Chapter 8

Temporal Radiance Caching

Very often global illumination methods aim at simulating the light/matter interactions within static scenes. Accounting for the displacement of objects and light sources either require a complete recomputation of the global illumination solution for each frame of the animation, or involve complex data structures and algorithms for temporal optimization. Furthermore, the global illumination solutions commonly exhibit low temporal quality when used in dynamic scenes: flickering, popping, ... In the context of computer-assisted effects for movies, high quality global illumination is obtained through temporal filtering: a 30 fps animation is first rendered at 60 fps by recomputing the global illumination for each frame. Then, each frame of the 30 fps animation is generated by averaging two frames of the 60 fps animation, hence reducing the temporal artifacts at the cost of high computational cost. For interactive applications such as video games, the illumination must be computed interactively. In this case, approximate models are generally preferred, such as the precomputation of a static global illumination solution, and the update of direct lighting only at runtime.

This chapter describes a simple and accurate method based on temporal caching for the computation of global illumination effects in animated environments [GBP07], where viewer, objects and light sources move. This approach focuses on a temporal optimization for lighting computation based on the irradiance caching [WRC88] technique. As this algorithm leverages the spatial coherence of indirect lighting to reduce the cost of global illumination, we consider here an extension of these methods for sparse temporal sampling and interpolation. In [WRC88], Ward *et al.* propose a reuse an irradiance value in the neighborhood of the actual computation point. While the weighting function and gradients of [WRC88] account for the spatial change of irradiance, Temporal Radiance Caching considers the temporal change of the indirect lighting (Figure 8.2).

8.1 Temporal Irradiance Caching

8.1.1 Irradiance Caching in Dynamic Scenes

The irradiance caching algorithm leverages the spatial coherency of the indirect lighting, and thus reduces the computation time of global illumination. In dynamic scenes, 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 8.1 illustrates the consequence of the change of record distribution: since the gradients extrapolate the change of incoming radiance, they are not completely accurate compared to the ground truth. Therefore, the accuracy of the lighting reconstructed by irradiance caching is not constant over a surface: the accuracy is maximum at the record location, and decreases as the extrapolation point gets away from the record. 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, a significant amount of computational effort is wasted.

The remainder of this chapter describes 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.

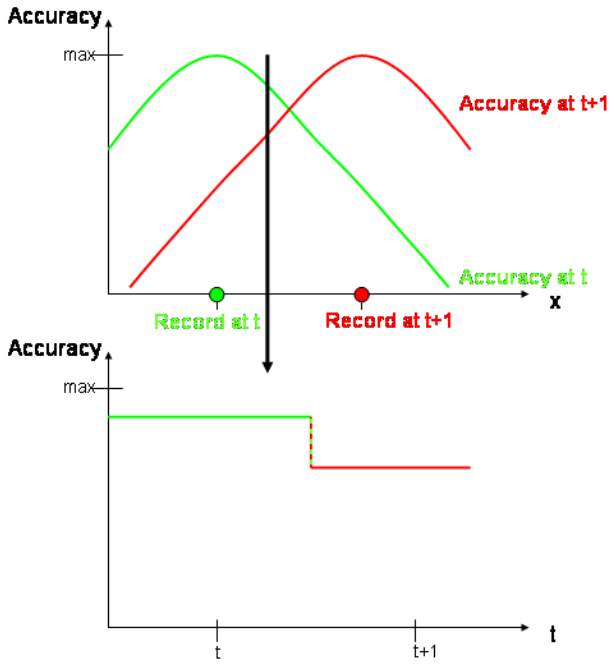
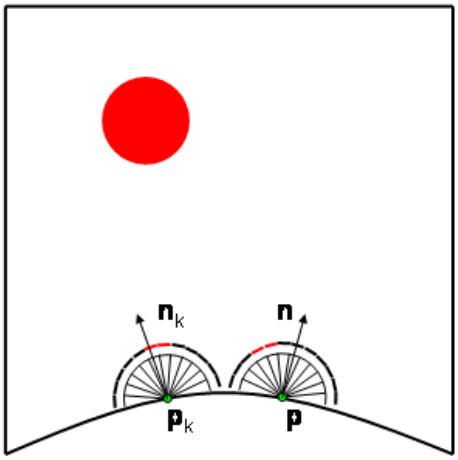


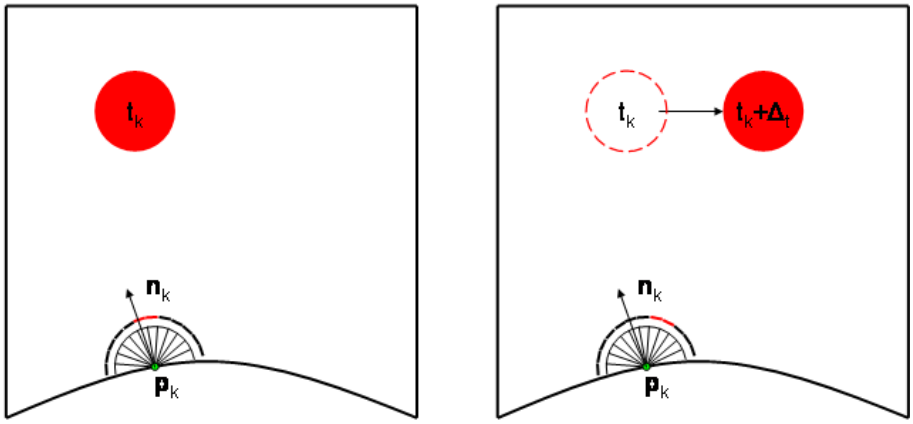
Figure 8.1: Changing the location of the records between successive frames can significantly modify the accuracy of the lighting at a given point (marked by the arrow). Therefore, the incoming radiance at this point can change noticeably between two frames, yielding flickering artifacts. Note that the maximum accuracy is obtained at the point and time of actual computation of the incoming radiance.

8.1.2 Overview of Temporal Radiance Caching

Following the spatial interpolation scheme of Ward *et al.* [WRC88, WH92], our aim is to reuse records across frames by performing a sparse temporal sampling and temporal interpolation of the irradiance (Algorithm 1). When a record k is created at frame n, the future incoming lighting is estimated. This estimate is first used to compute the temporal change rate of the irradiance. In the spirit of [WRC88], the temporal weighting function w_k^t is defined as the inverse of the temporal change. The number of frames in which k can contribute is then inversely proportional to the future change of incoming radiance. Since the actual computation of the future incoming lighting may be expensive, we use a simple reprojection technique for estimating the future lighting using the data sampled at the current frame. As the irradiance at a point is extrapolated using the irradiance and gradients of neighboring records, the temporal caching scheme also features temporal gradients for smooth temporal interpolation and extrapolation of the irradiance.



(a) Spatial change



(b) Temporal change

Figure 8.2: 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.

8.1.3 Temporal Weighting Function

The temporal weighting function expresses the validity of a given record over time. Using a derivation similar to that of [WRC88], the temporal change ϵ^t of incoming radiance between time t and t_0 is:

$$\epsilon^t = \frac{\partial E}{\partial t}(t_0) \quad (t - t_0) \tag{8.1}$$

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.

$$\frac{\partial E}{\partial t}(t_0) \approx \frac{E_1 - E_0}{t_1 - t_0} \tag{8.2}$$

$$= \frac{\tau E_0 - E_0}{t_1 - t_0} \quad \text{where } \tau = E_1/E_0 \tag{8.3}$$

Algorithm 1 Temporal Radiance Caching

```
for all frames n do
  for all existing records k do
    if w_k^t(n) is sufficient then
      Use k in frame n
    end if
  end for
  for all points p where a new record is needed do
    Sample the hemisphere above p
    Estimate the future incoming lighting (Section 8.1.4)
    Generate w_k^t (Section 8.1.3)
    Compute the temporal gradients (Section 8.1.5)
    Store the record in the cache
  end for
end for
```

$$= \quad E_0 \frac{\tau - 1}{t_1 - t_0} \tag{8.4}$$

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], the temporal weighting function is the inverse of the change, excluding the term E_0:

$$w_k^t(t) = \frac{1}{(\tau - 1)(t - t_0)} \tag{8.5}$$

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

$$w_k^t(t) \geq 1/a^t \tag{8.6}$$

The temporal weighting function is used 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. The temporal radiance change rate is defined as:

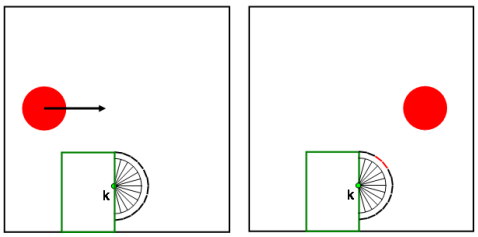
$$\tau_{radiance} = max\{\lambda_1^i/\lambda_0^i, 0 \geq i < n\} \tag{8.7}$$

where λ_0^i and λ_1^i are respectively the i^{th} projection coefficient of the current and future incoming lighting. The maximum rate of change is used to account for directional change of incoming radiance without loss of accuracy.

However, Equation 8.3 shows that if the environment remains static starting from frame t_0, we obtain $\tau = 1$. Therefore, Equation 8.6 shows that w_k^t is infinite for any frame, and hence 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 8.3). This is a limitation of this technique for estimating the temporal change of incoming radiance, which determines the lifespan of a record by only considering the change between E_t and E_{t+1}. A user-defined value δt_{max} limits the length of the validity time segment associated with each record to a reasonable, *ad hoc* value, such as 5 to 10 frames depending on the scene. If Equation 8.8 does not hold, we decide that the record cannot be reasonably used.

$$t - t_0 < \delta_{t_{max}} \tag{8.8}$$

This reduces the risk of having records from being used while they are obsolete, hence controlling the artifacts due to residual global illumination effects also known as "ghosts", which commonly appear in interactive methods. However, as a and a^T, this value must be user-defined by trial and error to obtain the best results. If $\delta_{t_{max}}$ is too low, many records may be recomputed unnecessarily. Setting $\delta_{t_{max}} = 1$ implies the recomputation of each record for each frame. In this case, the resulting performance would be similar to the classical per-frame computation. Nevertheless, this would completely avoid the ghosting artifacts, the indirect lighting being permanently computed from scratch. If $\delta_{t_{max}}$ is set too high, the same records might be reused in too many frames. Hence artifacts due to the residual global illumination effects are likely to appear in the vicinity of the moving objects, degrading the quality of the rendered frame. Hence such a high value significantly reduces the rendering time at the cost of accuracy.



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

Figure 8.3: When record k is created at time t_k, the surrounding environment is static ($\tau_k = 1$). However, the red sphere is visible from the k n frames later. The user-defined value $\delta_{t_{max}}$ prevents the record from contributing if $n > \delta_{t_{max}}$, reducing the risk of using obsolete records.

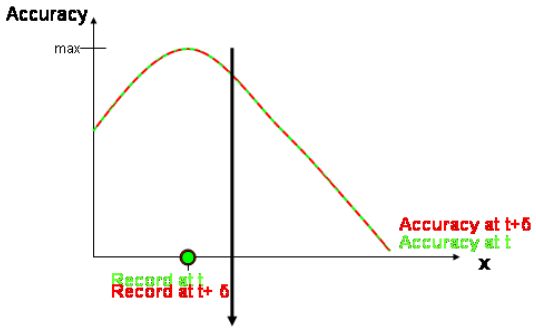
However, the flickering problem described in Section 8.1.1 still appears when a record is suddenly discarded. As proposed in [TMS02], this issue is limited by keeping track of the location of the records over time. Let us consider a record k located at point $\mathbf{p_k}$. If k was allowed to contribute to the 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 8.4(b)). Since the location of visible records remains constant in space, 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 8.5).

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. However, as shown in Equation 8.3, the determination of our temporal weighting function relies on the knowledge of the incoming radiance at the next time step, E_{t_0+1}. Since the explicit computation of E_{t_0+1} would introduce a huge computational overhead, this value is estimated using a simple reprojection.

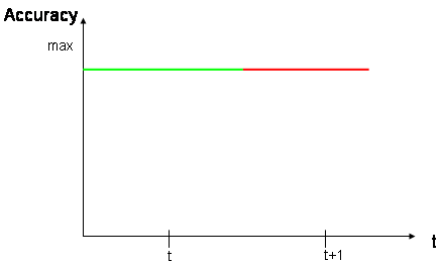
8.1.4 Estimating E_{t_0+1}

This method follows the reprojection approach proposed in [WDP99, WDG02]. However, while Walter *et al.* use reprojection for interactive visualization using ray tracing, the aim here 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.

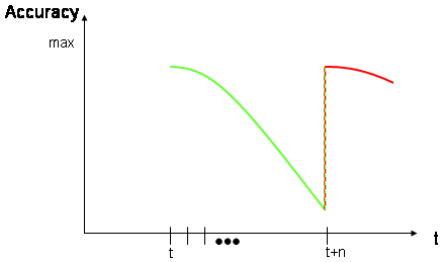
In the context of predefined animation, the changes in the scene are known and accessible at any time. When a record k is created at time t_0, the hemisphere above $\mathbf{p_k}$ is sampled (Figure 8.6(a)) to compute the incoming radiance and gradients at this point. Since the changes between times t_0 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 8.6(b)). The outgoing radiance



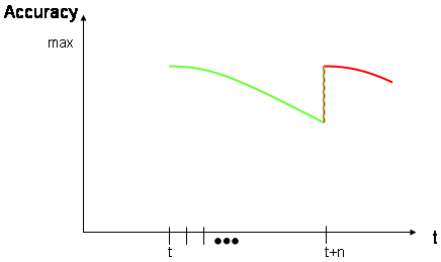
(a)



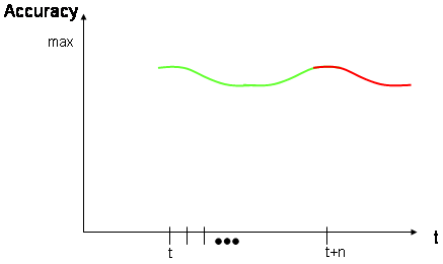
(b) New record after $\delta = 1$ frame



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



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



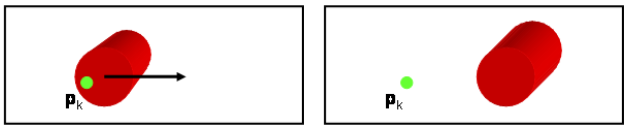
(e) New record after $\delta = n$ frames, interpolated temporal gradient

Figure 8.4: Empirical shape of the accuracy at a fixed time with respect to space (a), and at a fixed location with respect to time (b,c,d,e). If records are located at the same point between successive frames (a), the temporal accuracy is improved (b). However, flickering artifacts due to the temporal discontinuities of accuracy may appear when a record is reused in several frames, then recomputed (c). Extrapolated temporal gradients decrease the amplitude of the discontinuity in one pass, reducing flickering (d). Interpolated temporal gradients use two passes to eliminate the discontinuity by smoothing out the temporal changes (e).

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 accounts for the occlusion change (Figure 8.6(c)).

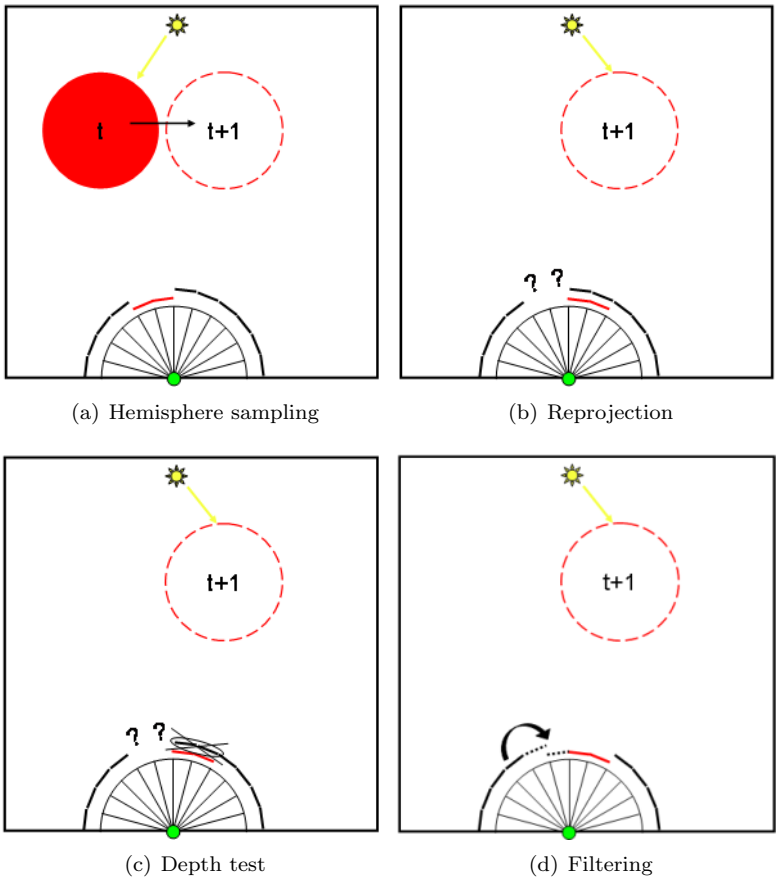
However, some parts of the estimated incoming radiance may be unknown (holes) due to displacement and overlapping of visible objects (Figure 8.6(d)). As proposed in [WDP99], we use a simple hole-filling method: each hole is filled using the background values, yielding a plausible estimate of the future indirect lighting.

Note that we use reprojection only to obtain an approximate of the incoming radiance at a



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

Figure 8.5: Record k created at time t_k remains at point $\mathbf{p_k}$ even though it lies on a dynamic object.



(a) Hemisphere sampling (b) Reprojection

(c) Depth test (d) Filtering

Figure 8.6: 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 the farthest neighboring values (d).

given point at time $t_0 + 1$. As shown in Figure 8.7 and Table 8.1, the reprojection reduces the error in the estimate of the future incoming lighting. Those errors were measured by comparing the irradiances at time $t + 1$ with the irradiances estimated using records created at time t.

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 8.4(c) shows that the accuracy of the computation is still not continuous in the course of time. Replacing obsolete records by new ones creates a discontinuity

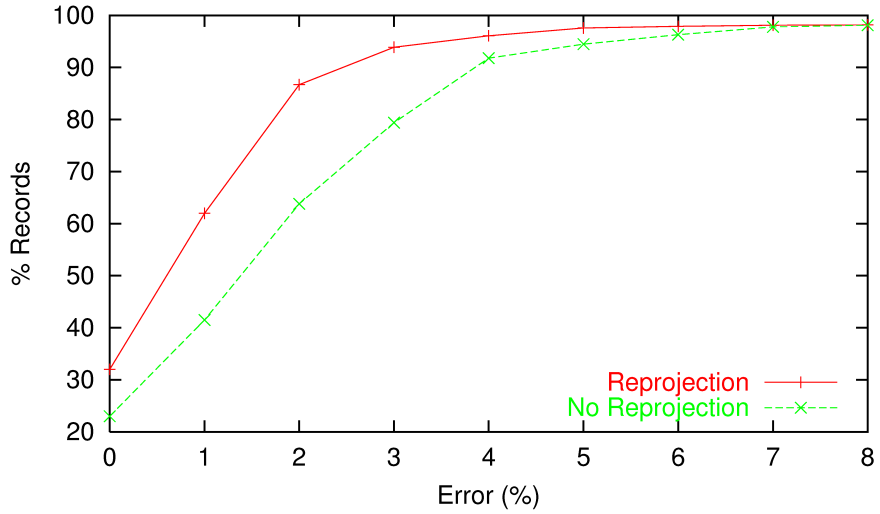


Figure 8.7: Percentage of records for which the estimate of the future incoming lighting is below a given RMS error level. The reprojection reduces the overall error in the estimate compared to a method without reprojection (i.e. the where the lighting is considered temporally constant). (Errors computed using 4523 values.)

Error	Reprojection	No Reprojection
Min	0%	0%
Max	30%	32%
Mean	2.9%	3.7%
Median	1.6%	2.4%

Table 8.1: RMS error of reprojection (Based on 4523 values).

of accuracy, which causes the visible flickering artifacts. The degree of the temporal irradiance interpolation is then raised using *temporal gradients*.

8.1.5 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, the irradiance at a 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 the formulae proposed in [WH92]:

$$\nabla^{\mathbf{t}}(\mathbf{p}) = \frac{\partial}{\partial t}(E_k + (\mathbf{n_k} \times \mathbf{n}) \cdot \nabla_{\mathbf{r}} + (\mathbf{p} - \mathbf{p_k}) \cdot \nabla_{\mathbf{p}}) \tag{8.9}$$

where:

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

As described in section 8.1.3, the location of all records is kept constant over time. Let us 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. The equation for temporal gradients becomes:

$$\nabla^{\mathbf{t}}(\mathbf{p}) = \nabla^{\mathbf{t}}_{\mathbf{E_k}} + (\mathbf{n_k} \times \mathbf{n}) \cdot \nabla^{\mathbf{t}}_{\nabla_\mathbf{r}} + (\mathbf{p} - \mathbf{p_k}) \cdot \nabla^{\mathbf{t}}_{\nabla_\mathbf{p}} \tag{8.10}$$

where:

- $\nabla^{\mathbf{t}}_{\mathbf{E_k}} = \frac{\partial E_k}{\partial t}$ is the *temporal gradient of irradiance*
- $\nabla^{\mathbf{t}}_{\nabla_\mathbf{r}} = \frac{\partial \nabla_\mathbf{r}}{\partial t}$ is the *temporal gradient of rotation gradient*
- $\nabla^{\mathbf{t}}_{\nabla_\mathbf{p}} = \frac{\partial \nabla_\mathbf{p}}{\partial t}$ is the *temporal gradient of translation gradient*

Using Equation 8.10, the contribution of record k created at time t_k to the incoming radiance at point $\mathbf{p}$ at time t is estimated by:

$$\begin{aligned} E_k(\mathbf{p}, t) \quad = \quad & E_k + \nabla^{\mathbf{t}}_{\mathbf{E_k}}(t - t_k) + \\ & (\mathbf{n_k} \times \mathbf{n}) \cdot (\nabla_\mathbf{r} + \nabla^{\mathbf{t}}_{\nabla_\mathbf{r}}(t - t_k)) + \\ & (\mathbf{p_k} - \mathbf{p}) \cdot (\nabla_\mathbf{p} + \nabla^{\mathbf{t}}_{\nabla_\mathbf{p}}(t - t_k)) \end{aligned} \tag{8.11}$$

This formulation represents 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 translation and rotation gradients over time. The values of these vectors can be easily computed using the information generated in Section 8.1.4. Since our method estimates the incoming radiance at time $t+1$ using the information available at time t, we define *extrapolated* temporal gradients as:

$$\nabla^{\mathbf{t}}_{\mathbf{E_k}} \approx E(t_k + 1) - E(t_k) \tag{8.12}$$
$$\nabla^{\mathbf{t}}_{\nabla_\mathbf{r}} \approx \nabla_\mathbf{r}(t_k + 1) - \nabla_\mathbf{r}(t_k) \tag{8.13}$$
$$\nabla^{\mathbf{t}}_{\nabla_\mathbf{p}} \approx \nabla_\mathbf{p}(t_k + 1) - \nabla_\mathbf{p}(t_k) \tag{8.14}$$

However, as illustrated in Figure 8.4(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 8.1.3, 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:

$$\nabla^{\mathbf{t}}_{\mathbf{E_k}} \approx (E_l - E_k)/(t_l - t_k) \tag{8.15}$$
$$\nabla^{\mathbf{t}}_{\nabla_\mathbf{r}} \approx (\nabla_\mathbf{r}(t_l) - \nabla_\mathbf{r}(t_k))/(t_l - t_k) \tag{8.16}$$
$$\nabla^{\mathbf{t}}_{\nabla_\mathbf{p}} \approx (\nabla_\mathbf{p}(t_l) - \nabla_\mathbf{p}(t_k))/(t_l - t_k) \tag{8.17}$$

As illustrated in Figure 8.4(e), the temporal gradients enhance the continuity of the accuracy, hence removing the flickering artifacts. However, the gradients only account for the first derivative of the change of incoming lighting, hence temporally smoothing the changes. While this method proves accurate in scenes with smooth changes, it should be noted that the gradient-based temporal interpolation may introduce ghosting artifacts when used in scenes with very sharp changes of illumination. In this case, a^t and $\delta_{t_{max}}$ must be reduced to obtain a sufficient update frequency.

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. In the next section, we discuss the implementation of our method on a GPU for increased performance.

8.2 GPU Implementation Details

This section first details the implementation of the incoming radiance estimate by reprojection (Section 8.1.4). 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.

Reprojection of Incoming Radiance As shown in section 8.1.3, 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 8.1.4), 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 hemisphere sampling method described in the previous chapter, 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 $t+1$.

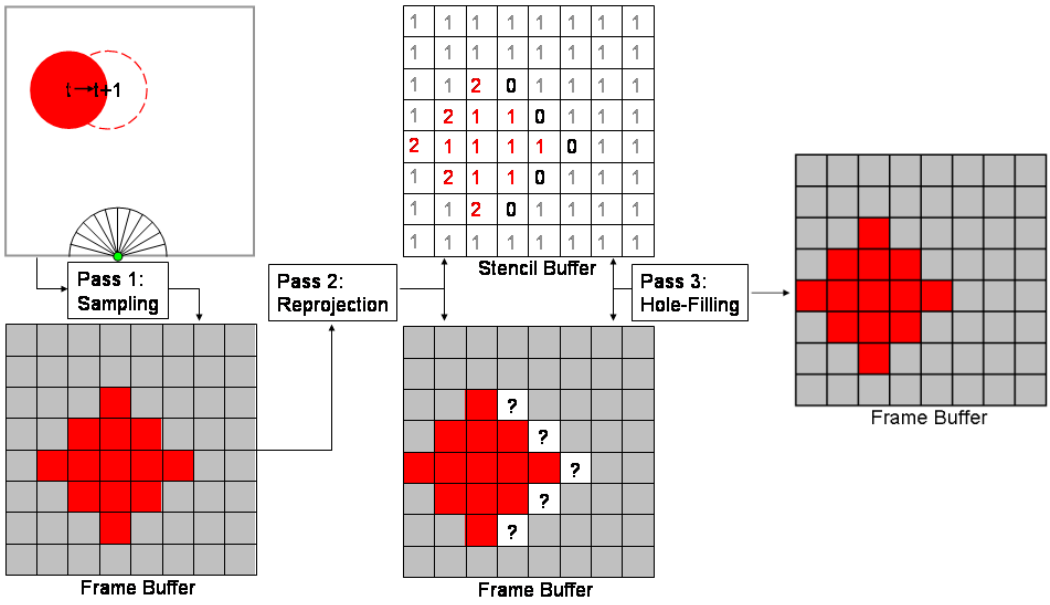


Figure 8.8: Reprojection using the GPU. The first pass samples the scene to gather the required information. Then, the visible points are reprojected to their estimated position at next time step. During this pass, each rendered fragment increments the stencil buffer. Finally, the holes (i.e. where the stencil value is 0) are filled using the deepest neighboring values.

This estimate is obtained in a second pass: each projected visible point generated in the first pass is considered as a vertex. This operation can be easily implemented using either the OpenGL Pixel Buffer Objects or OpenGL PBuffer rendering and per-vertex texture mapping. 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 8.6(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 also small. As described in Section 8.1.4, a third pass fills the holes using the local background (that is, the neighboring value with the highest depth). 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 is still 0. The final result of this algorithm is an estimate of the incoming radiance at time $t + 1$, generated entirely on the GPU (Figure 8.8). This estimate is used in the extrapolated temporal gradients and the temporal weighting function. As shown in Equation 8.6, this latter defines a maximum value of the lifespan of a given record, and triggers its recomputation. However, this recomputation is not always necessary.

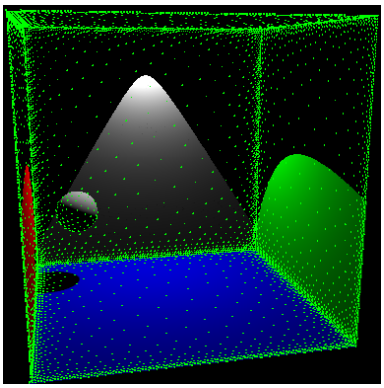
Culled Replacement/Deletion As described in the previous sections, the flickering artifacts of the lighting come from the temporal discontinuities of the accuracy. Therefore, if a record cannot contribute to the current image (i.e. out of the view frustum, or occluded), it can be simply deleted instead of being replaced by a novel, up-to-date record. This avoids the generation and update of a "trail" of records following dynamic objects (Figure 8.9), hence reducing the memory and computational costs. In the context of radiance cache splatting, 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. If a 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 appeared to be negligible (less than 0.1% of the overall computing time). Besides, in the following test scenes, this method reduces the storage and computational costs by up to 25-30%.

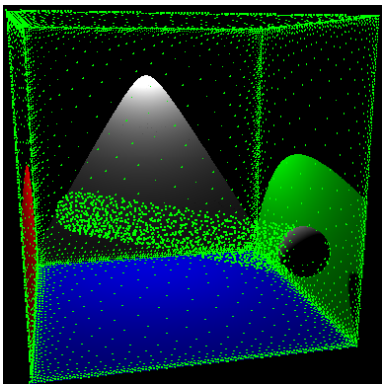
8.3 Results

This section discusses the results obtained using temporal radiance caching compared to the classical approach in which a new cache is computed for each frame. This 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 512MB. The scene details and timings are summarized in Table 8.2.

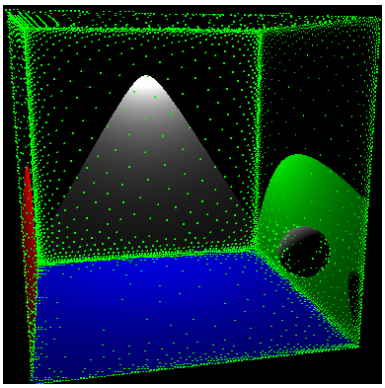
Cube in a Box This very simple, diffuse scene (Figure 8.12(a)) exhibits high flickering when no temporal gradients are used. Along with a significant speedup, our method reduces the flickering artifacts by using extrapolated temporal gradients. Such artifacts are unnoticeable with interpolated gradients. The animations are generated using a temporal accuracy parameter $a^t = 0.05$, and a maximum lifespan of 20 frames. The per-frame computation requires about 138 MB (772K records) to store all the irradiance records. In our method, the memory load is only 12.4 MB (50K records). Figure 8.10 shows the accuracy values obtained with and without temporal gradients. The remaining flickering of the extrapolated temporal gradients are due to the discontinuities of accuracy. Since our aim is high quality rendering, the following results focus on interpolated temporal gradients which avoid discontinuities.



(a) Time 1



(b) Time 100, systematic update



(c) Time 100, our record removal method

Figure 8.9: 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. When the sphere moves, new records are created to evaluate the incoming radiance on the sphere. If every record is permanently kept up-to-date, a "trail" of records lies on the path of the dynamic sphere (b). Using culling, only useful records are updated (c).

Moving Light A similar scene (Figure 8.12(b)) illustrates the behavior of our algorithm in the context of dynamic light sources. The bottom of the box is tiled to highlight the changes of indirect lighting. Due to the highly dynamic indirect lighting, the lifespan of the records is generally very short, yielding frequent updates of irradiance values. Compared to per-frame computation, our method renders the animation with higher quality in a comparable time.

Flying Kite In a more complicated, textured scene (Figure 8.12(c)), our algorithm also provides a drastic quality improvement while significantly reducing the computation time. In the beginning of the animation the change of indirect lighting is small, and hence 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. Using our temporal weighting function, the global illumination solution of this zone is updated at a fast pace, avoiding ghosts in the final image (Figure 8.11).

Japanese Interior In this complex scene (Figure 8.12(d)), the glossy and diffuse objects are rendered using respectively the radiance and irradiance caching algorithms. The animation illustrates the features of our method: dynamic nondiffuse environment, and important changes

of indirect lighting. In the beginning of the animation, the scene is lit by a single, dynamic light source. In this case, temporal gradients suppress the flickering artifacts present in per-frame computation, but 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 the indirect illumination. Our temporal gradients 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. During the generation of this animation, the average latency introduced by occlusion queries is 0.001% of the overall rendering time.

Spheres This scene features complex animation with 66 diffuse and glossy bouncing spheres and a glossy back wall (Figure 8.12(e)). Temporal radiance caching eliminates the flickering while reducing the computational cost of a factor 4.24. The temporal accuracy value is $a^t = 0.05$ and the maximum record lifespan $\delta_{t_{max}} = 5$.

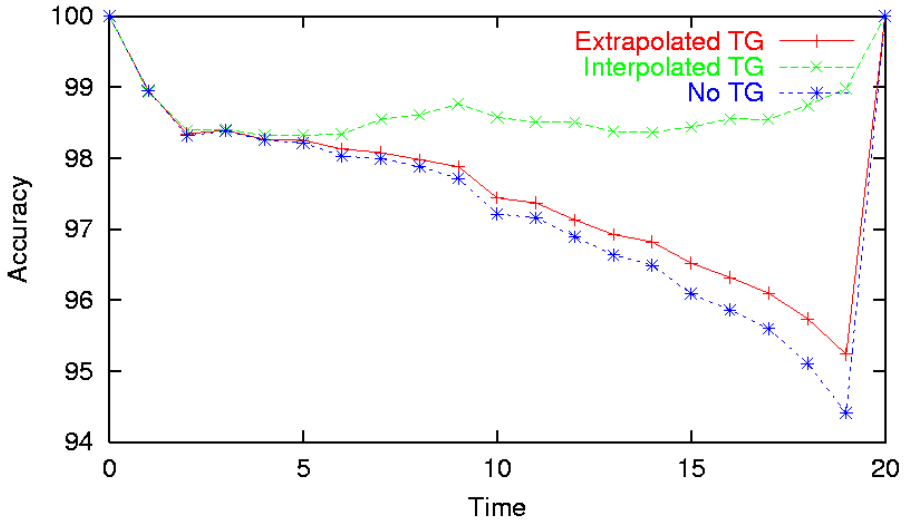
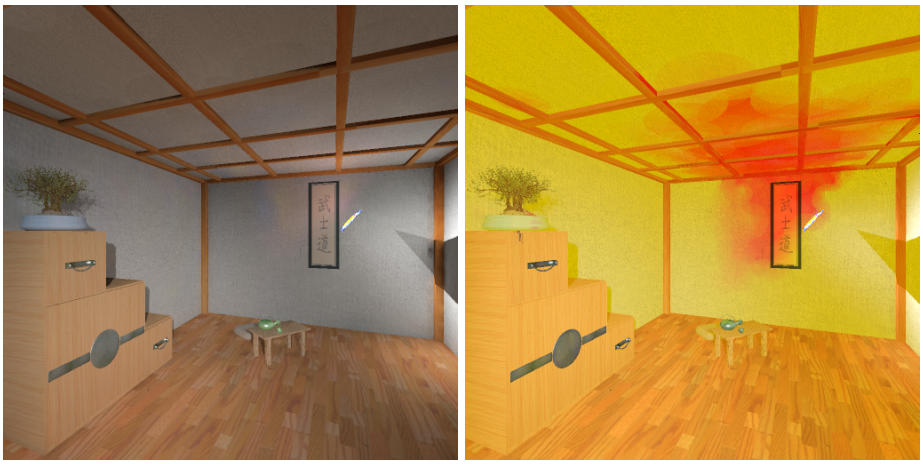


Figure 8.10: Temporal accuracy values obtained in scene Cube in a Box by creating records at time 0 and extrapolating their value until time 19. New records are recomputed at time 20. The temporal gradients (TG) provide a better approximation compared to the approach without those gradients. Using interpolated gradients, the accuracy is continuous and remains above 98%.

Scene	Nb. Poly	Nb. Frames	Per-Frame Comp.	Our Method	Speedup
Cube in a Box	24	400	2048s	269s	7.62
Moving Light	24	400	2518s	2650s	0.95
Flying Kite	28K	300	5109s	783s	6.52
Japanese Interior	200K	750	13737s	7152s	1.9
Spheres	64K	200	3189s	753s	4.24

Table 8.2: Test scenes and timings

Computational Overhead of Reprojection During the computation of a record, this method evaluates the value of the incoming lighting for both current and next time steps. As



(a) Actual sampling frame (b) Records lifespan

Figure 8.11: When computing the global illumination solution for the current frame (a), the temporal caching scheme estimates where the lighting changes. The lifespan of each generated record is computed by estimating the future change of lighting (b). Green and red colors respectively represent long and short lifespan.

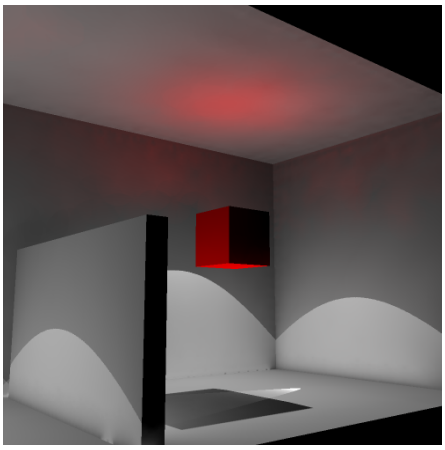
shown in Section 8.1.4, the estimation of the future incoming lighting is performed by simple reprojection. Therefore, the related computational overhead is independent of the scene geometry. In our tests, each record was computed at resolution 64×64. On our system, the reprojection is performed in approximately 0.46 ms. For comparison, the time required to compute the actual incoming lighting at a given point in our 200K polygons scene is 4.58 ms. In this case, the overhead due to the reprojection is only 10% of the cost of the actual hemisphere sampling. Even though this overhead is not negligible, this estimate reduces the overall rendering time by reusing the records in several frames.

8.4 Conclusion

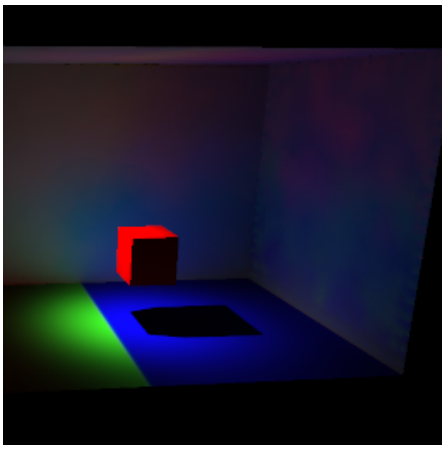
This chapter presented a novel method for exploiting temporal coherence in the context of irradiance caching. When used to render each frame of an animation, irradiance and radiance caching create the temporal discontinuities of the accuracy of the lighting, resulting in disturbing flickering artifacts. This approach is based on sparse sampling and interpolation of the indirect lighting in the temporal domain. The irradiance and radiance records are adaptively reused in several frames using our temporal weighting function and temporal gradients. The temporal weighting function determines the lifespan of each record depending on the change rate of the local incoming radiance: if strong changes of incoming radiance are detected, the corresponding lifespan is reduced to ensure a sufficient temporal sampling rate. At the end of the record lifespan, a new record is generated at the same location. This method reduces the flickering artifacts while allowing for the computation of interpolated temporal gradients for smooth temporal interpolation of the indirect lighting.

The evaluation of the temporal weighting function and of the extrapolated temporal gradients rely on an approximate knowledge of the future incoming radiance, which can be performed using GPU-accelerated reprojection. The GPU is also used to determine which records must be updated at the end of their respective lifespans.

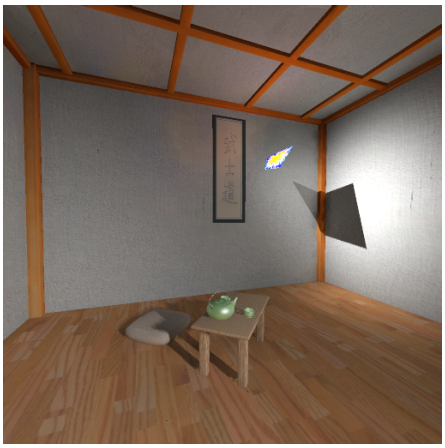
The results show both a significant speedup and an increased quality compared to per-frame computation. Due to sparse temporal sampling, the incoming radiance values for the entire animation segment can be stored within main memory.



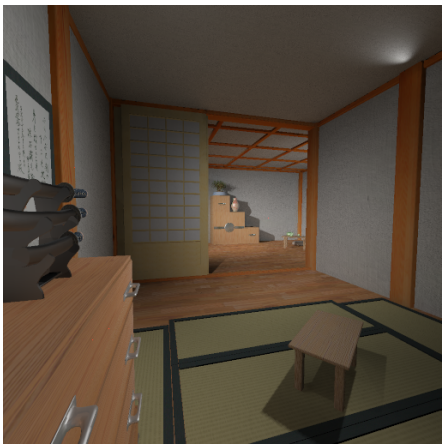
(a) Cube in a Box



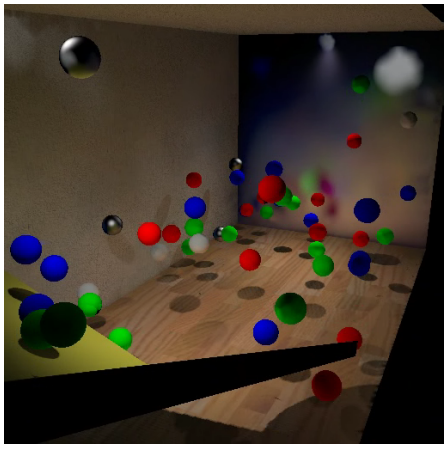
(b) Moving Light



(c) Flying Kite



(d) Japanese Interior



(e) Spheres

Figure 8.12: Images of scenes discussed in Section 8.3.

Bibliography

[GBP07] Pascal Gautron, Kadi Bouatouch, and Sumanta Pattanaik. Temporal radiance caching. *IEEE Transactions on Visualization and Computer Graphics*, 2007.

[TMS02] Takehiro Tawara, Karol Myszkowski, and Hans-Peter Seidel. Localizing the final gathering for dynamic scenes using the photon map. In *VMV*, 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 Heckbert. Irradiance Gradients. In *Proceedings of Eurographics Workshop on Rendering*, pages 85–98, Bristol, UK, May 1992.

[WRC88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A Ray Tracing Solution for Diffuse Interreflection. In *Proceedings of SIGGRAPH*, volume 22, pages 85–92, August 1988.