I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N° 1796

TEMPORAL RADIANCE CACHING

PASCAL GAUTRON , KADI BOUATOUCH , SUMANTA PATTANAIK

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Temporal Radiance Caching

Pascal Gautron [*], Kadi Bouatouch [**], Sumanta Pattanaik [***]

Systèmes cognitifs
Projets SIAMES

Publication interne n1796 — Avril 2006 — 27 pages

Abstract: We present a novel method for fast, high quality computation of glossy global illumination in complex animated environments. Building on the irradiance caching and radiance caching algorithms, our method leverages temporal coherence by introducing temporal gradients. Using our approach, part of the global illumination solution computed in previous frames is adaptively reused in the current frame. Our simple adaptive reusing scheme allows to obtain fast rendering times while avoiding the presence of flickering artifacts and global illumination ghosts. By reusing data in several frames, our method yields a significant speedup compared to classical computation in which a new cache is computed for every frame. Moreover, temporal gradients do not rely on any new, complicated data structure. This method can be straightforwardly included in any existing renderer based on irradiance and radiance caching. Furthermore, our method can be easily implemented using GPUs for improved performance.

Key-words: irradiance and radiance caching, global illumination, dynamic environments

(Résumé : tsvp)

[*] pgautron@irisa.fr
[**] kadi@irisa.fr
[***] sumant@cs.ucf.edu

Centre National de la Recherche Scientifique
(UMR 6074) Université de Rennes 1 – Insa de Rennes

Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Cache de Luminance Temporel

Résumé : Nous présentons une nouvelle méthode pour le calcul d'illumination globale rapide et de haute qualité dans des environnements animés. En nous basant sur les algorithmes de cache d'éclairement et de luminance, notre méthode exploite la cohérence temporelle par le biais de gradients temporels. Dans notre approche, une partie de l'éclairage calculé dans les images précédentes est réutilisée de manière adaptative pour le rendu de l'image courante. Cette méthode de réutilisation permet de réduire fortement les temps de calcul ainsi que les clignotements et autres artéfacts souvents constatés dans les autres méthodes. De plus, notre méthode n'introduit aucune structure de données complexe. Par conséquent, les gradients temporels peuvent être très simplement intégrés dans un logiciel de rendu basé sur l'algorithme de cache d'éclairement ou de luminance. Dans cette publication, nous intégrons les gradients temporels dans un logiciel de rendu basé sur le cache de luminance et exploitant la puissance des cartes graphiques.

Mots clés : cache d'éclairement et de luminance, illumination globale, environnements dynamiques

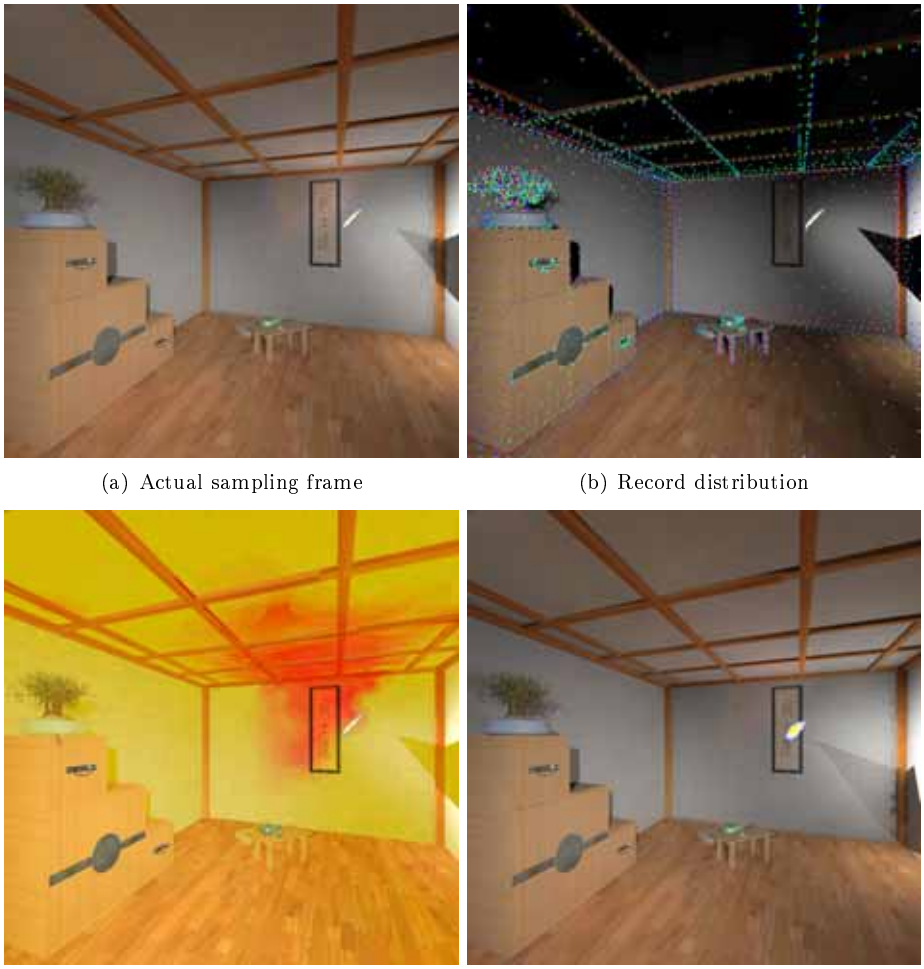
1 Introduction

Efficient computation of global illumination in complex animated environments is one of the most important research challenges in computer graphics. Achievements in this field have many applications, such as visual effects for computer-assisted movies and video games. Many approaches for such computation have been proposed during the last 20 years. Most of these methods are based on radiosity, such as [BWCG86, PTMG97, BP01, DDM02]. However, radiosity methods are prone to visual artifacts due to undersampling, or high memory consumption and computational cost due to high-quality sampling. Therefore, other approaches such as an extension of photon mapping [DBMS02] and irradiance caching [TMS04, SKDM05] have been proposed.

We propose a simple and accurate method based on temporal caching for efficiently computing global illumination effects in complex animated environments. Our method allows rapid generation of image sequences for environments in which viewer, objects and light sources move.

Our approach focuses on a temporal optimization for lighting computation based on irradiance caching [WRC88] and radiance caching [KGPB05] techniques. These algorithms leverage spatial coherence of indirect lighting to drastically reduce the computational cost of global illumination. They are based on sparse sampling and interpolation of the indirect lighting. However, when used to render globally illuminated dynamic scenes, those algorithms are prone to disturbing flickering artifacts. In this paper, we show that the irradiance interpolation scheme described in [WRC88] can be easily extended to the temporal domain. We define a temporal weighting function for temporal interpolation. As in [WRC88], this weighting function is based on an estimate of the change of incoming radiance. However, estimating the change of incoming radiance in the course of time requires a basic knowledge of the incoming radiance at future time steps. Therefore, we propose a inexpensive and accurate method based on reprojection to compute this information. Classical irradiance and radiance caching rely on gradients for smooth and accurate spatial interpolation of incoming radiance [WH92, KGPB05, KGBP05]. In spite of the smooth appearance of the static frames, animation sequences rendered using irradiance and radiance caching often suffer from flickering artifacts. Therefore, we propose *temporal gradients* for flicker-free rendering. These gradients can be computed either by extrapolation for on-the-fly computation, or by temporal interpolation for high quality rendering. The computation method for temporal gradients can be user-selected depending on the desired rendering quality. Unlike many previous approaches, our method does not introduce any new data structures and adds very little overhead to the existing data requirements. Since a same record in the cache can be used in several frames, both the computational cost and the memory required to store the records are significantly reduced. This allows to keep the records used to render an animation segment within a small memory space. Once records are computed in the scene, our GPU-based renderer can display the dynamic globally illuminated scene in real-time.

This paper is organized as follows: Section 2 presents some significant previous works in the field of global illumination for animations in dynamic scenes. Section 3 describes briefly the irradiance and radiance caching algorithms. Section 4.1 highlights the problems



(a) Actual sampling frame (b) Record distribution

(c) Records lifespan (d) Future frame

Figure 1: Our temporal caching method allows the reuse of irradiance and radiance records in multiple frames. When computing the global illumination solution for the current frame (a), our method estimates where the lighting changes using simple observations. For each record generated in current frame (b), our temporal weighting function estimates the future change of lighting and defines the lifespan and temporal gradients of each record (c), green and red colors respectively representing long and short lifespan. Since a record is reused in several frames, the computation cost of the global illumination solution for next frames is drastically reduced.

related to using those algorithms for animation rendering and describes our solution to these problems. Section 4.2 presents the derivation of our temporal weighting function. Since this function is based on an estimate of the change of incoming radiance in the course of time, Section 4.3 describes a inexpensive and accurate method for computing this information. A smooth temporal interpolation using temporal gradients is proposed in Section 4.4. Section 5 describes how our method can be efficiently implemented using graphics hardware. Finally, we present results obtained using our method in Section 6.

2 Previous Work

This section describes the most significant approaches to the computation of high quality global illumination in dynamic scenes. We also describe several methods aiming at computing approximate solutions at interactive frame rates. A thorough description of many existing methods can be found in [DDM02].

2.1 Interactive Methods

The methods described in this paragraph focus on interactive approximate rendering in which the quality may be compromised to enhance interactivity.

The Render Cache [WDP99, WDG02] is based on the high coherence of the points visible between two subsequent frames: most of the points visible in frame n are also visible in frame $n+1$. Therefore, the render cache stores the points visible in frame n, and reprojects them onto frame $n+1$. As the visibility may change between two successive frames, some pixels may not have any corresponding visible point stored in the cache. In that case, those pixels are either filled by computing the actual corresponding hit point, or by hole-filling. Even though this method provides interactive frame rates, the artifacts due to missing information makes the Render Cache usable only in the context of fast approximate rendering. Note that we use a similar reprojection technique to determine the lifespan of irradiance and radiance records.

Another approach based on interactive ray tracing is the Frameless Rendering [BFMZ94]. This method is based on parallel, asynchronous ray tracing. Each processor is assigned a fixed set of pixels to render continuously. The computation of each pixel updates the resulting image immediately. The computational power of several processors working in parallel allows interactive frame rates. However, the resulting image is likely to contain artifacts: a given pixel may have the same value during several frames, yielding spatially incoherent shading.

In the Shading Cache method [TPWG02], the irradiance at the vertices of a tesselated scene is adaptively computed and cached using a parallel computing cluster. The scene is rendered and displayed at interactive frame rates using hardware-based interpolation. The vertices within the region affected by object and camera movement are localized using an image-space sampling scheme and updated. The rendering shows ghosting artifacts in regions affected by moving objects; for example, color bleeding due to an object may remain

behind even though the object has moved away. These artifacts eventually disappear if the viewer and objects remain static during several seconds.

A similar drawback is also present in [DBMS02]. This method is based on photon mapping [Jen01] and quasi Monte Carlo. The photon map is updated when the user moves an object. For each frame, the algorithm detects which photon paths should be recomputed. The update consists in using quasi Monte Carlo to trace a group of photons following a similar path. This method is well-suited for interactive manipulation of objects, but suffers from delays between the object motion and the update of the photon map.

2.2 Radiosity

Radiosity [GTGB84] is a very popular method for computing global illumination. Consequently, many attempts have been made to optimize the computation of high quality radiosity solutions in animated environments.

A decomposition of the computation into *static radiosity* and *dynamic radiosity* images is proposed in [PTMG97]. The former represents the light transfers between static patches, while the latter accounts for light transfers involving dynamic objects. This approach shows significant speedups compared to frame-by-frame computation. However, it is limited to a fixed viewpoint, reducing its usability.

The approaches described in Damez et al. [DSH01, Dam01] are an extension of hierarchical radiosity to dynamic environments. A scene is discretized both in space and time, and the radiosity solution is stored using wavelets. In order to reduce the artifacts due to temporal discretization, a piecewise linear wavelet basis is used to represent temporal changes in radiosity. Although these methods show significant reduction of temporal discontinuities, it suffers from meshing artifacts and high memory consumption.

Besuievsky et al. [BS96, BP01] present the *Multi-Frame* lighting method. In a first pass, dynamic objects are replicated for each frame, yielding a single scene representation for the whole animation. In a second pass, rays are traced so that the visibility tests are performed only once for static objects, and once per frame for dynamic objects. This allows to reduce the overall cost of ray-object intersection. However, this method requires to store the radiosity for each mesh element and each frame. Since this method does not allow adaptive meshing, high quality can be obtained only by using dense meshing, implying prohibitive memory requirements.

The approach described in [MPT03] leverages temporal coherence by computing a hierarchy of texture movies for visible surfaces. These movies are generated by computing final gathering at least once for each pixel of the final image. The line-space hierarchy [DS97, GD04] is used to identify and recompute links that are affected by the motion of objects in the scene. This method requires a large amount of memory to store the texture movies. Moreover, popping artifacts may appear when switching the resolution levels of the hierarchical texture movies.

As any finite element method, radiosity methods inherently rely on meshing and subdivisions. Therefore, these methods are prone to meshing artifacts (due to the discontinuities of the lighting) and to high memory consumption due to the very fine subdivision required to

obtain high quality results. This is why meshless methods such as photon mapping [Jen01] and irradiance caching[WRC88] have also been investigated in the context of animation rendering.

2.3 Irradiance Caching and Final Gathering

Several approaches have been proposed to accelerate the final gathering process used to render photon maps in dynamic scenes. The methods described in this section exploit temporal coherence in the context of final gathering using irradiance caching [WRC88].

Tawara *et al.* [TMS02] propose a two step method for computing indirect lighting in dynamic scenes. In the first step, they compute an irradiance cache using only the static objects of the scene. Then, for each frame, they update the irradiance cache by introducing the dynamic objects at their frame specific positions. This method shows significant speedup, but is restricted to animations during which "lighting conditions do not change significantly". Otherwise, the static irradiance cache has to be recomputed from scratch, leading to the temporal artifacts inherent in this method.

Tawara *et al.* [TMS04] propose to detect the incoming radiance changes related to the displacement of objects. The rays traced for computing irradiance records are stored and updated using either a user-defined update rate, or an adaptive rate based on the number of rays hitting a dynamic object. This method requires an high amount of memory to store the rays, making it difficult to use in complex scenes where many records have to be computed.

The method described in [SKDM05] is based on a data structure called *anchor*. This structure allows to detect changes in visibility and incoming radiance for each irradiance record during an animation sequence. Therefore, the irradiance values stored in the cached records can be efficiently updated. However, even though this method provides high-quality results, it requires the explicit storage of the rays used to compute each irradiance record, and hence is highly memory intensive.

In this paper, we present a fast, simple and memory efficient method exploiting temporal coherence for irradiance and radiance caching [WRC88, KGPB05] for illuminating dynamic scenes. The following section presents a brief overview of irradiance and radiance caching.

3 Irradiance and Radiance Caching: an Overview

The irradiance and radiance caching algorithms are based on the following observation: "the indirect illuminance tends to change slowly over a surface" [WRC88]. These methods take advantage of spatial coherence by sparsely sampling and interpolating indirect incoming radiance. Irradiance caching [WRC88] is designed for the computation of indirect diffuse lighting. Radiance caching [KGPB05] extends the approach proposed by Ward *et al.* to glossy interreflections. The incoming radiance function at a record location is represented using hemispherical harmonics [GKPB04] and interpolated at the neighboring locations. The outgoing radiance values of visible points is computed by multiplying this interpolated radiance function by the reflectance function of the visible surfaces.

The irradiance and radiance caching algorithms are very similar. In irradiance caching, the record stores the irradiance at a given point on a surface. In radiance caching, the record stores the coefficients of projection of the incoming radiance into hemispherical harmonics basis. The interpolation schemes are similar: irradiance stored in the irradiance cache records and coefficients stored in the radiance cache records undergo weighted interpolation. The method described in this paper is similarly applicable to both caching techniques. For the simplicity of explanation we only consider irradiance in most of the following discussions. We show the results of application of our method to both irradiance and radiance caching in Section 6.

Irradiance Interpolation Let us consider a point $\mathbf{p}$ in a scene. In [WRC88], an estimate of the irradiance $E(\mathbf{p})$ at point $\mathbf{p}$ is given by:

$$E(\mathbf{p}) = \frac{\sum_{k \in S_r} w_k(\mathbf{p}) E_k(\mathbf{p})}{\sum_{k \in S_r} w_k(\mathbf{p})} \tag{1}$$

In Eq. 1, S_r represents the set of irradiance records surrounding $\mathbf{p}$. For each record $k \in S_r$, $w_k(\mathbf{p})$ is the weighting function of record k evaluated at point $\mathbf{p}$. $E_k(\mathbf{p})$ is the contribution of record k to the computed incoming lighting at point $\mathbf{p}$. In the next two paragraphs, we first detail the irradiance weighting function before presenting how the contributions of the records are computed.

Weighting function The irradiance caching algorithm is based on the observation that the indirect diffuse lighting changes slowly over a surface. A given record contributes to the indirect lighting of points with similar lighting. If the incoming lighting at the record location changes rapidly in the neighborhood of the record, its weight should be small in the surrounding area. Conversely, if the lighting changes slowly, the weight of the record should be high. Therefore, the weighting function is based on an approximation of the change in incoming radiance with displacement and rotation.

The weighting function is chosen as the inverse of this estimated change. At a point $\mathbf{p}$ with normal $\mathbf{n}$, the weighting function of record k is defined as:

$$w_k(\mathbf{p}) = \frac{1}{\frac{\|\mathbf{p}-\mathbf{p_k}\|}{R_k} + \sqrt{1 - \mathbf{n} \cdot \mathbf{n_k}}} \tag{2}$$

where $\mathbf{p_k}$, $\mathbf{n_k}$ and R_k are respectively the location of record k, its normal and the harmonic mean distance to the objects visible from $\mathbf{p_k}$.

This weighting function is not only used for interpolation, but also to determine the set of records $S_r(\mathbf{p})$ contributing to the incoming radiance at point $\mathbf{p}$:

$$S_r(\mathbf{p}) = \{k | w_k(\mathbf{p}) \geq a\} \tag{3}$$

where a is a user-defined accuracy value.

Irradiance Gradients for Accurate Extrapolation The irradiance contribution of record k to the irradiance at point $\mathbf{p}$ depends on the translational and rotational gradients of the irradiance. The gradient computations proposed in [WH92, KGPB05, KGBP05] present several methods to estimate the change in incoming radiance with respect to translation and rotation. Figure 1a and 1b of [WH92] illustrate the quality improvement obtained using gradients. Ward and Heckbert [WH92] define the contribution of record k to point $\mathbf{p}$ with normal $\mathbf{n}$ as:

$$E_k(\mathbf{p}) = E_k + (\mathbf{n_k} \times \mathbf{n}) \cdot \nabla_\mathbf{r} + (\mathbf{p} - \mathbf{p_k}) \cdot \nabla_\mathbf{p} \tag{4}$$

where E_k, $\nabla_\mathbf{r}$ and $\nabla_\mathbf{p}$ are respectively the irradiance, the rotation gradient and the translation gradient stored in record k.

The irradiance and radiance caching algorithms provide an accurate and efficient way of computing global illumination in static scenes. However, problems arise when using this approach to global illumination computation in dynamic environments. Next section presents the typical problems encountered while using irradiance caching in dynamic scenes, and describes our temporal optimization method.

4 Temporal Irradiance Caching

4.1 Irradiance Caching in Dynamic Scenes

As described in previous section, irradiance caching leverages spatial coherency of indirect lighting, and thus reduces the computation time of global illumination. In dynamic scenes, due to the presence of dynamic objects, a straightforward and commonly used method consists in computing the global illumination solution from scratch for every frame. Thus, the distributions of record location in frames n and $n+1$ are likely to be different. Figure 2 illustrates the consequence of the change of record distribution. Since the gradients extrapolate the change of incoming radiance, they are not 100% accurate compared to the ground truth. Therefore, the dependability (also named accuracy) of the lighting reconstructed by irradiance caching is not constant over a surface. The accuracy of the extrapolation is maximum at the record location, and decreases as the extrapolation point gets away from the record. Therefore, changing the distribution of records also changes the distribution of the accuracy, yielding flickering artifacts. Thus, simple use of irradiance caching in dynamic scenes gives rise to poor animation quality. Additionally, as the record computation is performed from scratch for every frame, significant amount of computational effort is wasted.

In this paper, we propose a simple and unified framework for view-dependent global illumination in dynamic environments with predefined animation in which objects, light sources, and cameras can move. We propose a temporal weighting function based on an estimate of the change of incoming radiance with respect to time. This weighting function, described in section 4.2, allows to decide whether a record contributes to the indirect lighting of points visible in a given frame.

Classical irradiance caching uses irradiance gradients for smooth, high quality spatial extrapolation of the incoming radiance around irradiance records. While the weighting

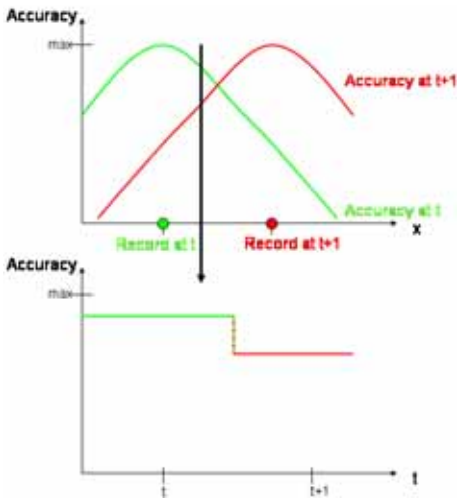


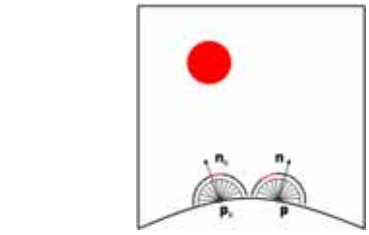
Figure 2: The location of the records has an impact on the spatial accuracy of the lighting reconstructed by irradiance caching compared to a reference solution. Since irradiance caching is based on sparse sampling and extrapolation of incoming radiance, the accuracy of the solution at a given point (marked by the black arrow) is not constant over a surface. When records are located at different points in successive frames, the accuracy at a given point can change abruptly. Therefore, the incoming radiance can change noticeably between two frames, yielding flickering artifacts. The maximum accuracy is obtained at the point and time of actual computation of the incoming radiance.

function roughly describes the *amount* of change of the incoming radiance, the gradients are computed by estimating more precisely *how* the incoming radiance changes with respect to displacement and normal divergence. As for our approach, it accounts for the temporal change of incoming radiance over time by making use of temporal gradients, based on a simple estimate of the change of incoming radiance in the course of time. The combination of our temporal weighting function with temporal irradiance gradients allows to simply extend the irradiance caching interpolation scheme to the temporal domain.

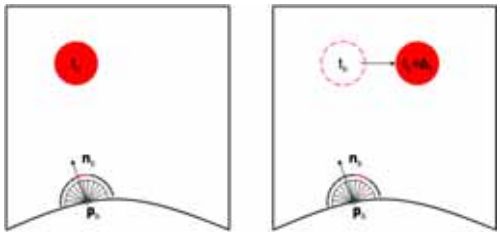
4.2 Temporal Weighting Function

In this section, we derive a formula expressing the validity of a given record over time. Using a derivation similar to that of [WRC88], we define the temporal change ϵ^t of incoming radiance between time t and t_0 as:

$$\epsilon^t = \frac{\partial E}{\partial t}(t_0) \quad (t - t_0) \tag{5}$$



(a) Spatial change



(b) Temporal change

Figure 3: Change of incoming radiance with respect to space and time. The black-red zone above the hemisphere represents the incoming radiance function. Classical irradiance caching (a) estimate the change of incoming radiance with respect to displacement and normal divergence. In the case of temporal irradiance caching (b), the incoming radiance can change between two frames even though the record location remains constant.

This derivative $\frac{\partial E}{\partial t}(t_0)$ can be approximated using estimates of incoming radiance at two successive times t_0 and t_1, denoted E_0 and E_1.

$$\frac{\partial E}{\partial t}(t_0) \approx \frac{E_1 - E_0}{t_1 - t_0} \tag{6}$$

$$= \frac{\tau E_0 - E_0}{t_1 - t_0} \quad \text{where } \tau = E_1/E_0 \tag{7}$$

$$= E_0 \frac{\tau - 1}{t_1 - t_0} \tag{8}$$

In our method, the time range of the animation is discretized into integer frame indices. Therefore, we always choose $t_1 - t_0 = 1$, i.e. E_1 and E_0 represent the estimated irradiance at two successive frames.

As in [WRC88], we define the temporal weighting function as the inverse of the change, excluding the term E_0:

$$w_k^t(t) = \frac{1}{(\tau - 1)(t - t_0)} \tag{9}$$

where $\tau = E_1/E_0$ is the *temporal irradiance change rate*.

This weighting function expresses the confidence of the incoming radiance value estimated at time t_0 in subsequent frames. This function can be evaluated and tested against a user-defined accuracy value a^t. A record k created at t_0 is allowed to contribute to the image at t if

$$w_k^t(t) \geq 1/a^t \tag{10}$$

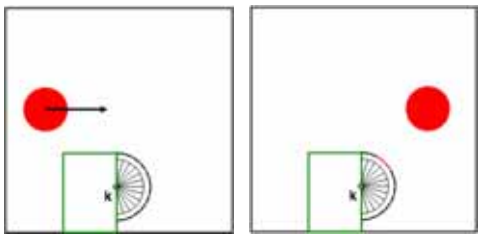
The temporal weighting function allows to adjust the time segment during which a record is considered as valid. Since a given record can be reused in several frames, the computational cost can be significantly reduced. However, Eq. 7 shows that if the environment remains static starting from frame t_0, we obtain $\tau = 1$. Therefore, w_k^t becomes infinite:

$$w_k^t(t) = \infty \quad \forall t \qquad \text{if } \tau = 1 \tag{11}$$

In this case, record k is allowed to contribute at any time $t > t_0$. However, since part of the environment is dynamic, the inaccuracy becomes significant when $t - t_0$ gets high (see Figure 4). Therefore, we introduce a user-defined value δt_{max} limiting the length of the validity time segment associated with a given record. If Equation 12 does not hold, the record cannot be used.

$$t - t_0 < \delta_{t_{max}} \tag{12}$$

This reduces the risk of having records from being used while they are obsolete, hence controlling the apparition of global illumination ghosts.



(a) Time t_k (b) Time $t_k + n$

Figure 4: When record k is created at time t_k, the moving red sphere is invisible. Consequently, $\tau_k = 1$. However, the red sphere becomes visible from the location of record k n frames later. The user-defined value $\delta_{t_{max}}$ prevents the record from contributing if $n > \delta_{t_{max}}$, guaranteeing the use of up-to-date records only.

However, the flickering problem described in Section 4.1 still appears when a record is suddenly discarded. As proposed in [TMS02], we avoid this problem by keeping track of the location of the records over time. Let us consider a record k located at point $\mathbf{p_k}$. If k was allowed to contribute to previous frame and cannot be reused in current frame, a new record l is created at the same location, i.e. $\mathbf{p_l} = \mathbf{p_k}$ (Figure 5(b)). Therefore, the location

of visible records remains constant in space. As a consequence, the distribution of accuracy has less temporal variations, which reduces flickering artifacts. Note that the location of records remain constant even though they lie on dynamic objects (Figure 6).

The temporal weighting function provides a simple and adaptive way of leveraging temporal coherence by introducing an aging method based on the change of incoming radiance. Our approach allows to reuse records in several frames, reducing both the computational cost and the flickering artifacts. However, as shown in Equation 7, the determination of our temporal weighting function relies on the knowledge of the incoming radiance at the next time step, E_{t_0+1}. In complex scenes, explicitly computing both E_{t_0} and E_{t_0+1} for each record would introduce a huge computational overhead. In the next section we propose a simple and accurate method for estimating E_{t_0+1}.

4.3 Estimating E_{t_0+1}

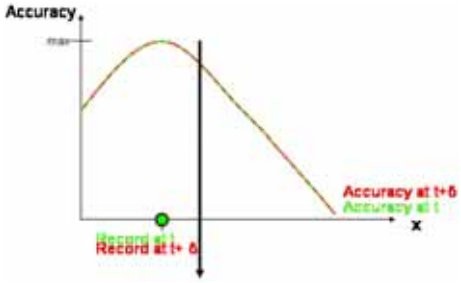
Our method follows the reprojection approach proposed in [WDP99]. However, while Walter *et al.* use reprojection for interactive visualization using ray tracing, our aim is to provide a simple and reliable estimate of the incident radiance at a given point at time $t_0 + 1$ by using the data acquired at time t_0 only.

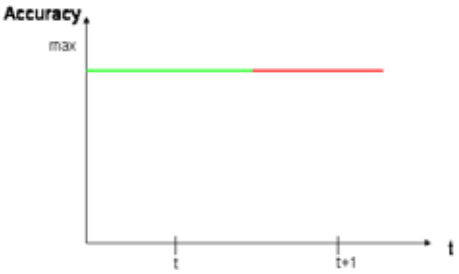
In the context of predefined animation, the changes in the scene are known and accessible at any time. When a record k is created at time t_0, the hemisphere above $\mathbf{p_k}$ is sampled (Figure 7(a)) to compute the incoming radiance and gradients at this point. Since the changes between times t_0 and $t_0 + 1$ are known, it is possible to reproject the points visible at time t_0 to obtain an estimate of the visible points at time t_0+1 (Figure 7(b)). The outgoing radiance of reprojected visible points can be estimated by accounting for the rotation and displacement of both objects and light sources. In overlapping areas, a depth test takes occlusion change into account (Figure 7(c)).

However, some parts of the estimated incoming radiance may be unknown (holes) due to displacement and overlapping of visible objects (Figure 7(d)). As proposed in [WDP99], we use a simple hole-filling method: a 3×3 filter fills the unknown areas with plausible values.

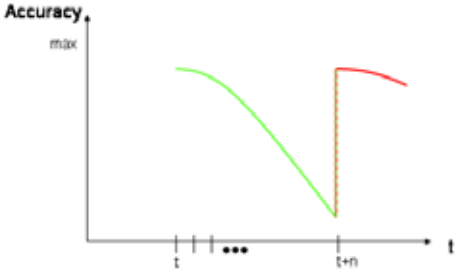
As shown in [WDP99], this method cannot be used for direct visualization of scenes for high quality rendering: holes and/or blurred zones may appear, causing noticeable artifacts. However, we do not use this method for the rendering of the final image. Instead, we use reprojection to obtain an approximation of the incoming radiance at a given point at time $t_0 + 1$. In our test scenes, this method is 99% accurate for estimating the irradiance at the next time step. This high accuracy is due to the fact that the displacements are generally small between two successive frames. Furthermore, the incoming radiance values are summed either to compute the irradiance or the projection of incoming radiance function into the hemispherical harmonics basis.

The temporal weighting function and record replacement scheme, along with an estimation of future incoming radiance, allow to improve both the computational efficiency and the animation quality. Nevertheless, Figure 5(c) shows that the accuracy of the computation is not continuous in the course of time. Replacing obsolete records by new ones creates a discontinuity of accuracy, which causes the visible flickering artifacts. Therefore, we propose

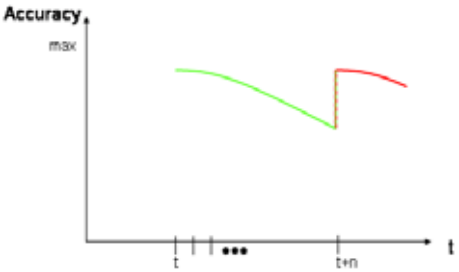




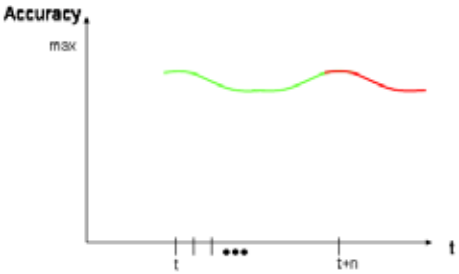
(b) New record after $\delta = 1$ frame



(c) New record after $\delta = n$ frames, no temporal gradient



(d) New record after $\delta = n$ frames, extrapolated temporal gradient



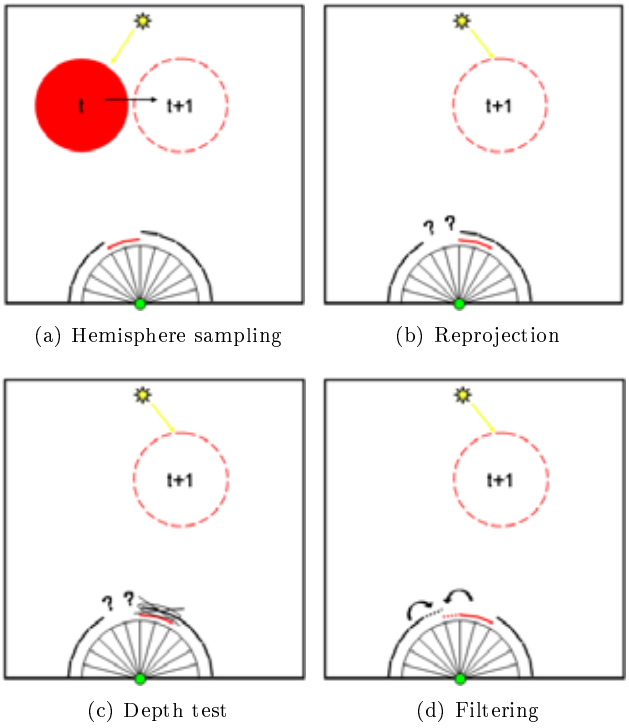
(e) New record after $\delta = n$ frames, interpolated temporal gradient

Figure 5: Empirical shape of the accuracy with respect to space (a) and time (b,c,d,e). If records are located at the same point between successive frames, the temporal accuracy is improved (a,b). However, flickering artifacts (i.e. temporal discontinuity of accuracy) may appear when a record is used without temporal gradients during several frames, then recomputed (c). Extrapolated temporal gradients drastically decrease the amplitude of the discontinuity in one pass, reducing flickering (d). Interpolated temporal gradients use two passes to eliminate the discontinuity (e).

temporal gradients to generate a smooth and less noticeable transition between successive records.

(a) Time t_k (b) Time $t_k + n$

Figure 6: Record k is created at time t_k on the dynamic object. Even though the object moves in subsequent frames, record k remains at the same position.



(a) Hemisphere sampling (b) Reprojection

(c) Depth test (d) Filtering

Figure 7: The hemisphere is sampled at time t as in the classical irradiance caching process (a). For each ray, our method estimates where each visible point will be located at time $t+1$ by reprojection (b). Distant overlapping points are removed using depth test (c), while resulting holes are filled using neighboring values (d).

4.4 Temporal Gradients

Temporal gradients are conceptually equivalent to classical irradiance gradients. Instead of representing the incoming radiance change with respect to translation and rotation, those gradients represent how the incoming radiance gets altered over time.

In the context of irradiance caching, Eq. 4 shows that the irradiance at point $\mathbf{p}$ is estimated using rotation and translation gradients. The temporal irradiance gradient of record k at a given point $\mathbf{p}$ with normal $\mathbf{n}$ is derived from Eq. 4 as:

$$\nabla^{\mathbf{t}}(\mathbf{p}) = \frac{\partial}{\partial t}(E_k + (\mathbf{n_k} \times \mathbf{n}) \cdot \nabla_{\mathbf{r}} + (\mathbf{p} - \mathbf{p_k}) \cdot \nabla_{\mathbf{p}}) \quad (13)$$

$$= \frac{\partial E_k}{\partial t} + \frac{\partial}{\partial t}\left[(\mathbf{n_k} \times \mathbf{n}) \cdot \nabla_{\mathbf{r}}\right] + \frac{\partial}{\partial t}\left[(\mathbf{p} - \mathbf{p_k}) \cdot \nabla_{\mathbf{p}}\right] \quad (14)$$

where:

- $\nabla_{\mathbf{r}}$ and $\nabla_{\mathbf{p}}$ are the rotation and translation gradients
- $\mathbf{p_k}$ and $\mathbf{n_k}$ are the location of the record k and the surface normal

As described in section 4.2, we choose to keep the location of the records constant over time even though they lie on dynamic objects. Moreover, we choose any point of interest $\mathbf{p}$ with normal $\mathbf{n}$ which is also constant over time. Therefore, $\mathbf{n_k} \times \mathbf{n}$ and $\mathbf{p} - \mathbf{p_k}$ are constant with respect to time. Hence the equation for temporal gradients becomes:

$$\nabla^{\mathbf{t}}(\mathbf{p}) = \frac{\partial E_k}{\partial t} + (\mathbf{n_k} \times \mathbf{n}) \cdot \frac{\partial \nabla_{\mathbf{r}}}{\partial t} + (\mathbf{p} - \mathbf{p_k}) \cdot \frac{\partial \nabla_{\mathbf{p}}}{\partial t} \quad (15)$$

$$= \nabla^{\mathbf{t}}_{\mathbf{E_k}} + (\mathbf{n_k} \times \mathbf{n}) \cdot \nabla^{\mathbf{t}}_{\nabla_{\mathbf{r}}} + (\mathbf{p} - \mathbf{p_k}) \cdot \nabla^{\mathbf{t}}_{\nabla_{\mathbf{p}}} \quad (16)$$

where:

- $\nabla^{\mathbf{t}}_{\mathbf{E_k}}$ is the *temporal gradient of incoming irradiance* at $\mathbf{p_k}$
- $\nabla^{\mathbf{t}}_{\nabla_{\mathbf{r}}}$ is the *temporal gradient of rotation gradient* at $\mathbf{p_k}$
- $\nabla^{\mathbf{t}}_{\nabla_{\mathbf{p}}}$ is the *temporal gradient of translation gradient* at time t_k
- t_k is the time at which record k has been created

Using Eq. 16, the contribution of record k created at time t_k to the incoming radiance at point $\mathbf{p}$ at time t is estimated by:

$$\begin{aligned} E_k(\mathbf{p}, t) = \; & E_k + \nabla^{\mathbf{t}}_{\mathbf{E_k}}(t - t_k) + \\ & (\mathbf{n_k} \times \mathbf{n}) \cdot (\nabla_{\mathbf{r}} + \nabla^{\mathbf{t}}_{\nabla_{\mathbf{r}}}(t - t_k)) + \\ & (\mathbf{p_k} - \mathbf{p}) \cdot (\nabla_{\mathbf{p}} + \nabla^{\mathbf{t}}_{\nabla_{\mathbf{p}}}(t - t_k)) \end{aligned} \quad (17)$$

This formulation allows to store the temporal change of the incoming radiance around p_k as 3 vectors. These vectors represent the change of the incoming radiance at point $\mathbf{p_k}$ and the change of the translational and rotational gradients over time. The values of these vectors can be easily computed using the information generated in section 4.3. Since our

method estimates the incoming radiance at time $t+1$ using information available at time t, it is straightforward to compute *extrapolated* temporal gradients as:

$$\nabla^{\mathbf{t}}_{\mathbf{E_k}} \approx E(t_k+1) - E(t_k) \quad (18)$$
$$\nabla^{\mathbf{t}}_{\nabla_\mathbf{r}} \approx \nabla_\mathbf{r}(t_k+1) - \nabla_\mathbf{r}(t_k) \quad (19)$$
$$\nabla^{\mathbf{t}}_{\nabla_\mathbf{p}} \approx \nabla_\mathbf{p}(t_k+1) - \nabla_\mathbf{p}(t_k) \quad (20)$$

However, as illustrated in Figure 5(d), these temporal gradients do not remove all the discontinuities in the animation. When a record k is replaced by record l, the accuracy of the result exhibits a possible discontinuity, yielding some flickering artifacts. As explained in section 4.2, this problem can be avoided by keeping track of the history of the records: when record k gets obsolete, a new record l is created at the same location. Since $t_l > t_k$, we can use the value of incoming radiance stored in l to compute *interpolated* temporal gradient for record k:

$$\nabla^{\mathbf{t}}_{\mathbf{E_k}} \approx (E_l - E_k)/(t_l - t_k) \quad (21)$$
$$\nabla^{\mathbf{t}}_{\nabla_\mathbf{r}} \approx (\nabla_\mathbf{r}(t_l) - \nabla_\mathbf{r}(t_k))/(t_l - t_k) \quad (22)$$
$$\nabla^{\mathbf{t}}_{\nabla_\mathbf{p}} \approx (\nabla_\mathbf{p}(t_l) - \nabla_\mathbf{p}(t_k))/(t_l - t_k) \quad (23)$$

As illustrated in Figure 5(e), these gradients enhance the continuity of the accuracy, hence removing the flickering artifacts.

This method provides a simple way of extending irradiance caching to dynamic objects and light sources, by introducing a temporal weighting function and temporal gradients. This algorithm can be simply included within any existing renderer based on irradiance caching. In the next section, we discuss the implementation of two important features of our method on a GPU: the estimation of incoming radiance for next frame, and the decision of recomputing the value of a record.

5 GPU Implementation

Our method has been implemented within a GPU-based renderer for irradiance and radiance caching. First, we detail the implementation of the incoming radiance estimate by reprojection (Section 4.3). Then, we describe how the GPU can be simply used in the context of radiance cache splatting to discard useless records and avoid their replacement.

Radiance Cache Splatting The radiance cache splatting method [GKBP05] is based on a simple observation of the spatial weighting function described in [WRC88]: an (ir)radiance record k cannot contribute to the lighting of points located outside a sphere of influence centered at p_k. Therefore, the indirect lighting at visible points can be obtained by splatting the sphere corresponding to record k on the image plane. For each fragment within the splatted sphere, a fragment program evaluates the weighting function for record k and

verifies whether record k is allowed to contribute to the indirect lighting of the visible point corresponding to this fragment (Equation 3). The weighted average described in Equation 1 is computed using simple hardware blending. This method allows to display high quality global illumination at interactive frame rates.

Reprojection of Incoming Radiance As shown in section 4.2, the computation of the temporal weighting function and temporal gradients for a given record k requires an estimate of the radiance reaching point $\mathbf{p_k}$ at the next time step. This estimate is obtained through reprojection (section 4.3), provided that the position of the objects at next time step is known. Therefore, for a given vertex v of the scene and a given time t, we assume that the transformation matrix corresponding to the position and normal of v at time $t+1$ is known. We assume that such matrices are available for light sources as well. Using the method described in [GKBP05], a record k can be generated by rasterizing the scene on a single plane above point $\mathbf{p_k}$. In a first pass, during the rasterization at time t, shaders can output an estimate of the position and incoming lighting at time $t+1$ of each point visible to $\mathbf{p_k}$ at time t. This output can be used to reconstruct an estimate of the incoming radiance function at time $t+1$.

This estimate is obtained in a second pass: each projected visible point generated in the first pass is considered as a vertex. Each of those vertices is sent to the graphics pipeline as a pixel-sized point. The result of the rasterization process is an estimate of the incoming radiance function at time $t+1$. Since the size of the sampling plane is usually small (typically 64×64), this process is generally much faster than resampling the whole scene.

During the reprojection process, some fragments may overlap. Even though the occlusion can be simply solved by classical Z-Buffer, the resulting image may contain holes (Figure 7(d)). These holes are created at the location of dynamic objects. Since the time shift between two successive frames is very small, the holes are generally very small. Therefore, as described in Section 4.3, we use a third pass to apply a filter on the pixels surrounding a hole to obtain a plausible value. This computation can be performed efficiently on the GPU using the stencil buffer with an initial value of 0. During the reprojection process, each rasterized point increments the stencil buffer. Therefore, the hole-filling algorithm must be applied only on pixels where the stencil buffer has not been incremented. The final result of this algorithm is an estimate of the incoming radiance at time $t+1$, generated entirely on the GPU (Figure 8).

This method allows inexpensive estimation of the incoming radiance function at next time step. This result is used to compute both the temporal gradients and the temporal weighting function. As shown in Equation 10, this latter defines a maximum value of the lifespan of a given record k. At the end of this lifespan, section 4.2 indicates that a new record l is generated at the same location to replace the obsolete record k. However, if k does not contribute to the indirect lighting of visible points, we do not need to generate record l. Therefore, we propose a simple method to detect whether a record has to be replaced, or simply discarded.

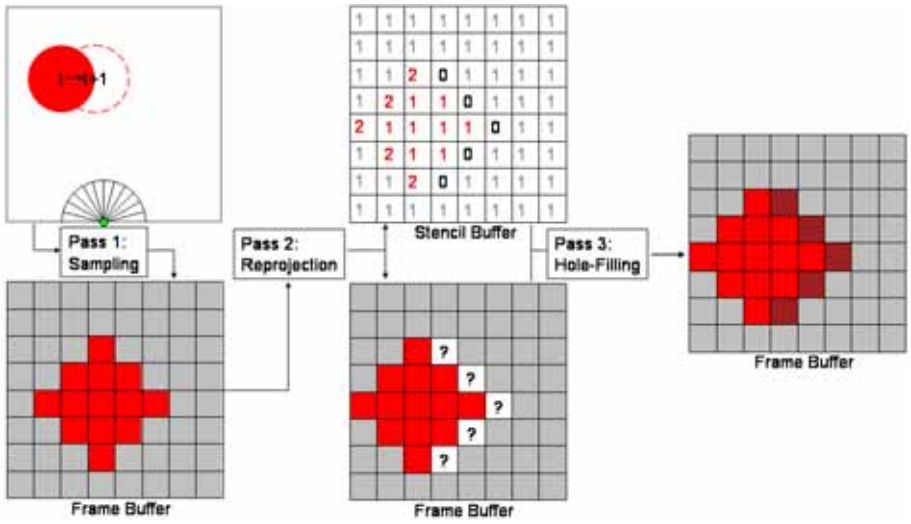
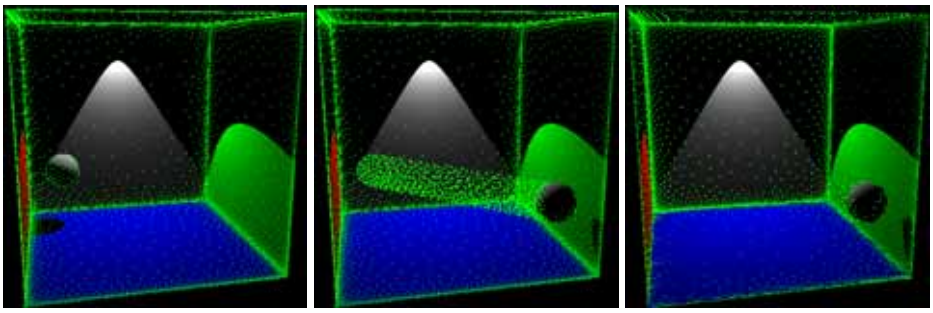


Figure 8: Reprojection using the GPU. The first pass samples the scene to gather the required information. The second pass reprojects the visible points to obtain an estimate of the incoming radiance function at next time step. During this pass, each rendered fragment increments the stencil buffer (0 corresponds to the absence of fragment, 1 to one fragment, and 2 to two overlapping fragments). The last pass fills the holes (i.e. where the stencil value is 0) using a simple 3×3 filter, yielding a plausible estimate of the radiance incoming from newly visible points.

Replacement/Deletion Method As described in previous sections, the flickering artifacts of the lighting come from the temporal discontinuities of the accuracy. Therefore, if a record cannot contribute to the current image (i.e. out of the view frustum, or occluded), it can be simply deleted instead of being replaced by a novel, up-to-date record. As shown in Figure 9, this avoids the generation and update of a "trail" of records following dynamic objects. Therefore, both the computing time and memory consumption are reduced. In the context of radiance cache splatting [GKBP05], this decision can be easily made using hardware occlusion queries: during the last frame of the lifespan of record k, an occlusion query is issued as the record is rasterized. In the next frame, valid records are first rendered. Since record k is now obsolete, the result of the occlusion query is read from the GPU. If the number of covered pixels is 0, the record is discarded. Otherwise, a new record l is computed at location $\mathbf{p_l} = \mathbf{p_k}$.

The hardware occlusion queries are very useful, but they suffer from high latency. However, in our method, the result of a query is not needed immediately. Between the query issue and the reading of the record coverage, the renderer renders the other records, then switches the scene to next frame and renders valid records. In practice, the average latency



(a) Time 1 (b) Time 100, systematic update (c) Time 100, our record removal method

Figure 9: In this scene, the sphere moves from the left to the right of the Cornell Box. At time 1 (a), records (represented by green points) are generated to compute the global illumination solution. Several records lie on the dynamic sphere. Therefore, when the sphere moves, new records are created to evaluate the incoming radiance on the sphere. If every record is kept up-to-date during the entire animation segment, a "trail" of records lie on the path of the dynamic sphere (b). Using our method, only useful records are updated, avoiding the storage and update of useless records (c).

appeared to be negligible (less than 0.1% of the overall computing time). Besides, in our test scenes, this method reduces the storage cost by 25-30%, and the computational cost by 20-30%.

6 Results

This section discusses the results obtained using our method and compares them with the results obtained using the classical method in which a new cache is computed for each frame. This latter method is referred to as *per-frame computation* in the remainder of this section. The images, videos and timings have been generated using a 3.8GHz Pentium 4 with 2 GB RAM and an nVidia GeForce 7800 GTX 256MB. The timings are summarized in Table 1.

Cube in a Box This very simple, diffuse scene (Figure 11(a)) exhibits high flickering when no temporal gradients are used (see accompanying video). This is mainly due to the simplicity of the scene and the absence of textures. However, the accompanying video shows that flickering is drastically reduced by using extrapolated (1-pass) temporal gradients, and is unnoticeable with interpolated (2-pass) gradients. Moreover, the 400 frame animation is rendered in 268.7 s using temporal gradients, while a complete computation using a new cache per every frame requires 2048.2 s. Therefore, for this scene, our method is 7.6 times faster. In this example, images are generated using a temporal accuracy parameter $a^t = 0.05$,

and a maximum lifespan of 20 frames. Figure 10 shows the accuracy values obtained with and without temporal gradients. Our method reduces the loss of accuracy, hence decreasing the number of flickering artifacts. In the case of per-frame computation the memory required to store all the irradiance cache records is about 138 MB (772K records in the cache). In our method, the memory load is about 11 times smaller (12.4 MB, i.e. 50K records in the cache)). As shown in Figure 10, the extrapolated (1-pass) temporal gradients still generate discontinuities of accuracy, hence flickering artifacts may be visible. Since our aim is high quality rendering, the following results focus on interpolated (2-pass) temporal gradients which do not introduce discontinuities.

Moving Light A similar scene (Figure 11(b)) illustrates the behavior of our algorithm in the context of dynamic light sources. The bottom of the box is tiled to highlight the change of indirect lighting when the light source moves. Using our method, the indirect lighting is stable while per-frame computation yields important flickering artifacts. Since the lighting is provided by a single, dynamic light source, the indirect lighting changes very quickly. Therefore, the lifespan of the records is generally very short, yielding frequent updates of irradiance values. Compared to per-frame computation, our method allows to render the animation with higher quality in a comparable time.

Flying Kite In a more complicated scene containing textures (Figure 11(c)), our algorithm also provides a drastic quality improvement. The video shows that temporal gradients yield flicker-free animations. Moreover, the rendering speed is 6.52× as compared to per-frame computation. In the beginning of the animation, the motion of the kite does not have much impact on the indirect lighting of the surrounding objects. Therefore, most of the records can be reused in several frames. However, when the kite gets down, its dynamic reflection on the ceiling and wall is clearly noticeable. Our algorithm updates records in this zone very frequently. Therefore, the global illumination solution of this zone is updated at a fast pace and thus avoids ghosts in the final image. As shown in Figure 1, our temporal weighting function adapts the validity period of the records to the actual change of indirect lighting, ensuring both reactivity and computational efficiency.

Japanese Interior (200K polygons) This scene contains highly detailed, dynamic objects (Figure 11(d)). In this scene, glossy objects are rendered using radiance caching, while irradiance caching is used to render diffuse objects. The animation illustrates the features of our method: dynamic diffuse and glossy objects, and important changes of indirect lighting. In the beginning of the animation, the scene is lit by a single, dynamic light source. Therefore, records have a very short lifespan. In this case, temporal gradients suppress the flickering artifacts present in per-frame computation, but they do not provide a significant speedup (1.25×). In the remainder of the animation, most of the environment is static, even though some dynamic objects generate strong changes in indirect illumination. Our temporal gradients allow to take advantage of this situation by adaptively reusing records in several frames. The result is the elimination of flickering and a significant speedup (up

to 9×) compared to per-frame computation. In the whole animation segment, our method yields an overall speedup of 1.9×. Note that during the generation of this animation, the average latency introduced by occlusion queries is 0.001% of the overall rendering time.

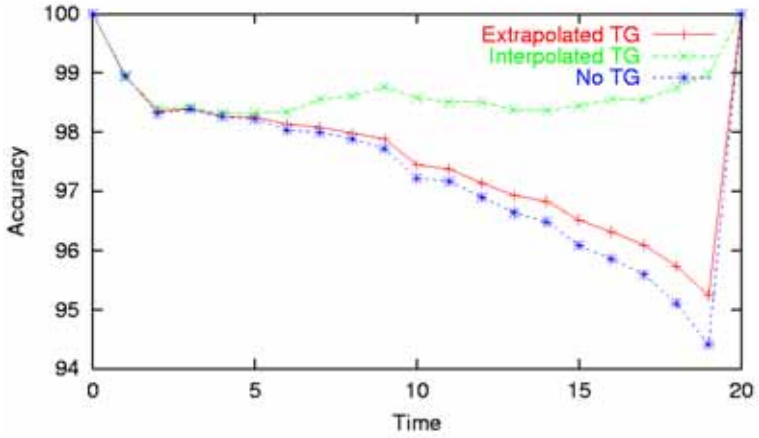
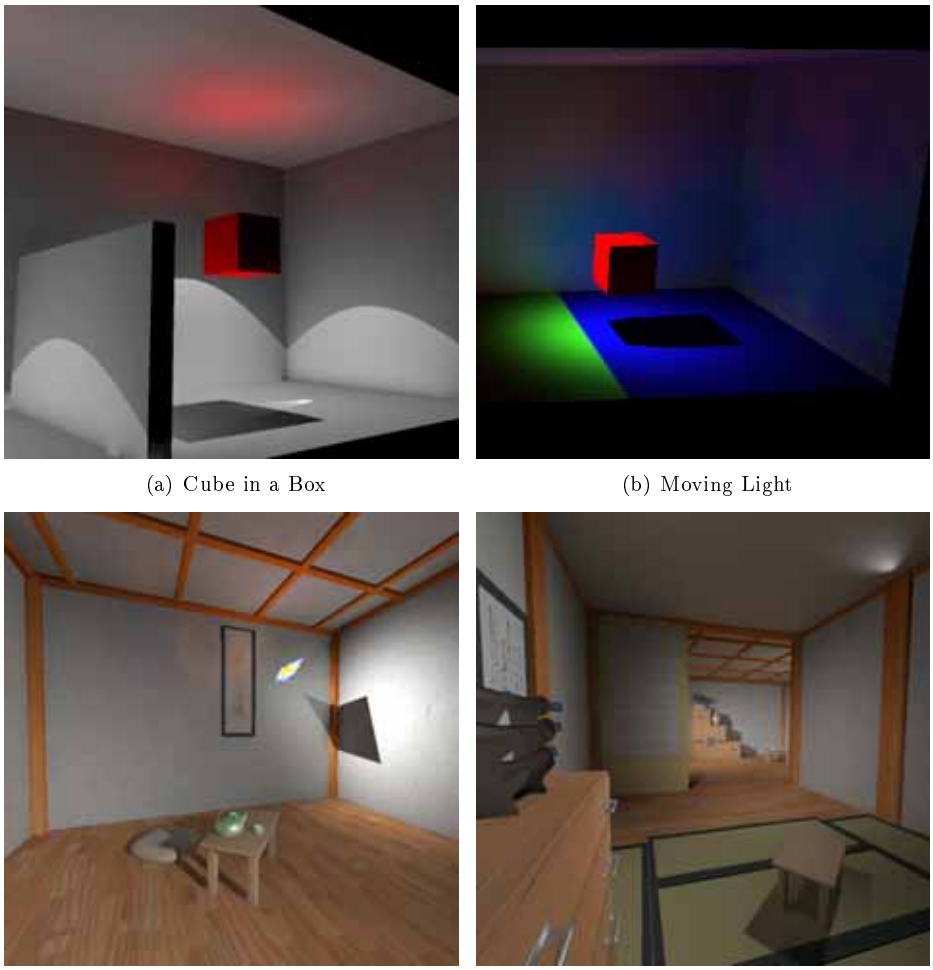


Figure 10: Temporal accuracy values obtained by creating records at time 0 and extrapolating their value until time 19. New records are recomputed at time 20. Our method based on temporal gradients (TG) yields a better approximation compared to the approach without those gradients. Note that the accuracy remains above 98% during the whole time frame using interpolated (2-pass) gradients.

Scene	Per-Frame Computation (s)	Our Method (s)	Speedup
Cube in a Box	2048	269	7.62
Moving Light	2518	2650	0.95
Flying Kite	5109	783	6.52
Japanese Interior	13737	7152	1.9

Table 1: Timings obtained using classical per-frame computation and our method.

Computational Overhead of Reprojection During the computation of a record, our method evaluates the value of the incoming lighting for both current and next time steps. As shown in Section 4.3, the estimation of the incoming lighting at next time step is performed by simple reprojection. Therefore, the related computational overhead is entirely independent of the scene geometry. In our tests, each record was computed using a 64×64 grid. Independent of the scene, the time required to estimate the future incoming lighting



(a) Cube in a Box (b) Moving Light

(c) Flying Kite (d) Japanese Interior

Figure 11: Images of scenes discussed in Section 6. Animation rendering for each of these scenes is shown in the accompanying video.

at a given point is approximately 0.46 ms. As a comparison, the time required to compute the actual incoming lighting at a given point in a 200K polygons scene is 4.58 ms. In this case, the additional cost due to the reprojection is only 10% of the cost of the actual record computation. Even though this overhead is not negligible, our estimate allows to reuse many records in several frames, thus reducing the overall rendering cost.

7 Conclusion

In this paper, we presented a novel method for exploiting temporal coherence in the context of irradiance and radiance caching. We proposed an approach for sparse sampling of incoming radiance in the temporal domain. We defined a temporal weighting function and temporal gradients, allowing a simple and accurate temporal interpolation of incoming radiance values. The results show both a significant speedup and an increased quality compared to per-frame computation. Moreover, the sparse sampling in the temporal domain allows to store the incoming radiance values for the entire animation segment within main memory. Due to the significative quality improvement and the ease of implementation, we believe that our approach can be easily integrated in production renderers for efficient rendering of animated scenes.

Future work includes the design of a more accurate estimation method for extrapolated (1-pass) temporal gradients. Such a method will find use in on-the-fly computation of indirect lighting during interactive sessions. Another improvement would consist in designing an efficient method for faster aging of the records located near newly created records for which important changes have been detected. This would avoid the need for a user-defined maximum validity time, while guaranteeing the absence of global illumination ghosts.

References

[BFMZ94] Gary Bishop, Henry Fuchs, Leonard McMillan, and Ellen J. Scher Zagier. Frameless rendering: double buffering considered harmful. In *Proceedings of SIGGRAPH*, pages 175–176, 1994.

[BP01] Gonzalo Besuievsky and Xavier Pueyo. Animating radiosity environments through the multi-frame lighting method. *Journal of Visualization and Computer Graphics*, 12:93–106, 2001.

[BS96] Gonzalo Besuievsky and Mateu Sbert. The multi-frame lighting method: A monte carlo based solution for radiosity in dynamic environments. In *Proceedings of Eurographics Workshop on Rendering*, pages 185–194, 1996.

[BWCG86] Daniel R. Baum, John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. The back-buffer algorithm: An extension of the radiosity method to dynamic environments. *The Visual Computer*, 2(5):298–306, 1986.

[Dam01] Cyrille Damez. *Simulation Globale de l'Eclairage Pour Des Sequences Animees Prenant En Compte la Coherence Temporelle*. PhD thesis, Universite Joseph Fourier, Grenoble, France, 2001.

[DBMS02] Kirill Dmitriev, Stefan Brabec, Karol Myszkowski, and Hans-Peter Seidel. Interactive global illumination using selective photon tracing. In *Proceedings of Eurographics Workshop on Rendering*, pages 25–36, 2002.

[DDM02] Cyrille Damez, Kirill Dmitriev, and Karol Myszkowski. Global illumination for interactive applications and high-quality animations. In *Proceedings of Eurographics*, pages 55–77, September 2002.

[DS97] George Drettakis and Francois X. Sillion. Interactive update of global illumination using a line-space hierarchy. In *Proceedings of SIGGRAPH*, volume 31, pages 57–64, 1997.

[DSH01] Cyrille Damez, Francois X. Sillion, and Nicolas Holzschuch. Space-time hierarchical radiosity with clustering and higher-order wavelets. In *Proceedings of Eurographics*, pages 129–141, September 2001.

[GD04] Xavier Granier and George Drettakis. A final reconstruction approach for a unified global illumination algorithm. *ACM Transactions on Graphics*, 23(2):163–189, 2004.

[GKBP05] Pascal Gautron, Jaroslav Křivánek, Kadi Bouatouch, and Sumanta Pattanaik. Radiance cache splatting: A GPU-friendly global illumination algorithm. In *Proceedings of Eurographics Symposium on Rendering*, June 2005.

[GKPB04] Pascal Gautron, Jaroslav Křivánek, Sumanta Pattanaik, and Kadi Bouatouch. A novel hemispherical basis for accurate and efficient rendering. In *Proceedings of Eurographics Symposium on Rendering*, pages 321–330, 2004.

[GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modelling the interaction of light between diffuse surfaces. In *Proceedings of SIGGRAPH*, pages 212–222, July 1984.

[Jen01] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, July 2001.

[KGBP05] Jaroslav Křivánek, Pascal Gautron, Kadi Bouatouch, and Sumanta Pattanaik. Improved radiance gradient computation. In *Proceedings of SCCG*, pages 149–153, 2005.

[KGPB05] Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):550–561, 2005.

[MPT03] Ignacio Martín, Xavier Pueyo, and Dani Tost. Frame-to-frame coherent animation with two-pass radiosity. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):70–84, 2003.

[PTMG97] Xavier Pueyo, Dani Tost, Ignacio Martín, and Blanca Garcia. Radiosity for dynamic environments. *The Journal of Visualization and Computer Animation*, 8(4):221–231, 1997.

[SKDM05] Miloslaw Smyk, Shin-ichi Kinuwaki, Roman Durikovic, and Karol Myszkowski. Temporally coherent irradiance caching for high quality animation rendering. In *Proceedings of Eurographics*, volume 24, pages 401–412, 2005.

[TMS02] Takehiro Tawara, Karol Myszkowski, and Hans-Peter Seidel. Localizing the final gathering for dynamic scenes using the photon map. In *VMV*, 2002.

[TMS04] T. Tawara, K. Myszkowski, and H.-P. Seidel. Exploiting temporal coherence in final gathering for dynamic scenes. In *Proceedings of Computer Graphics International*, pages 110–119, June 2004.

[TPWG02] Parag Tole, Fabio Pellacini, Bruce Walter, and Donald P. Greenberg. Interactive global illumination in dynamic scenes. In *Proceedings of SIGGRAPH*, pages 537–546, 2002.

[WDG02] Bruce Walter, George Drettakis, and Donald P. Greenberg. Enhancing and optimizing the render cache. In *Proceedings of Eurographics Workshop on Rendering*, pages 37–42, 2002.

[WDP99] Bruce Walter, George Drettakis, and Steven Parker. Interactive rendering using the render cache. In *Proceedings of Eurographics Workshop on Rendering*, pages 235–246, 1999.

[WH92] Gregory J. Ward and Paul S. Heckbert. Irradiance gradients. In *Proceedings of Eurographics Workshop on Rendering*, pages 85–98, 1992.

[WRC88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *Proceedings of SIGGRAPH*, pages 85–92, 1988.