Fast HDR Image-Based Lighting Using Summed-Area Tables

Justin Hensley[1], Thorsten Scheuermann[2], Montek Singh[1] and Anselmo Lastra[1]

[1]University of North Carolina, Chapel Hill, NC, USA — {hensley, montek, lastra}@cs.unc.edu
[2]ATI Research, Marlborough, MA, USA — thorsten@ati.com

Abstract

We introduce a technique that uses summed-area tables to dynamically compute image-based lighting from high-dynamic-range environment maps. A combination of first order and second order summed-area tables are used to approximate various BRDFs such as the Phong BRDF. A key feature of the approach is "offsetting" of the original image with respect to its mean, in order to significantly increase the precision of our computation. On current graphics hardware, we are able to render at over 40 frames per second while still re-computing all necessary summed-area tables on the fly to allow for dynamic lighting. When the environment map is static, we are able to render at over 130 frames per second, but still allow for dynamic, spatially varying filtering of the static environment map.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism: Color, shading, shadowing, and texture I.4.3 [Image Processing and Computer Vision]: Enhancement: Filtering

1. Introduction

There are many applications in computer graphics where spatially varying filters are useful. One example is the rendering of glossy reflections. Unlike perfectly reflective materials, which only require a single radiance sample in the direction of the reflection vector, glossy materials require integration over a solid angle. Blurring by filtering the reflected image with a support dependent on the surface's BRDF can approximate this effect. This is currently done by pre-filtering off line, which limits the technique to static environments.

In this paper we present a method to rapidly generate higher-order summed-area tables —e.g., summed-area table of a summed-area table— that is efficient enough to allow multiple tables to be generated every frame while maintaining interactive frame rates. We extend the technique presented by Hensley et. al. [HSC*05] to approximate reflections generated using a Phong BRDF and high dynamic range environment maps.

Summed-area tables, originally introduced by Crow [Cro84], were proposed as a technique to enable more general texture filtering than was possible with mip maps. Once generated, a summed-area table provides a means to evaluate a spatially varying box filter in a constant number of texture reads. Heckbert [Hec86] extended Crow's work to handle more complex filter functions, such as Bartlett and cubic filter kernels. Finally, Hensley et al. [HSC*05] introduced an approach to dramatically improve computational precision of summed-area tables. This paper builds upon their approach.

The paper is organized as follows. Section 2 provides background information on summed-area tables. Next, in Section 3 we present our technique for rendering glossy reflections. Section 4 presents our results. Finally, Section 5 discusses future work.

2. Background

Originally introduced by Crow [Cro84] as an alternative to mip maps, a summed-area table is an array in which each entry holds the sum of the pixel values between the sample location and the bottom left corner of the corresponding input image, i.e., the integral of the pixel values from the image origin to the current point.

Summed-area tables enable the rapid calculation of the sum of the pixel values in an arbitrarily sized, axis-aligned rectangle at a fixed computational cost. Figure 2 illustrates

Figure 1: *The left image shows the Hebe model shaded with a Phong BRDF approximated by filtering a high dynamic range environment map in real time using summed-area tables. The right image shows the Hebe model shaded with an approximation of a diffuse BRDF using a second-order summed-area table. Both models rendered using the Grace Cathedral light probe.*

how a summed-area table is used to compute the sum of the values of pixels spanning a rectangular region. To find the integral of the values in the dark rectangle, we begin with the pre-computed integral from (0,0) to (x_R, y_T). From this value, we subtract the integrals of the rectangles (0, 0) to (x_R, y_B), and (0, 0) to (x_L, y_T). The integral of the hatched box is then added to compensate for having been subtracted twice.

The average value of a group of pixels can be calculated by dividing their sum by the area they cover. Crow's technique amounts to convolution of an input image with a box filter. Its power lies in the fact that the filter support can be varied at a per pixel level without increasing the cost of the computation. Unfortunately, since the value of the sums (and thus the dynamic range) can get quite large, the table entries require extended precision. The number of bits of precision needed per component is

$$P_s = log_2(w) + log_2(h) + P_i$$

where w and h are the width and height of the input image. P_s is the precision required to hold values in the summed-area table, and P_i is the number of bits of precision of the input. Thus, a 256x256 texture with 8-bit components would require a summed-area table with 24 bits of storage per component.

Another limitation of Crow's summed-area table technique is that it is only capable of implementing a simple box filter. This is because only the sum of the input pixels is

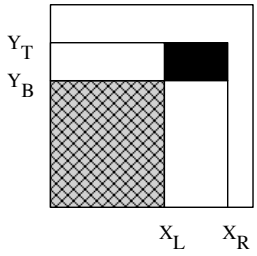


Figure 2: *(after [Crow84]) An entry in the summed-area table holds the sum of the values from the lower left corner of the image to the current location. To compute the sum of the dark rectangular region, evaluate $T[X_R,Y_T] - T[X_R,Y_B] - T[X_L,Y_T] + T[X_L,Y_B]$ where T is the value of the entry at (x, y)*

stored; therefore it is not possible to directly apply a generic filter by weighting the inputs.

In [Hec86], Heckbert extended the theory behind summed-area tables to handle more complex filter functions. Heckbert made two key observations. The first is that a summed-area table can be viewed as the integral of the input image, and the second that the sample function introduced by Crow was the same as the derivative of the box filter function. By taking advantage of those observations and the following convolution identity

$$f \otimes g = f'^n \otimes \int^n g$$

it is possible to extend summed-area tables to handle higher order filter functions, such as the Bartlett filter, or even a Catmull-Rom spline filter. The process is essentially one of repeated box filtering. Higher order filters approach a Gaussian, and exhibit fewer artifacts.

For instance, Bartlett filtering requires taking the second-order box filter, and weighting it with the following coefficients:

$$f = \begin{matrix} 1 & -2 & -1 \\ -2 & 4 & -2 \\ 1 & -2 & -1 \end{matrix}$$

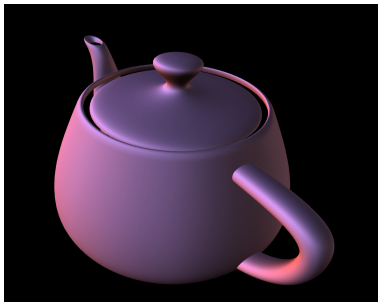
In general, the precision requirements of Heckbert's method can be determined as follows:

$$P_s = n * (log_2(w) + log_2(h)) + P_i$$

where w and h are the width and height of the input texture, n is the degree of the filter function, P_i is the input image's precision, and P_s is the required precision of the n^{th}-degree summed-area table.

Hensley et al. [HSC*05] presented a technique to mitigate the error due to a loss of precision called *offset* summed-area tables. In particular, by offsetting the input image with respect to its mean pixel value, the summed-area table is no longer monotonic, which dramatically reduces its peak

(a) Box filter



(b) Bartlett filter

Figure 3: *Image (a) shows an approximation to a diffuse BRDF using a simple box filter. Image (b) shows an approximation using a Bartlett filter, which clearly eliminates the banding artifacts seen in Image (a).*

value, thereby mitigating any precision loss. Furthermore, offsetting the image by its mean allows them to take advantage of the sign bit of the floating-point representation, thereby gaining another bit of precision. A final contribution of their paper is a method based on recursive doubling, to rapidly generate floating-point offset summed-area tables using graphics hardware.

A technique introduced by [AG02, YP03] combines multiple samples from different levels of a mip map to approximate filtering. This technique suffers from several problems. First, a small step in the neighborhood around a pixel does not necessarily introduce new data to the filter; it only changes the weights of the input values. Second, when the inputs do change, a large amount of data changes at the same time, due to the mip map, which causes noticeable artifacts. In [Dem04], the authors added noise in an attempt to make the artifacts less noticeable; the visual quality of the resulting images was noticeably reduced.

3. Implementation

By using higher-order offset summed-area tables, it is possible to filter a dual-paraboloid high-dynamic range environment map in constant time, irrespective of the filter size.. Figure 1 shows the image of a model rendered with an approximate Phong BRDF computed dynamically using offset summed-area tables. The diffuse component is approximated by sampling a second order summed-area table with a large filter kernel in the direction of the normal, and the specular component is approximated by sampling a first order summed-area table with a small filter kernel in the direction of the reflection direction. The second order filter is used for the diffuse component to prevent noticeable artifacts from box filtering, shown in Figure 3. The first order filter is used for the specular component to increase performance, since it only requires four texture reads, and to limit the effect of precision loss due to the use of single precision floating point textures.

For generating summed-area tables on the GPU, we apply the technique described in [HSC*05]. The rendering steps necessary for fully dynamic image-based lighting are:

- Update the dual-paraboloid map faces with the current lighting environment.
- For each dual-paraboloid map face
  - compute the offset summed-area table SAT^1
  - compute the 2nd-order summed-area table SAT^2
- Render the object using the SAT^1 and SAT^2 textures for both dual-paraboloid faces

3.1. Practical Issues in the Computation of Higher-Order SATs

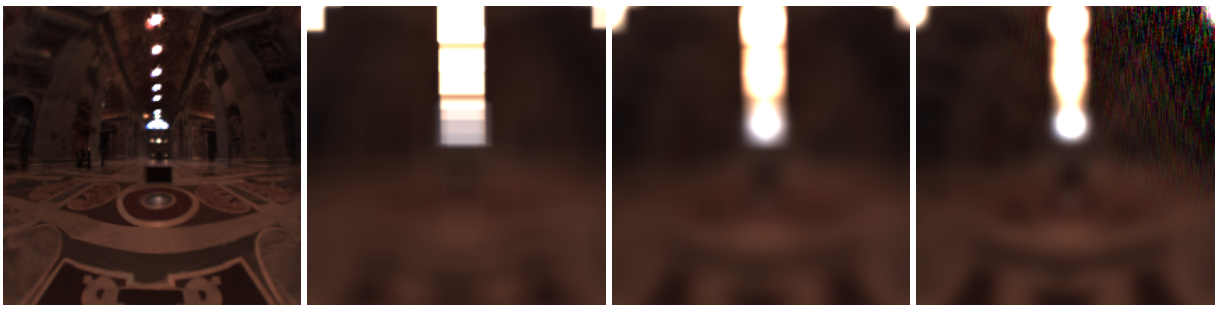
An essential portion of our algorithm requires filtering large pixel regions, which increases the chance that a filter kernel will extend beyond the boundary of the image being filtered. Traditional filtering approaches handle such a situation by effectively padding the image with values at its boundary pixels.

There is a simple optimization that traditional approaches use, which is no longer applicable for computing higher-order summed-area tables. Traditionally, padding by extending boundary pixel values is actually achieved by remapping read accesses to the padded region back to the boundary pixels. Thus, no actual padding is done, but its effect is simulated by "clamping" the coordinates to the boundary. However, for computing higher order summed-area tables, one cannot use clamping on the first-order summed-area table. In particular, the correct padding is not simply a replication of its boundary values; it is actually the integral of the underlying padded image.

Therefore, for filtering from a second-order summed-area table, instead of clamping, our approach is to actually pad the original image, and then do all computations on the padded image. This technique ensures that all higher-order summed-area tables are computed accurately.

4. Results

Figure 4 shows the result of filtering an image with a first, second, and third order offset summed-area table. Figure



(a) original image (b) offset SAT^1 (box) (c) offset SAT^2 (Bartlett) (d) offset SAT^3 (cubic)

Figure 4: *Comparison of images filtered using repeated offset summed-area tables. Image (a) shows the original image, half of a dual-paraboloid map computed from the St. Peter's Basilica light probe [Deb98]. Image (b) is filtered with a box filter using a first order offset summed-area table. Image (c) is filtered with a Bartlett filter using a second order offset summed-area table. Image (d) is filtered with a cubic filter using a third order offset summed-area table. The error in upper right corner of image (d) is the result of a loss of precision.*

4(a) shows the original image, which is one half of a dual-paraboloid map computed from the St. Peter's Basilica light probe [Deb98]. Figure 4(b) shows the results of filtering the image with a box filter using a first order offset summed area table. Figure 4(c) shows the results of filtering the image with a Bartlett filter using a second order offset summed area table, and Figure 4(d) shows the results of filtering the original image with a cubic filter using a third order offset summed area table. The noise in upper right-hand corner of the image is due to precision loss, since a single-precision floating-poing number is not capable of directly storing a third order summed-area table. For very small filter sizes even the second order summed area table is unable to reconstruct an image without noise. For larger filter sizes, the error is averaged out. Note that this noise is significantly better than what would be produced if offsetting was not used in summed-area table calculation.

As can be seen from the images in Figure 4, the box filter produces noticeable artifacts which are exacerbated by the high dynamic range image. As the bright regions of the image enter the filter kernel they saturate it, making the clearly defined "boxes" in the resulting image. The cubic filter smoothly handles the bright portions of the input image, but introduces noise in the output image since a single-precision floating point texture cannot adequately store a third order offset summed area table. Additionally, the cubic filter requires sixteen reads to compute each output pixel, irrespective of filter size. The Bartlett filter provides a reasonable compromise between the box and cubic filters, and only requires nine reads to reconstruct each output pixel.

Using a second-order summed-area table, along with the image-padding technique of Section 3.1, we are able to render the image shown in Figure 1 at 40 frames per second on a 3.2 GHz Pentium 4 with an ATI Radeon X1900XT graphics card. Even though the summed area tables are not changing each frame, they are recomputed on the fly to emulate a dynamic high-dynamic range environment map. The rendering speed increases to 130 frames per second if we do not recompute the summed-area tables every frame.

5. Conclusion

In this paper, we have presented a technique to approximate a Phong BRDF using high-dynamic range environment maps. The technique is efficient enough to be re-computed on the fly every frame. Our future work involves improving the precision requirements of offset summed-area tables by extending the offsets to non-constant values, and taking advantage of better branching support in modern shader architectures to reduce the need to sample both the front and back dual-paraboloid maps for all fragments.

6. Acknowledgments

Financial support was provided by an ATI Research fellowship, the National Science Foundation under grants CCF-0306478, and CCF-0205425. Equipment was provided by NSF grant CNS-0303590.

References

[AG02] ASHIKHMIN M., GHOSH A.: Simple blurry reflections with environment maps. *J. Graph. Tools 7*, 4 (2002), 3–8.

[Cro84] CROW F. C.: Summed-area tables for texture mapping. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1984), ACM Press, pp. 207–212.

[Deb98] DEBEVEC P.: Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of SIGGRAPH 98* (July 1998), pp. 189–198.

[Dem04] DEMERS J.: *GPU Gems*. Addison Wesley, 2004, ch. "Depth of Field: A Survey of Techniques", pp. 375–390.

[Hec86] HECKBERT P. S.: Filtering by repeated integration. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), ACM Press, pp. 315–321.

[HSC*05] HENSLEY J., SCHEUERMANN T., COOMBE G., SINGH M., LASTRA A.: Fast summed-area table generation and its applications. In *Computer Graphics Forum* (2005), vol. 24, pp. 547–555.

[YP03] YANG R., POLLEFEYS M.: Multi-resolution real-time stereo on commodity graphics hardware. In *In Conference on Computer Vision and Pattern Recognition* (2003).