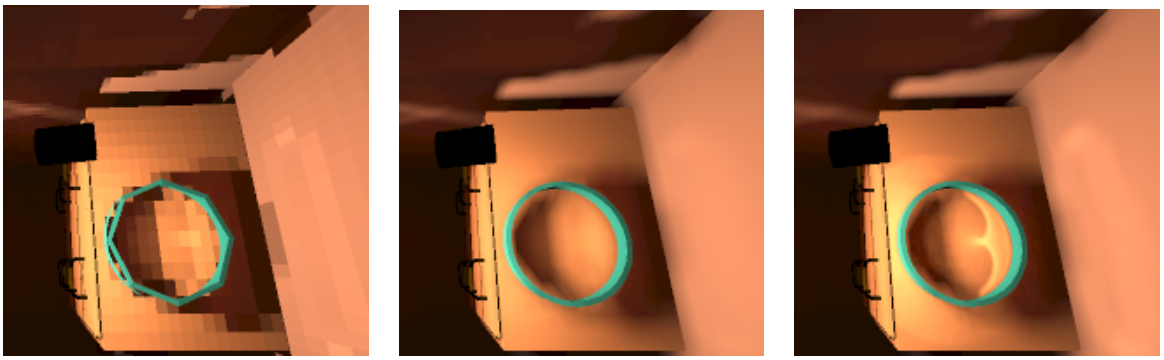


Figure 2: *A dresser with caustics. (left) constant (flat) reconstruction, (middle) reconstructed smoothly using the radiosity mesh, (right) the same solution reconstructed by caustic textures.*

rily by the HRC mesh as in the original unified algorithm[10]. We thus develop a criterion to determine whether to use the mesh or a texture. Second, if we decide that a texture is required, we need to determine its resolution; we use a variance-based approach to achieve this. Finally, we need to reconstruct the illumination function before creating the texture used for display, and restrict it to the part which is useful.

Note that the use of caustic textures does not affect the light transport resolution problem as performed by the uniform algorithm. Particles are still stored and integrated into the radiosity hierarchy as in the unified algorithm[10], and contribute normally to the solution. The difference is that the surfaces are subdivided much more coarsely and as a consequence particles are often placed higher up in the hierarchy. The result is of course faster computation.

The integrated use of textures in this context enables the treatment of complex scenes with involved secondary global illumination, which also contain many secondary specular effects such as caustics. The restriction of textures to the part which is useful (see Section 3.3), further enhances this capacity, since we only build small textures where needed.

3.1. Choice of representation

Intuitively, we will use textures if a large number of particles have arrived on an object, and if they do not have uniform power values.

After particle tracing, particles are accumulated on the hierarchy elements. We traverse the hierarchy, and at each top-level object (input polygon etc.), we know the number of particles which have arrived. If this number is above a certain threshold (typically 10), we create a "pre-texture" or histogram, which we use to evaluate whether it is appropriate to use a caustic texture. The dimensions of the histogram are determined by the method described below (Section 3.2), which is also the method for choosing the caustic texture resolution.

We accumulate the particle flux arriving at each element of the histogram. Each particle p has power ϕ_p. For a histogram of dimension n x m, the element $(i,\ j)$ of the histogram contains:

$$p(i,j) = \sum_{p \in (i,j)} \frac{\phi_p}{A_{tex}/(n\,m)} = \sum_{p \in (i,j)} \frac{\phi_p\, n\, m}{A_{tex}} \qquad (1)$$

where A_{tex} is the area of the texture, $p \in (i,j)$ denotes that the position of particle p is in the texel (i,j) and $A_{tex}/(n\,m)$ is the area of a "pixel" of the histogram. For quadrilaterals A_{tex}, is equal to the area of the polygon; for all other objects a bounding quadrilateral is used.

We estimate uniformity of particle power as follows:

$$V = \frac{(p_{min} - p_{max})}{p_{max}} \qquad (2)$$

where p_{max} and p_{min} are the maximum and minimum values of the histogram. We use the following test to determine whether or not to create a texture:

$$V < \eta \qquad (3)$$

where η is a user defined threshold, representing percent acceptable variation. If Eq. (3) is true, we do not create the texture.

The advantage of using caustic textures can be seen immediately. Consider Fig. 2, where on the left we show a caustic reconstructed on the mesh of the drawer and on the right we show the same result using a caustic texture. For the same number of particles and the same quality unified solution, we show a clear gain in visual quality.

3.2. Finding the caustic texture resolution

The choice of caustic texture resolution is critical, since it has direct impact on memory and computation cost, as well as on display speed and visual quality. Choice of caustic texture resolution can be considered as part of the reconstruction process. In essence, a large texture resolution requires a large number of particles to obtain a low-noise result. Alternatively, we can consider a low-resolution texture as a smoothing filter, thus diminishing noise, but increasing bias.

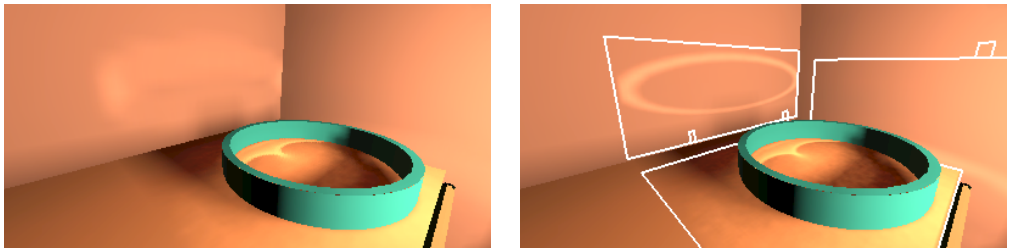


Figure 3: *(Left) Caustic shown without restriction (right) the restricted caustic. The restricted regions are outlined in white.*

Building on this idea, we decide on caustic texture resolution using variance in density estimation context, as developed by Walter et al.[29].

If the function we are estimating is f, variance is given as follows:

$$\forall\, x, Var(f(x)) \;\leq\; \frac{\int K^2}{n\,h^2} f(x) \;\leq\; \frac{\int K^2}{n\,h^2} \tag{4}$$

where K is the normalized kernel, n is the number of particles and h is the size of the kernel.

Since we are on a finite texture, we take the size of the kernel to be half the size of a texel:

$$h \;=\; \frac{1}{2}\sqrt{\frac{1}{2^{2d}}}, \tag{5}$$

where d is the resolution of the texture. We will choose the resolution so that variance is less than a given threshold ε:

$$\frac{K\,2^{2d}}{n} \;\leq\; \varepsilon. \tag{6}$$

Thus the resolution d is chosen as follows:

$$d \;\geq\; \frac{ln(\frac{n\varepsilon}{K})}{2ln2} \;-\; 1. \tag{7}$$

The value of K is a constant[29], depending on the chosen kernel. We use a box filter in our examples. The threshold ε is defined by the user, and can be thought of as a "variance threshold". We typically use a value of 1. This process can be applied to both dimensions of the texture.

3.3. Reconstruction and Restriction

If we decide to create a caustic texture, we first choose its resolution as described above. We then apply a reconstruction process to create a caustic texture which will be used for display. Again, we are in the context of a trade-off between noise and bias.

We use a smoothing kernel similar to that used in photon maps[15]. We first find the maximum value of the histogram p_{max}. For each texel we spiral out, accumulating particles in an auxiliary buffer p' as follows:

$$p'(i,j) \;=\; \frac{\sum_{r\leq R} p(i,j)}{N} \tag{8}$$

where R is a maximum permitted radius, r is the current radius, and N is the number of texels under the kernel used. However, this accumulation is terminated if the $p'(i,j) \;\geq\; p_{max}$.

Constructing a texture on an entire input surface such as a polygon can often be wasteful. Consider for example the case shown in Fig. 3, where only small regions of the wall and the dresser top are lit by the reflection from the caustics and specular reflections due the spot light off the specular ring. Instead, we can restrict the texture to the part of the surface which is actually affected by the caustic.

To do this, we use the maximum texture value as explained above. We start traversing the texture until we find a value which is at least as large as a percentage of the maximum defined by the user (typically between 1-5%). We then flood fill the texture for as long as we are above this value. This process gives us the maximum and minimum 2D texture coordinates in which "valid" caustic values exist. We use these to construct a bounding box, and then recreate the texture using the same resolution, but limited to this region. If multiple regions exist on a surface, we restart the process at the point we previously started accumulation, after setting all pixels already collected to zero.

The result of the restriction is evidently higher quality caustics for the same texture resolution. This can be seen in Fig. 3(right) on the wall.

4. Incremental Particle Tracing

An important goal of our new approach is to restrict the number of particles traced when moving an object, to achieve rapid updates. In the original unified algorithm[10], all particles were re-shot during object motion. While this is a feasible solution for small scenes, it quickly becomes impractical for even moderately complex environments. To achieve fast or even interactive updates, we first use the line-space hierarchy for incremental updates of the diffuse transport, and then restrict the number of particles which are re-sent during object motion.

As with diffuse transfer, the direct effect of object motion on specular transfer can be limited in space (although less so than for diffuse). The idea is to find all the particle paths which are affected by object motion, and only re-emit these

particles. We have chosen to achieve this efficiently using an adaptive spatial data structure, i.e., an octree. We will store references to links in the octree, from which we can reconstruct the particle paths.

4.1. Dynamic Octree and Initialization

Initially, the octree is empty. When an object is selected for motion ("dynamic object"), we subdivide the octree cells around it up to a maximum depth. The octree is thus adapted to the motion and the position of the dynamic object. This is illustrated in Fig. 4.

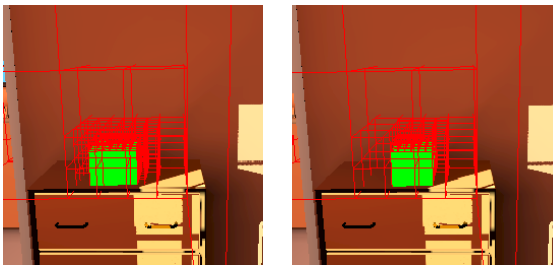


Figure 4: *(Left) The Octree in initial position (Right) Octree after movement. Notice simplification for the previous object position.*

During particle tracing, links are inserted into the octree. For each octree cell, we maintain a hash table, indexed by receiver element r on which the link is stored, and the link l. For the initial particle direction, and every subsequent reflected or refracted particle, we find the cells of the octree affected. The link in question is then inserted in the hash tables of the corresponding cells. Evidently, we only consider links which involve a diffuse to specular (DS^*) transfer. When accessing the hash tables to identify links affected by movement, the additional element key permits immediate identification of the element affected, which is required to remove the previous particle information and perform reemission. It also saves memory since we do not have to store the element explicitly.

4.2. Incremental Update

For a given movement of an object, we know the previous and new positions. The dynamic object is re-inserted into the octree in the new position. The octree is traversed to perform this insertion, and each cell is marked either unaffected, changed to contain the object, or changed to no longer contain the object. If a cell contained the object previously, but no longer contains it, its children are simplified (see Fig. 4(right)).

For changed cells we collect the links in the corresponding hash tables. For each such link, we remove the particles and their contribution on the elements affected. A chain of elements is maintained with each particle, and the particles are removed from all following elements which were affected by reflection(s) and/or refraction(s) of this particle. The particles associated with this link are then re-emitted, and the elements marked as changed. As is the case for the diffuse line-space hierarchy (see Section 2.1 and 5), we mark the part of the line-space hierarchy affected as changed. As a result, during push-pull radiosity values are updated consistently. Recall that only part of the hierarchy is updated. To obtain secondary effects due to DS^*D transfer, multiple gather/push-pull steps must be applied.

4.3. Interactive Update with Quality Degradation

Despite an order of magnitude speedup compared to rerunning the entire solution, the update rates for the solution presented above are still quite high (around 20 seconds for the indoor scene). To achieve quasi-interactive rates we can limit the number of particles re-emitted during motion, with some loss in quality.

During motion, we keep track of all links which should be updated, but only re-emit a small, user-controllable percentage of them. When motion stops, we simply re-emit all particles which were collected during the motion, but not emitted. The results of an example are shown in Fig. 5.

5. Improved Incremental Diffuse Solution

As mentioned in Section 2.1, we use the line-space hierarchy[6] to perform rapid incremental updates for diffuse illumination. We have nonetheless introduced a few improvements to that approach.

One problem with the original method was the requirement that shafts be computed and stored for the entire link hierarchy. This problem was also identified by Schoeffel and Pomi [19], who used a prediction approach to limit the precomputation.

We simply do not store shafts. We use the line-space coding used in[9], which does not require the numbering scheme of the original approach[6], and traverse the hierarchy by keeping track of the receiver and sender elements of the links. During this traversal we maintain a list of potential blockers created by the shaft-culling process. Shafts are thus built incrementally only on the list of current potential blockers, making the operation much cheaper. The memory savings are very significant, since the storage of all shafts results in a prohibitive memory overhead, and renders the approach inappropriate for complex scenes (see[19] for analysis of the memory requirements of LSH).

Another drawback of the original line-space approach was that when treating clusters, irradiance had to be "immediately pushed" down to the surfaces. This results in multiple hierarchy traversals, with consequent computation overhead. Instead, we maintain a list of the add and remove irradiance operations at the level of the link being modified. During push-pull we accumulate these lists which are pushed down

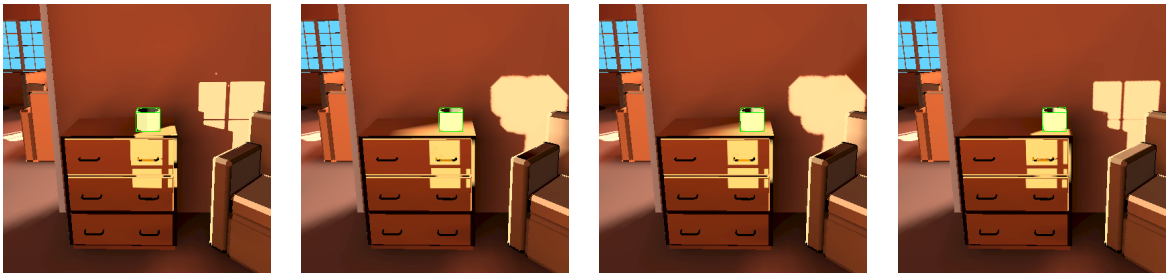


Figure 5: *(Left) Initial position with good quality shadow. (Middle) two intermediate positions during degraded quality motion at 3-4 sec/frame. (Right) Motion has ended, 20 seconds later we obtain the quality shown, equivalent to normal updates.*

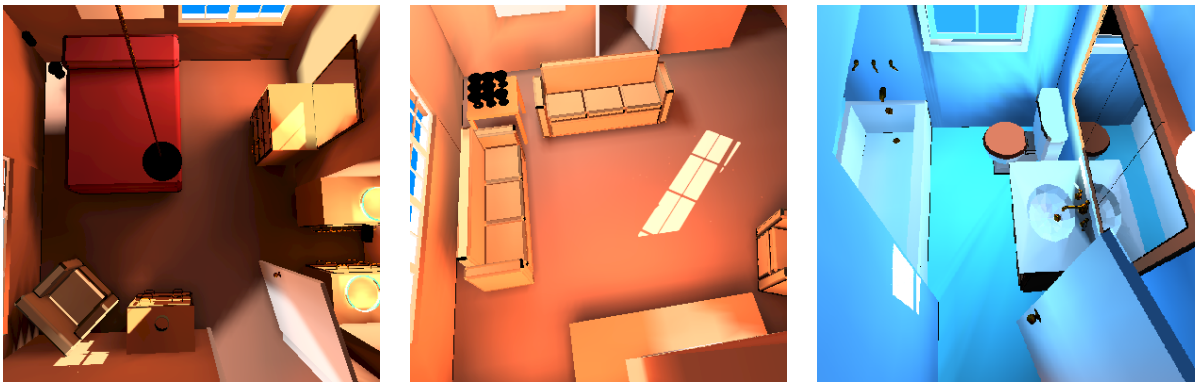


Figure 6: *Indoors scene. (a) Bedroom, with caustics over the two chest of drawers due to spotlights (lower right), and the caustics due to the reflection of sunlight from window (top) through the mirror (right) onto the wall and dresser (bottom). (b) Living room, reflection of sunlight off the mirror (unseen at the lower left). (c) Bathroom, caustics due to the mirror above the sink on the left wall. All images hardware rendered with the Render Cache[28] for specular eye path pixels.*

the hierarchy, and the add/remove operations are performed in place, avoiding multiple hierarchy traversals.

6. Implementation issues

6.1. Caustic Texture Rendering

To render the caustic textures with the illumination represented in the mesh, we use two passes. The textures are mapped on the input level object, for example the unsubdivided input polygon. For a given input polygon, we first disable the depth *writing* and render a textured polygon for each caustic texture. We then render the subdivided sub-polygons with the correct illumination values in a standard fashion, by blending the values. The blend function used is *GL_ADD*, giving the desired result. This allows us to render an unlimited number of caustics for a polygon.

For restricted textures, we use the *GL_CLAMP* texture parameter and set the border color to zero to avoid the repetition of the last pixel color.

6.2. Caustic texture parameterization

To create caustics textures, we transform the 3D particle impact to a 2D texture coordinate. We use an implicit 2D $[0..1]x[0..1]$ parameterization for each object type in our system (polygon, sphere, cone, cylinder).

For restricted textures we use a matrix transformation, mapping the new positions of the restricted texture limits to the interval $[0..1]$, thus allowing direct mapping from particle 3D coordinates to the restricted 2D texture coordinates.

7. Results

We have tested our approach on two scenes, one of an interior including artificial lights and receiving sunlight through the windows, and one of an outdoors scene in which indirect non-diffuse reflections provide dominant illumination.

The interior scene is a cabin with three rooms, the bedroom, the bathroom and the living room (Fig. 6). When tessellated by a traditional radiosity program, the scene contains approximately 180,000 polygons. We use the approach described in[10], which recognizes high-level primitives, such as spheres and cylinders. There are about 4,000 such initial objects, while after subdivision there are about 60,000 subdivided elements. The global illumination solution includes several caustics due to lights in the rooms and early morning sun light from the window. Images of the three rooms of the house are shown in Fig. 6, where we can see that several specular reflections and caustics exist in different rooms.

Scene	ITTC	TNP	OM
Cabin	35mn44s	1,600,000	370 KB
Outdoors	1h19mn	800,000	2.3 MB

Scene	UT	PT	PPT
Cabin	14.5s	0-97,000	0-6%
Outdoors	96s	5,700-41,000	0.8-17%

Table 1: *ITTC: initial time to convergence. TNP: total number of particles, OM: memory used by octree UT: average update time to move an object (container/car). PT: particles traced during an update, PPT: percentage of total particles.*

These specular reflections are created through the unified hierarchical radiosity/particle tracing approach.

To achieve convergence (i.e., when the level of energy no longer changes), our system required 6 iterations, for a total of 35 minutes and 44 seconds. The total number of particles traced to create this solution was 1.6 million. We use 25 caustics textures for specular effects with an average size of 85x85 pixels.

We translate the cylindrical container on top of the dresser in the bedroom, so that it will move into the reflection of the sun due to the mirror. Four positions of the container are shown in Fig. 7. The update times and number of particles retraced are shown in the figure.

As we can see, the number of particles retraced is around 3-4% of the total number in the scene, resulting in more than an order of magnitude speedup compared to the complete solution. As a consequence we can move the object at around 20 seconds per frame, compared to the 35 min. needed to completely recompute the solution.

Detailed results are shown in table 1. The additional memory used by the octree is 370Kb. Note that on average only 3% of the particles are re-emitted, and that we use a modest amount of memory for the octree storage.

The second example is an outdoors scene, shown in Fig. 8. The sun is on the left, shines on the metal building on the right, which creates the bright shapes on the left-hand building and on the street. A car drives between the two building, casting the shadow seen on the left. The scene contains 47,000 initial polygons, and 118,000 subdivided polygons after the global illumination solution.

Convergence of the initial solution took 1h19mn, with 800,000 particles at each iteration. In Fig. 9 we see four positions of the car and the moving shadow as a consequence. Update times and number of particles are shown in this figure. Note that in this case, the diffuse update is dominant in the update time, due to the large number of links created by our refiner.

It is important to note that all the images presented here are rendered using hardware acceleration for view-independent lightning part, combined with the RenderCache for the view-dependant part, and thus one can interactively walkthrough the results. When moving objects, it is necessary to carefully handle the cost of re-creating display lists and reconstructing illumination. Interactivity can be maintained for the most part if the changes are local enough.

8. Conclusions

We have presented a new approach for rapid updates of global illumination in complex scenes, including caustic and specular effects.

We first showed how to separate the reconstruction of high-frequency specular phenomena such as caustics from the reconstruction of the slowly varying diffuse illumination, using caustic textures. Automatic methods were presented to determine when to prefer the textures over the mesh-based reconstruction, to choose the resolution of the textures and to perform appropriate reconstruction.

We next presented a novel approach to restrict the recasting of particles during incremental updates in the scene. We use an efficient octree structure with references to the links which generate the particles to identify those which move efficiently. It is thus possible to update the global illumination solution very rapidly (for example in 20 seconds for a solution requiring 35 minutes to solve the entire system).

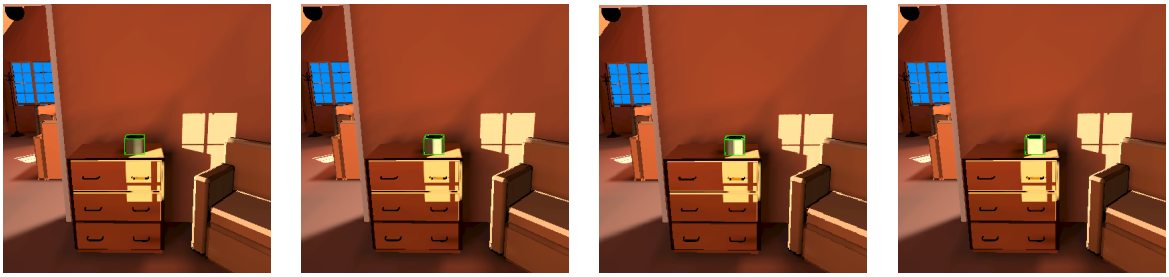
Finally, we developed an quasi-interactive method which degrades quality, but achieves much faster updates, permitting rapid positioning of objects including specular and caustic effects (3-4 seconds per frame, again for a solution which initially required 35 minutes).

One drawback of our approach is that we have introduced a number of user-controlled thresholds, which should be automatically chosen by the system. Although the default values of these thresholds work well for the scenes tested, a more consistent approach which adapts to different input scenes is clearly desirable. The abundance of texture memory and multitexturing hardware on PC platforms clearly argues in favour of also representing shadows by textures in the spirit of Myszkowski and Kunii[16], either with the convolution approach of Soler and Sillion[25] or some other shadow generation method. Again, a consistent methodology for doing this should be developed.

Finally, we are investigating a more sophisticated display approach to consistently handle the use of polygons and textures, and the book-keeping required to keep track of the parts of the hierarchy which are modified or not.

References

1. R. Bastos, M. Goslin, and H. Zhang. Efficient radiosity rendering using textures and bicubic reconstruction. In



19s/47,000 part. 19s/47,000 part. 20s/47,000 part. 22s/57,000 part.

Figure 7: *Four positions of the container on the dresser in the bedroom, the update times are shown beneath each image. Note the increase in brightness on the container and the moving shadow behind*

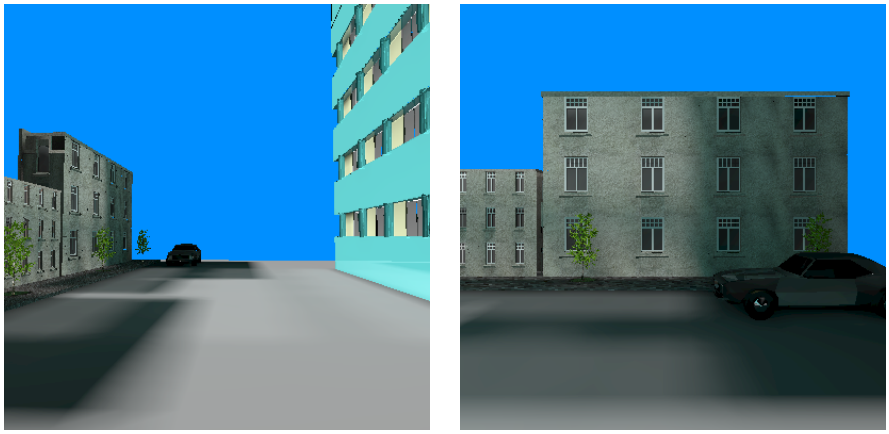


Figure 8: *Outdoors scene (47,000 polygons). The sunlight comes from the left and the building on the right has specular walls.*

Michael Cohen and David Zeltzer, editors, *1997 Symposium on Interactive 3D Graphics*, pages 71–74. ACM SIGGRAPH, April 1997.

2. S. E. Chen. Incremental radiosity: An extension of progressive radiosity to an interactive image synthesis system. In *Computer Graphics (SIGGRAPH '90 Proceedings)*, pages 135–144, August 1990.

3. S. E. Chen, Holly E. Rushmeier, G. Miller, and D. Turner. A progressive multi-pass method for global illumination. In *Computer Graphics (SIGGRAPH '91)*, volume 25, pages 165–174, July 1991.

4. M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A progressive refinement approach to fast radiosity image generation. In *Computer Graphics (SIGGRAPH 88)*, volume 22, pages 75–84, August 1988.

5. P. J. Diefenbach and N. I. Badler. Multi-pass pipeline rendering: Realism for dynamic environments. *1997 Symposium on Interactive 3D Graphics*, pages 59–70, April 1997.

6. G. Drettakis and F. Sillion. Interactive update of global illumination using a line-space hierarchy. In *Proc. SIGGRAPH'97*, Annual Conference Series, pages 57–64, August 1997.

7. D. W. George, F. X. Sillion, and D. P. Greenberg. Radiosity redistribution for dynamic environments. *IEEE Computer Graphics and Applications*, 10(4), July 1990.

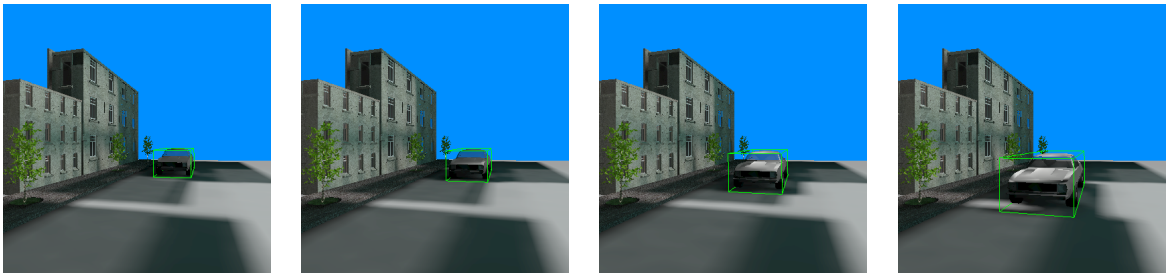
8. C. M. Goral, K. K. Torrance, D. P. Greenberg, and B. Battaile. Modelling the interaction of light between diffuse surfaces. In *Computer Graphics (SIGGRAPH'84)*, volume 18, pages 213–222, 1984.

9. X. Granier and G. Drettakis. Controlling memory consumption of hierarchical radiosity with clustering. In *Graphics Interface*, pages 58–65, June 1999.

10. X. Granier, G. Drettakis, and B. Walter. Fast global illumination including specular effects. In *11th EG Workshop on Rendering, "Rendering Techniques 2000"*, pages 47 – 59. Springer Wien, 2000.

11. E. A. Haines and J. R. Wallace. Shaft culling for efficient ray-traced radiosity. *2nd EG Workshop on Rendering (Photorealistic Rendering in Computer Graphics)*, 1994.

12. P. Hanrahan, D. Salzman, and L. Aupperle. A hierar-



71s/11,000 part. 93s/23,000 part. 136s/139,000 part. 114s/24,000 part.

Figure 9: *Four positions of the car in the outdoors scenes. Update times and particles used are shown beneath each image. Note the increase in brightness of the car and the moving shadow of the car onto the floor on the left.*

chical radiosity algorithm. In *Computer Graphics (SIGGRAPH '91)*, volume 25, pages 197–206, July 1991.

13. P. S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *Computer Graphics (SIGGRAPH'90)*, volume 24, pages 145–154, August 1990.

14. D. S. Immel, M. F. Cohen, and D. P. Greenberg. A radiosity method for non-diffuse environments. *Computer Graphics (SIGGRAPH'86)*, 20(4):133–142, August 1986.

15. H. Wann Jensen. Global illumination using photon maps. In *7th EG Rendering Workshop, "Rendering Techniques '96"*, pages 21–30. Springer Wien, 1996.

16. K. Myszkowski and T. L. Kunii. Texture mapping as an alternative for meshing during walkthrough animation. In *Photorealistic Rendering Techniques*, Eurographics, pages 389–400. Springer-Verlag, 1994.

17. S. Parker, W. Martin, P-P J. Sloan, P. Shirley, B. Smits, and C. Hansen. Interactive ray tracing. *1999 ACM Symposium on Interactive 3D Graphics*, pages 119–126, April 1999.

18. E. Reinhard, B. Smits, and C. Hansen. Dynamic acceleration structures for interactive ray tracing. In *11th EG Workshop on Rendering, "Rendering Techniques 2000"*, pages 299–306. Springer Wien, 2000.

19. F. Schoeffel and A. Pomi. Reducing memory requirements for interactive radiosity using movement prediction. In *10th EG Workshop on Rendering, "Rendering Techniques '99"*. Springer Wien, 1999.

20. E. Shaw. Hierarchical radiosity for dynamic environments. *Computer Graphics Forum*, 16(2):107–118, 1997.

21. F. X. Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Trans. on Visualization and Computer Graphics*, 1(3):240–254, September 1995.

22. F. X. Sillion, J. R. Arvo, S. H. Westin, and D. P. Greenberg. A global illumination solution for general reflectance distributions. In *Computer Graphics (SIGGRAPH'91)*, volume 25, pages 187–196, July 1991.

23. F. X. Sillion and C. Puech. A general two-pass method integrating specular and diffuse reflection. In *Computer Graphics (SIGGRAPH'89)*, volume 23, pages 335–344, July 1989.

24. B. Smits, J. Arvo, and D. P. Greenberg. A clustering algorithm for radiosity in complex environments. In *Proc. SIGGRAPH'94*, Annual Conference Series, pages 435–442, July 1994.

25. C. Soler and F. X. Sillion. Texture-based visibility for efficient lighting simulation. *ACM Transactions on Graphics*, October 2000. To appear.

26. W. Stürzlinger and R. Bastos. Interactive rendering of globally illuminated glossy scenes. In *8th EG Workshop on Rendering, "Rendering Techniques '97"*, pages 93–102. Springer Wien, 1997.

27. J. R. Wallace, M. F. Cohen, and D. P. Greenberg. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. In *Computer Graphics (SIGGRAPH'87)*, volume 21, pages 311–320, July 1987.

28. B. Walter, G. Drettakis, and S. Parker. Interactive rendering using the render cache. In *10th EG Workshop on Rendering, "Rendering Techniques'99"*. Springer Wien, June 1999.

29. B. Walter, P. M. Hubbard, P. Shirley, and D. P. Greenberg. Global illumination using local linear density estimation. *ACM Trans. on Graphics*, 16(3):217–259, July 1997.