EUROGRAPHICS 2005 / M. Alexa and J. Marks
(Guest Editors)

Volume 24 (2005), Number 3

Temporally Coherent Irradiance Caching for High Quality Animation Rendering

Miłosław Smyk[1,2], Shin-ichi Kinuwaki[3], Roman Ďurikovič[4], and Karol Myszkowski[1]

[1] MPI Informatik, Saarbrücken, Germany
[2] Szczecin University of Technology, Poland
[3] The University of Aizu, Fukushima, Japan
[4] University of Saint Cyril and Metod, Trnava, Slovakia

Abstract

In rendering of high quality animations that include global illumination, the final gathering and irradiance caching are commonly used. However, the computational cost they incur is high enough to discourage their wide use in production rendering. We introduce a data structure called anchor, which lets us permanently link cache locations to points intersected by their final gathering rays. Consequently, we can cheaply probe and transfer the (ir)radiance by exploiting the temporal coherence of successive animation frames, resulting in half an order of magnitude acceleration and reduced temporal artifacts. Additionally, our anchor structure lets us render moderately glossy surfaces at the cost much lower than the traditional importance sampling techniques. We also describe an efficient, perceptually motivated and independent scheme for limiting the growth in the number of irradiance caches. Finally, an implementation in a practical rendering system is demonstrated.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism I.3.6 [Computer Graphics]: Methodology and Techniques

1. Introduction

The rendering of high-quality CG movies featuring global illumination effects is considered very expensive. The main reason for the cost is the design of global illumination algorithms, which are optimized for static scenes only. In practice this means that when such algorithms are used for a dynamic scene, all computations should be repeated from scratch even when only minor changes take place.

An important issue in terms of frame quality is temporal aliasing in animations. Many typical errors in the lighting computation and the 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.

In recent years, several global illumination algorithms have been proposed that exploit temporal coherence between frames to improve the computation efficiency and reduce temporal aliasing (refer to [DDM03] for a recent survey on this topic). In this work we focus on the photon mapping algorithm proposed by Jensen [Jen96, Jen01], which is commonly used for the global illumination computation both in academia as well as in the industry [CLF*03, CB04]. The method works well even for complex environments and all possible light paths can easily be simulated. The computation is performed in two stages. In the first stage, photons are traced from light sources toward the scene and photon hit points are registered in a kd-tree structure, the so called photon map. The map is used in the second rendering stage for the lighting reconstruction using nearest neighbor density estimation [Hec90]. Usually the direct lighting and the specular effects are explicitly computed for each pixel, and the soft indirect lighting is obtained through the costly *final gathering* procedure [Rei92, Jen01], which relies on the integration of the incoming radiances over the hemisphere. Those costs can be reduced by using the *irradiance cache* data structures [WRC88, WH92] to store irradiance samples sparsely in the object space. The cached values are used to interpolate the indirect lighting for each pixel and are computed lazily.

In animation rendering the irradiance cache computation is a real computation bottleneck [CLF*03, CB04, TL04] that prevents a more widespread use of global illumination effects in a standard movie production pipeline. The main goal of this work is the improvement of the irradiance cache performance through exploiting temporal coherence between cache locations and their irradiance samples for the subsequent animation frames. This naturally leads to a significant reduction of popping artifacts as well. We introduce a lazily build data structure, which monitors local changes of illumination across the scene at the so-called anchor points, whose density adapts to changes in the lighting distribution. Since anchor points are directly linked to hemispherical samples (strata) for each irradiance cache, lighting updated at those points for each frame is immediately propagated to the corresponding strata (many strata belonging to different caches can be linked to the same anchor). A visibility map computed using graphics hardware is used to detect occlusions/disocclusions between anchors and a given cache location, which are caused by dynamic objects in the scene. All affected strata are identified in the visibility map and ray tracing is used to assign another nearest anchor point or possibly to create a new one. Thus, in our approach costly ray tracing is only used to update visibility changes caused by dynamic objects and it is not repeated for all strata and for each frame as in traditional approaches. Moreover, our technique is conservative in detecting scene illumination changes with a controllable accuracy, which is imposed by the density of anchor points and the resolution of visibility maps. Since irradiance values are known for each stratum in our algorithm, we can easily extend the irradiance cache algorithm to handle moderately glossy surfaces (for specular surfaces importance sampling is more efficient than stratified hemispherical integration) by reprojecting (warping) strata from neighboring cache locations to the corresponding strata at a given point in the scene.

The paper is organized as follows. In Section 2 we discuss previous work on final gathering and related interpolation schemes in spatial and temporal domains which improve the computation performance. In Section 3 we present our anchoring approach and we demonstrate its efficiency in the visibility computation and lighting reconstruction. We outline our animation rendering framework in Section 4 and we discuss the extensions of irradiance caching that are needed to take advantage of our anchor data structures. In Section 5 we present our approach to handle glossy surfaces using anchors. We demonstrate the results obtained using our techniques in Section 6 and we conclude this paper and discuss future work directions in Section 7.

2. Previous Work

Final gathering is the method of choice in the high quality rendering of indirect lighting for Lambertian surfaces, in which case BRDF-weighted importance sampling [LW95, Jen95] does not work and a full hemispherical integration of the incoming radiance is needed. Final gathering enables the reconstruction of fine lighting details for each image pixel even for a very coarse solution of the global illumination on the scene surfaces. Final gathering has originally been introduced for radiosity methods, which use the mesh for the light transport computation, and the mesh granularity limits the size of lighting details that can be directly rendered [Rei92, Smi94]. Final gathering is also commonly used in photon mapping because a direct rendering of the diffuse illumination based on the density estimation of photons leads to a poor image quality [Jen01].

In the following two sections we discuss final gathering solutions which exploit spatial lighting coherence for Lambertian and glossy surfaces. Then, we present final gathering extensions into the temporal domain.

2.1. Diffuse Surfaces

Since the lighting distribution is smooth over Lambertian surfaces, various interpolation schemes have been developed, which work in the object space and lead to high quality of lighting reconstruction [WRC88, BDT99]. Ward et al. proposed to cache the irradiance at adaptively chosen locations on the scene surfaces and then to interpolate the cached values to reconstruct the radiance for each pixel [WRC88]. The quality of interpolated lighting can be improved by using translational and rotational irradiance gradients to weight the contribution of neighboring caches [WH92]. Tabellion and Lamorlette [TL04] found that considering the projected pixel size in heuristics controlling the spatial layout of caches leads to a more favorable irradiance sampling density and a better image quality. Kato [Kat02] proposes an image-space driven approach for controlling the cache density, in which image features, such as object silhouettes, are taken into account. Kato explicitly stores irradiance values for each stratum and reprojects them to new cache locations, which leads to a significant reduction in the amount of traced final gather rays and improves the image quality with respect to a simple interpolation between the cache values.

Since the incoming radiance contributions can be significantly different as a function of direction in the hemispherical integration, the computation accuracy can be relaxed for less important directions. In the hierarchical radiosity such directions can easily be determined through link data structures that connect all surface pairs in the scene. Scheel et al. [SSS02] and Granier and Drettakis [GD04] use a less accurate mesh-based interpolation for slowly changing lighting components and explicit sampling of the most contributing emitters. Link data structures, inherent in the hierarchical radiosity, are used to speedup the visibility and the form factor computation for those emitters. Arikan et al. [AFO05] split the incoming radiance integral into two separate components for distant and nearby emitters. In the former case, smooth and slow changes of the incident radiance are expected, therefore the number of directional samples and the density of cache locations (they use spherical harmonics to store and interpolate incoming radiance) can be relaxed. The nearby surfaces must be sampled more densely to approximate the rapid changes of the radiance. To reduce the sampling cost, ray tracing towards the nearby surfaces is not performed and full visibility of those surfaces is assumed, which may lead to image artifacts.

In this work, we address the cache density control and efficient schemes of hemispherical integration for dynamic scenes.

2.2. Glossy Surfaces

Final gathering can easily be used for surfaces with arbitrary scattering functions. Here we narrow our discussion to those approaches that aim towards relaxing the requirement of pixel-by-pixel computation for glossy surfaces. Christensen et al. [CLSS97] mention the possibility of reusing the indirect lighting results over several pixels with a low variation, but they do not provide any specific algorithm to select such pixels and do not report any results concerning the efficiency of such an approach. Mostefaoui et al. [MDG99] build a hemispherical representation of the incoming radiance along with a lists of clusters (patches) that may cause a high variation of lighting within the currently rendered mesh element. To compute the shading of a pixel, the incoming radiance stored at the hemispheres is bilinearly interpolated while the contributions of all clusters (patches) from the list are explicitly sampled using stochastically traced rays. Mostefaoui et al. demonstrate the effectiveness of their algorithm for the rendering of surfaces with Bidirectional Texture Functions (BTFs).

Recently, a successful attempt to extend the irradiance cache technique to handle moderately glossy surfaces has been proposed [GKPB04] by Gautron et al. In their approach incoming radiance values at each cache location are projected to the spherical harmonic basis functions. To reconstruct the outgoing radiance at a given point, spherical harmonic coefficients for neighboring cache locations are interpolated (as in [AFO05]) and the reconstructed irradiance field is convolved with the surface BRDF.

In this work we propose a solution that extends the irradiance cache algorithm for glossy surfaces in dynamic scenes.

2.3. Exploiting Temporal Coherence

For animated sequences it is desirable to keep possibly the same cache locations for subsequent frames and to update cached values so that they are consistent with dynamic changes in the scene.

Martin et al. [MPT03] use the final gathering algorithm to compute texture movies that are dynamically assigned to visible surfaces. The resolution of movie textures is adaptively chosen so that at least one texel stores an irradiance value for each pixel in the final frame. This may lead to popping artifacts when the camera or textured objects are moving. Martin et al. found that usually around ten times more texels than pixels are sufficient to obtain high quality frames, which leads to a huge number of final gathering steps. The temporal coherence in the radiosity solution is obtained using the line-space hierarchy [DS97, GD04], which is designed to efficiently identify links that are affected by changes in the scene and to redistribute the lighting energy for the current scene configuration. In the final gathering step, the expensive visibility computation is performed once for a given time interval for links that are classified as static, but is repeated for dynamic links in each frame. The method is limited to the Lambertian surfaces.

Tawara et al. [TMS04] extended the irradiance cache technique into the temporal domain. They stored incoming radiance samples for all strata and stochastically updated a fixed percentage of those samples using a simple aging criterion. Changes in the scene have not been taken into account in the sample updating process, which potentially could lead to the use of invalid samples. A fixed percentage of updated samples was chosen manually and has not been dynamically correlated to the changes in the scene. Each sample update was performed by shooting a ray as in the traditional final gathering technique.

In this work, we detect all visibility changes for a given cache location and selectively shoot rays only for those strata which are affected by those changes (due to direct occlusions/disocclusions). We also take into account changes in illumination that are not accompanied by visibility changes, in which case ray shooting is not required to update incoming radiance samples. In the following section we describe our approach to achieve these goals.

3. Anchor Data Structure

Selective update of irradiance cache strata requires an explicit storage of the incoming radiance samples for each stratum. Such a storage is feasible even for complex scenes and high resolution frames as it was recently demonstrated [TMS04, Kat02]. A problem is to decide how to monitor local illumination changes in the scene and how to identify strata that they affect.

We address this problem within the framework of photon mapping by introducing a new data structure which we call the *anchor points*. Anchor points are lazily created at intersection points of final gathering rays with scene surfaces. Explicit links for the energy transfer (similar as in [BF89]) between the anchor points and the corresponding strata are created (see Figure 1). Inserting a new anchor point is avoided if suitable (see Section 3.1) existing anchor points in the local neighborhood can be found (we discuss the problem of anchor density control in Section 3.3). As a result of such an anchor inserting procedure, a number of strata from different caches can be linked to a given anchor point. To speed up the search of existing anchor points, an *anchor kd*-tree is built. Each node of the tree refers indirectly to the anchor point by a pointer to the following record in the *anchor list*:

```
struct Anchor{
 float3 pos;       \\ Anchor position
 uchar theta, phi;\\ Object normal at anchor
 RGBE   Li;        \\ Anchor irradiance value
 Material* mat;    \\ Anchor material
};
```

When calculating the radiance of a diffuse surface, the local albedo is multiplied by irradiance during the final gathering stage. Since the anchor is directly accessed without space traversing (refer Section 4.2), the material information at the anchor position should be stored in anchor's data structure.

The anchor list is maintained through subsequent animation frames. New anchor points are inserted when the visibility between scene surfaces is affected by moving objects. Also, the camera animation can lead to the exposition of previously invisible scene fragments that must be covered by irradiance caches, which in turn may require the insertion of new anchor points. Existing anchor points are removed from the list as well when they are not linked to any cache location (in Section 4.3 we discuss our approach to the cache insertion and removal). However, a vast majority of anchor points can be maintained for the subsequent animation frames and only anchor irradiance values `Li` must be updated. For this purpose photon tracing is performed for each frame and `Li` values are computed for all anchor points using the standard nearest neighbor procedure [Jen01]. All changed `Li` are immediately propagated to strata that are linked with a given cache. We use quasi monte carlo (QMC) sequences for temporally coherent photon tracing and strata sampling [Chr02].

Non-random insertion of nodes into the *anchor kd*-tree during final gathering results in the tree becoming unbalanced. In our implementation over 80% of all anchors are placed deeper than the optimal depth of a balanced tree and the maximum tree depth is three times larger than the optimal one. While a slightly skewed tree does not influence the performance very much, we measure the average number of nodes visited during the anchor insertion/search and if it reaches a certain threshold, we rebalance the entire *anchor kd*-tree before proceeding with the next animation frame.

In the following section we elaborate on heuristics used to decide on anchor's suitability for the reusage as a target location of a particular stratum. Then, we discuss an important problem of handling visibility changes between strata and corresponding anchor points due to moving objects. Finally, we present our approach to controlling the density of anchor points.

3.1. Anchor Reuse Heuristics

Before a stratum is linked to a preexisting anchor, it is checked against a set of criteria that are chosen to exclude anchors that probably do not correctly represent the irradiance at the exact point of the final gather ray intersection. Involved heuristics include a normal vector check equivalent to the one used in photon mapping to qualify a photon for the irradiance estimation [Jen01], a surface albedo check to eliminate the influence of high-frequency textures, and finally a visibility check (performed by sending a shadow ray), which guards against using anchors that could be located behind a wall, e.g., in the neighboring room. The anchor's verification tests are conducted in the cheapest to the most expensive order. If three consecutive anchors do not pass the tests, a new anchor is created at the exact final gather ray intersection point and used instead.

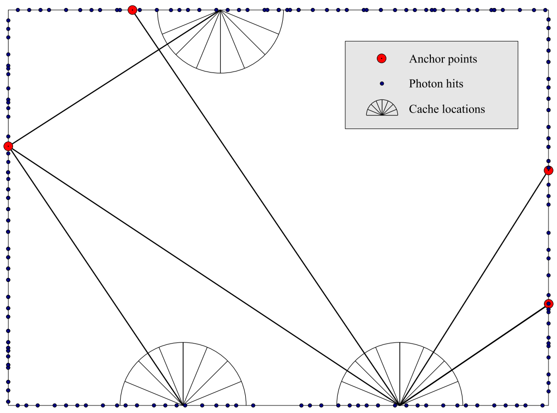

Figure 1: *Cache and anchor data structures: each cache stratum is linked to an anchor point in the scene. One anchor point can be linked by many caches. Neighboring strata of a cache are usually linked to different anchor points, possibly with the exception of scene corners which are handled separately.*

3.2. Handling Changes in Visibility

Our anchor data structure, presented in the Section 3, requires an update when links between strata and anchor points are broken due to the occlusion by moving objects. For the same reason new links must be established when moving objects expose some scene fragments that were invisible in the previous frame from a given cache position (the disocclusion problem). A common solution to this problem is to use shafts [HW91] that represent the set of line segments joining two surfaces. A simple test for the intersections of shafts by the bounding box of a dynamic object allows to rapidly identify changes in the visibility [DS97, GD04]. Since in our application the links originating at a given cache position are diverging over the full hemisphere, many shafts would be required.

Currently, the film industry is evaluating the use of graphics hardware to support accelerated rendering, as it reduces the cost of traditional effects and widens the application area of computationally intensive techniques like global illumination [nVI05]. While the visibility determination is a general problem and temporally coherent irradiance caching can use any of the preexisting methods to tackle it, we explore a hardware-assisted route to demonstrate its benefits.

Graphics hardware can be employed for very efficient visibility testing. Two steps are involved in the visibility test: the z-buffer test on dynamic objects only, and the mapping of occluded pixels to the corresponding strata.

In the first step, all dynamic objects in the current frame are projected from a given cache position to a hemisphere using a vertex shader program. This step is repeated for all caches in the scene. Note, that when a dynamic object is hidden behind a static one, the occlusion will not be detected. However, this case does not cause any particular problem because all rays shot from strata, marked as dynamic strata, are tested for the object type they hit, either dynamic or static. In this framework we can speed up the visibility detection because all dynamic objects are loaded into an *OpenGL display list* or a *vertex buffer object* only once. Loaded dynamic objects — either rigid or soft — can then be used for all cache projections within the currently processed frame.

In the second step, we assume that the visibility sphere map is defined over the discretized space. Based on our experiments, the adequate resolution of a sphere map $n \times n$ is approximately $n = 2 \times N_s$, where N_s is the total number of final gather strata. The visibility sphere maps are transferred from the video memory to the CPU memory space and then each stratum element's center is projected back to the sphere map coordinates. The sphere coordinates are converted into a 2D strata array, which corresponds to the direction of the final gather rays in the cache.

Finally, we detect the strata with dynamic object(s) over them by checking the color labels at the sphere map pixels. For every such stratum we need to retrace a new ray in two consecutive frames: in the current one to detect a possible intersection with the dynamic object, and in the following one, to properly handle the case where an object has moved away, exposing static geometry behind it. To achieve this, for each stratum we maintain a boolean `flag` field which is set to TRUE whenever the visibility test for this direction determined a possible overlap with a dynamic object. Later, this field is checked when the irradiance of the cache holding the strata is computed and if it is found to be TRUE, a final gather ray is shot instead of just using a link to stratum's anchor. The ray can either hit a dynamic object and return irradiance computed in a traditional way, or hit a static geometry, at which point we repeat the anchor attachment procedure for the stratum and set its `flag` field to FALSE.

3.3. Anchor Density Control

The question arises how to control the density of anchor points? For example, should we apply a higher density at bright regions of the scene, or should they be denser in regions that are often accessed by final gather rays, or at places with rapid illumination changes? When a final gather ray intersects an object at location P we search for the nearest anchor within the $D_{\max}$ radius of P and if it is unsuccessful, we insert a new one exactly at P, thus putting an upper limit on the distances between anchors. At the same time, we have to make sure that anchors are not placed too densely, as it may negatively impact the performance without increasing the quality. $D_{\min}$ is the minimum valid distance between anchors. In the following sections we discuss our strategies for the $D_{\max}$ and $D_{\min}$ selection.

3.3.1. Maximum Anchor Distance $D_{\max}$

We propose two approaches for estimating radius $D_{\max}$.

In the first method, anchors are distributed with a constant radius $D_{\max}$, specified by the user. This is a simple and effective method to control the number of anchors in the scene. Our measurements indicate that even for a fast moving camera or quickly changing strong indirect illumination, the method constantly manages to maintain a low error and the lowest computation time (see Figure 7).

In the second approach to determine anchor density, we use "global" importons, as described by Suykens [Suy02]. They are emitted from the camera location towards the screen and stored on the second diffuse surface they encounter on their path. As a result, they tend to amass in places which will be frequent targets of final gather rays or, to put it in other words, in places which will have a higher than average contribution to the reconstructed irradiance. It thus makes sense to lower the search radius $D_{\max}$ in such areas (and hence increase the density of anchors) by making it inversely proportional to the local importon density estimate.

We also experimented with an anchor placement strategy that considers the local irradiance value for the $D_{\max}$ estimation. Unfortunately, according to our study, such an approach does not produce noticeable improvement of the final image quality in spite of increased computation time.

Figure 2 demonstrates the actual placement of anchors in the SPONZA scene (refer to Figure 6) with the constant radius and importance driven placement.

Finally, it is necessary to observe that any non-random modifications to the final gather ray hit locations introduce bias. While in our testing we did not encounter significant bias-related artifacts, some other $D_{\max}$ determination strategies may require scaling the weights of irradiances coming along different final gather rays based on solid angles they represent. One solution to this problem has been described in [SP01].

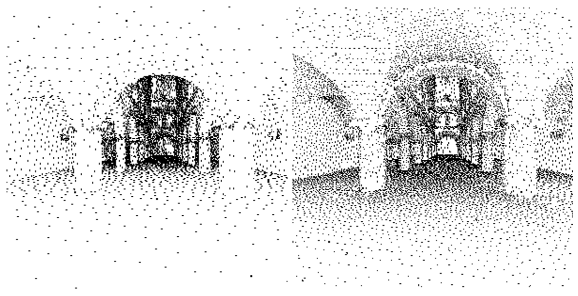

Figure 2: *Anchor density control with constant radius (left) and importance driven (right) anchor placement. The same number of anchors is distributed for the two density control methods. Dots represent the actual positions of anchors.*

3.3.2. Minimum Anchor Distance $D_{\min}$

Enforcing a lower limit for the anchor density becomes important in places where the number of final gather ray hits has a potential to exceed the number of stored photons. One such situation, where this is especially evident, occurs when surfaces are located close next to each other, e.g., in the corners. The photon density there is usually lower than the average, while many caches are present and thus many gathering rays are intersecting small areas.

As a result, both density control methods proposed above place many anchors close to the corners. However, as the anchor density approaches the photon density, the value of information gained by adding further anchors rapidly decreases. Finally, when the densities equalize, there is nothing more to learn about region's irradiance and hence there is no point in having more than one anchor per photon (this is also why the precomputed irradiance works [Chr99]). We thus decide to select $D_{\min}$ on the fly and tie it with a local photon density. We achieve this by letting the photons themselves control the placement of anchors (see Figure 3).

Assume that stratum S is responsible for the shooting of a final gather ray that intersects a static geometry at point P. The procedure of locating a suitable anchor, to link S with, starts with the search for a photon ξ closest to P within a user-settable radius. If ξ is found and it already contains a link to an anchor, we reuse the anchor and link the stratum S to it.

However, if ξ is not found or does not contain a link to any anchor, we revert to behavior described in Section 3.3.1. The anchor resulting from this operation is then linked from photon ξ if it was found earlier.

Since the anchors are reused in subsequent frames and the photons are always reshot from scratch, the approach described above requires one additional step to work correctly: after the photons for the next frame are shot, for each anchor a search is made for the nearest photon and a link is made from this photon to the anchor. Without it, the anchor density could be underestimated in some areas, leading to wasted computational effort.

Pseudocode 1 summarizes the use of $D_{\max}$ and $D_{\min}$ by our method.

Pseudocode 1 AttachToAnchor(stratum)

```
Find the nearest photon $\xi$ from the final gather hit point P.
if Photon $\xi$ has a link to anchor then
  Link stratum to the photon's anchor.
else
  if Nearest anchor within $D_{\max}$ neighborhood of P was found. then
    Link stratum to this anchor.
  else
    Create new anchor.
    Link stratum and photon $\xi$ to the newly created anchor.
  end if
end if
```

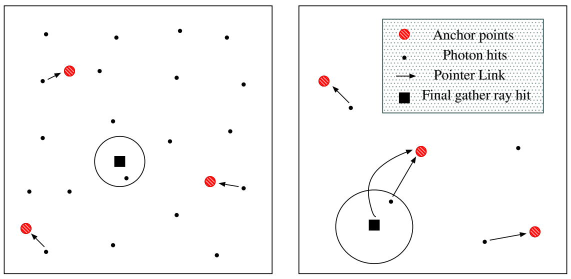

Figure 3: *If the photon that is nearest to the requested location has a pointer to an anchor, it is likely that the density of photons and anchors are similar in this scene region. Then, we do not perform a search but rather link photon's anchor directly. The density of photons is much higher than that of anchors (left). The densities of photons and anchors are similar (right).*

4. Rendering Algorithm

The complete temporally coherent rendering framework based on irradiance caching must not only support the reuse of information obtained during the final gathering, but also efficiently manage auxiliary data structures. In this section, we demonstrate extensions to the irradiance caching, which enable it to work hand in hand with our anchoring technique, presented in Section 3. We also describe a perceptually motivated scheme for avoiding the unlimited growth of the irradiance cache nodes, which can be used independently from other parts of our framework.

4.1. Cache Data Structure

For each incoming direction at the cache location (stratum) we store a pointer to an anchor structure that contains (among other things) the incoming radiance. The distance to the nearest intersection point is preserved to enable the reevaluation of the harmonic mean distance [WRC88] or to find the closest distance to the intersected surfaces [TL04] in the upper hemispherical directions, which is required to compute well-distributed cache locations. An indication of whether the sample hits a moving object is stored in the `flag` field. The same field is also used to store the information whether the sample hits any geometry in the scene. This information is employed to adaptively estimate the number of rays to be updated. The data structure for strata (`RadianceSample`) is designed as follows:

```
struct RadianceSample{
 Anchor* pA;     \\ The nearest anchor
 float16 t;      \\ Distance to the nearest
                 \\ intersection point
 uchar   flags;  \\ The flag of hitting
                 \\ a dynamic object
                 \\ and the flag of hitting
                 \\ any geometry combined
};
```

4.2. Cache Update for the Animation

Since irradiance caching distributes caches in scene space, it is tempting to reuse available information in successive animation frames. Let us first consider a cache located on a static part of the scene geometry, which contributes to one or more pixels and thus should be updated. Final gather rays shot from such cache can be categorized into two groups: the group that intersects a static geometry and the group that intersects a dynamic object. In the former group, the cache strata hit static geometry at the same intersection points for two consecutive frames (see Section 3.2) and thus space traversal is not needed as we only have to read the incoming radiances from the updated anchor points. In the latter group, the intersection points may change and thus the rays need to be retraced.

If the cache that is updated is located on the dynamic object we do not try to reuse any information from previous frames. Therefore, all old dynamic caches are discarded and new ones are distributed over the dynamic object in their place.

The algorithm implementing these ideas divides the rendering into two phases. In the first phase the photons are shot using Quasi Monte Carlo (QMC) sequences to determine their paths and surface events. One of benefits of using QMC is the ability to easily recreate these photon paths where the geometry remains unchanged.

Then, as the rendering progresses and successive caches are queried for the irradiance at their location, they actually reconstruct it with the help from the anchor tree. For each cache stratum an anchor nearest to the intersection points is located as described in Section 3 and its irradiance is multiplied by the surface albedo. The resulting value is used as the irradiance arriving at the strata. If no suitable anchor is found (see Section 3.1), a new one is created and inserted into the anchor tree, with the irradiance value estimated at its location.

Pseudocode 2 summarizes the steps taken by our method to render a frame of animation.

Pseudocode 2 Animate(frame)

```
Shoot photons
if first frame then
  for each static cache do
    for each stratum do
      Shoot this final gather ray
      AttachToAnchor(stratum)
      Read $Li$ from anchor
    end for
  end for
else
  for all anchors do
    Recompute irradiance at anchor location
    Relink nearest photons
  end for
  for each static cache do
    Detect dynamic objects in the scene, Section 3.2
    for each stratum do
      if dynamic object was hit in current or previous frame then
        Reshoot this final gather ray
        AttachToAnchor(stratum)
      end if
      Read $Li$ from anchor
    end for
  end for
end if
```

4.3. Cache Density Control

As caches are added due to viewpoint changes and the object movement, their number will continuously grow, gradually decreasing the caching scheme's efficiency in assisting the irradiance reconstruction. We therefore need a method that will delete the caches that are no longer necessary or useful. Caches that can be removed fall into three categories that need to be treated separately:

- Caches that are no longer visible due to camera location changes.
- Caches on dynamic objects.
- Caches that were added when a dynamic object neared, but are no longer needed because the object is gone, and the ones caused by viewpoint change/zoom (in the cache allocation scheme proposed by [TL04]).

The first category can be handled by keeping a use marker C_{frame} in each cache. When rendering, all caches that contributed to a particular pixel are marked with the current frame number. After the frame is completed, we remove all caches which are older than a user-defined age. Note, that it is not enough to simply discard the caches that are not visible directly, as off-screen caches can also contribute to the image.

The caches that fall into the second category are simply discarded.

Finally, the last category consists of caches whose influence radius will grow when the distance to nearest objects increases. Because the cache density remains constant, pixels in the surrounding area have more and more caches contributing to them. Thus, we keep a cache use counter in each pixel (i.e., the number of caches used to extrapolate the irradiance at this pixel), as well as a list of caches and their respective contribution weights. After the frame is complete, we first find pixels that have more than N caches contributing to them (we empirically derived six to be a good value for N).

Then, for each such cache we measure the total effect of its potential disappearance: in each pixel it affected we remove its contribution and renormalize the contributions of the remaining caches, calculate the Weber's fraction by dividing the result by the original pixel value (Equation 1).

$$\delta_i = \sum_{pixels \in i}^{j} \frac{|P_{j,i} - \overline{P}_{j,i}|}{P_{j,i}} \tag{1}$$

where δ_i is the total influence of the cache i on the image, $P_{j,i}$ is the value of pixel j including the influence of cache i, and $\overline{P}_{j,i}$ is the value of pixel j excluding it.

In the end, we remove the caches with δ below some predefined threshold, i.e., the ones having the lowest influence on the entire image. The example results of this process are demonstrated in Figure 4.

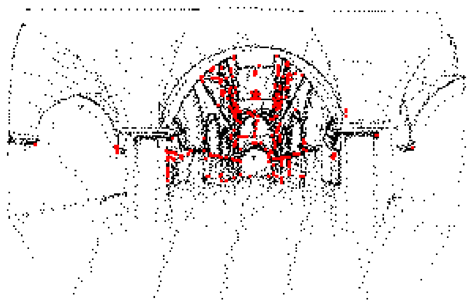

Figure 4: *The visualization of our cache removal scheme. Larger red dots indicate the 5% of caches with the lowest influence on the visible reconstructed irradiance.*

5. Rendering Moderately Glossy Surfaces

At this point we will expand the anchor based method to handle moderately glossy surfaces. Moderately glossy surfaces have a surface bi-directional reflectance function (BRDF) close to the Lambertian BRDF. The approach proposed by Gautron [GKPB04] employed hemispherical harmonics (HSH) to interpolate both the BRDF and first few low frequency coefficients of incoming radiance. However, HSH based approaches and cache based interpolation approaches may have a problem with higher frequencies as pointed out in [GKPB04].

To overcome the above weakness and to evaluate incoming irradiance rapidly and accurately, we propose an extension of our anchor scheme, which enables more anchors to contribute to a single stratum (*anchor bucketing*).

5.1. Anchor Bucketing

For all pixels representing moderately glossy surfaces we must evaluate the outgoing radiance using the final gathering (rather than to interpolate it as for Lambertian surfaces) because the pixel radiance values can be very different.

Let us consider a final gather sample location C (the ray intersection point) in object space and nearby sample cache points A and B. We can predict the final gathering result at C by hemispherical projection of all anchor points visible from A and B onto a sample C (refer to Figure 5), similarly to [Kat02].

The hemisphere at C is also discretized into strata, but some anchor points can be occluded by an object during the projection to a hemisphere stratum. In general, such a case can be resolved by choosing the anchor point nearest of all anchors projected to a given strata, similarly to how the Z-buffer algorithm works. Unfortunately, each stratum would hold only a single irradiance value while we would clearly prefer more than one in order to increase the accuracy of the incoming radiance evaluation.

Therefore, we introduce *a bucket of anchors* that will be projected to a single stratum. The *bucket of anchors* contains all anchors above the stratum whose distance to hemisphere center C is less than D_{bucket}. All anchors in the bucket are linked to the stratum and used for the evaluation of incoming radiance.

For fast calculation of D_{bucket} we employ the GPU hemisphere mapping of all scene objects and store the depth values. The hemisphere map is then transformed to the 2D strata array. Finally, distance D_{bucket} for a given stratum is computed by increasing its depth value by a small offset.

6. Results

We demonstrate the implementation of the proposed algorithms in our custom rendering system by using it to compute two animation segments — SPONZA and SIBENIK

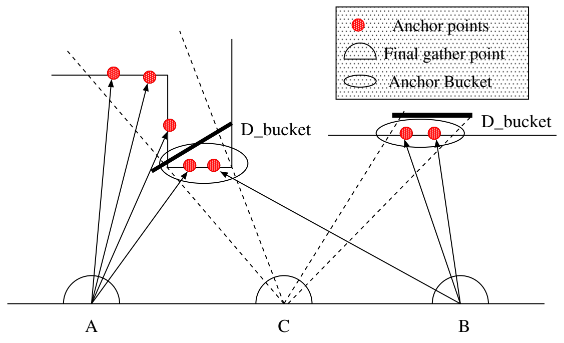

Figure 5: *Final gathering at points A, and B. The radiance value is interpolated at C by reprojecting the anchors created from cache locations A and B.*

(see Figure 6). The scenes contain respectively 60,000 and 70,000 polygons and are illuminated by large amounts of diffuse light. In both cases, in addition to camera movement, we also animate several objects which, due to their strong diffuse reflectance, noticeably influence nearby surfaces.

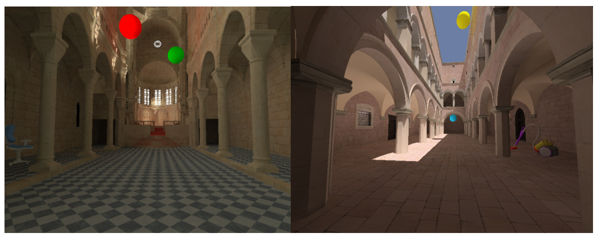

Figure 6: *Example frames shaded with full global illumination and using constant anchor distribution:* SIBENIK *(left) and* SPONZA *(right).*

Figure 7 summarizes the error measurements for the three methods: anchor placement with constant density, anchor placement driven by importance, and precomputed irradiance [Chr99]. In order to provide a fair comparison, we arranged the number of anchors to be the same as the number of precomputed irradiance points and also used the same cache distribution for all methods. The error was calculated relative to the reference sequence computed with brute-force k-nearest neighbors method for all final gathering rays, again with the same cache distribution as in the compared methods.

Relatively high RMS error present in the images rendered with precomputed irradiance demonstrates that for spatially large scenes with a lot of detail even 10^6 photons may not be enough. As only $1/4$ of them is actually used to hold irradiance estimates, they are too sparsely distributed to be effectively sampled by final gather rays. On the other hand, anchor search always ends up with a valid link, noticeably improving the overall quality of irradiance reconstruction.

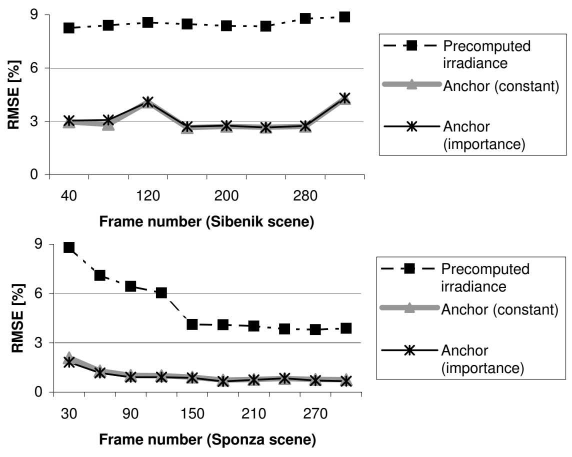

Figure 7: *Error measurements of radiance values for* SIBENIK *and* SPONZA *scene. RMS error in [%].*

Also, observe that the type of anchor density control strategy has a negligible impact on the measured error.

What this graph does not show is the inter-frame temporal coherence achieved by our technique, and manifested by a significantly lower number of popping artifacts and reduced image flickering (as demonstrated by video materials from [**?**]).

Statistics regarding the reuse of anchors appear in Figure 8. Reuse of irradiance estimates stored at photon locations with precomputed irradiance is also shown for comparison purposes. Observe that it has higher reuse count than anchor-based methods only at the extreme ends of the chart, where either almost no reuse happens (1–10) or the reuse is excessive due to many final gather rays in the corners getting their (redundant) irradiance estimate from the same photon. In other cases we are getting better reuse with anchors, especially when they are distributed according to importance (Section 3.3.1). Note, that this is an important feature when recomputing anchor irradiance lazily, as in such scenario large numbers of frequently reused anchors directly translate to performance gains.

Figure 9 shows how rendering of moderately glossy surface done with our technique compares to importance sampling. 128 and 196 final gathering rays are used respectively.

6.1. Timings

Average timings for frames from both sequences are shown in Table 1. These and further measurements were performed on an AthlonXP 1900+ PC machine running Windows 2000 and equipped with an nVidia GeForce 6800 graphics card. Frames were rendered in 800×600 resolution using 1,800 rays in final gathering. The average number of anchors used is 270,000 for SPONZA and 310,000 for SIBENIK.

Table 2: *Statistics concerning the GPU-based visibility algorithm.*

# of polygons	Size	GPU [%]	GPU → CPU [%]	Conversion [%]	Total relative time
Subdivided box (300)	$SA_{8\times32}$, $VSM_{32\times32}$	40.78	55.30	3.91	1.0
	$SA_{20\times80}$, $VSM_{64\times64}$	29.84	56.63	14.63	1.391
Chair (4000)	$SA_{8\times32}$, $VSM_{32\times32}$	47.77	51.17	1.10	3.417
	$SA_{20\times80}$, $VSM_{64\times64}$	43.75	50.81	5.42	3.827

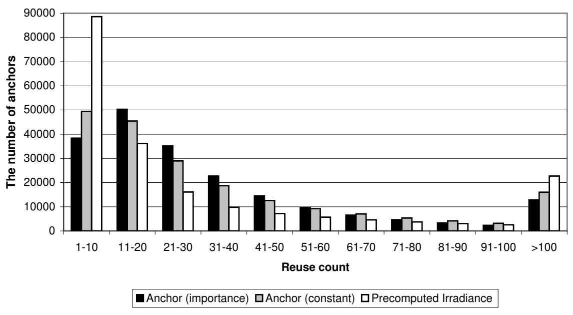

Figure 8: *Breakdown of anchor reuse frequencies.*

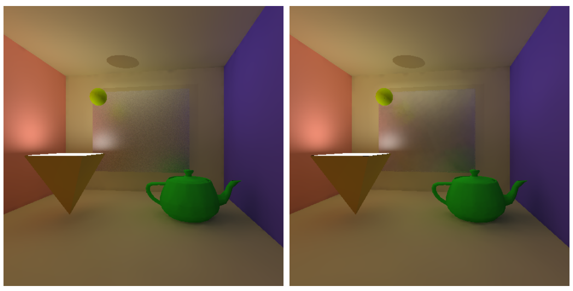

Figure 9: *Rendering of moderately glossy surfaces: importance sampling — 254s (left) and our method — 108s (right).*

Table 1: *Single frame timings breakdown for the* SIBENIK *and* SPONZA *scenes.*

	Brute force [s] SIBENIK/SPONZA	Proposed algorithm [s] SIBENIK/SPONZA
Photon tracing	34 / 28	34 / 28
Irradiance est.	23 / 21	6 / 6
Final gathering	1040 / 901	122 / 115
Cache interpol.	33 / 37	36 / 41
Total	1130 / 987	198 / 186

The average cache count is around 12,000 for SPONZA and 15,000 for SIBENIK. The photon count is 0.82 M and 1.07 M, respectively. The number of rays shot is 21 millions for SPONZA and 27 millions for SIBENIK. Only 1%–1.5% of total rays are reshot for both scenes in the animation framework. Each stratum and anchor point requires 7 and 22 bytes of storage, respectively. The memory use by cache strata (151 MB and 189 MB) overwhelms the cost of storing anchors (5.9 MB and 6.8 MB for SPONZA and SIBENIK, respectively).

As evidenced by Table 1, we achieve total speedup by a factor of 5.7 for SIBENIK and 5.3 for SPONZA. Not surprisingly, the largest reduction in computational effort happens in final gathering (8.5×), but there are also savings during irradiance estimation due to anchor reuse. In fact, final gathering performance is increased so much that typically less critical stages become significant factors in the total result. Moreover, irradiance cache interpolation is slower than in the classical approach because of additional computations required for cache removal (Section 4.3).

Table 2 summarizes the statistics of the GPU visibility algorithm. The first column shows the name and number of polygons used in the test (i.e., the moving objects in SIBENIK scene), the second column shows the size of strata array *SA* in CPU and visibility sphere map *VSM* on GPU, the third column shows the GPU time, the fourth column shows the time needed for loading GPU map to CPU, the next column shows the time needed for the conversion of *VSM* to *SA*, and the last column shows the relative time, where time of the first test is scaled to 1.0 (which roughly corresponds to one second for 10,000 caches).

7. Conclusions

In this paper we have presented a novel technique for high quality animation rendering. By introducing and effectively exploiting the concept of anchors — a data structure that links irradiance caches to final gather ray hit points — we achieved a dual goal of enabling inter-frame temporal coherence, while simultaneously obtaining a significant performance boost. We also demonstrated how anchors can be employed to provide efficient rendering of moderately glossy surfaces.

Although we proposed the method for final gathering on glossy surfaces using anchor bucketing, the problem of efficient querying the incoming radiance from surrounding glossy surfaces remains unsolved. Essentially, the anchor points should be placed on diffuse surfaces only. Brute force photon density estimation is needed in final gathering for each intersection point that involves a surface with an arbitrary BRDF. Also, a better scheme for the cache density control over glossy surfaces is required to improve the computation efficiency.

As future work we intend to explore a multi-resolution strata representation, which would considerably accelerate the hemispherical reprojection used for glossy surfaces.

Acknowledgments

We would like to thank Robert Herzog, Takehiro Tawara, and Katsuhiro Numata for fruitful discussions and comments on our work. We are grateful to Jun Yamadera from Eyes Japan co. for providing us with 3D models and to Marko Dabrovic from RNA Studio for making the SPONZA ATRIUM and SIBENIK CATHEDRAL scenes publicly available. We would also like to thank Grzegorz Krawczyk, Rafał Mantiuk and Krzysztof Kobus for providing mentally stimulating work environment. This research was partially sponsored by the following grants: VEGA 1/0174/03 from the Ministry of Education, Slovakia and UCM Trnava, Slovakia.

References

[AFO05] ARIKAN O., FORSYTH D., O'BRIEN J.: Fast and Detailed Approximate Global Illumination by Irradiance Decomposition. *ACM Transactions on Graphics 24*, 3 (2005).

[BDT99] BALA K., DORSEY J., TELLER S.: Radiance Interpolants for Accelerated Bounded-Error Ray Tracing. *ACM Transactions on Graphics 18*, 3 (1999), 213–256.

[BF89] BUCKALEW C., FUSSELL D.: Illumination Networks: Fast Realistic Rendering with General Reflectance Functions. In *Computer Graphics (Proceedings of SIGGRAPH 89)* (July 1989), vol. 23, pp. 89–98.

[CB04] CHRISTENSEN P., BATALI D.: An Irradiance Atlas for Global Illumination in Complex Production Scenes. In *Rendering Techniques 2004* (June 2004), pp. 133–141.

[Chr99] CHRISTENSEN P. H.: Faster Photon Map Global Illumination. *Journal of Graphics Tools 4*, 3 (1999), 1–10.

[Chr02] CHRISTENSEN P.: Photon Mapping Tricks. In *SIGGRAPH 2002, Course Notes No. 43,A Practical Guide to Global Illumination Using Photon Mapping* (2002), pp. 93–121.

[CLF*03] CHRISTENSEN P., LAUR D., FONG J., WOOTEN W., BATALI D.: Ray Differentials and Multiresolution Geometry Caching for Distribution Ray Tracing in Complex Scenes. In *Eurographics* (2003), pp. 543–552.

[CLSS97] CHRISTENSEN P., LISCHINSKI D., STOLLNITZ E., SALESIN D.: Clustering for glossy global illumination. *ACM Transactions on Graphics 16*, 1 (1997), 3–33.

[DDM03] DAMEZ C., DMITRIEV K., MYSZKOWSKI K.: State of the Art in Global Illumination for Interactive Applications and High-quality Animations. *Computer Graphics Forum 22*, 1 (2003), 55–77.

[DS97] DRETTAKIS G., SILLION F.: Interactive Update of Global Illumination Using A Line-Space Hierarchy. In *Proceedings of SIGGRAPH 97* (1997), Computer Graphics Proceedings, Annual Conference Series, pp. 57–64.

[GD04] GRANIER X., DRETTAKIS G.: A Final Reconstruction Framework for a Unified Global Illumination Algorithm. *ACM Transactions on Graphics 23*, 2 (2004), 163–189.

[GKPB04] GAUTRON P., KRIVANEK J., PATTANAIK S., BOUATOUCH K.: A Novel Hemispherical Basis for Accurate and Efficient Rendering. In *Rendering Techniques 2004* (June 2004), pp. 321–330.

[Hec90] HECKBERT P.: Adaptive Radiosity Textures for Bidirectional Ray Tracing. In *Computer Graphics (ACM SIGGRAPH '90 Proceedings)* (August 1990), pp. 145–154.

[HW91] HAINES E., WALLACE J.: Shaft culling for efficient ray-traced radiosity. In *Second Eurographics Workshop on Rendering (EGWR)* (1991), pp. 122–138.

[Jen95] JENSEN H.: Importance Driven Path Tracing using the Photon Map. In *Rendering Techniques '95 (Proceedings of the 6th Eurographics Workshop on Rendering)* (1995), Springer-Verlag, pp. 326–335.

[Jen96] JENSEN H.: Global Illumination Using Photon Maps. In *Rendering Techniques '96 (Proceedings of the 7th Eurographics Workshop on Rendering)* (1996), Springer-Verlag, pp. 21–30.

[Jen01] JENSEN H.: *Realistic Image Synthesis Using Photon Mapping*. AK, Peters, 2001.

[Kat02] KATO T.: Photon Mapping in Kilauea. In *SIGGRAPH 2002, Course Notes No. 43, A Practical Guide to Global Illumination Using Photon Mapping* (2002), pp. 122–191.

[LW95] LAFORTUNE E., WILLEMS Y.: A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing. In *Eurographics Rendering Workshop 1995* (1995), pp. 11–20.

[MDG99] MOSTEFAOUI L., DISCHLER J.-M., GHAZANFARPOUR D.: Rendering Inhomogeneous Surfaces with Radiosity. In *Eurographics Rendering Workshop 1999* (1999), pp. 283–292.

[MPT03] MARTIN I., PUEYO X., TOST D.: Frame-to-frame Coherent Animation with Two-pass Radiosity. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (2003), 70–84.

[nVI05] NVIDIA: Gelato: GPU-Accelerated Film Renderer. In *http://www.nvidia.co.uk/page/gelato.html* (2005).

[Rei92] REICHERT M.: *A Two-Pass Radiosity Method Driven by Lights and Viewers Position*. M.Sc. thesis, Cornell University, 1992.

[Smi94] SMITS B.: *Efficient Hierarchical Radiosity in Complex Environments*. Ph.D. thesis, Cornell University, 1994.

[SP01] SERPAGGI X., PÉROCHE B.: An adaptive method for indirect illumination using light vectors. In *EG 2001 Proceedings*, Chalmers A., Rhyne T.-M., (Eds.), vol. 20(3). Blackwell Publishing, 2001, pp. 278–287.

[SSS02] SCHEEL A., STAMMINGER M., SEIDEL H.-P.: Grid based Final Gather for Radiosity on Complex Clustered Scenes. *Computer Graphics Forum 21*, 3 (2002), 547–556.

[Suy02] SUYKENS F.: *On Robust Monte Carlo Algorithms for Multi-pass Global Illumination*. Ph.D. thesis, Dept. Computer Science, Faculty of Engineering, K.U.Leuven, 2002.

[TL04] TABELLION E., LAMORLETTE A.: An Approximate Global Illumination System for Computer-Generated Films. *ACM Transactions on Graphics 23*, 3 (2004), 469–476.

[TMS04] TAWARA T., MYSZKOWSKI K., SEIDEL H.-P.: Exploiting Temporal Coherence in Final Gathering for Dynamic Scenes. In *Computer Graphics International (CGI 2004)* (2004), IEEE Computer Society, pp. 110–119.

[WH92] WARD G., HECKBERT P.: Irradiance Gradients. In *Proceedings of the 3rd Eurographics Workshop on Rendering* (1992), pp. 85–98.

[WRC88] WARD G., RUBINSTEIN F., CLEAR R.: A Ray Tracing Solution for Diffuse Interreflection. In *Computer Graphics (ACM SIGGRAPH '88 Proceedings)* (1988), pp. 85–92.