Statistical Acceleration for Animated Global Illumination

Mark Meyer John Anderson
Pixar Animation Studios

Figure 1: *An example of our statistical acceleration technique on a production shot: Our technique accelerates rendering by quickly computing noisy, low sample indirect illumination and using the variation over time to statistically denoise it. The left image shows a zoomed view of the shot using the noisy, unfiltered indirect illumination while the middle image uses the statistically filtered indirect illumination (contrast has been enhanced slightly to better illustrate the noise in print). The rightmost image shows the final comped frame using the statistically filtered indirect illumination. See the video for the full animation sequence.*

Abstract

Global illumination provides important visual cues to an animation, however its computational expense limits its use in practice. In this paper, we present an easy to implement technique for accelerating the computation of indirect illumination for an animated sequence using stochastic ray tracing. We begin by computing a quick but noisy solution using a small number of sample rays at each sample location. The variation of these noisy solutions over time is then used to create a smooth basis. Finally, the noisy solutions are projected onto the smooth basis to produce the final solution. The resulting animation has greatly reduced spatial and temporal noise, and a computational cost roughly equivalent to the noisy, low sample computation.

CR Categories: I.3.7 [Computing Methodologies]: Three-Dimensional Graphics and Realism

Keywords: Global Illumination, Rendering, Denoising, Filtering

1 Introduction

Global illumination effects provide important visual cues and have proved to be extremely important for many fields including computer animation, visual effects and computer games. However, simulating global illumination is a complex and computationally expensive task.

One of the most general techniques for computing global illumination is to use Monte Carlo integration (using stochastic ray tracing - or one of its variants) to solve the *Rending Equation* [Kajiya 1986]:

$$L_r(\mathbf{x},\omega_r) = L_e(\mathbf{x},\omega_r) + \int_\Omega f_r(\omega_r,\mathbf{x},\omega_i)\, L_i(\mathbf{x},\omega_i)\, cos\ \theta\ \delta\omega_i \quad (1)$$

Solving this equation is computationally expensive as it requires a large number of sample rays to reduce the noise to an acceptable level. This expense is exacerbated when rendering animated global illumination since the number of sample rays must be increased to capture the animated illumination as well as to reduce the temporal noise. In fact, typical errors and noise levels that are acceptable in static global illumination images often produce unpleasant flickering, shimmering and popping in animations when they are not coherent in time.

Since global illumination is so important, it should come as no surprise that it has attracted significant attention from researchers. We will only list the most relevant works and refer the reader to one of the many excellent survey papers (such as [Damez et al. 2003]) for a more in depth review.

Most techniques for computing global illumination attempt to reduce both the computational overhead as well as the noise by interpolating from a sparse (either in space, time, or both) set of samples. Ward *et al.*'s *irradiance caching* technique [Ward et al. 1988; Ward and Heckbert 1992], for instance, creates a cache containing all of the irradiance samples computed while integrating the global illumination for a frame. When a new irradiance sample is needed, the density of nearby samples in the cache is used to determine if the irradiance can be interpolated from the cache samples or must be computed from scratch. This interpolation both reduces the number

of times that the expensive illumination integral must be computed as well as reduces the noise. *Irradiance Filtering* [Kontkanen et al. 2004], on the other hand, computes a sparse set of low sample irradiance values and then uses a spatially varying filter to reduce the noise and smooth the results.

The predominant technique for computing global illumination in both academia and industry is the photon mapping technique of Jensen [Jensen 1996]. This method is efficient, even in complex environments, and allows for the simulation of all possible light paths. In the first pass, photons are emitted from the light sources and traced through the scene. As the photons hit surfaces, they are stored in a k-d tree, thus creating the photon map. In the second pass, the image is rendered using a Monte Carlo ray tracer in conjunction with the photon map. The photon map is sampled using a nearest neighbor density estimation technique when fast approximate radiance computations are required by the ray tracer. Using this technique, the noise and number of radiance samples is greatly reduced, resulting in impressive images and computational efficiency. Extensions [Cammarano and Jensen 2002] also allow time varying photon maps to be used and correctly handle motion blur.

It should be noted that the computation times for photon mapping are often dominated by a stochastic ray tracing process known as *final gathering*. Our technique is complimentary to photon mapping and can be used to accelerate final gathering. In fact, all of the indirect illumination examples in this paper were computed using final gathering from a photon map-like data structure.

Nimeroff *et al.*[Nimeroff et al. 1996] uses a range-image based framework in which the indirect illumination is sampled sparsely in time and interpolated. The keyframes at which the indirect illumination is computed are found by recursively subdividing the timeline. The indirect illumination is computed at the key time steps and the solutions for consecutive keyframes are compared. If a large percentage of the vertices contain differences greater than a threshold, the time sequence is subdivided and the process repeats. The temporal interpolation of the indirect illumination is possible due to the observation that time, like space, often has slow, smooth indirect illumination changes. Interpolating through time takes advantage of this smoothness both reducing the computation required as well as avoiding flicking and popping effects. However, the accuracy of the solution varies with the distance to the keyframes and improper keyframe placement can cause global illumination effects to be missed.

Myskowski *et al.*[Volevich et al. 2000; Myszkowski et al. 1999; Myszkowski et al. 2001], on the other hand, sparsely samples the indirect lighting on every frame - reducing the chances of missing an illumination event. The samples from preceding and following frames are used to reduce the noise to an acceptable level. The number of frames to collect these samples from is controlled by statistics such as the photon density on each mesh element. Again, reusing these samples over several frames greatly improves the computation efficiency as well as reduces the temporal aliasing.

Like previous work, our technique accelerates the indirect illumination computations using the correlation of the illumination samples to reduce both the number of samples required as well as the noise in the resulting solution. Unlike previous work however, we achieve this speed up by computing the indirect illumination for the entire sequence using a small number of sample rays when computing the integral in equation 1. This produces a noisy indirect illumination sequence. This sequence is then statistically analyzed to produce a smooth indirect illumination basis which captures the structure in the sequence. Finally, the noisy sequence is projected onto the smooth basis producing a smooth indirect illumination sequence whose computational cost is roughly the same as the initial noisy computation. The following sections will explain these steps in more detail.

2 Statistical Filtering

In this section, we describe our technique for rendering an animation with global illumination. We will focus on computing the indirect illumination as its cost dominates most scenes containing global illumination.

Let us for the moment assume that the camera and all objects in the scene are not moving - therefore, the only changes to the illumination are a result of lighting changes or surface material changes. While this seems like a harsh restriction, it will allow us to describe the technique working solely in image space - and it is not a limitation of our algorithm. This restriction to static cameras and objects will be removed in section 3 and is used here only for ease of exposition.

2.1 Overview

We wish to compute an animated sequence of indirect illumination images quickly. One simple way of doing this is to reduce the number of ray samples used to integrate equation 1. However, as mentioned previously, this will produce noisy images that flicker when played in an animation. A key observation is that although the individual pixels are noisy, correlation in the temporal domain still provides us with important information. For instance, if the illumination were static over the animation, we could simply average the pixel value over time to get a more accurate value for the pixel. This is similar to the technique of negative stacking used by astronomers. By taking 2 or more original negatives of the same object and stacking them, the signal in the resulting image is increased while the noise, being independent, actually cancels out and is reduced.

A slightly more complex case would be when the lighting scales linearly over time. Here we could simply find the linear lighting animation that best fits the noisy images and use this as our final animation. This suggests a more general approach. If we have a basis for the illumination in the animation, we can simply project the noisy animation onto this basis to produce our final, smooth animation. The next step is to choose such a basis.

2.2 Choosing the basis

Given our noisy image sequence $\hat{\mathbf{I}}(\mathbf{x},t)$, where $\mathbf{x}$ is the pixel location and t is time, we use *principle component analysis* (PCA) to represent the noisy animation sequence:

$$\hat{\mathbf{I}}(\mathbf{x},t) = \sum_{i=1}^{N} w_i(t)\, \mathbf{B}_i(\mathbf{x}) \tag{2}$$

where N is the number of images in the noisy sequence, and $\mathbf{B}_i(\mathbf{x})$ are the basis functions computed by PCA. An example of the PCA basis functions (or modes) for a noisy image sequence is shown in figure 2(middle). Notice how the noise in the basis images increases as we increase the basis number. This is due to the fact that in a PCA constructed basis, the low numbered functions capture the slowly varying components of the signal. Since the indirect

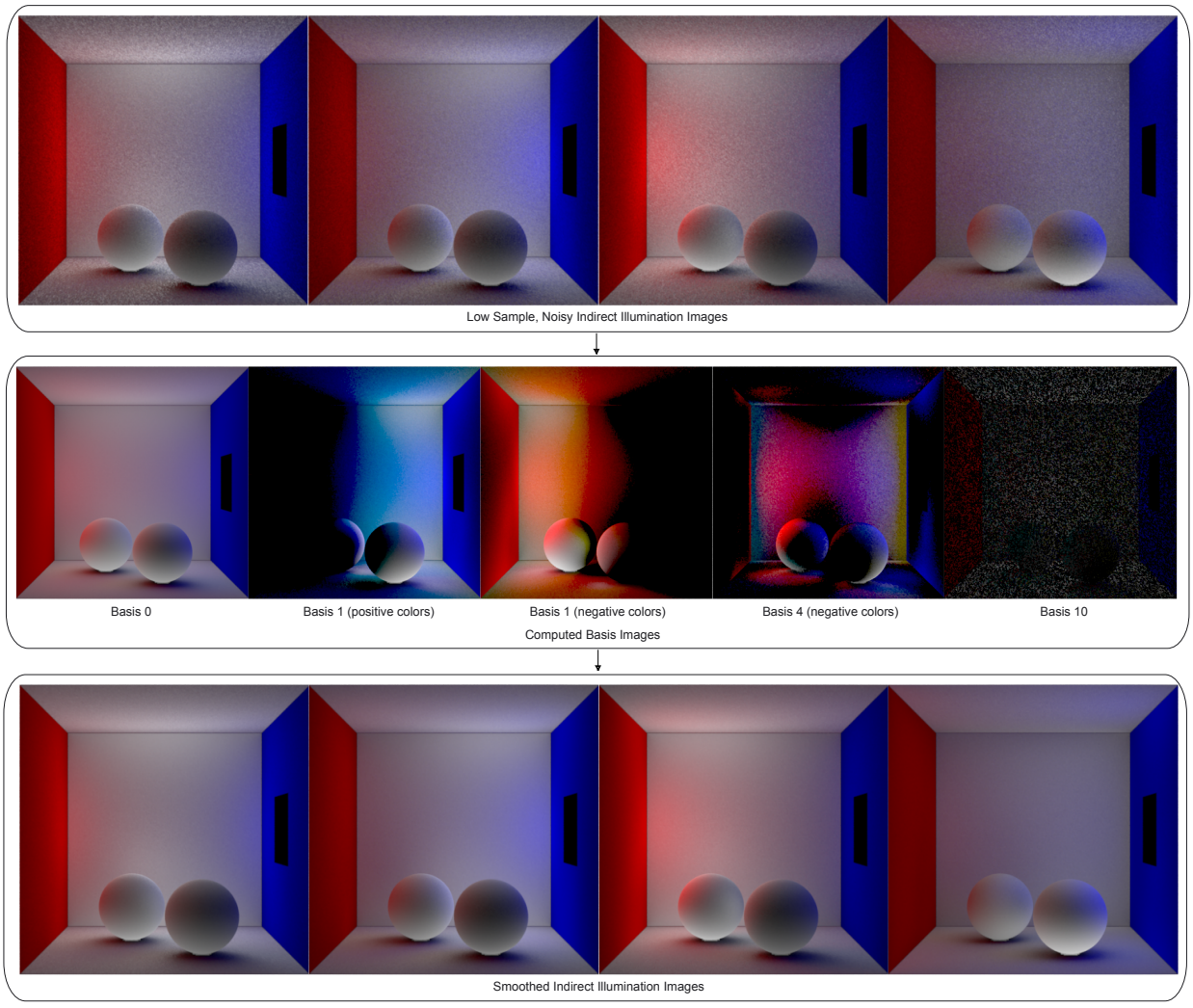

Figure 2: *Overview of our global illumination acceleration algorithm: (top) The indirect illumination for each frame of the animation is computed using a small number of ray samples (32) resulting in a sequence of noisy indirect illumination images. (middle) A basis is computed from this noisy sequence using principle component analysis. Notice how the higher numbered basis functions contain more of the sequence's noise. A truncation of the basis is computed to remove the higher numbered basis functions that reconstruct the noise. (bottom) The noisy images are projected onto the truncated basis (using 5 functions) resulting in a smooth sequence of indirect illumination images.*

illumination varies slowly compared to the noise, most of the indirect illumination is contained in the lower numbered basis functions while the higher numbered basis functions are mainly needed to reconstruct the noise of the original image sequence. This suggests that if we choose a subset of the PCA basis functions, say the first M, we can project the noisy image sequence onto this truncated basis to produce our smooth image sequence.[1].

2.3 Choosing the truncation

We now need to compute M such that reconstructing the noisy image sequence, $\hat{\mathbf{I}}(\mathbf{x},t)$, using the truncated basis $\mathbf{B}_i$, results in a smooth image sequence $\tilde{\mathbf{I}}(\mathbf{x},t)$ that captures the indirect illumination effects:

$$\hat{\mathbf{I}}(\mathbf{x},t) \approx \tilde{\mathbf{I}}(\mathbf{x},t) = \sum_{i=1}^{M} w_i(t)\,\mathbf{B}_i(\mathbf{x}) \tag{3}$$

Choosing a value of M that is too small will lead to the loss of some of the indirect illumination effects, while choosing a value that is too large will result in added noise structure in the solution.

Since the reconstruction is extremely fast once the initial noisy image sequence is generated, we can allow a user to interactively choose M by viewing the reconstructed sequence and adjusting a slider. However, for times when user intervention is undesirable, we have devised an automatic solution for choosing M. The automatic system is based on the variance unexplained by the truncated PCA reconstruction. Knowing the percentage of unexplained variance when using x basis functions ($percentVar(x)$), we choose M

[1]This projection can be viewed as a filtering operation using the optimal spatial filter kernels for the sequence [Capon et al. 1967]

as the lowest x such that:

$$(percentVar(x) \leq \varepsilon)$$

and

$$(percentVar(x) - percentVar(x+1)) \leq \varepsilon_{change}$$

for a user defined ε and ε_{change}. Intuitively, these criteria represent the fact that we want to be reasonably close to the original image sequence, and we want to stop adding basis functions when adding the next one doesn't produce much improvement (it is probably mostly reconstructing the noise at that point).

2.4 Animated Indirect Illumination Algorithm

Combining what we have learned from the previous sections, our final animated indirect illumination algorithm is as follows:

1. Render the sequence of images $\hat{\mathbf{I}}$ using a small number (e.g. 16-32) of ray samples at each spatial location. (Figure 2(top))
2. Use PCA on the sequence of images $\hat{\mathbf{I}}$ to construct a basis $\mathbf{B}_i$ and truncation M in which the sampling noise cannot be expressed. (Figure 2(middle))
3. Project the image sequence $\hat{\mathbf{I}}$ onto the truncated basis $\mathbf{B}_i$ to produce the final smoothed image sequence $\tilde{\mathbf{I}}$. (Figure 2(bottom))

Using this algorithm, we can produce animation sequences that contain no noisy flickering or popping for roughly the cost of a low sample, noisy animation.

3 Extensions

While section 2 described the basic filtering algorithm, there are several extensions that we can add to make the algorithm much more useful.

3.1 Moving cameras and objects

The most important extension is to remove the image space restriction and to allow for moving cameras and objects. The difficulty with naively storing the illumination at pixel locations when using a moving camera is that the temporal changes at a pixel would encode both illumination changes as well as changes due to the camera motion (such as visibility changes). Although our algorithm could still be used on the resulting pixels, the additional, non-illumination variations make the denoising process much more difficult. A better solution would be to compute the indirect illumination at the same set of object space positions for each frame, and then store these values in a point cloud or texture map. Since the object space points are fixed, the temporal variation of each value is due only to changes in the illumination (in addition to the noise). Therefore, the point clouds or textures can be denoised using the same basis projection technique used for images in the previous section. When the indirect illumination is needed for the final render, it can be accessed via a lookup into these smoothed point clouds or textures.

Rigidly moving objects can be handled in the same manner as a moving camera by storing the results in an object space point cloud, texture map or similar structure. Deforming objects require the use of a rest or reference object with a static set of sample points. The indirect illumination should be computed for each frame at points on the deformed object that correspond to the points on the reference/rest object. By storing these illumination values at the reference sample positions (using either a point cloud or texture map), these deforming objects can be denoised similarly to rigid objects. Figure 1 (and the accompanying video) shows an example of our statistical acceleration technique on an animation containing both a moving camera and objects.

Although the result shown here stores irradiance in a point cloud, directionally varying results could easily be incorporating using a structure in the spirit of [Kristensen et al. 2005]: each sample location would store a spherical harmonic representation of the lighting and the denoising would be performed on the coefficients of the spherical harmonic basis functions (as opposed to just on the single irradiance value).

3.2 Per Pixel Truncation and Importance Sampling

While the truncation selection method described in section 2.3 does a good job of computing M for the entire image, there are times when M should be chosen more locally. One example of this is Figure 3 which displays an indirect illumination frame from a sequence in which a ball is moving along the floor creating a translating contact shadow. In this sequence, it is perfectly reasonable to use a small number of modes (1 or 2) for the background walls and ceiling since they do not vary much. However, the contact shadow on the floor requires many more modes. The problem that arises when using the larger number of modes is that the back wall, which was well described by 1 or 2 modes, gains only noise structure in the higher modes (see Figure 3(b)). Therefore, we may wish to choose M on a pixel by pixel basis.

Choosing the truncation on a pixel by pixel basis allows each pixel to decide how many basis functions are required for proper reconstruction. Unfortunately, if neighboring pixels don't use similar truncations artifacts can occur. To remedy this the truncation map should be smoothed using a technique such as [Kontkanen et al. 2004] to produce a smooth floating point truncation map. An example of a truncation map for the moving ball example is shown in Figure 3(c) - where blue indicates a small value of M and red indicates a large value of M. A non-integer truncation simply indicates that an interpolation should be performed between solutions with $floor(M)$ and $ceil(M)$ (a truncation of 4.7 would simply be an interpolation of 0.7 between the 4 basis solution and the 5 basis solution). Figure 3(d) shows the reconstruction using the truncation map in (c). Notice how the structure on the back wall is significantly reduced.

The truncation map can also be used to drive importance sampling. Pixels with a larger value of M usually require more ray samples than those with a smaller value of M since such pixels have higher variation in the sequence. Noting this, we can use the truncation map to determine how many ray samples to use in each pixel. We have had success with a very simple scheme in which pixels with M less than a threshold value M_{min} are reconstructed using the truncated basis as normal, while those with $M \in (M_{min}, M_{max}]$ are interpolated between the basis reconstruction, $\tilde{I}$, and a high sample computation, I_{hi}. The pixel value is thus:

$$\begin{cases} \tilde{I}, & M \leq M_{min} \\ (1-\alpha)\tilde{I} + \alpha I_{hi}, & M \in (M_{min}, M_{max}] \\ I_{hi}, & M \geq M_{max} \end{cases}$$

where $\alpha = \frac{M - M_{min}}{M_{max} - M_{min}}$.

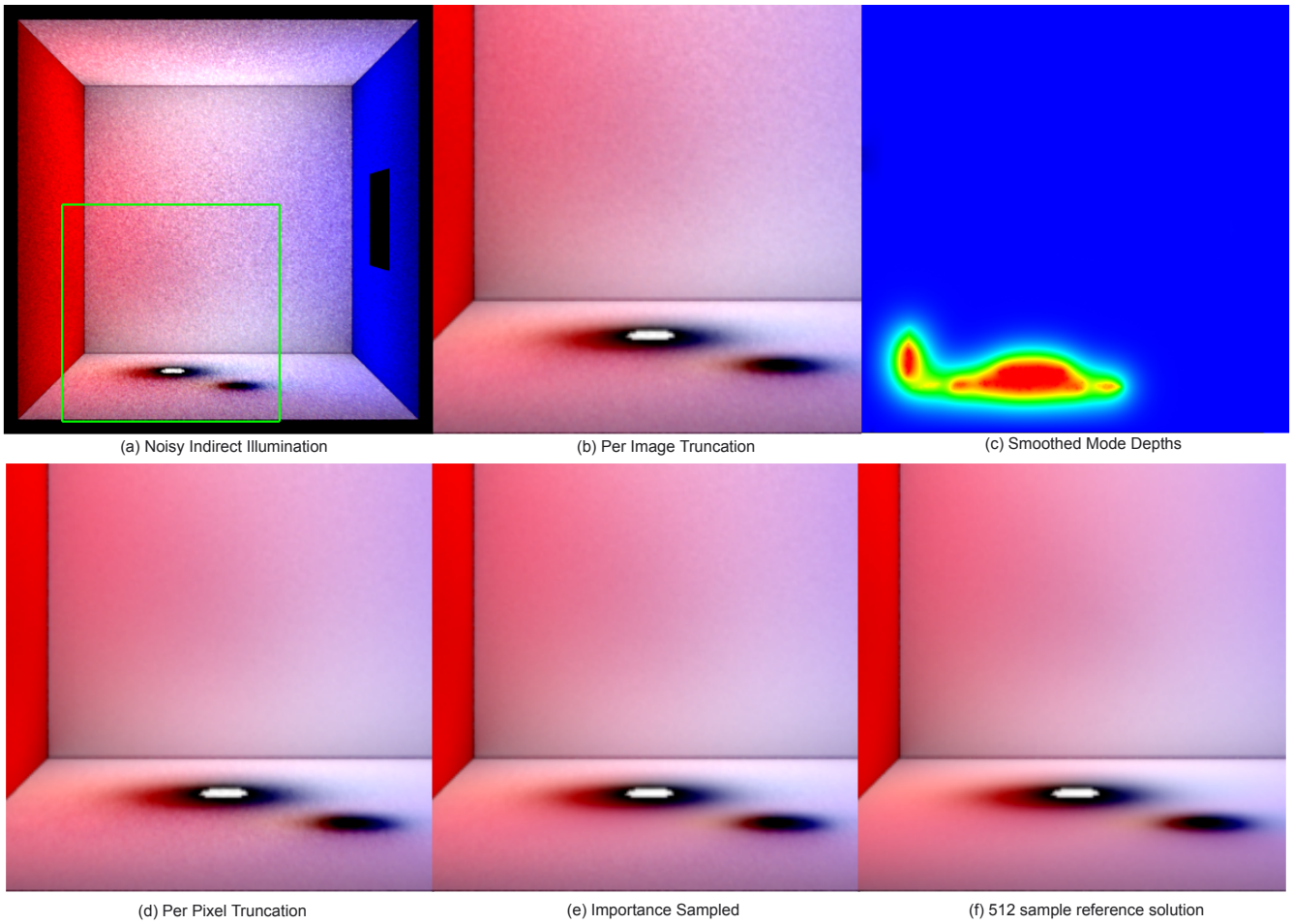

Figure 3: *An example of denoising with a moving contact shadow: In this sequence, the front ball translates from right to left (see the video for the full animation). (a) A single frame of the indirect illumination on the set computed using 32 ray samples per hemispherical integration. (b) The smoothed image using 7 basis functions. The translating contact shadow requires all 7 basis functions, but using 7 basis functions actually introduces additional structure on the back wall (which only required 1-2 basis functions). (c) The smoothed, per pixel basis truncations allow us to reconstruct using a different number of basis functions in each pixel - as in (d) - greatly reducing the structure on the back wall. (e) Adding additional samples where needed (based on the basis truncation) allows us to reduce the structure on the floor even further producing a result very close to the 512 sample reference image in (f). Note that all images were significantly contrast enhanced to make the structure more visible.*

While more intelligent schemes could be derived (such as interpolating between statistically denoised solutions with different ray samples), we have found this simple scheme to be sufficient in our cases. Figure 3(e) shows this scheme on the moving ball example (here $M_{min} = 4$, $M_{max} = 7$, and 512 samples were used when computing I_{hi}). Notice how the structure on the floor is reduced such that it approaches the look of the 512 sample reference solution in Figure 3(f). Using this simple importance sampling technique does increase the average ray samples per pixel to 51.845, up from the base value of 32.

4 Results

Several results using our statistical acceleration system are shown in Figures 1, 2, 3 and the accompanying video. In all examples, we set $\varepsilon = 0.005$, and $\varepsilon_{change} = 0.001$. All timings (see Table 1) were performed on a 3.4GHz Intel Xeon.

Figure 2 and the first video segment show an example of indirect illumination in the Cornell box resulting from moving a single point light in a circular path along the ceiling. A sample of the noisy indirect illumination images is shown in Figure 2(top). The computed basis functions are shown in Figure 2(middle) and the resulting smoothed images (projecting onto the first 5 basis functions) in Figure 2(bottom). Using our acceleration scheme on this 100 frame sequence results in an 8.09x speedup over an irradiance cached, 512 sample render.

Figure 3 and the second video segment show an example on the indirect light in the Cornell box resulting from moving the front ball from left to right (note that this is only the indirect illumination on the box as the two balls have been removed from the image). Although the translating contact shadow causes this sequence to require more basis functions than the previous moving light example (7 versus 5), our denoising time is still reasonable (68 seconds). Using our acceleration scheme on this 150 frame sequence results in an 11.8x speedup over the irradiance cached, 512 sample render.

Figure 1 and the third video segment show an example of our sta-

Scene	Num Frames	Noisy Render	Denoising	High Sample Render	Speedup Factor
Cornell Light (Figure 2)	100	1174s	56s	9948s	8.09x
Cornell Ball (Figure 3)	150	1802s	68s	21960s	11.8x
Moving Car (Figure 1)	126	15125s	522s	345870s	22.1x

Table 1: *Results of applying our statistical acceleration technique to several animated sequences. All of the high sample renders used 512 samples while the noisy renders used 32 samples (except for the Moving Car which used 16 samples). The denoising times include both the basis function computation and the projection time.*

tistical acceleration technique applied to a production quality sequence. This example contains both a moving camera and object, as well as complex shaders and geometry. Additionally, this sequence contains a significant illumination discontinuity when the light is turned on. Unlike a naive temporal smoothing, our basis projection technique does not smooth the transition when the light turns on. Using our statistical acceleration technique allows us to render the indirect illumination for this 126 frame sequence using only 16 samples in just over 4 hours and 20 minutes, while the 512 sample render requires over 96 hours to render. This represents a significant benefit in turnaround time for a production pipeline.

5 Conclusions and Future Work

We have presented an easy to implement technique for accelerating indirect illumination for animated sequences. Our technique first computes low sample, noisy solutions and then uses temporal variation to compute a smooth sequence adapted basis. Finally, the noisy solutions are projected onto the basis resulting in the final smooth solution. The resulting cost of computing the indirect illumination is therefore greatly reduced - roughly equivalent to computing the low sample, noisy solution. Although there may be structure remaining in the denoised illumination sequence, it is often imperceivable especially when combined with surface texture and direct lighting contributions. Moreover, the resulting animation sequence does not contain temporal noise such as flicker or popping. This fact, combined with the decreased rendering times make this technique a useful tool for decreasing turnaround time in a production pipeline.

As future work we wish to explore different basis computation strategies (and truncation rules), as well as different error metrics, and basis localization strategies. Additional processing of the basis functions is also of interest. We would like to explore the effects of different smoothings on the basis functions themselves, as well as what it means to allow for editing of the basis functions.

References

CAMMARANO, M., AND JENSEN, H. W. 2002. Time Dependent Photon Mapping. In *Proceedings of the 13th Eurographics Workshop on Rendering*.

CAPON, J., GREENFIELD, R. J., AND KOLKER, R. J. 1967. Multidimensional maximum-likelihood processing of a large aperture seismic array. *Proc. IEEE 55*, 2, 192–211.

DAMEZ, C., DMITRIEV, K., AND MYSZKOWSKI, K. 2003. State of the art in global illumination for interactive applications and high-quality animations. 55–77.

HEINRICH, S., AND KELLER, A. 1994. Quasi-Monte Carlo Methods in Computer Graphics Part II: The Radiance Equation. Tech. Rep. 243/94.

JENSEN, H. W. 1996. Global Illumination Using Photon Maps. In *Rendering Techniques '96 (Proceedings of the 7th Eurographics Workshop on Rendering)*.

KAJIYA, J. 1986. The Rendering Equation. In *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, 143–150.

KONTKANEN, J., RSNEN, J., AND KELLER, A. 2004. Irradiance Filtering for Monte Carlo Ray Tracing. In *MC2QMC 2004*.

KRISTENSEN, A. W., AKENINE-MOELLER, T., AND JENSEN, H. W. 2005. Precomputed local radiance transfer for real-time lighting design. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3, 1208–1215.

MYSZKOWSKI, K., ROKITA, P., AND TAWARA, T. 1999. Perceptually-Informed Accelerated Rendering of High Quality Walkthrough Sequences. In *Proc. of the 10th Eurographics Workshop on Rendering*, 5–18.

MYSZKOWSKI, K., TAWARA, T., AKAMINE, H., AND SEIDEL, H.-P. 2001. Perception-Guided Global Illumination Solution for Animation Rendering. In *Computer Graphics Proc. (ACM SIGGRAPH '01 Proceedings)*, 221–230.

NIMEROFF, J., DORSEY, J., AND RUSHMEIER, H. 1996. Implementation and Analysis of an Image-Based Global Illumination Framework for Animated Environments. *IEEE Transactions on Visualization and Computer Graphics 2*, 4, 283–298.

VOLEVICH, V., MYSZKOWSKI, K., KHODULEV, A., AND E.A., K. 2000. Using the Visible Differences Predictor to Improve Performance of Progressive Global Illumination Computations. *ACM Transactions on Graphics 19*, 2, 122–161.

WARD, G., AND HECKBERT, P. 1992. Irradiance Gradients. In *Proceedings of the 3rd Eurographics Workshop on Rendering*, 85–98.

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