Global Illumination for Interactive Applications and High-Quality Animations

Cyrille Damez, Kirill Dmitriev, Karol Myszkowski

Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken Germany

Abstract

One of the main obstacles to the use of global illumination in image synthesis industry is the considerable amount of time needed to compute the lighting for a single image. Until now, this computational cost has prevented its widespread use in interactive design applications as well as in computer animations. Several algorithms have been proposed to address these issues. In this report, we present a much needed survey and classification of the most up-to-date of these methods. Roughly, two families of algorithms can be distinguished. The first one aims at providing interactive feedback for lighting design applications. The second one gives higher priority to the quality of results, and therefore relies on offline computations. Recently, impressive advances have been made in both categories. Indeed, with the steady progress of computing resources and graphics hardware, and the current trend of new algorithms for animated scenes, common use of global illumination seems closer than ever.

1. Introduction

Synthesis of images predicting the appearance of the real world has many important engineering applications including product design, architecture, and interior design. Even less rigorous applications, such as special effects, film production, or computer games would benefit from an increase in realism. One of the major components of such predictive image synthesis is the simulation of the interaction of light with objects in the scene. The ever-increasing processing power and memory size, as well as recent algorithmic developments, have made global illumination computations affordable even for complex scenes.

A vast majority of the existing global illumination algorithms were designed for rendering static scenes. In practice this means that when such algorithms are used for a dynamic scene, all computations have to be repeated from scratch even when only minor changes are performed. The use of those algorithms in practical, real-life applications is therefore seriously limited. Indeed, the excessive cost of redundant computations cannot be tolerated in core computer graphics applications such as the production of animated sequences or interactive rendering.

In recent years, several new algorithmic solutions have been proposed that address explicitly the problem of global illumination computations in changing environments. Significant effort has been devoted to the reduction of required resources. Indeed, considering the large number of frames necessary even for a brief movie sequence, shortening the production time carries an important economic value for the movie industry. Moreover, the ability to provide a fast feedback to the user in response for interactive changes in the scene greatly improves the usability of lighting simulation algorithms.

Another important issue is temporal aliasing in animations. Many typical errors in lighting computation and reconstruction cannot be perceived by the human observer when they are coherent in the temporal domain. However, they may cause unpleasant flickering and shimmering effects when this coherence is lost. Such artifacts are more difficult to combat efficiently if temporal processing of global illumination is not performed. In such a case, the traditional approach of reinforcing the accuracy of global illumination computation for each frame proves extremely inefficient. Moreover, it is very difficult to locally control the perceivability of temporal aliasing. Clearly, temporal coherence in the scene changes must be taken into account in modern global illumination algorithms to make them practical.

If one considers the recent resurgence of this topic in the computer graphics literature, it seems that both the hardware and the algorithms at hand are now mature enough to consider lighting simulations for animated environments as feasible. This state-of-the-art report is therefore intended:

- to define a classification of global illumination techniques in animated scenes, developed in Section 2.
- to recall and discuss the most up-to-date development in this field. We discuss interactive and off-line global illumination algorithms in Sections 3 and 4, respectively.
- to evaluate the respective performances, pros and cons of these new approaches: in Section 5 we provide a synthetic comparison taking into account some relevant criteria, describing their strengths and weaknesses in practical applications. Finally, we conclude this report and speculate on possible directions of further research.

2. Content of this Report

Global illumination algorithms for animations must be designed to deal efficiently with many different types of changes in the scene. The most typical scene changes may involve:

- the camera - its position and orientation in the scene, the viewing direction, the viewing angle,
- surface attributes - light reflectance and transmission characteristics, textures,
- light sources - their positions in the scene, spatial dimensionalities and shapes, goniometric diagrams,
- the geometry - moving rigid objects, articulated figures, shape animation.

While not exhaustive, this list gives an idea of the challenges imposed on a versatile global illumination algorithm. In practice, many algorithms are specialized to handle only a subset of these scene changes. A common example of such specialized algorithms are walkthrough techniques dedicated for efficient handling of camera animation, a difficult problem in itself for non-diffuse environments. A good survey of relevant interactive rendering techniques for such environments was presented by Heidrich[33] (refer to Sections 2–4 of his report).

In this state-of-the-art report we focus on general purpose algorithms. As a consequence, specialized walkthrough techniques are not discussed in this report. The common denominator for most of the algorithms we detail in this report is their ability to handle modifications in the scene geometry. Visibility changes between objects in the scene make it one of the most difficult aspects of the global illumination computation.

Global illumination algorithms for animated scenes can be grouped in two categories, with different scopes of application:

- The *interactive* methods, which are designed to trade the image quality for the response speed. Their main objective is to provide a fast response for frame-to-frame changes in the environment, but not to a sequence of such changes. A fixed frame rate is required by some applications, which splits the illumination update into equal time intervals and results in progressive refinement of resulting images.
- The *off-line* methods, for which complete knowledge of trajectories and upcoming events is required. Those methods are usually aimed at producing high quality animations. In this case, differences in the rendering time of particular frames are perfectly acceptable.

Since the working hypothesis for the former methods is less restrictive, any interactive method can in theory be applied in place of an off-line method, i.e. in a non-interactive application, whereas the contrary generally is not possible. However, the quality of the animations they provide, as we show in the report, usually cannot match their off-line counterparts. The temporal coherence of lighting can also be exploited much better when longer image sequences are considered. This requires the knowledge of changes in the environment in advance, which is impossible in the interactive scenario.

For the sake of completeness, we chose to mention in this state-of-the-art report even pioneering animated global illumination algorithms, though we mainly focus on the most recent ones. For the same reason we briefly mention some techniques dedicated for efficient display of the lighting simulation results during the rendering phase. This topic was extensively covered for interactive solutions by Heidrich[33] (refer to Section 5 of his report) and an interested reader can find there a more detailed discussion of relevant techniques. In this report we discuss more extensively only those rendering solutions, which explicitly emphasize on the temporal coherence between the subsequent animation frames.

3. Interactive Methods

For many applications, such as lighting design, it is desirable to be able to move one or more objects, or change their local reflection properties, and witness *as fast as possible* the effects of such modifications on the scene appearance. Without an appropriate, interactive enough feedback, fine tuning of the lighting is a tedious process. Unfortunately, such modifications of the scene, because of the very nature of global illumination, may change the appearance of every surface in the scene. The methods reviewed in this section usually address this problem by trying to determine a minimum set of recomputation to be made to provide the user a "precise enough" feedback, while still maintaining a "sufficient" frame rate.

Two separate problems can be distinguished:

- First, the actual simulation of global illumination has to be performed and updated with respect to the user changes. Relevant methods will be reviewed in Sections 3.1– 3.5. We start with the incremental algorithms that evolved from the finite element radiosity method in Section 3.1. We then proceed to a review of stochastic methods in Sections 3.2–3.4, and of an hybrid approach in Section 3.5.

- Then the results of the simulation also have to be displayed. Straightforward hardware polygon rasterisation can be used if only diffuse reflections are taken into account, but as soon as specular reflections are to be simulated, more sophisticated display methods are called for. Therefore, we describe relevant real-time display approaches that can be applied to global illumination solutions in Sections 3.6– 3.7, though they do not address the question of *how* global illumination is to be computed.

3.1. Interactive Update of Radiosity Solutions

The Radiosity Method[25] and its derivatives[11, 31, 67, 63] allow the efficient computation of global illumination in perfectly diffuse environments. As a consequence, they are particularly suited for applications such as lighting design or walkthroughs of architectural models. Moving the camera away from the original point of view does not require any work other than reprojecting the polygons that compose the finite element mesh on the image plane and drawing them.

However, for the radiosity methods, changes in the surfaces reflectances require new iterations to compute the repropagation of light in the scene. Moreover, geometric modifications in the scene imply an expensive reevaluation of many form factors, and as a consequence light needs to be repropagated until convergence. Various independent algorithms based on the radiosity method have been proposed to address these issues :

Incremental Progressive Radiosity Chen[9] and George *et al.*[23] independently proposed two similar algorithms, based on Progressive Radiosity[11]. Recall that Progressive Radiosity differs from the original "full-matrix" radiosity algorithm in that it iteratively selects an element in the mesh (starting from the light sources) and *shoot* its unpropagated energy, therefore allowing the progressive display of images with increasing accuracy.

Chen's and George's algorithms allow the update of a converged radiosity solution after the displacement of an object in the following way. First, the radiosity of the dynamic object is repropagated toward all static surfaces. Then, one static mesh element is chosen. Its radiosity is shot at the dynamic object, and the inverse of its radiosity is shot at the surfaces that are inside the shadow volume cast by the dynamic object. This process is repeated iteratively until a satisfying approximation is computed. Müller *et al.*[48] later extended Progressive Radiosity further, building shadow-lists to speed up modified interactions computation.

Unfortunately, as Progressive Radiosity requires a computation time which is quadratic with respect to the number of mesh elements, its use (and as a consequence, the use of Chen's and George *et al.*'s algorithm) is limited to extremely simple scenes. As a consequence, it cannot reconstruct illumination solutions with high frequency features, such as sharp shadows, without using multiple levels of precision for its computation.

Dynamic Hierarchical Radiosity Hierarchical Radiosity was first introduced by Hanrahan *et al.*[31] to reduce the cost of radiosity computations. This method is based on the use of a hierarchical mesh, representing the surfaces radiosity at various precision levels. Light exchanges between elements are represented by a link structure that specifies at which level of precision the form factor computation and light gathering should be performed. This method can be naturally extended to handle dynamic scenes, as the link structure makes it easier to localize the form factors that have potentially been affected by the changes in the geometry.

Forsyth *et al.*[22] first proposed an extension of hierarchical radiosity to update a converged hierarchical radiosity solution with respect to such a change. For every new solution that has to be computed, a given link can be occluded or validated (if the dynamic object passes between the corresponding sender and receiver), promoted or pushed down the hierarchy (when the displacement of one of the two links extremities makes the corresponding exchange representation excessively precise or too approximate, respectively).

To limit the computational cost, visibility changes for static elements are ignored during the reevaluation of the links. The form factors involving moving surfaces are only evaluated for a couple of key frames and interpolated in between. Furthermore, the link reevaluation procedure presented by Forsyth *et al.* can progressively refine the mesh, but does not provide any simplification mechanism. As a consequence, the size of the mesh is continuously increasing as the interactive session goes on, which makes it impractical for long interactive sessions. As a consequence, Shaw[61] proposed an alternative link structure allowing to "unrefine" the mesh, through the use of special *shadow links* keeping track of blocker information, and a method to localize occlusion caused by the dynamic object using motion volumes.

However, even though these approaches provide significant improvements over the Progressive Radiosity method presented earlier, they still need a couple of seconds to update even very simple scenes. Moreover, as pointed by Drettakis and Sillion[20], they lack a mechanism to offer a consistent trade-off between accuracy and speed to be able to reach interactive frame rates with fairly complex scenes.

Line-Space Hierarchy In order to achieve interactive frame rates, Drettakis and Sillion[20] proposed to augment the Hierarchical Radiosity link structure with shafts[29] that represent the set of line segments joining two linked hierarchical elements. Traversing the link structure to test for intersections of shafts by the bounding box of the dynamic object allows to rapidly identify interactions that have to be recomputed, further refined or simplified. Note that the gathering of light, as well as link structure simplification are performed *in-place* during the traversal, whereas the mechanism proposed by Shaw required multiple traversals of the hierarchy to do so. An history of the refinement is kept, through the use of *passive links* (*i.e.* interactions that are represented with more precision further down in the hierarchy, but are kept nonetheless in case simplifications are later required). This is similar to the idea of *shadow links* discussed previously.

The Line Space Hierarchy method also gives the user control over the time vs. quality trade-off. During the traversal, links that need to be refined are stored in a heap, with respect to the total potential energy transfer they represent. When the sum of the number of links already refined and those in the heap exceeds a certain limit, no further links are scheduled for refinement. The budget limit of links allowed for refinement has to be fixed with respect to the desired frame rate.

Similarly, the update of modified links is controlled in order to guarantee a certain frame rate. During this traversal, passive links corresponding to a modified interaction are recursively considered. The fraction of the remaining time alloted to the update of a certain part of the hierarchy is given by the sum of the number of *active* links arriving at the corresponding hierarchical element E and the number of active sub-links below E. At a given level, if the total number of children links exceed the current remaining budget, it is not possible to update them in the alloted time. As a consequence, the passive link is reinstated and the underlying links (and if necessary, mesh elements) are removed. To avoid repeated traversal, the counting of links is done during the *push-pull* phase and recorded. With a total update budget of 330 links, the authors reported a framerate of approximately 0.3 sec. per frame.

As the hierarchy of objects in the Line-Space Hierarchy algorithm uses clustering[67, 63], a consistent solution is always computed. No matter how fast the frame rate required is, the set of active links in the hierarchy always account for the complete line space. Of course, when fast updates are required, all interactions will be placed at a rather high level. The computed solution will therefore be a rather crude approximation.

However, when the scene becomes more complex, or the requested precision is high, the weight of the link hierarchy becomes overwhelming. The memory required for the storage of shafts is particularly significant. In order to reduce the memory requirements of the Line-Space Hierarchy algorithm, Schoeffel *et al.*[60] proposed to clean up the link hierarchy by removing shafts. Shafts that are likely to be needed in the next couple of frames are predicted by a simple movement extrapolation scheme and built by a separate process. This reduces the adverse effect on performance that would be caused by "on-the-fly" reconstruction of shafts. Typical memory savings range from 80% to 90%. When the dynamic object is moved along a continuous path with a "reasonable" speed (which admittedly might not always be the case during interactive design session), a performance similar to the original line-space algorithm can be obtained.

3.2. Radiosity Computation Using Graphics Hardware

In the previous section we discussed the radiosity algorithms which rely completely on the software-based computation for which graphics hardware is used only at the image display stage. However, the rapidly improving speed and functionality of graphics hardware makes it more and more attractive to relegate some expensive computation from the CPU to graphics hardware[33].

In this section we discuss two example algorithms in which the radiosity computation is supported by graphics hardware. First, we briefly overview an algorithm proposed by Udeshi and Hansen[71] which is a hybrid of hardware radiosity and ray tracing. Then, we discuss the Instant Radiosity algorithm developed by Keller[42] in which the use of graphics hardware is emphasized even more during the lighting computation.

Hybrid Hardware Radiosity and Ray Tracing. Udeshi and Hansen[71] use graphics hardware to obtain the direct illumination on diffuse surfaces. Shadows are computed using a shadow volume algorithm improved by the authors . Also, approximate one-bounce diffuse interreflection computation is performed, in which bright directly illuminated surface elements become secondary (virtual) light sources. A single-plane version of the hemi-cube algorithm is used to compute the form factors. However, visibility is not taken into account for virtual lights, which may lead to incorrect indirect illumination. The CPU is mostly involved in ray tracing of pixels through which specular and refractive objects can be seen. The authors report an achievement of several frames per second on a graphic supercomputer (64 processors and 8 graphics pipelines) for scenes of moderate complexity.

Instant Radiosity. Keller[42] proposes an algorithm that uses the CPU in a first pass, and classical OpenGL lighting functions in a second pass. First, photons are emitted from light sources and are traced in the scene to produce an approximation of the diffuse radiance on object surfaces in this scene. Instead of the classical random walk terminated by Russian Roulette, a quasi-random walk algorithm, based on fractional absorption, is used[41]. The algorithm applies knowledge of the average diffuse reflectivity ρ of the scene to restrict the length of photon paths in a physically plausible way. Let for instance N be the total number of walks the user wants to generate. Then exactly $\lfloor \rho N \rfloor$ walks will have zero length, $\lfloor \rho^2 N \rfloor$ walks will have length equal to 1, $\lfloor \rho^3 N \rfloor$ - length equal to 2, and so on. The maximum walk length will then be equal to $\lfloor log_\rho(1/N) \rfloor$. All photons start with the same energy. When a photon hits the surface, its energy is scaled by a factor $f_d(y)/\rho$, where $f_d(y)$ is the diffuse reflectivity at the point of intersection.

As a result of the photon tracing stage, an approximation of the diffuse radiance is obtained. Graphics hardware is then cleverly used to render the image with shadows. Each photon hit point on the scene surfaces is treated as a virtual point light source. Since standard graphics hardware can render only one light source with shadows at a time, the accumulation buffer is used to sum up the individual images. The luminance of every pixel is usually affected by many photons, thus a very crude radiance approximation is enough. Keller shows that a number of quasi-random walks varying from 32 to 128 already yields visually smooth images.

Two problems of the algorithm are discussed in [42]. Since only a crude approximation of radiance is generated, a good spatial distribution of photons is required to achieve plausible results. Partially this problem is solved by applying uniform quasi-random sequences for photon shooting. However, in the regions where the contribution of a single photon is too strong (e.g. close to its hit point), objectionable bright spot can appear. This problem is hidden by OpenGL, which clamps intensity values to the 0–1 range, limiting the effect of such artifacts. The second problem is that each photon, colored by a texture, has a large influence on the overall color of the scene.

Dynamic environments are handled by explicitly storing an individual image, corresponding to each quasi-random walk. Every time a walk is renewed, its image is replaced by a new one. The current N images are then accumulated and displayed, thus implicitly performing temporal antialiasing. This, however, restricts the number of quasi-random walks since a large amount of memory is required to store individual images, and the accumulation of images for all random walks is costly for large N.

The indisputable advantage of the approach is that a solution is obtained with pixel precision. Images thus avoid visually objectionable artifacts, such as light leaks, inherent to finite element approaches. However, since the technique is view-dependent, the objects entering the field of view may be insufficiently sampled before enough light paths involving those objects have been accumulated. Handling objects outside the field of view but visible in mirrors is also difficult. Some of these issues are successfully addressed by using a Distributed Ray Tracing technique, which we describe in Section 3.4.

3.3. Selective Photon Tracing

In the Instant Radiosity algorithm[42] discussed in the previous section, photon paths are successively replaced by new ones to update lighting in dynamic environments. Ideally, at first only the photon paths, which clearly interfere with dynamic objects in the scene should be replaced. Also, since Instant Radiosity is a view-dependent algorithm, image recalculation is required for any change of the camera parameters. In this section we discuss an algorithm called Selective Photon Tracing (SPT) developed by Dmitriev *et al.*[19] which addresses those important issues.

The SPT algorithm uses graphics hardware to compute direct lighting with shadows using the shadow volume algorithm in a way similar to Udeshi and Hansen[71] (refer to Section 3.2). Using the functionality of modern graphics hardware Dmitriev *et al.* can process up to 4 light sources with goniometric diagrams per one rendering pass. The indirect lighting is computed asynchronously using a quasi-random photon tracing (similar to Keller[41]) and density estimation techniques (similar to Volevich[74]). The indirect lighting is reconstructed at vertices of a precomputed mesh and can be readily displayed using graphics hardware for any camera position. The most unique feature of the SPT algorithm is exploiting the temporal coherence of illumination by tracing photons selectively into the scene regions that require illumination update.

In the SPT algorithm, pseudo-random sequences, typically used in photon shooting, are replaced by the quasi-random Halton sequence[30]. This proves to be advantageous. Firstly, as shown by Keller[41], quasi-random walk algorithms converge faster than classical random walk ones. Secondly, a periodicity property of the Halton sequence[51] provides a straightforward way of updating indirect illumination as the scene changes. Let us briefly discuss this periodicity property, which is fundamental for efficient searching of invalid photon paths in dynamic scenes. If h_{ij} is j-th coordinate of the Halton point with index i, it can be shown that[19]:

$$|h_{ij} - h_{(i+mN_g)j}| < \frac{1}{b_j^k}, \text{ if } N_g = l b_j^k \qquad (1)$$

where b_j is a base, used to compute the j-th coordinate of Halton points, k, l, and m are integers such that $k \geq 0$ and $l > 0$. For instance, setting $N_g = b_0^{k_0} b_1^{k_1} b_2^{k_2} b_3^{k_3}$ yields points in which the first four coordinates closely match. The closeness of this match is governed by the corresponding powers k_0, k_1, k_2, k_3 (the larger power values are selected the closer match is obtained). If the scene surfaces and BSDFs are reasonably smooth, quasi-random points with similar coordinates produce photons with similar paths, which is depicted in Figure 1.

By selecting quasi-random points with such an interval N_g, that the similarity of coordinates $j = 0$ and $j = 1$ is assured, photons are emitted coherently inside a pyramid with its apex at light source position. However, those photons do not reflect coherently (Figure 1a). By additionally increasing N_g to obtain a similarity of coordinates $j = 2$ and $j = 3$, photons are selected so that they reflect coherently inside a more complex bounding shape, as shown in Figure 1b.

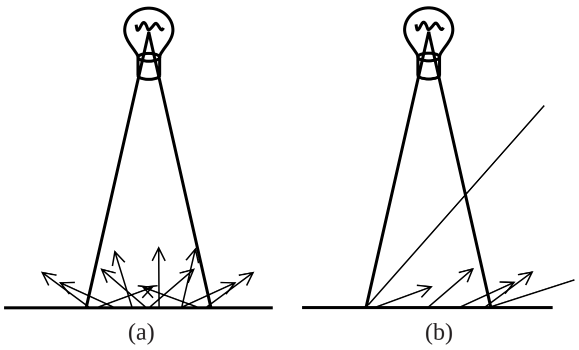

Figure 1: *Structure of photon paths, corresponding to quasi-random points with similar coordinates (a) similarity of first two coordinates. (b) similarity of first four coordinates.*

The algorithm considers a pool of photons of fixed size Z for the whole interactive session. The photons are traced during the initialization stage. Then the paths which become invalid due to changes in the scene are selectively updated. This is performed by tracing photons for the previous scene configuration with negative energy and tracing photons for the new scene configuration with positive energy. If the scene modifications are local enough and only a small number of photon paths are affected, such a double effort is justified compared to computing the solution anew. Since it would be too expensive to update ray tracing acceleration data structures between the previous and current scene configurations for every photon, photons are reshot in groups corresponding to the same (frozen) scene configuration. The number of photon groups in pool Z is set to be N_g and their structure is depicted in Figure 2.

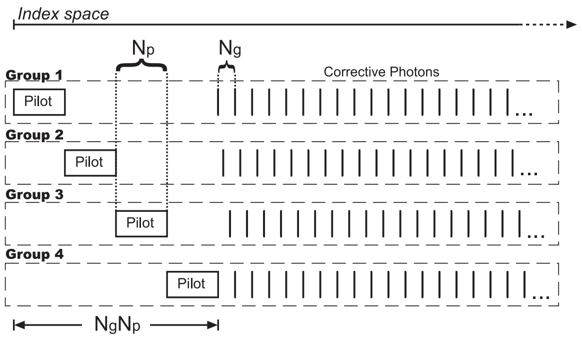

Figure 2: *Distribution of photons between groups (for the purpose of this figure $N_g = 4$ is assumed). Each row represents one group. Each group contains N_p pilot photons (shown as rectangles) as well as a number of corrective photons (shown as vertical lines) whose indices are spaced with interval N_g.*

Each group consists of N_p pilot photons and $(Z - N_pN_g)/N_g$ corrective photons. The pilots photons are emitted in the scene with respect to a distribution corresponding to the light sources goniometric diagrams. Based on the pilot photons which hit dynamic objects and therefore require updating, the remaining photons with similar paths in the scene can be found immediately. This is possible due to the periodicity property inherent to the multi-dimensional Halton sequence, which is used to generate those photons (refer to Equation 1). The corrective photons of the same group are coherent in space and their paths have the structure depicted in Figure 1. The goal of the corrective photons is to update illumination selectively in the scene regions affected by dynamic objects.

The photon update computations are performed iteratively. Each iteration consists of reshooting one photon group. The order in which the photon groups are updated is decided using an inexpensive energy- and perception-based criterion whose goal is to minimize the perceivability of outdated illumination. While reshooting pilot photons of the group, the priority of other groups is estimated. It is shown[19] that if the pilot photon with index i hits a dynamic scene region, then it is likely that corrective photons belonging to the group with index $i_g = N_g \ mod \ i$ also traverse this region. This means that group i_g may require updating photons and the priority of i_g for such an update is increased by increment Δp. The importance increment Δp can be reduced for a given photon hitting a textured surface due to visual masking[13, 21]. The reduction of visual sensitivity for lighting patterns is estimated for each texture in a preprocessing stage, which means that taking into account the visual masking during the interactive rendering stage does not introduce any significant overhead. This way, the photon group update scheduling also accounts for perceptual importance of detected illumination changes.

The frame rate in an interactive session using the SPT algorithm is mostly determined by the OpenGL performance. For rendering with shadows from multiple light sources with goniometric diagrams, frame rates ranging from 1.1 to 8 fps. are reported (refer to Figure 3). Indirect illumination is updated incrementally and the result of each update is immediately delivered to the user. Most lighting artifacts created by outdated illumination are removed in the first 2–3 seconds after the scene change. If photons are updated in a random order at least 2–4 times longer computations are needed to obtain images of similar quality. Better performances can be expected for more complex scenes, or when user modifications have more localized consequences. The unique feature of SPT is that while the temporal coherence in lighting computations is strongly exploited, no additional data structures storing photon paths are required.

In the case study examples (refer to Figure 3) provided by Dmitriev *et al.*[19] the coherence of photons was considered only up to their second bounce. This proved to be sufficient to efficiently update the illumination for the studied scenes.

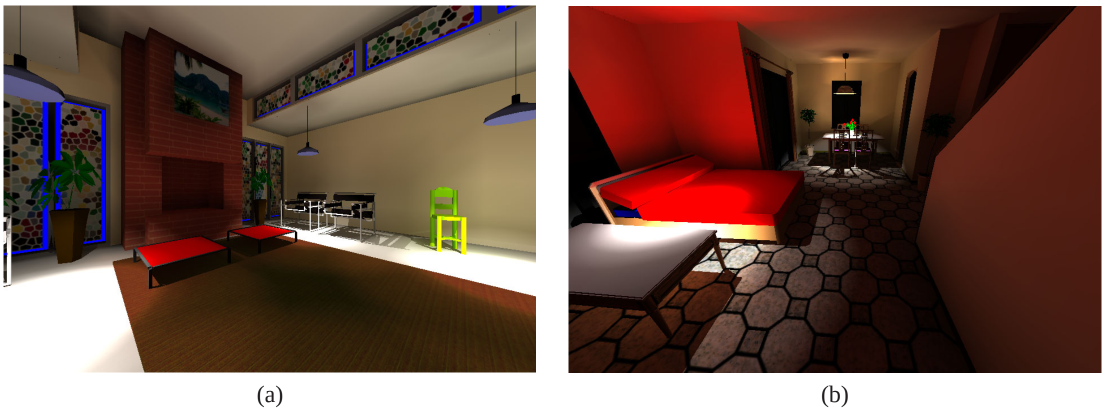

Figure 3: *Example frames with global illumination effects obtained using Selective Photon Tracing for scenes composed of (a) 12,400 and (b) 377,900 triangles. On a 1.7 GHz Dual P4 Xeon computer with an NVIDIA GeForce3 64 MB video card the frame rates 8 fps. and 1.1 fps. have been achieved, respectively. In order to remove all visible artifacts in lighting distribution resulting from object changes in the scene, 2–3 seconds of the selective photon update computation are usually required.*

When such a coherence for a higher number of bounces is required, the number of groups N_g must be increased. This results in a proportional increase of the photon pool size Z. Interactive update of such an increased pool of photons might not be feasible on a single PC-class computer. The major problem with the algorithm proposed by Dmitriev *et al.* is lack of adaptability of the mesh used to reconstruct the indirect illumination in the scene. Also, only point light sources are explicitly handled in their hardware-supported algorithm for the direct illumination computation.

Selective photon tracing can be used in the context of off-line global illumination computation as well. In particular, this technique seems to be attractive in all those applications which require local reinforcement of computations based on some importance criteria. An example of such an application is the efficient rendering of high quality caustics, which usually requires a huge number of photons. After identifying some paths of caustic photons, more coherent particles can easily be generated using this approach. The drawback of many existing photon based methods is that too many particles are sent into well illuminated scene regions with a simple illumination distribution, and too few photons reach remote scene regions. This deficiency could easily be corrected using the SPT approach, by skipping the tracing of redundant photons and properly scaling the energy of the remaining photons.

3.4. Interactive Global Illumination Using a Distributed Ray-Tracing Engine

Remarkable advances in ray tracing speed have recently been made. Through the exploitation of image and object space coherence, and better use of computational resources, such as processor caches and SIMD instructions, ray tracing at interactive speed was proved feasible[76, 77]. Though for small scenes it still cannot outperform graphics hardware on a single PC, comparable performance can be obtained for the rendering of complex scenes which are composed of hundreds of thousands of triangles. Moreover, ray tracing is fairly easy to parallelize, which makes it possible to achieve even greater performance on a cluster of PCs. Those remarkable improvements in ray tracing performance made it possible to develop an interactive global illumination system, which extends the earlier discussed Instant Radiosity approach to handle more general lighting effects at even higher speed[75].

Similarly to the Instant Radiosity algorithm[42], in the approach proposed by Wald *et al.*[75] a coarse representation of radiance is generated by shooting particles from the light sources and treating their hit points as virtual point light sources. The computations are distributed over a cluster of PCs. Each PC generates its own set of quasi-random walks and as a result considers a different set of virtual lights. Since the network bandwidth is not sufficient to collect large resolution antialiased images from every client, each PC computes only selected samples of the final image. The image plane is split into image tiles and the same (fixed) sample pattern is chosen for all tiles using the interleaved sampling technique[43]. The samples within each tile are computed by different PCs, which means that neighboring samples are obtained for different sets of virtual lights.

Though such image tiling solves the network bandwidth problem, it results in structured noise, since sample values computed on different PCs may differ significantly. To remove this noise, discontinuity buffering is used. The idea is to apply a 3×3 neighbor filter to each pixel, by taking into account only those pixels that represent scene surfaces with similar normal vectors and depth values (measured with respect to the observer). This selection of similar pixels for filtering purposes is important to avoid excessive smearing of fine geometric details, e.g. object silhouettes. The proposed criteria of pixel similarity do not prevent the blurring of sharp shadow boundaries, which cannot be distinguished from the noise in reconstructed lighting. Wald *et al.* demonstrated that their 3×3 discontinuity buffer works well in practice and effectively removes noise caused by interleaved sampling.

The algorithm proposed by Wald *et al.* can render many important lighting effects such as diffuse interreflections, specular reflection and refraction, and even caustics (refer to Figure 4b). The latter effect is obtained using an approach similar to Photon Mapping, in which photons are shot from light sources in the direction of specular objects that are expected to cast caustics. To speed up the caustics computation, the kd-tree structure, typically used for photons lookup, is replaced by a uniform grid. Instead of n-nearest neighbor queries, all photons inside a sphere of a fixed radius are collected and used to estimate pixel color. The caustics computation is embedded into the general parallelization strategy. The whole pool of caustic photons is distributed evenly among all computers of the cluster. Image tiles, delivered by each client, include both diffuse and caustic illumination and are filtered by the server. Noise, caused by an insufficient number of caustic photons is partially removed by the discontinuity buffer procedure, described above. 500–1500 caustic photons per light source are shown to be sufficient for the generation of good looking caustics (refer to Figure 4b)

The proposed approach solves many problems, inherent to the Instant Radiosity algorithm. For example, objects reflected in mirrors are now rendered without any problem, and caustics can be computed as well. As well as the Instant Radiosity, this algorithm performs per pixel computations, which limits discretization error and enables rendering of parametric geometry. Since a full illumination recomputation is done after each object or camera movement, there is no outdated illumination. Such a recomputation, however, involves huge computational efforts and requires a cluster of PCs to support interactive frame rates. Also, the temporal coherence between frames is lost which may cause temporal aliasing artifacts which will appear as slight image flickering. The server capacity of collecting the image tiles currently limits the frame rate by 5–6 fps. at a video resolution.

3.5. Incremental Update of Glossy Global Illumination

Admittedly, Hierarchical Radiosity is one of the most efficient global illumination method for diffuse scenes. However, the extension of deterministic, finite element methods to glossy surfaces is too computationally and memory intensive, even if it was proved feasible on scenes of limited complexity[64, 10, 68]. In particular, the explicit storage in memory of a directional representation of outgoing radiances for all mesh elements prevents the use of these methods for scenes whose complexity corresponds to industry requirements, even on todays most expensive high-end hardware. On the other end, various non deterministic methods[72, 37, 73] have demonstrated their efficiency for computing lighting effects involving specular reflections and refractions, such as caustics, but are far less efficient when it comes to diffuse interactions.

To keep the best from both worlds, Granier *et al.*[27] proposed a *Unified Hierarchical Algorithm* that cleanly separates purely diffuse light paths from those involving specular reflections and/or refractions. The former are computed by a classical hierarchical radiosity approach, while the latter are handled using stochastic particle shooting. Clever use of shafts allows limitation of this stochastic shooting to portions of line space where they are really needed. For a proper integration of both methods within the hierarchical radiosity paradigm, particles' energies are added at an appropriate level in the hierarchical mesh during the push-pull traversal.

Rapid updates[26] of solutions computed by the Unified Hierarchical Algorithm are possible using a framework similar to the one of the Line Space Hierarchy (see Section 3.1). In this algorithm, all transfers to diffuse surface are recorded in the mesh or in caustic textures and displayed using straightforward polygon rasterization hardware. Automatic methods to choose between particle storage in mesh or in textures are provided, as well as means to restrict the texture size. Specular paths to the eye are interactively displayed using the Render Cache method (refer to Section 3.6).

To obtain fast updates during object motion, the number of particles resent during object motion has to be reduced. As for diffuse transfer in the Line-Space Hierarchy algorithm, the particle paths that are affected by the dynamic object movement have to be quickly identified. This is done using an octree which is subdivided up to a user-specified maximum depth around the dynamic object position. Links that are used to transfer particles are inserted in the octree cells they traverse (in hash tables for quicker access) with the corresponding receiver element. During object motion, the octree is consulted to access the affected links. Only a small, user controllable percentage of them is reemitted. Figure 5 shows a typical caustic quality degradation during object motion.

This algorithm allows quasi interactive update of global illumination solutions including specular effects within a couple seconds per frame. As a comparison, the initial frame solution for the example scene of Figure 5 took 35 minutes to compute. However, this method is controlled using several user fixed thresholds. There is no automatic method yet to choose their value to guarantee a given frame rate. This strongly reduces its usability for non-expert users.

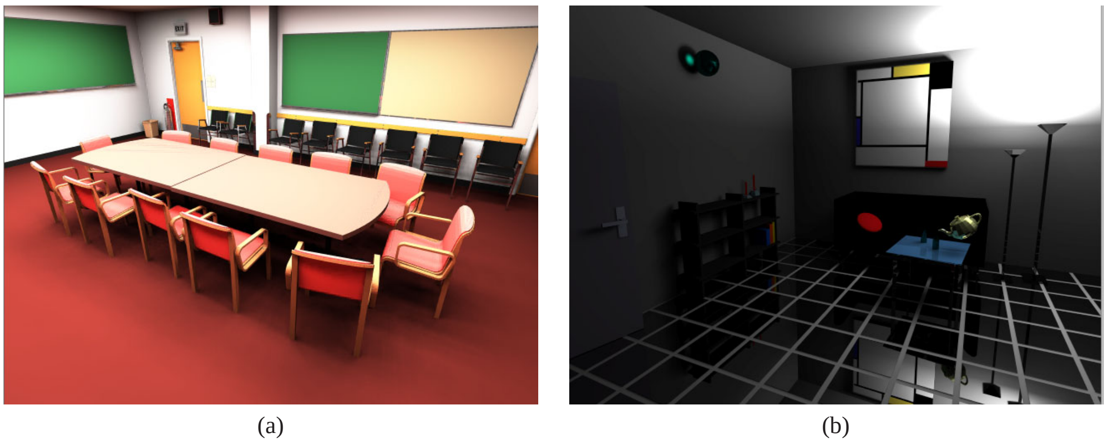

Figure 4: *Example frames with global illumination effects obtained using distributed ray-tracing engine for scenes composed of a) 300,000 and b) 34,000 triangles. On a cluster of 16 CPUs (Athlon1800+) the frame rates 1.5 fps. and 1.84 fps. have been achieved, respectively. To fully eliminate all visible image artifacts resulting from changes of the view direction or other changes in the scene 1–2 seconds of the computation are usually required. (Images courtesy of Ingo Wald, Thomas Kollig, Carsten Benthin, Alexander Keller, and Philipp Slusallek.)*

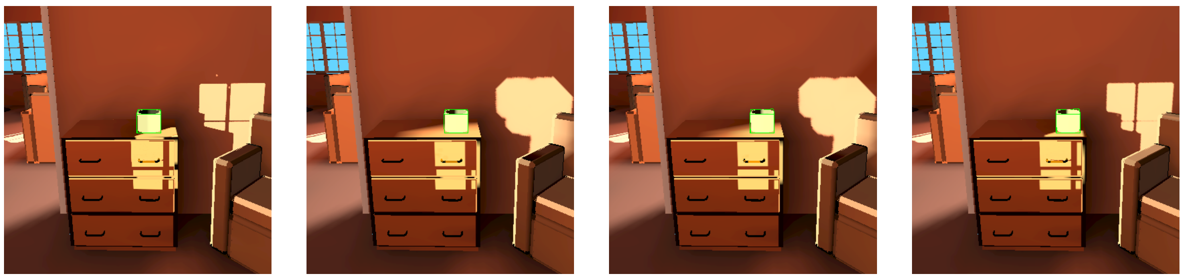

Figure 5: *Update of caustic textures during object motion. From left to right: initial position with good quality, two intermediate positions with degraded quality (3 sec/frame), final image quality 20 sec. after the end of motion. (Images courtesy of Xavier Granier and George Drettakis.)*

3.6. Reconstructing Illumination from a Sparse Set of Samples

As acknowledged in the previous sections, high quality global illumination methods allowing specular effects are usually computationally intensive. In particular, the interactive methods reviewed earlier usually rely on ray tracing to display specular paths to the eye. Maintaining interactive frame rates when taking into account such view dependent specular effects usually requires a considerable amount of processing power (see Section 3.4). Moreover, if updating the geometry requires a couple seconds, interactive fine tuning of objects positions, as for example needed in scene design applications, becomes tedious.

Extending the work of Bergman *et al.*[4] and Bishop *et al.*[7], the algorithms reviewed in this section propose to decouple global illumination computations from geometric projections and image rendering. As such, these algorithms are not actual global illuminations algorithms: they do not aim at computing the illumination, but at providing an interface for the interactive display and update of global illumination solutions, when the actual underlying algorithm cannot provide the required frame rate.

Given a sparse set of high quality illumination samples computed asynchronously by a separate process (typically using Ray Tracing or Monte Carlo Path Tracing[38, 45, 72]) the illumination is progressively reconstructed, either in image space (see Section 3.6.1) or in object space (see Section 3.6.2). Therefore, when performing a change, these algorithms always offer a very fast feedback even if the resulting image accuracy may be poor. Eventually, when no changes are performed, the output will be refined until the maximum quality is obtained.

3.6.1. Reconstruction in Image Space

Walter *et al.*[79] proposed a mechanism based on point reprojection and progressive, adaptive sampling of the illumination to provide fast feedback for the interactive use of a pixel or ray based renderer (e.g. Ray Tracing or Monte Carlo path tracing). Images are generated by reprojecting samples previously stored in the so-called *Render Cache*. A depth culling heuristic is used to remove a large number of points that should have been occluded. Linear interpolation between reprojected points and filtering on 3×3 neighborhood are used to fill in the remaining gaps in the image.

Samples in the Render Cache age at a constant rate, and are progressively replaced in the cache by newer ones. New samples are requested in areas most likely to change. A priority image is generated, based on the samples' age and the number of neighboring samples used during the interpolation step (in order to give higher priority to image regions which have been generated using a lower density of samples from the cache in the previous frame). An error diffusion dithering algorithm is used to transform the greyscale priority image into a binary image, encoding whether a sample for a given pixel is to be generated or not. This dithering ensures that the sample distribution is uniform in similar priority regions.

Several heuristics are used to identify samples that are likely to be invalid. For instance, the authors advise premature aging of those in the neighborhood of a pixel that has suddenly changed color. However, this color change heuristic can be falsely triggered by Monte Carlo noise. In this case, when using a stochastic global illumination method, it is therefore necessary to provide an estimate of the noise floor that is to be expected from the sampling process.

When the rendered sequence of images offers enough time coherence, the quality of intermediate images is sufficient to give the user a clear idea of what is happening. However, the artifacts caused by the progressive update of objects or shadow positions may sometimes be distracting. Moreover, when the desired image is rapidly changing (e.g. when the camera is quickly moving, or when stepping through a wall), most samples collected from the previous frame are invalid, and the display needs some time to be correctly updated. Due to the fact that the samples are requested according to the last frame computed, some screen areas cannot be covered by samples , for certain camera motions (e.g. backward). As a consequence, the intermediate reconstructed images will not cover the whole screen.

As the illumination samples can be computed asynchronously, multiple rendering process can be used in parallel. Obviously, as more samples are provided to build the image, the aforementioned artifacts are consequently reduced. For a single rendering process, potential frame rates of about 14 fps. on a R10000 195 MHz processor are reported *for Ray Tracing* with a screen resolution of 256×256. Substantial improvements have been reported using up to 60 processors on an SGI Onyx2 computer.

Some additions to the original Render Cache algorithm have been proposed recently[78]. An additional filtering pass and the use of predictive sampling reduce the amount of visual artifacts in intermediate images. Moreover, the authors show that an implementation carefully optimized for current industry standard processors, paying particular attention to memory access coherence, makes the performance scalable with higher display resolutions. Frame rates of about 12 fps. are reported on a single 1.7 GHz processor *for Ray Tracing* with a 512×512 resolution.

The performance of the Render Cache is linearly dependent on the resolution of the display. As noted previously, when the sample density is not sufficient, the reconstructed intermediate images may not cover the whole screen, which will cause holes in the images. As a consequence, rendering higher resolution images requires more samples to be computed if such holes are to be avoided. Moreover, the original Render Cache algorithm is a purely software based approach. While this can be seen as positive, it would probably benefit from the use of some of the facilities commonly offered by most graphics cards (e.g. z-buffering).

3.6.2. Reconstruction in Object Space

In order to avoid the holes that can be witnessed with point based approaches such as the Render Cache, reconstruction in object space, using a polygonal mesh and classical rasterising hardware to display it, seems to be a reasonable alternative.

In order to allow interaction with objects in the scene, and at the same time ensure that the objects' geometric representation is always correct, Tole *et al.*[69] proposed a new object-space caching structure: the *Shading Cache*. As in the Render Cache approach, high quality illumination samples are provided by a separate asynchronous process. Those samples are used to progressively update the shading of patch vertices in a 3D mesh. As this polygonal mesh is displayed using the graphic pipeline, interactive modification of the scene geometry is always possible in real time. However, the shading of surfaces may be incorrect for intermediate images, and is progressively updated to take into account the objects' new configuration.

This algorithm starts from an unrefined mesh, all geometric primitives in the scene formed of one single patch. This mesh is progressively refined, either to provide a better quality for the current camera and objects position, or to take into account modifications in the scene. As in the Render Cache algorithm, updates are made according to new illumination samples that are requested based on a priority image. In the Shading Cache algorithm, this priority image consists of the difference between the maximum and minimum tone-mapped color of each patch in the current view. However, in this case, patches, instead of points in image space, are to be selected for update. Therefore, here the sample selection cannot rely on the dithering of a greyscale image. Instead, greyscale values are normalized and used as probabilities to select a given patch in the current view for update.

A flood filling algorithm is provided to ensure that small patches with high priority, such as sharp shadow borders, have fair enough chances to be selected.

Selected patches can be updated either by re-evaluating their vertices, or by subdividing them in four and evaluating the children's vertices. Subdivision of patches is possible until they reach sub-pixel size. To ensure that the size of the mesh remains reasonable, patches that have not contributed to the images for several frames, because they moved out of the field of view or their projected area has become smaller than a pixel, are removed. Double buffering is used to avoid concurrent accesses to the vertex array by the progressive refinement loop and the display process.

To take into account the potentially changing view-dependent appearance of specular surfaces, aging is used to progressively recompute their illumination. The aging speed of surfaces is attributed using a perceptually uniform gloss scale[53]. As a consequence, patches whose shading changes more rapidly with the viewpoint are updated more often.

In order to provide fast updates of the illumination, as in Granier and Drettakis' Unified Algorithm (see Section 3.5), a maximum subdivision depth has to be fixed during object or camera motion. When the motion stops, refinement can be continued up to pixel level.

As the shading samples are computed by separate, independent processes, this algorithm can easily benefit from multiprocessing. On a setup consisting of 9 dual 1.7 GHz Pentium IV, for scenes composed of up to $10,000$ primitives, updating the geometry can be done at 40–60 fps., while the illumination update to maximum quality may require 10–20 sec. Performance comparisons provided by the authors show that the Shading Cache requires ten times less samples on average than the Render Cache to effectively render an image. This allows much faster updates of illumination in response to user changes, when a very high-quality, slow sample renderer is used.

3.7. Other High Quality Display Methods

In the previous section we discussed a number of solutions which make possible efficient image display with glossy and specular effects through progressive refinement of sparsely distributed samples in image space (Section 3.6.1) or object space (Section 3.6.2). An interesting alternative is offered by the use of rapidly developing graphics hardware (note that in Section 3.2 we also discuss the use of graphics hardware but in a different context of the lighting computation for radiosity algorithms). Thanks to the speed and flexibility of current graphic cards, many of those effects can be produced almost instantly without the need of progressive refinement.

In this section we only briefly overview hardware-supported techniques for the rendering of glossy and specular effects. More elaborate discussion of those techniques can be found in a specialized survey on similar topics presented by Heidrich[33].

Pioneering work on the rendering of mirror reflections for planar surfaces using multipass pipeline rendering was done by Diefenbach[18, 17] and Wojdala *et al.*[82]. In their methods every mirror plane requires a separate rendering pass, using the mirror reflection of the real observer parameters as the camera position. Higher levels of reflections, i.e. a mirror reflected in another mirror and so on, can be easily computed by recursion of the basic algorithm. Handling light refraction is more complex because it involves a non-linear transformation, which cannot be directly implemented on graphics hardware. However, Snell's law can be well approximated for paraxial rays (small incidence angles) using the tangent law[17], which results in a linear transform.

For handling curved surfaces, a better performance can be achieved using the standard environment mapping technique, which hardware implementation is now available on most graphics cards. Complex reflectance models can be rendered using this technique through their decomposition into simpler parts which can be evaluated in multipass rendering with alpha blending[34]. Using prefiltered environment maps[34, 39, 8] a complete global illumination of an object in the scene can be rendered with realtime performance. Each entry of the prefiltered map contains an outgoing radiance value in a given direction, which is precomputed as a BRDF weighted integral of irradiance incoming from the whole scene.

For dynamic environments it is essential to update prefiltered maps at interactive speed. Kautz *et al.*[40] use hardware accelerated convolution operation to efficiently filter the map for the Phong reflectance model, which is feasible due to radial symmetries inherent for this model. The costly convolution operation can be avoided based on prefiltering technique proposed by Ramamoorthi and Hanrahan[55]. They assumed that the irradiance function is smooth and continuous, so that only 9 spherical harmonics coefficients are enough to give accurate approximation of irradiance for diffusely reflecting surfaces. Those coefficients can be computed in linear time with respect to the number of pixels in the map. It is therefore possible to prefilter environment maps for diffuse objects on the fly.

All these techniques properly consider only illumination by distant objects. Recently, Sloan *et al.*[66] introduced a transfer function, which maps incoming radiance to outgoing radiance and can be precomputed for any rigid object. The function can be used to render self-interreflection and self-shadowing effects on-the-fly for an object illuminated by dynamic, local, and low frequency lighting. The self-transfer function can also be generalized to a neighborhood-transfer function in order to find the local influence of a given object on the illumination of neighboring objects.

For rendering of glossy effects, interactive ray tracing[76, 77] is a viable solution as well. As we discussed in Section 3.4

ray tracing can be an algorithm of choice especially for complex scenes. Also, ray tracing is very flexible in handling various types of geometry and data structures for storing lighting information. Pixel data for rendered images can be easily processed with a floating precision. This might be required for tone mapping operators[70] that are necessary to display visual information in a perceptually plausible way on a given display device.

4. Offline methods

In the previous section we discussed interactive global illumination and rendering solutions, which usually trade the image quality for the response speed. In this section we focus on off-line solutions used in the final production of high quality animations. Such high quality is required in applications such as entertainment, advertisement, education, engineering, architecture, urban planning, and many others. Adding global illumination effects in the aforementioned applications greatly improve their capacity to reproduce reality and therefore their credibility.

A vast majority of the algorithms reviewed in this section rely on the *a priori* knowledge of all modifications in the scene to ensure an uniform image quality, both spatially and temporally. The main objective of those methods is to exploit the spatio-temporal coherence of illumination distribution in the subsequent animation frames in order to improve the computation efficiency and reduce the temporal aliasing.

Again, as in Section 3 two separate problems can be distinguished:

- First, the actual simulation of global illumination has to be performed according to the animation script, which describes all changes in the scene as a function of time. Relevant methods will be reviewed in Sections 4.1– 4.5. We start with the off-line solutions evolving from progressive (Section 4.1), hierarchical (Section 4.2), and Monte Carlo (Section 4.3) radiosity algorithms, extended into the time domain. We then proceed to a review of the image-based framework for lighting computation in animations (Section 4.4). Finally, we discuss some applications of human perception models to improve the efficiency of density estimation and irradiance cache computations for dynamic environments (Section 4.5).
- Second, the results of the global illumination computation also has to be carefully displayed to obtain the high quality animation frames. Basically all algorithms discussed in Sections 4.1– 4.5 have inherent problems to store the results of the global illumination computation with a spatial resolution that is suitable for their immediate display for an arbitrary camera view. The algorithm of choice to improve the spatial resolution of lighting details for a given camera view is the so-called *final gathering*. During the final gathering stage, the global illumination computation is not explicitly performed, but rather the results of such computation are spatially integrated at the selected sample points which are located in the object space. Nevertheless, for the sake of completeness of this report, we review in Section 4.6 some final gathering solutions, however, we limit ourselves only to those solutions which exploit the temporal coherence between frames.

4.1. Radiosity for animations

As recalled previously in Section 3.1, finite element methods, such as the radiosity methods, are particularly well suited to the computation of view independent global illumination solutions when all surfaces are considered diffuse. Please refer to Section 3.1 for a brief reminder of the principle of those methods.

The first attempts at efficiently using radiosity methods to compute high-quality animations focused on adapting the full-matrix and Progressive radiosity algorithm to take advantage of time coherence. Baum *et al.*[1] proposed to project the bounding box of the moving objects on hemicubes to determine the form-factors that need to be updated. More recently, Pueyo *et al.*[54] proposed to follow the principle of cell animations and compute every frame as the sum of a fixed image and a frame dependent one. As a consequence, this method requires that the camera position should be fixed.

Since these methods are based on full-matrix or Progressive Radiosity, they are somewhat inefficient when compared to more recent algorithms, and scale poorly when the complexity of the produced mesh exceeds a couple thousand patches.

4.2. Space-Time Hierarchical Radiosity

The Space-Time Hierarchical Radiosity method is aimed at computing the radiosity function B for every point of a scene S composed of rigid objects, all potentially moving (including the primary light sources), for every possible time of a given finite time interval T. Every surface is assumed to be perfectly diffuse, and all object paths are supposed to be known in advance.

Like the Dynamic Hierarchical Radiosity and the Line-Space Hierarchy (see Section 3.1), the *Space-Time Radiosity* method proposed by Damez and Sillion[16] is based on Hierarchical Radiosity. However, instead of solving a sequence of equations (one for each frame of the animation), this algorithm proposes to solve one integral equation that models all light exchanges between the surfaces of the scene S during the animation interval T:

$$\begin{aligned} \forall X &= (x,t) \in S \times T \\ B(X) &= E(X) + \int_{Y=(y,t')\in S\times T} B(Y)K(X,Y)\,dY \end{aligned}$$

Therefore, instead of updating the hierarchical spatial

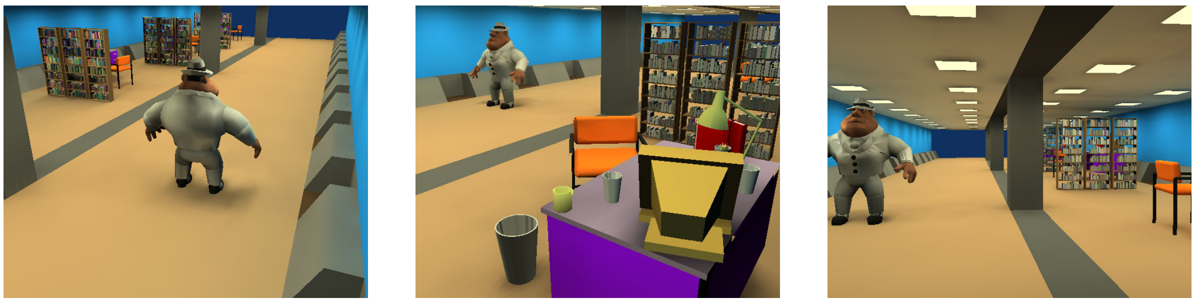

Figure 6: *Example stills from a* 500 *frames long animation computed using Space-Time Hierarchical Radiosity. The scene was build using* 35,000 *input surfaces. Rendering was approximately* 20 *times faster than a frame per frame computation, but required* 7 *times more memory.*

mesh on a frame-by-frame basis, this method builds a mixed spatial and temporal mesh which describes the variation of radiosity on all surfaces, during all the animation.

Each element of the space-time mesh is the Cartesian product of a polygon and a certain portion of the animation time interval, called the *validity interval* of the element. The potential interaction between two elements i and j, with validity interval T_i and T_j, is determined by a "space-time" form factor $F_{i,j}$. If constant radiosity is assumed over each patch of the mesh, this space-time form factor has the following form:

$$F_{i,j} = \frac{1}{(t_i^{final} - t_i^{initial})} \int_{T_i \cap T_j} f_{i,j}(t)\, dt$$

where $f_{i,j}(t)$ is the classical 3D form factor between the geometric supports of i and j at time t.

As in the classical Hierarchical Radiosity, the mesh is built as a side effect of a link refinement procedure, which ensures that all links are placed at an appropriate level of precision. This is done by recursively evaluating the links precision, pushing coarse links down the hierarchy, and eventually splitting the emitter element or the receiver when more precision is needed. However, in the space-time case, two questions have to be answered by the so-called refinement oracle for each is link in the hierarchical mesh:

1. Is this link an accurate enough representation of the light exchange between the emitter and the receiver?
2. If it is not, does the precision need to be improved along the time dimension, or in the spatial dimension?

Hanrahan *et al.*'s implementation of Hierarchical Radiosity uses a refinement criterion based on an estimate of the amount of energy exchanged through the link, because it is directly linked to the amplitude of the approximation made. Such criterion cannot answer our second question. Therefore, a refinement oracle based on direct sampling of the radiosity equation kernel function was proposed[16]. The mean value of this set of samples is compared to a precision threshold to evaluate the precision of the link. The variances, with fixed time and varying sample points and with varying point and fixed time are compared to decide if the considered link should be split along the temporal or the spatial dimension. This criterion was later improved[15] using an estimate of the radiosity error on the receiver (similar to the criterion proposed by Bekaert *et al.*[2]).

As this method is a straightforward extension of the classical Hierarchical Radiosity algorithm, it can benefit from most of the additions that have been proposed through the last decade. For instance, it has been shown[15] that the Space-Time Hierarchical radiosity can successfully use clustering approach, and therefore deal with scenes composed of tens of thousand polygons, and wavelets basis to improve the temporal continuity of the solutions and reduce the overall weight of the mesh. Also, it can probably easily be extended to benefit from a multiprocessor architecture, using an approach similar to the one proposed by Sillion *et al.*[65].

However, this similarity also causes this method to suffer from the same problem as the classical Hierarchical Radiosity algorithm. In particular, meshing artifacts become obvious, both in the spatial and temporal dimensions, if a proper reconstruction pass or final gathering is not applied. Using elements with radiosity varying linearly over time greatly improves the solution temporal continuity[15].

As this method heavily relies on temporal coherence in the light exchanges, its performance depends essentially on the the expected continuity of the radiosity function. The amount of memory required by the mesh can grow very rapidly with an increasing complexity in direct illumination. On the other hand, when the illumination is only changing slowly over time, or very locally, time subdivisions are very few. In such a case, the speedup (compared to a frame-by-frame hierarchical radiosity computation of equivalent precision) is large, and the increase in memory consumption is less important.

The speedup can vary from 2 (for a simple 3 sec. animation in a scene composed of less than 100 polygons) to almost 20 (for 24 sec. for the animation shown in Figure 6), while moderately complex scenes (about 10k input surfaces) will lead to a typical acceleration factor of 6 to 8. However, the memory used to compute the animation (compared to the memory needed for one frame) will be multiplied by a factor varying typically between 7 to 12. Clearly, a method to reduce the RAM requirements (e.g. allowing to cache large unused portions of the mesh on disk) is called for[14].

4.3. Multi-Frame Lighting Method

This algorithm, presented by Besuievsky *et al.*[5, 6], also aims at computing view independent radiosity solutions for diffuse animated scenes, where all motions are known in advance. All objects, including light sources, can be animated.

It is a finite-element method that uses global Monte Carlo estimates of the diffuse light transport[57]. Global lines are cast independently from surface positions, with an uniform density all over the scene. Such lines can be generated e.g. by joining random pairs of point taken in a sphere bounding the whole scene.

The Multi-Frame Lighting method benefits from time coherence by using the same set of global lines for the whole animation. Global lines are tested for intersection once against all static objects, and against every frame position of the dynamic objects. Visibility lists are built for each global line. Static and dynamic lists are then merged and sorted for every frame, and used to perform the light transfer[57].

As a result, the longer the animation, and the higher the ratio of static surfaces versus the total number of surfaces, the better this method performs[6]. However, as the algorithm stores in memory the radiosities for all mesh elements and all frames, the memory consumption of the algorithm grows quickly, whereas the increase of the speedup ratio with the length of the animation is bounded. The authors report that a good compromise between memory consumption and speedup can be obtained by splitting the animations in segments of about 60 frames. In this case, typical speedup reported, when compared to a frame-by-frame computation, ranges from 7 to 25. Solutions for a possible distributed processing version of this algorithm are also presented[6], though they have not been actually implemented.

A major limitation of this method is that it does not allow adaptive meshing. As a consequence, the same mesh is used through all the animation. High spatial frequency variations of the radiosity function (e.g. sharp shadow boundaries) are poorly reconstructed. If a very dense mesh is used, the overall memory consumption of the algorithm and the computing time will raise dramatically as more lines are needed to avoid noise artifacts. However, this issue can probably be addressed using a hierarchical approach[3].

4.4. Image-Based Method Handling Dynamic Environments

Frame-to-frame changes in lighting are often slow, and for the indirect lighting, usually quite smooth as well. Based on this observation, the global illumination can be computed only for a limited number of images, and then reconstructed using those images for an arbitrary camera position and an arbitrary moment in time. A significant step toward this goal was done by Nimeroff *et al.*[36] who proposed a powerful range-image based framework for handling global illumination in dynamic environments. In this framework images for arbitrary camera locations can be interpolated within a "view-space" spanned by the base range images. Nimeroff *et al.* discuss two major problems that must be addressed in this framework:

1. Where to place the camera in order to cover the whole scene and ensure the highest possible quality of the resulting images?
2. How to choose the snapshots of global illumination in the temporal domain to capture its changes in dynamic environments?

For the initial step, Nimeroff *et al.* propose to place the keyframe cameras at the corners of the "view-space" and to compute the corresponding base images. For simplicity only 2D placement of the camera in the scene is investigated, and the vertical positioning of camera is ignored. The quality is evaluated by computing the percentage of pixels whose item buffer identifiers are identical for all these images. If the percentage is below a predefined threshold value, the space is subdivided and additional keyframe cameras are inserted. This quality criterion captures well the occlusion relations between objects in the scene and is swiftly computed using graphics hardware. However, this criterion might be unreliable for several global illumination effects. For example, for scenes with mirrors and transparent objects the distortion of reflected/refracted patterns is not estimated, while it can be quite significant in the derived images due to interpolation. Also, the distortion of texture patterns is ignored.

Direct and indirect lighting are handled asynchronously and are sparsely sampled in time. To reconstruct the global illumination, they are interpolated between independently selected base images. The time steps for recomputing the lighting are found by recursive subdivision. At each time step the lighting is calculated for a number of vertices using wavelet radiosity. Then the differences between the corresponding vertices are computed. If differences larger than an assumed threshold are found for a certain percentage of vertices the time sequence is subdivided. Since such subdivisions are decided independently for direct and indirect lighting, it may happen that significant differences in the indirect lighting cause a subdivision, while those differences might be washed out in the resulting images by the direct lighting[24]. As a result, the criterion driving the time sequence subdivision might be too conservative. Also, tone reproduction[70] is not applied to the resulting lighting. This is difficult in the view-independent framework proposed by Nimeroff *et al.* because the eye adaptation conditions cannot be established. Both effects can affect the visibility of changes in lighting.

While the original goal of the framework proposed by Nimeroff *et al.* is off-line high quality rendering, with the current development of graphics hardware (image warping can be easily implemented with real-time performance), it might prove suitable for interactive applications as well. This algorithm ability to generate new images for new camera positions contained in the volume delimited by the original viewpoints is indeed useful for interactive applications. The framework by Nimeroff *et al.* can provide such views at selected moments in time for predefined scene changes. One possible application of this framework could be designing the camera paths while having a good approximation of global illumination changes as a result of objects motion in the scene.

The interpolation of lighting between two time steps is an important feature of Nimeroff's framework. The continuity of changes in the lighting distribution between time steps eliminates popping effects, which may result from switching between two distinct lighting distributions as in[83] (refer to Section 4.5). However, the accuracy of lighting reconstruction fluctuates between frames, achieving the highest level for the keyframes, and then gradually decreasing for inbetween frames.

4.5. Perception-Guided Animation Rendering

Nimeroff *et al.* proposed a simple energy-based metric to guide the keyframe placement in order to reduce the error of interpolated lighting for inbetween frames. However, it is not clear whether for a given error threshold the lighting distribution errors are perceivable or not. Ideally, some perception-based animation quality metrics are required that can decide whether the errors introduced by exploiting the temporal coherence are below the sensitivity level of human observers. Such error metrics are common in digital video compression and broadcasting applications (refer to a survey of such metrics resulting from the research performed by the Video Quality Experts Group[12]).

Myszkowski *et al.*[49] proposed a perception-based spatiotemporal Animation Quality Metric (AQM) which was designed specifically for handling synthetic animation sequences. We briefly overview the most important characteristics of the AQM and we discuss its application to guide the global illumination computation for dynamic environments[50] in Section 4.5.1.

The human visual system model used in the AQM can be considered as a quite advanced one, however, it is limited to the modeling of the early stages (from the retina and to the visual cortex V1) of the visual path. Since gains by adding further extensions to such early vision models are rather small[52], some attempts of using higher level perceptual and cognitive elements have been introduced in the context of animation. Yee *et al.*[83] and Haber *et al.*[28] showed successful applications of visual attention models to improve the efficiency of global illumination and rendering computation. We discuss those solutions in Section 4.5.2.

4.5.1. Animation Quality Metric

The Animation Quality Metric is an objective metric of image quality, which takes as input two animation frames and generates a map of perceivable differences between those frames. Based on the map it is possible to predict how strong and where on screen the differences between frames can be seen. The central part of the AQM is a model for the spatiovelocity Contrast Sensitivity Function (CSF), which specifies the detection threshold for a stimulus as a function of its spatial and temporal frequencies[44]. Also, visual masking is modeled, which affects the detection threshold of a stimulus as a function of the interfering background stimulus which has similar spatial frequency characteristic[21]. The AQM models temporal and spatial mechanisms (channels), which are used to represent the visual information at various scales and orientations, in a similar way as the primary visual cortex does[13].

Myszkowski *et al.* used the AQM to decide upon the computation stopping condition for the Density Estimation Photon Tracing (DEPT) algorithm[74]. The DEPT is similar to other stochastic solutions in which photons are traced from light sources towards surfaces in the scene. The energy carried by every photon is deposited at the hit point locations on those surfaces[32, 62, 80]. A simple photon bucketing on a dense triangular mesh is performed, and every photon is discarded immediately after its energy is distributed to the mesh vertices. Efficient object space filtering substantially reduces visible noise. Excessive smoothing of the lighting function can be avoided by adaptively controlling the local filter support, which is based on stochastically-derived estimates of the local illumination error[74, 80].

Myszkowski *et al.* extend the DEPT algorithm to handle animated objects, and the proposed extensions could be easily applied to other stochastic algorithms such as Photon Mapping[37]. Direct lighting is assumed to be computed for each frame. Initially, the indirect lighting function is sparsely sampled in space for all frames (not just for fixed keyframes[36, 83]) within a given animation segment. Then, based on the obtained results, a decision is made whether the segment can be expanded/contracted in the temporal domain. Since the validity of samples may depend on the particular region in the scene for which indirect lighting conditions change more rapidly, different segment lengths are chosen locally for each mesh element (used to store photon hits), based on the variations of the lighting function. Energy-based statistical measures of such local variations

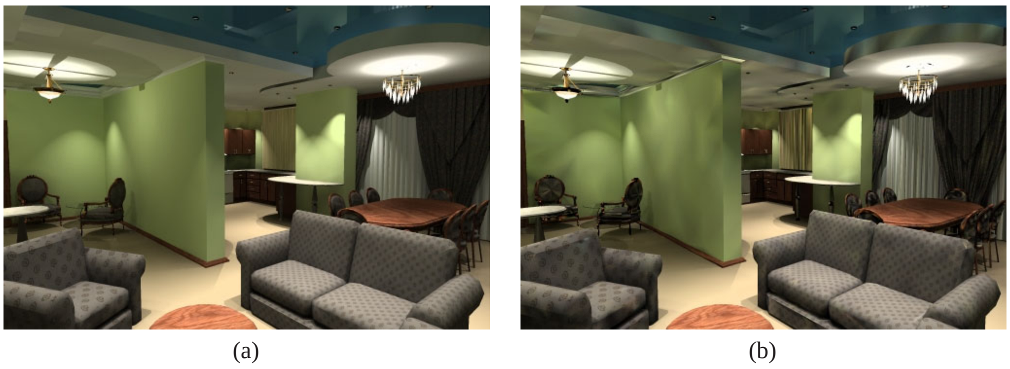

Figure 7: *Example animation frames (a) with and (b) without temporal processing for about 80,000 photons per frame. An animation segment length of 31 frames was considered for the temporal photon processing. The average time of the indirect lighting computation per frame was less than 3 seconds.*

are used to calculate the number of preceding and following frames for which samples can be safely used for a given region. More samples are generated if the quality of the frames obtained for a given segment length is not sufficient.

The perception-based AQM is used to choose an average number of photons per frame for each segment to prevent perceivable degradation of animation quality. The two animation frames, which are required as the AQM input, are obtained by splitting all temporally processed photons into two halves and generating the corresponding frames $I_1(K)$ and $I_2(K)$. Since the AQM test is costly it is performed only for the central frames K in each animation segment. The same mesh is used for the indirect lighting reconstruction in $I_1(K)$ and $I_2(K)$, so the solution bias is the same for both compared frames. Effectively the AQM is used to measure the perceivable differences between frames $I_1(K)$ and $I_2(K)$, which result from the stochastic noise. The AQM provides a conservative stopping condition for photon tracing when the noise falls below the sensitivity level of the human observer. Tracing more photons cannot improve the perceived quality of the indirect lighting reconstruction due to limitations in the spatial mesh resolution.

As the final step of the photon-based lighting reconstruction, spatial filtering[74] is performed for those scene regions in which a sufficient number of samples cannot be collected in the temporal domain. For the final rendering the indirect lighting is reconstructed using the techniques aforementioned, while specular effects and direct lighting are computed for every frame separately by ray tracing.

In the algorithm proposed by Myszkowski *et al.* sparse sampling of indirect lighting is performed for every frame, and the final lighting reconstruction is based on the processing of lighting distributions for a number of subsequent frames placed along the animation path. Thus, at each moment of time a similar level of accuracy of indirect lighting can be obtained. Such a framework is less prone to perceivable errors, and the probability of overlooking some important lighting events between keyframes is substantially reduced. The reported speedup of indirect lighting computation is over 20 times for tested scenes[50] and the resulting animation quality is much better than in the traditional frame-by-frame approach. While only scenes composed of less than 100,000 triangles have been tested, a similar speedup can be expected for more complex scenes because collecting photons in the temporal domain is always cheaper than shooting them from scratch for each frame.

The main drawback of this algorithm is the limited accuracy of indirect lighting reconstruction, which is imposed by the spatial resolution of the mesh used for collecting photons. Obviously, the mesh resolution can be set arbitrarily fine and more photons can be traced, however, some local adaptation of the mesh density driven by the lighting distribution would be far more efficient. For example, it is very difficult to obtain high quality caustics using the algorithm developed by Myszkowski *et al.*

Recently, a plug-in for the animation package 3D Studio Max was developed using a similar approach. The main difference is that in the original algorithm, photons are collected from both previous and following frames, whereas the new approach only considers photons collected from frames preceding the current one. Also, the spatio-temporal processing of photons is improved by applying an adaptive nearest neighbors algorithm, which operates both in the space and time domains. Figure 7 shows example animation frames obtained using the plug-in with and without the temporal photon processing for the same number of traced photons per frame. The differences in quality are even more pronounced in the context of animations, when additionally flickering effects can be seen in the case shown in Figure 7b.

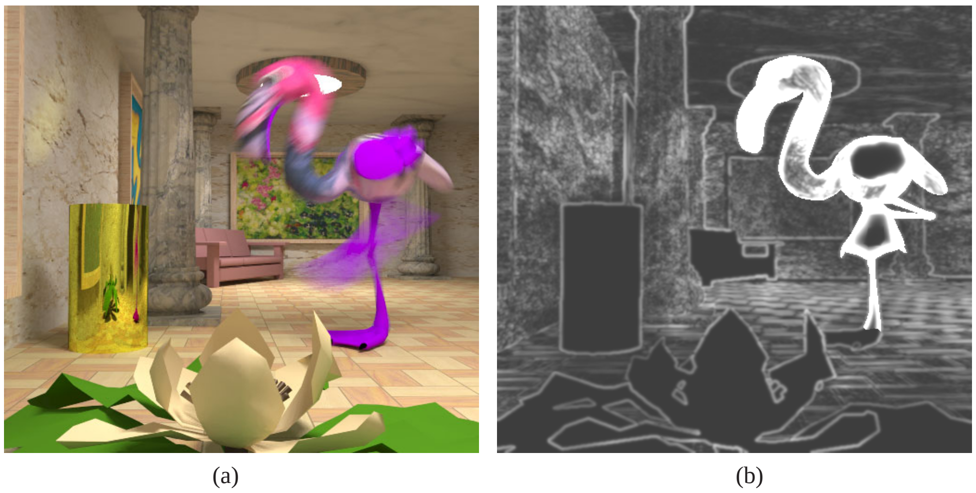

Figure 8: *Spatiotemporal error tolerance map processing. (a) An input frame with dynamic objects rendered for a moving camera and (b) the corresponding spatiotemporal error tolerance map obtained using the spatiovelocity CSF processing. Brighter areas on the map denote areas where less effort should be spent in computing the lighting solution. (Images courtesy of Hector Yee, Sumant Pattanaik, and Donald Greenberg.)*

4.5.2. Visual Attention Modeling

In this section we discuss two techniques which use visual attention modeling at the stage of lighting simulation and rendering to improve the computation performance. A basic idea is to reinforce the computation in those image regions that attract more attention of the observer.

Visual Attention Modeling in Lighting Computation. Yee *et al.*[83] proposes an interesting application of a visual attention model to improve the efficiency of indirect lighting computations in the RADIANCE system[81] for dynamic environments. A saliency map of the human visual system sensitivity based on the visual attention model[35] and spatiovelocity CSF[44, 49] is created (Figure 8 shows an example of the spatiotemporal error tolerance map resulting from the spatiovelocity CSF processing for a selected animation frame). The saliency map (refer to Figure 9b) is used to control the irradiance caching for secondary lighting in the RADIANCE system on a per pixel basis. For less salient image regions greater errors can be tolerated, and the indirect lighting can be interpolated for a larger neighborhood. This makes caching more efficient at the expense of blurring details in the lighting distribution. The reported speedup of irradiance caching falls into the range 3–9 times.

Further speedup is obtained by computing the indirect lighting only for selected keyframes and re-using the obtained lighting for the remaining inbetween frames. Interpolation of indirect lighting between the keyframes is not performed and the same number of inbetween frames (between each pair of keyframes) is always used. Since there is no verification of the validity of applying the same keyframe lighting to the inbetween frames, some popping effects due to switching the lighting between keyframes can be perceived. To reduce this effect, Yee *et al.* sampled the indirect lighting more densely, e.g. every ten frames.

The technique proposed by Yee *et al.* significantly improves the performance of the RADIANCE system for animation rendering applications. However, variability in the selection of the regions of interest (ROI) for different observers, or even for the same observer from session to session, can lead to some degradation of the animation quality. This is the case for those image regions that were not considered as important attractors of the visual attention in the saliency map. Yee *et al.* reports that such degradations of quality could be perceived when the same animation sequence is viewed more than once by the same observer. For those applications which require high quality animations, possibly viewed many times by a large number of observers, the approach proposed by Yee *et al.* might not be suitable. Yee *et al.* also ignore visual masking which plays an important role in hiding imperfections of reconstructed lighting[21].

Visual Attention Modeling in Rendering. Haber *et al.*[28] use the visual attention model developed by Itti[35] for interactive walkthrough applications with precomputed global illumination. They use graphics hardware for rendering the Lambertian surfaces (refer to Figure 9a), and allocate the remaining computational power to correct the appearance of non-diffuse objects on-the-fly using ray tracing (refer to Figure 9c). The question, how to obtain the best image quality as perceived by a human observer within a limited amount of time for each frame, arises. Haber *et al.* address this problem by enforcing corrective compu-

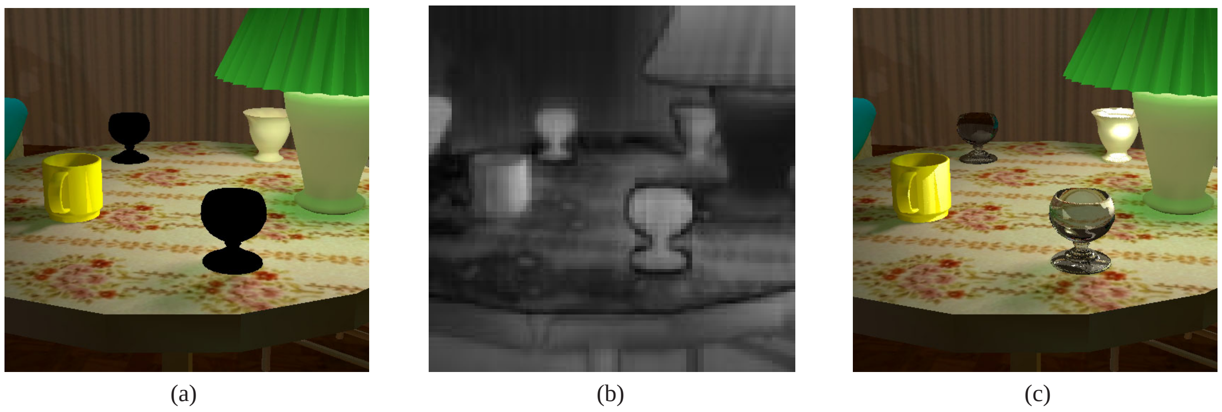

Figure 9: *Visual attention processing. (a) An input image showing a view-independent global illumination solution, which has been computed in a preprocessing step. (b) The resulting saliency map, where bright grey levels encode objects with strong saliency. (c) The fully converged solution after corrective splatting has been applied according to the results of the attention model.*

tation for those non-diffuse objects that are selected using the Itti model. This model is used to predict the saliency of selected objects (an example of such a saliency map is shown in Figure 9b) in the scene and to distribute the computational resources according to the predicted visual attraction of those objects. In this application, the model of visual attention is used just to order the computation, which is much safer than making irrecoverable decisions on the reduction of local image quality[83]. Haber *et al.* included the volitional, task driven component (under cognitive control) to the Itti model, which makes it possible to influence the computation priority according to the user preferences. The global illumination solution presented by Haber *et al.* is limited to walkthroughs and the main focus of their work was on the high quality interactive image display which we discuss in Sections 3.6 and 3.7.

4.6. High quality display/ Final gathering methods

Most of the mesh based off-line methods, though their results can be displayed "as is", generally require a second pass to obtain high quality images from the solution they computed. This second pass, when operating on a frame-by-frame basis, proves even more expensive than the first one. The algorithm of choice for the final high quality rendering is the so-called *final gathering* in which the lighting function is reconstructed at the level of image pixels. Usually the direct lighting is computed for a surface region represented by a given pixel, and the indirect lighting is obtained through irradiance integration over the environment surrounding this surface.

Final gathering has been initially introduced in the context of the radiosity algorithm to overcome the problems arising with the mesh-based storage of lighting[56, 46, 67]. Recently, some efficient versions designed specifically for the Hierarchical Radiosity with clustering have been proposed[58, 59]. Final gathering is also commonly used after Photon Mapping algorithms because a direct rendering of diffuse illumination based on the density estimation of photons leads to poor image quality[37]. All these final gathering approaches ignore temporal processing and therefore are suitable only for the rendering of static environments.

Recently, Martin *et al.*[47] proposed a final gathering algorithm for animated hierarchical radiosity. The temporal coherence in the radiosity solution is obtained using the line-space hierarchy[20, 60, 26], which is designed to identify efficiently links affected by changes in the scene and redistributing the lighting energy for the current scene configuration (we review the related techniques in Section 3.1).

Because of the final gathering step only a coarse global radiosity solution[46] is computed. The temporal changes in the lighting are updated by comparing the previous and current radiosity solutions. The radiosity solution for visible surfaces is stored in a hierarchical tree of textures representing the space-time illumination. Therefore, for each visible surface patch a texture movie is created and displayed during rendering in a given time interval.

The resolution of textures is adaptively chosen so that at least one texel stores lighting information for each pixel in the final frame. Martin *et al.* found that usually around ten times more texels than pixels are sufficient to obtain high quality final frames. The texture size is upper bounded and scene independent because the textures are maintained (resp. are newly created) only for the surfaces that remain visible (resp. just became visible) in the current frame, and are destroyed immediately when those surfaces become invisible. The texel lighting is computed during the gathering step using the hierarchical radiosity links. They are classified as *good* if the gathered lighting error is within given bounds, and *bad* otherwise. For good links a final gather step is not required, and resulting lighting is accumulated in a texture using linear interpolation within a given patch. For each shooter polygon associated with a bad link the graphics hardware is used to estimate the visibility of texels in a receiver patch using the projective shadows technique. In the temporal domain bad links are classified as *static* or *dynamic*. The costly visibility computation is performed once for a given time interval for static links, and is repeated for each frame for dynamic links.

The final gathering method proposed by Martin *et al.* leads to significant rendering speedup (1.7–7 times in the examples given by the authors). However, the method shares typical drawbacks of hierarchical radiosity solutions such as poor handling of non-Lambertian surfaces and significant storage costs required by the link data structures. Those costs are even more significant in the presented solution because the history of links is also stored, e.g. links are not deleted when refined for possible reuse in different time intervals. Also, in Martin *et al.*'s implementation surface clustering is not supported, which increases the quantity of stored links even further.

5. Summary and conclusion

In this section we present a summary of this report, in the form of synthetic Tables 1 and 2. We purposely omitted some of the algorithms (often pioneering ones that we reviewed for the sake of completeness) when a more recent one uses a similar approach, with improved results. Each approach included in these tables has its pluses and minuses. This synthetic classification should help the reader to quickly determine which one is the most appropriate for his needs.

As in the rest of this report, we distinguish between *interactive* and high-quality *offline* global illumination algorithms. Obviously, comparing their performance (in particular concerning speed) would not be meaningful. However, the criteria we use for comparison within the two sets of method are quite similar, the only difference being the criterion chosen for measuring speed. For interactive methods, the *frame rate* is used as a measure of the algorithm efficiency, whereas for offline methods, we report the *speedup* when compared to an equivalent frame by frame computation. Note that for the former, in order to offer a fair comparison basis, figures reported in the original articles by their authors should be weighted according to the hardware the algorithms were implemented on.

As acknowledged earlier in this report, several new algorithms aiming at computing global illumination in animated scenes have been proposed in the past couple of years. Obviously, even if significant progresses have been made, neither a seamless, truly real time, nor a reliable high quality, artifact free global illumination algorithm has been proposed. There is still room for improvement for the algorithms we presented in this report, all of which have their own set of strengths and weaknesses.

Often, the artifacts generated by these algorithms are similar to the one caused by their "static" counterparts. For example, almost all mesh-based method we presented in this report suffer from meshing artifacts that cause aliasing and poor shadow border reconstruction. Similarly, stochastic methods suffer from noise both in the spatial and in the temporal dimensions. Solutions have been proposed to reduce these problems for still images, but most of these approaches do not cure the specifically temporal effects on animations (e.g. popping and flickering).

The algorithms we presented in the this report indeed achieve impressive results. However, their ultimate objectives (real-time or affordable artifact free global illumination for animated scenes) obviously haven't been reached yet. For example, we think that further investigations will be worth pursuing for a better use of temporal coherence in order to reduce flickering artifacts in high quality global illumination. For interactive design scenarios, new methods to progressively add high frequency lighting details on request also seem to be called for. Therefore, we are looking forward new advances in this field and hope this state-of-the-art report will prove a useful reference for future research.

Acknowledgements

We would like to thank the authors of discussed papers who provided us with images illustrating their work. Thanks also to all those who read early drafts of the paper - Philippe Bekaert, Katja Daubert, Jörg Haber, Vlastimil Havran, Jan Kautz, Annette Scheel, and Philipp Slusallek for their helpful comments and suggestions. This work was supported in part by the European Community within the scope of the RealReflect project IST-2001-34744 "Realtime visualization of complex reflectance behavior in virtual prototyping".

	Lighting Effects Handled	**Average Scene Complexity**	**Frame Rate**	**Artifacts**	**Additional notes**
SPT	Diffuse	10^5 Polygons	1 fps. (dual P4 1.7 GHz)	Meshing artifacts + Delayed illumination update (0.25 fps.)	Only point light sources
DRT	All	10^5 Polygons	1.5 fps. (8 dual Athlon 1800+)	Overestimated lighting near photon impact + Delayed illumination update (0.5 fps.)	Requires a PC cluster
UHA	All	10^4 Primitives	0.25 fps. (P3 733 Mhz)	Meshing artifacts	Requires a static solution precomputation
RC	All	10^4 Polygons	0.5 fps. (dual P4 1.7 GHz)	Stale points + Holes in images	Requires a couple of minutes to obtain full quality static images
SC	All	10^4 Polygons	45 fps. (9 dual P4 1.7 GHz)	Meshing artifacts + Delayed illumination update (0.1 fps.)	

Table 1: *Synthetic comparison of interactive algorithms (refer to Section 3).*

	Lighting Effects Handled	**Average Scene Complexity**	**Speedup (average)**	**Artifacts**	**Additional notes**
STHR	Diffuse	10^4 Polygons	$\times 7$	Meshing artifacts + Popping	Very high memory requirements (up to 500MB)
MFLM	Diffuse	10^3 Polygons	$\times 12$	Blurred shadow boundaries	
ISI	All	10^4 Polygons	$\times 10$	Possible artifacts in texturing and specular effects	May require many base images for specular effects
TFPT	Diffuse	10^4 Polygons	$\times 20$	Meshing artifacts	Less popping than with a frame by frame computation
SSVA	All	10^4 Polygons	$\times 8$	Popping	Possible quality problems when submitted to different viewers
TPR	Diffuse	10^2 Polygons	$\times 5$	Popping may be visible when switching texture precision	High memory requirements + No surface clustering

Table 2: *Synthetic comparison of offline algorithms (refer to Section 4).*

SPT:	Selective Photon Tracing[19], Section 3.3
DRT:	Global Illumination based on Distributed Ray Tracing[75], Section 3.4
UHA:	Unified Hierarchical Algorithm[27, 26], Section 3.5
RC:	Render Cache[79, 78], Section 3.6.1
SC:	Shading Cache[69], Section 3.6.2

STHR:	Space-Time Hierarchical Radiosity[16, 14], Section 4.2
MFLM:	Multi-Frame Lighting Method[6], Section 4.3
ISI:	Image Space Interpolation[36], Section 4.4
TFPT:	Temporally Filtered Particle Tracing[50], Section 4.5.1
SSVA:	Spatiotemporal Sensitivity and Visual Attention driven Global Illumination [83], Section 4.5.2
TPR:	Two Pass Radiosity[47], Section 4.6

References

1. D. R. Baum, J. R. Wallace, M. F. Cohen, and D. P. Greenberg. The Back-Buffer Algorithm: An Extension of the Radiosity Method to Dynamic Environments. *The Visual Computer*, 2(5):298–306, September 1986. 12

2. P. Bekaert and Y. Willems. Error Control for Radiosity. In *Proceedings of the 7th Eurographics Workshop on Rendering*, pages 153–164, 1996. 13

3. Philippe Bekaert, Laszlo Neumann, Attila Neumann, Mateu Sbert, and Yves D. Willems. Hierarchical monte carlo radiosity. In *Proceedings of the 9th Eurographics Workshop on Rendering*, pages 259–268, 1998. 14

4. L. D. Bergman, H. Fuchs, E. Grant, and S. Spach. Image rendering by adaptive refinement. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 29–37, 1986. 9

5. G. Besuievsky and M. Sbert. The Multi-Frame Lighting Method: A Monte Carlo Based Solution for Radiosity in Dynamic Environments. In *Proceedings of the 7th Eurographics Workshop on Rendering*, pages 185–194, 1996. 14

6. G. Besuievsky and X.Pueyo. Animating radiosity environments through the multi-frame lighting method. *Journal of Visualization and Computer Animation*, 12:93–106, 2001. 14, 20

7. G. Bishop, H. Fuchs, L. McMillan, and E. J. Scher Zagier. Frameless rendering: Double buffering considered harmful. In *Computer Graphics (ACM SIGGRAPH '94 Proceedings)*, volume 28, pages 175–176, 1994. 9

8. B. Cabral, M. Olano, and P. Nemec. Reflection space image based rendering. In *Computer Graphics (ACM SIGGRAPH '99 Proceedings)*, pages 165–170, 1999. 11

9. S.E. Chen. Incremental Radiosity: An Extension of Progressive Radiosity to an Interactive Image Synthesis System. In *Computer Graphics (SIGGRAPH 90 Conference Proceedings)*, pages 135–144, 1990. 3

10. P. H. Christensen, E. J. Stollnitz, D. H. Salesin, and T. D. DeRose. Global Illumination of Glossy Environments Using Wavelets and Importance. *ACM Transactions on Graphics*, 15(1):37–71, January 1996. 8

11. M. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A Progressive Refinement Approach to Fast Radiosity Image Generation. In *Computer Graphics (ACM SIGGRAPH '88 Proceedings)*, volume 22, pages 75–84, 1988. 3

12. P.J. Corriveau, A.A. Webster, A.M. Rohaly, and J.M. Libert. Video Quality Experts Group: the Quest for Valid Objective Methods. In *Proc. of SPIE Vol. 3959*, pages 129–139, 2000. 15

13. S. Daly. The Visible Differences Predictor: An algorithm for the assessment of image fidelity. In *Digital Image and Human Vision*, pages 179–206, 1993. 6, 15

14. C. Damez. *Simulation globale de l'éclairage pour des séquences animées prenant en compte la cohérence temporelle*. PhD thesis, U.J.F. Grenoble, France, 2001. 14, 20

15. C. Damez, N. Holzschuch, and F. Sillion. Space-time hierarchical radiosity with clustering and higher-order wavelets. In *Eurographics 2001 Short Presentations*, pages 35–42, 2001. 13

16. C. Damez and F. Sillion. Space-time hierarchical radiosity. In *Proceedings of the 10th Eurographics Workshop on Rendering*, pages 235–246, 1999. 12, 13, 20

17. P. J. Diefenbach. *Pipeline Rendering: Interaction and Realism through Hardware-Based Multi-Pass Rendering*. Ph.D. thesis, University of Pennsylvania, 1996. 11

18. P.J. Diefenbach and N.I. Badler. Multi-pass pipeline rendering: Realism for dynamic environments. In *1997 Symposium on Interactive 3D Graphics*, pages 59–70. ACM SIGGRAPH, 1997. 11

19. K. Dmitriev, S. Brabec, K. Myszkowski, and H-P. Seidel. Interactive global illumination using selective photon tracing. In *Proceedings of the 13th Eurographics Workshop on Rendering*, 2002. 5, 6, 20

20. G. Drettakis and F. Sillion. Interactive update of global illumination using a line-space hierarchy. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, pages 57–64, 1997. 3, 18

21. J.A. Ferwerda, S. Pattanaik, P. Shirley, and D.P. Greenberg. A model of visual masking for computer graphics. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, pages 143–152, 1997. 6, 15, 17

22. D.A. Forsyth, C. Yang, and K. Teo. Efficient radiosity in dynamic environments. In *Proceedings of the 5th Eurographics Workshop on Rendering*, pages 313–323, 1994. 3

23. D. W. George, F. Sillion, and D. P. Greenberg. Radiosity Redistribution for Dynamic Environments. *IEEE Computer Graphics and Applications*, 10(4):26–34, July 1990. 3

24. S. Gibson and R. J. Hubbold. Perceptually-driven radiosity. *Computer Graphics Forum*, 16(2):129–141, 1997. 14

25. C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modelling the Interaction of Light Between Diffuse Surfaces. In *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, volume 18, pages 212–222, 1984. 3

26. X. Granier and G. Drettakis. Incremental updates for rapid glossy global illumination. In *Computer Graphics Forum (Proceedings of Eurographics 2001)*, volume 20, pages 268–277, 2001. 8, 18, 20

27. X. Granier, G. Drettakis, and B. Walter. Fast global illumination including specular effects. In *Proceedings of the 11th Eurographics Workshop on Rendering*, pages 47–58, 2000. 8, 20

28. J. Haber, K. Myszkowski, H. Yamauchi, and H.-P. Seidel. Perceptually guided corrective splatting. *Computer Graphics Forum*, 20(3):142–152, 2001. 15, 17

29. E. Haines and J. Wallace. Shaft culling for efficient ray-traced radiosity. In *Eurographics Workshop on Rendering*, pages 122–138, 1991. 3

30. J.H. Halton. *On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals*. Numerische Mathematik, 2:84-90, 1960. 5

31. P. Hanrahan, D. Salzman, and L. Aupperle. A Rapid Hierarchical Radiosity Algorithm. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, 1991. 3

32. P. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, pages 145–154, August 1990. 15

33. W. Heidrich. Interactive Display of Global Illumination Solutions for Non-Diffuse Environments - A Survey. *Computer Graphics Forum*, 20(4):225–243, 2001. 2, 4, 11

34. W. Heidrich and H.-P. Seidel. Realistic, hardware-accelerated shading and lighting. In *Computer Graphics (ACM SIGGRAPH '99 Proceedings)*, pages 171–178, 1999. 11

35. L. Itti, C. Koch, and E. Niebur. A model of Saliency-Based Visual Attention for Rapid Scene Analysis. *Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998. 17

36. J. Nimeroff J., J. Dorsey, and H. Rushmeier. Implementation and analysis of an image-based global illumination framework for animated environments. *IEEE Transactions on Visualization and Computer Graphics*, 2(4):283–298, 1996. 14, 15, 20

37. H. W. Jensen. Global Illumination Using Photon Maps. In *Proceedings of the 7th Eurographics Workshop on Rendering*, pages 21–30, 1996. 8, 15, 18

38. J.T. Kajiya. The rendering equation. In *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, pages 143–150, 1986. 9

39. J. Kautz and M.D. McCool. Interactive rendering with arbitrary brdfs using separable approximations. In *Proceedings of the 10th Eurographics Workshop on Rendering*, pages 247–260, 1999. 11

40. J. Kautz, P.-P. Vázquez, W. Heidrich, and H.-P. Seidel. A unified approach to prefiltered environment maps. In *Proceedings of the 11th Eurographics Workshop on Rendering*, pages 185–196, 2000. 11

41. A. Keller. Quasi-monte carlo radiosity. In *Proceedings of the 7th Eurographics Workshop on Rendering*, pages 101–110, 1996. 4, 5

42. A. Keller. Instant radiosity. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, pages 49–56, 1997. 4, 5, 7

43. A. Keller and W. Heidrich. Interleaved sampling. In *Proceedings of the 12th Eurographics Workshop on Rendering*, pages 269–276, 2001. 7

44. D.H. Kelly. Motion and Vision 2. Stabilized spatio-temporal threshold surface. *Journal of the Optical Society of America*, 69(10):1340–1349, 1979. 15, 17

45. E. P. Lafortune and Y. D. Willems. Bi-directional Path Tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, 1993. 9

46. D. Lischinski, F. Tampieri, and D.P. Greenberg. Combining hierarchical radiosity and discontinuity meshing. In *Computer Graphics (ACM SIGGRAPH '93 Proceedings)*, pages 199–208, 1993. 18

47. I. Martín, X. Pueyo, and D. Tost. Frame-to-frame coherent animation with two-pass radiosity. Technical Report (To appear in *IEEE Transactions on Visualization and Computer Graphics*) 99-08-RR, GGG/IIIiA-UdG, Jun 1999. 18, 20

48. S. Mueller and F. Schoeffel. Fast radiosity repropagation for interactive virtual environments using a shadow-form-factor-list. In *Proceedings of the 5th Eurographics Workshop on Rendering*, pages 339–356, 1994. 3

49. K. Myszkowski, P. Rokita, and T. Tawara. Perceptually-informed accelerated rendering of high quality walk-through sequences. In *Proceedings of the 10th Eurographics Workshop on Rendering*, pages 5–18, 1999. 15, 17

50. K. Myszkowski, T. Tawara, H. Akamine, and H.-P. Seidel. Perception-guided global illumination solution for animation rendering. In *Computer Graphics Proceedings (ACM SIGGRAPH '01 Proceedings)*, pages 221–230, 2001. 15, 16, 20

51. H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Chapter 4, SIAM, Pennsylvania, 1992. 5

52. W. Osberger. *Perceptual Vision Models for Picture Quality Assessment and Compression Applications*. Ph.D. thesis, Queensland University of Technology, 1999. 15

53. F. Pellacini, J.A. Ferwerda, and D.P. Greenberg. Toward a psychophysically-based light reflection model for image synthesis. In *Computer Graphics (ACM SIGGRAPH '00 Proceedings)*, pages 55–64, 2000. 11

54. X. Pueyo, D. Tost, I. Martin, and B. Garcia. Radiosity for Dynamic Environments. *The Journal of Visualization and Comp. Animation*, 8(4):221–231, 1997. 12

55. R. Ramamoorthi and P. Hanrahan. An efficient representation for irradiance environment maps. In *Computer Graphics (ACM SIGGRAPH '01 Proceedings)*, pages 497–500, 2001. 11

56. M.C. Reichert. *A Two-Pass Radiosity Method to Transmitting and Specularly Reflecting Surfaces*. M.Sc. thesis, Cornell University, 1992. 18

57. M. Sbert, X. Pueyo, L. Neumann, and W. Purgathofer. Global Multipath Monte Carlo Algorithms for Radiosity. *The Visual Computer*, 12(2):47–61, 1996. 14

58. A. Scheel, M. Stamminger, and H.-P. Seidel. Thrifty final gather for radiosity. In *Proceedings of the 12th Eurographics Workshop on Rendering*, pages 1–12, 2001. 18

59. A. Scheel, M. Stamminger, and H.-P. Seidel. Grid based final gather for radiosity on complex clustered scenes. *Computer Graphics Forum*, 21(3), 2002. 18

60. F. Schoeffel and P. Pomi. Reducing Memory Requirements for Interactive Radiosity Using Movement Prediction. In *Proceedings of the 10th Eurographics Workshop on Rendering*, pages 225–234, 1999. 4, 18

61. E. Shaw. Hierarchical radiosity for dynamic environments. *Computer Graphics Forum*, 16(2):107–118, June 1997. 3

62. P. Shirley, B. Wade, P. M. Hubbard, D. Zareski, B. Walter, and D. P. Greenberg. Global Illumination via Density Estimation. In *Proceedings of the 6th Eurographics Workshop on Rendering*, pages 219–230, 1995. 15

63. F. Sillion. Clustering and Volume Scattering for Hierarchical Radiosity Calculations. In *Proceedings of the 5th Eurographics Workshop on Rendering*, pages 105–117, 1994. 3, 4

64. F. Sillion, J. R. Arvo, S. H. Westin, and D. P. Greenberg. A Global Illumination Solution for General Reflectance Distributions. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, volume 25, pages 187–196, 1991. 8

65. F. Sillion and J. Hasenfratz. Efficient parallel refinement for hierarchical radiosity on a DSM computer. In *Proceedings of the Third Eurographics Workshop on Parallel Graphics and Visualisation*, 2000. http://www-imagis.imag.fr/Membres/Jean-Marc.Hasenfratz/PUBLI/EGWPGV00.html. 13

66. P.-P. Sloan, J. Kautz, and J. John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Computer Graphics (ACM SIGGRAPH '02 Proceedings)*, 2002. 11

67. B. Smits, J. Arvo, and D. Greenberg. A Clustering Algorithm for Radiosity in Complex Environments. In *Computer Graphics (ACM SIGGRAPH '94 Proceedings)*, pages 435–442, 1994. 3, 4, 18

68. M. Stamminger, A. Scheel, X. Granier, F. Perez-Cazorla, G. Drettakis, and F. Sillion. Efficient glossy global illumination with interactive viewing. *Computer Graphics Forum*, 19(1):13–25, 2000. 8

69. P. Tole, F. Pellaccini, B. Walter, and D. P. Greenberg. Interactive global illumination in dynamic scenes. In *Computer Graphics (ACM SIGGRAPH '02 Proceedings)*, 2002. To be published. 10, 20

70. J. Tumblin and H.E. Rushmeier. Tone reproduction for realistic images. *IEEE Computer Graphics and Applications*, 13(6):42–48, November 1993. 12, 15

71. T. Udeshi and C.D. Hansen. Towards interactive photorealistic rendering of indoor scenes: A hybrid approach. In *Proceedings of the 10th Eurographics Workshop on Rendering*, pages 63–76, 1999. 4, 5

72. E. Veach and L. Guibas. Bidirectional Estimators for Light Transport. In *Proceedings of the 5th Eurographics Workshop on Rendering*, pages 147–162, 1994. 8, 9

73. E. Veach and L. J. Guibas. Metropolis light transport. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, pages 65–76, 1997. 8

74. V. Volevich, K. Myszkowski, A. Khodulev, and Kopylov E.A. Using the Visible Differences Predictor to improve performance of progressive global illumination computations. *ACM Transactions on Graphics*, 19(2):122–161, 2000. 5, 15, 16

75. I. Wald, T. Kollig, C. Benthin, A. Keller, and P. Slusallek. Interactive global illumination. In *Proceedings of the 13th Eurographics Workshop on Rendering*, 2002. 7, 20

76. I. Wald, P. Slusallek, and C. Benthin. Interactive Distributed Ray Tracing of Highly Complex Models. In *Proceedings of the 12th Eurographics Workshop on Rendering*, pages 277–288, 2001. 7, 11

77. I. Wald, P. Slusallek, C. Benthin, and M. Wagner. Inter-

active rendering with coherent ray tracing. *Computer Graphics Forum*, 20(3):153–164, 2001. 7, 11

78. B. Walter, G. Drettakis, and D.P. Greenberg. Enhancing and Optimizing the Render Cache. In *Proceedings of the 13th Eurographics Workshop on Rendering*, 2002. 10, 20

79. B. Walter, G. Drettakis, and S. Parker. Interactive rendering using the render cache. In *Proceedings of the 10th Eurographics Workshop on Rendering*, pages 235–246, 1999. 9, 20

80. B.J. Walter. *Density estimation techniques for global illumination*. Ph.D. thesis, Cornell University, 1998. 15

81. G.J. Ward. The RADIANCE lighting simulation and rendering system. In *Computer Graphics (ACM SIGGRAPH '94 Proceedings)*, pages 459–472, 1994. 17

82. A. Wojdala, M. Gruszewski, and K. Dudkiewicz. Using hardware texture mapping for efficient image synthesis and walkthrough with specular effects. *Machine Graphics and Vision*, 3(1–2):137–151, 1994. 11

83. H. Yee, S. Pattanaik, and D.P. Greenberg. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Transactions on Graphics*, 20(1):39–65, January 2001. 15, 17, 18, 20