Problems & Solutions: Implementation details

Jaroslav Křivánek
Czech Technical University in Prague
This part of the course summarizes the common problems encountered when implementing irradiance caching. For each of the problems we describe one or more possible solutions to alleviate it. It should be stressed that this part describes the solutions used in my own implementation but slightly different solutions may be used in other implementations. Various tricks used at PDI/Dreamworks are described by Eric Tabellion.
Problems & Solutions: Implementation Details

• **Problem**: uneven interpolated irradiance
  – **Solution**: (1) two-pass image sampling, (2) bounded weighting function

• **Problem**: slow interpolation
  – **Solution**: (1) faster octree, (2) reuse last query result

• **Problem**: wrong gradients in scenes with glossy surfaces
  – **Solution**: ignore glossy term in indirect illumination sampling

• **Problem**: artifacts when caching used with motion blur
  – **Solution**: temporal re-projection

J. Křivánek
• **Problem:**
  small geometry missed/ignored

• **Solution:**
  neighbor clamping
A common problem in irradiance caching is that rays in the stochastic hemisphere sampling miss geometry features in the scene. This can produce clearly visible image artifacts, most often due to the exaggerated validity radius of some records.
Ideally, record spacing would be determined by the mean distance to the neighboring geometry. In practice, this distance is determined as the mean of the ray lengths in hemisphere sampling. If rays miss some geometry, the resulting mean distance is overestimated and we can see discontinuities in the resulting images due to interpolation.
A reliable way of detecting the over-estimated mean distance due to missing geometry in hemisphere sampling is based on two observations.

1. Because only few records usually suffer from the overestimated mean distance, we can use the distance estimate at other records to rectify the overestimated distance.

2. Distances obey the triangle inequality. If one record, $p_k$, is at the distance $d_k$ from some geometry feature, then another record, $p_j$, cannot be farther from this geometry feature than $d_k + ||p_j - p_k||$.

If we replace $d_k$ and $d_j$ by the mean distance $R_k$, $R_j$, respectively, we can detect suspicious cases by comparing $R_k$, $R_j$ to the distance between the two records to verify if the triangle inequality holds. Strictly speaking, this should only work if $R_k$, $R_j$, was the distance to the nearest geometry feature, but in practice this works fine even for mean distance. We call this heuristic 'neighbor clamping'.
Solution: Neighbor Clamping

- Upon addition of a new record, \( j \), in the cache:
  - for \( k \) in nearby records
    - \( R_j = \min\{ R_j, R_k + \| p_j - p_k \| \} \)

Clamp new record’s \( R \) by its neighbors’ \( R \)

- for \( k \) in nearby records
  - \( R_k = \min\{ R_k, R_j + \| p_j - p_k \| \} \)

Clamp neighbors’ \( R \) by the new record’s \( R \)

Here is how we proceed in practice. When a new record, \( j \), is added to the cache, we first locate all existing records whose area of influence overlap with the area of influence of the record being added. (That is to say, all records \( k \), such that \( \| p_j - p_k \| < R_j + R_k \). Then for all those records, we clamp the \( R_j \) value of the new record: \( R_j = \min\{ R_j, R_k + \| p_j - p_k \| \} \). This enforces the triangle inequality. After that, we use the clamped \( R_j \) value of the new record to clamp the \( R_k \) values of the nearby records. This second step enforces the transitivity of triangle inequality.
And voilà, the artifacts due to the over-estimated mean distance are gone.
With neighbor clamping, the spacing between the records is equalized. It falls off gradually as we move away from the geometry features.
In the Sponza atrium scene, the benefit of neighbor clamping is even more apparent.
Record spacing is equalized and indirect illumination is properly sampled around the cornices above the arches.

To conclude, irradiance caching produces image artifacts when the mean distance to geometry is overestimated. Neighbor clamping reliably detects and corrects the overestimated mean distance, thereby suppressing these artifacts.
• **Problem:**
  rays leaking through geometry

• **Solution:**
  neighbor clamping
Another serious problem of irradiance caching occurs when handling scenes with small cracks between polygons. Such scenes are quite common in practice – either because the scene is not well modeled or because of a limited numerical precision in the exported scene.

If a primary ray happens to hit such a crack, then most of the secondary rays used in hemisphere sampling “leak” through the crack. As a result, the irradiance estimate is completely wrong and, more seriously, the mean distance is greatly overestimated. (If there were no ray leaking, all those leaking rays would actually be very short.)
Consequences of Ray Leaking

- Wrong illumination estimate
- Extrapolated over large area

The incorrect irradiance estimate is then extrapolated over a large area.
A partial remedy is quite simple – just turning on neighbor clamping. Neighbor clamping detects and rectifies overestimated mean distance, so the wrong irradiance estimate is not extrapolated over such a large area. However, neighbor clamping cannot do anything about the wrong irradiance estimate itself.
• **Problem:**
  too many records in corners

• **Solution:**
  impose minimum record spacing
Remember that the spacing of irradiance records is given by the mean distance to the neighboring geometry (and also by the object curvature, which we disregard in this discussion). If you do not impose any minimum limit on the spacing, irradiance caching will spend most of the time generating too many records around edges and corners. To avoid this problem, it is a good idea to impose a minimum distance between the records. This can be done by setting some minimum threshold $R_{\text{min}}$ on the $R_i$ value of a record.
One possibility is to limit the spacing in world space, by fixing the threshold, $R_{\text{min}}$, to the same value all over the scene. In Radiance, $R_{\text{min}}$ is specified as a fraction of the scene size. This way of limiting the record spacing tends to generate too few radiance cache records near the camera and too many records far away.
A better idea, proposed by Tabellion and Lamorlette [2004], is to use a multiple of the projected pixel size for the threshold $R_{min}$. Good values for $R_{min}$ range between 1.5x and 3x the projected pixel size.
Especially for exterior scenes, it is also important to limit the maximum value of $R_i$. In *Radiance*, this maximum is 64 times the minimum. Tabellion and Lamorlette [2004] use the maximum of 10x the projected pixel size.
• **Question:**
  use mean or minimum ray length for record spacing?

• **Answer:**
  – minimum without neighbor clamping
  – mean with neighbor clamping
Minimum or Mean Distance for Record Spacing?

- Record spacing

- $R_i$...
  - Harmonic mean of ray lengths [Ward et al. 88]
  - Minimum ray length [Tabellion and Lamorlette 04]

In the irradiance caching implementation in Radiance [Ward et al. 1988], the record spacing is based upon the *mean* distance to neighboring geometry. This approach tends to miss some small geometry features. At EGSR in 2006 [Křivánek et al. 2006], we proposed neighbor clamping to resolve these problems. Tabellion and Lamorlette [2004] use the minimum distance instead of the mean distance to resolve these problems. Let us see how irradiance caching behaves when using the minimum and mean distance.
First, when there is no gradient limit on record spacing and neighbor clamping is not used, then the minimum distance indeed produces much better images than the mean distance.

(See Greg Ward’s slides on Radiance implementation for more information about the gradient limit on record spacing.)
Minimum or Mean?
no gradient limit, no neighbor clamping

MIN  a=0.5, #recs 14k
MEAN a=0.07, #recs 14k
If we limit record spacing by the translational gradient, we get much more records in the high-gradient areas around the cornices.

This is also a nice example of the gradient limit on record spacing.
But still, even with the gradient limited spacing, using the mean distance leaves some artifacts around the cornices.
However, when we turn on neighbor clamping, the artifacts are gone.
Looking at the record distribution, we see that with the mean distance, record spacing falls off gradually as we move away from the geometry, whereas with the minimum distance, records tend to concentrate in the corners.
Minimum or Mean – Conclusion

• Without neighbor clamping
  – minimum is better
  – mean can miss sources of indirect light

• With neighbor clamping
  – mean is better
  – small sources if indirect reliably detected for both
  – minimum suffers from record clumping in corners

In our experience, the gradual falloff of the spacing is desirable. In our practice, when gradient limit on record spacing and neighbor clamping is used, then the mean distance usually produces better images with the same number of records than the minimum distance.
• **Problem**: translational gradients explode in corners

• **Solution**: clamp translational gradient magnitude
A common problem in irradiance caching implementation are the dark and bright patches in corner areas due to the exaggerated gradient magnitude. This happens because of the division by the ray length in gradient computation, which happens to be is very tiny in the corners.
Similarly to other IC implementations, I limit the gradient magnitude in the corner areas, although the heuristic I use slightly differs from the Greg Ward’s Radiance implementation or Eric Tabellions’s PDI/Dreamworks implementation. Mine is quite straightforward:

```plaintext
if ( R < R' )
   \text{grad}_t *= \frac{R}{R'}
```

Here:

- \( R \) is the minimum/mean ray length computed by the hemisphere sampling, possibly reduced by the gradient magnitude.
- \( R' \) is the value of \( R \) after clamping between the minimum and maximum spacing thresholds, \( R' = \max \{ \min \{ R, R_{\text{max}} \}, R_{\text{min}} \} \)
- \( \text{grad}_t \) is the translational gradient (one 3D vector per R,G,B)
This slide summarizes the actions taken to add a new irradiance record into the cache. First, we sample the hemisphere by casting a number of secondary rays. This gives the irradiance estimate, the translational and the rotational gradients, and the estimate of the mean (or minimum) distance to the neighboring geometry, $R$. We then limit the value of $R$ by the translational gradient as follows: $R = \min \{ R, 1/\|\text{grad}_t\| \}$. After that we clamp $R$ by the minimum and maximum threshold (determined from the projected pixel size). This produces the clamped value $R' = \max \{ \min \{ R, R_{\max} \}, R_{\min} \}$. If $R$ was increased by this clamping (i.e. if $R$ was originally less than $R_{\min}$), we decrease the gradient magnitude accordingly, in order to avoid negative values in extrapolation.

The next step is neighbor clamping. For its correct functionality, it is essential to use the original, unclamped value of $R$ (not $R'$) in the neighbor clamping procedure and redo the clamping every time $R$ is reduced. It also means that each record must store the value of both $R$ and $R'$ and also $R_{\max}, R_{\min}$ used for the clamping.

Finally, we insert the new record into the cache.
• **Problem:**
  uneven interpolated irradiance

• **Solution:**
  (1) two-pass image sampling
  (2) bounded weighting function
Lazy evaluation of irradiance values is a great feature of irradiance caching that makes the algorithm very flexible. However, if not used carefully, it has a negative impact on the image quality. This slide shows the kind of artifacts you may expect when generating image pixels in the scanline order, adding new irradiance values lazily as needed.
Using hierarchical image traversal instead of the scanline order improves the image quality (and, actually, decreases the number of records needed to cover the whole image), but image artifacts still remain.
In our experience, the best solution is a two pass traversal of the image. In the first pass, the irradiance cache is filled so that all pixels are covered, but no image is generated. In the second pass, arbitrary pixel traversal can be used to generate the image.
The order of pixel traversal in the first pass has an impact both on the image quality and also on the distribution of records in the scene and, consequently, on the number of records required to cover the whole image. A good solution is to use a best-candidate pattern pixel traversal. It generates the best image quality with the fewest records.
Another way to achieve smooth indirect illumination is to increase the area over which a record can be reused by increasing the ‘a’ value in the second pass.
### Increasing ‘a’ for the Second Pass

<table>
<thead>
<tr>
<th>Pass</th>
<th>Rays per Hemi</th>
<th>a Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1(^{st}) pass</td>
<td>400</td>
<td>0.25</td>
</tr>
<tr>
<td>2(^{nd}) pass</td>
<td></td>
<td>0.35 (1.4x)</td>
</tr>
</tbody>
</table>
Using a very large value of 'a' in the second pass makes it possible to obtain relatively smooth indirect illumination even for a small number of rays in hemisphere sampling (here 400 per hemisphere), at the cost of blurring some of its details. However, blurring is rarely a problem since indirect illumination is mostly smooth by its nature anyhow.
To achieve the desired smoothing effect when increasing the `a’ value, we have to pay attention to using a suitable weighting function for interpolation.
The original weighting function proposed by Ward et al. [1988] has two undesirable properties. First, it goes to infinity when the distance between the point of interpolation $p$ and the location of a record, $p_i$, goes to zero. Second, there is a discontinuity at the border of the influence area of a record. The discontinuity tends to produce some visible seams in the images. One approach to suppressing the seams, used in Radiance, is to randomize the acceptance of a record for interpolation. In our experience, a better way is to make the weight function zero at the border of the influence area – two possibilities are shown on the slide.
Even when the weight is zero at the border, it may still be virtually impossible to get smooth indirect illumination. The reason is the high value of the weight at the record location.
Using a weighting function that is bounded at the record location, such as the one proposed by Tabellion and Lamorlette [2004], the interpolated illumination becomes much smoother.
Tens of millions of irradiance cache queries are used to generate a production-quality image. Therefore, it is important to keep the interpolation time as low as possible. We use two techniques to improve the interpolation speed: faster octree for indexing the records and reuse of last query result.
The octree structure proposed for irradiance caching by Ward et al. [1988] (used in the Radiance implementation) references one record from exactly one node. As a consequence, searching for all records contributing to interpolation at a given location involves traversing several path in the tree, which means recursion. This makes the tree lookup relatively slow. A better option is to reference each record from all octree nodes that the record may overlap. The tree lookup then reduces to a simple traversal form the root to the leaf. A simple implementation of this improved octree is a part of PBRT [Pharr and Humphreys 2004]. The price to pay for the faster lookup is a slightly increased memory consumption and slower insertion.
Our second trick to amortize the cost of irradiance cache lookups only applies to ray tracing-based renderers (unlike REYES & comp.). When generating a production-quality image with ray tracing, 16x or higher pixel super sampling is quite common in practice. However, the color returned by each irradiance cache query within the same pixel is usually virtually identical. We exploit this observation by retaining the result of the IC query and returning the retained value for all other queries within the same pixel (subject to some further checks as shown on the slide).
• **Problem:**
  wrong gradients in scenes with glossy surfaces

• **Solution:**
  ignore glossy term in indirect illumination sampling

When rendering scenes with some glossy or specular surfaces, the results of the translational gradient computation are often incorrect, resulting in artifacts in the image. The most common solution to this problem is to ignore the glossy term for shader evaluations launched by the rays generated by hemisphere sampling. Doing so ignores a part of indirect illumination but most often this is preferable to visible image artifacts.
In many scanline renderers, such as Pixar’s PRMan, shading is computed before resolving visibility and motion blur. It means that irradiance caching is not affected by the presence of the motion blur in the scene and works fine with motion blurred sequences.

However, for ray tracing-based renderers, the situation is more complicated. Each generated primary ray is assigned a randomly generated time. When tracing this ray, transformation / deformation of animated geometry used for intersection testing is adjusted to that time. If an irradiance cache record is generated for some time value, its irradiance will be incorrect for any other time during the frame. Moreover, lots of records are created because of the movement of the animated geometry.

The workaround for this problem is to reproject any intersection position to time t=0 before the irradiance cache query. This is somewhat equivalent to what happens in the PRMan.