SIGRAD 2012
A. Kerren and S. Seipel (Editors)

Real-time Image Based Lighting with Streaming HDR-light Probe Sequences

Saghi Hajisharif[†], Joel Kronander[‡], Ehsan Miandji[§], and Jonas Unger[¶]

Linköping University, Sweden

Abstract

We present a framework for shading of virtual objects using high dynamic range (HDR) light probe sequences in real-time. Such images (light probes) are captured using a high resolution HDR camera. In each frame of the HDR video, an optimized CUDA kernel is used to project incident lighting into spherical harmonics in real time. Transfer coefficients are calculated in an offline process. Using precomputed radiance transfer the radiance calculation reduces to a low order dot product between lighting and transfer coefficients. We exploit temporal coherence between frames to further smooth lighting variation over time. Our results show that the framework can achieve the effects of consistent illumination in real-time with flexibility to respond to dynamic changes in the real environment.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Image based lighting, pre-computed radiance transfer, high dynamic range video

1. Introduction

Image Based Lighting (IBL), [Deb98], is a widely used technique for photo-realistic rendering of virtual objects so that they can be seamlessly composited into still or video footage captured in real scenes. The key idea of IBL is to capture the lighting present in the real scene and use this information to illuminate the virtual objects. The scene lighting in traditional IBL is measured by capturing an omni-directional High Dynamic Range (HDR) image, or HDRi, at a single point in space. Such a panoramic HDRi is generally called a *light probe*. Since the HDRi captures the full dynamic range in the scene (from the direct light sources to the parts of the scene that are in shadow), the light probe can be thought of as a measurement of the scene radiance incident at the point in space where the panorama was captured, and can be used as an approximation of the lighting in the scene during rendering. The ease of use and level of realism attainable have now made IBL a standard tool in most production pipelines, and even in real-time applications (based on approximations).

The key challenge in real-time IBL rendering is that the scene lighting is described as an image with no explicit information about where light sources and other high intensity regions are located. When each fragment in the scene is shaded during rendering, this leads to a significant sampling problem of the spherical radiance distribution described by the light probe image. This has led to the development of techniques for Pre-computed Radiance Transfer (PRT), for an overview see [Ram09]. In PRT, the light transport is approximated using basis projections, e.g. spherical harmonics or wavelets, in which the light probe HDRi, material properties, and local self occlusion on the virtual objects can be represented with only a small number coefficients, and the interaction between lighting, material and geometry can be efficiently computed in the transformed space.

Traditional IBL has been limited to only static lighting environments. This is due to the fact that there are no cameras available on the market, that can capture true HDR images in a single shot. HDR images are commonly captured using exposure bracketing, a series of differently exposed images covering the dynamic range of the scene that are combined into a final HDR image. This limits the capture to

† sagha198@student.liu.se
‡ joel.kronander@liu.se
§ ehsan.miandji@liu.se
¶ jonas.unger@liu.se

static scenes and still images. However, recent developments in sensor and imaging hardware, [UG07, TKTS11, KUG12], have now made it possible to capture HDR-video (HDRv). This in turn also enables the capture of light probe video sequences, and thus IBL with dynamic real world lighting environments and moving light probes.

Contributions In this paper, we present a technique and system overview for real-time IBL using HDRv light probe sequences. Real world scene lighting is recorded using a high quality and high resolution 4Mpixel HDRv camera running at 25 or 30 frames per second (fps). Utilizing a real-time CUDA kernel, the input HDR images are processed and the spherical radiance distribution described by each frame in the video sequence is projected onto a low order spherical harmonics basis. Using precomputed techniques, real-time image based rendering of synthetic objects is achieved. and used for image based rendering of synthetic objects.

2. Background and Previous Work

The way in which illumination varies in a scene as a function of location in space (x,y,z), time t, direction (ϕ,θ) and wavelength λ is described by the plenoptic function $P(x,y,z,\phi,\theta,\lambda,t)$, Adelson and Bergen [AB91]. In computer graphics image synthesis, this corresponds to the radiance distribution $L(\mathbf{x},\vec{\omega}_i)$ incident at a surface point $\mathbf{x}$ from an angular region subtended by a set of directions $\vec{\omega}_i$, as described by the rendering equation, Kajiya [Kaj86]:

$$L(\mathbf{x},\vec{\omega}_o) = \int_\Omega L(\mathbf{x},\vec{\omega}_i)\rho(\mathbf{x},\vec{\omega}_i,\vec{\omega}_o)d\vec{\omega}_i \qquad (1)$$

where ρ describes how the surface material at $\mathbf{x}$ transforms radiance incident from a direction $\vec{\omega}_i$ towards the outgoing direction $\vec{\omega}_o$ from which the point $\mathbf{x}$ is observed. In computer graphics, the time t is usually fixated, and the spectral characteristics, λ, of the plenoptic function, P, are usually described using three spectral bands for red, green and blue colors respectively.

Based on the observation that the lighting plays a key role in the visual realism of computer generated images, Debevec [Deb98] introduced image based lighting; a technique that enables virtual objects to be rendered into real scenes and appear as if they were actually there. In traditional IBL the incident radiance distribution $L(\mathbf{x},\vec{\omega}_i)$ is described by a panoramic HDR image I_j captured in a real scene, where j denotes the linear index for each pixel in the image. Each pixel j in I_j can be thought of as the radiance contribution from the solid angle of direction $\vec{\omega}_i$ subtended by the pixel area, i.e. $L(\mathbf{x},\vec{\omega}_i) \approx I_j$. Since the panoramic image I is captured at a single point in space, the spatial coordinate $\mathbf{x}$ vanishes, and the image I_j describes only the angular variations in the incident radiance distribution. This corresponds to the approximation that the lighting environment captured in I_j is infinitely far away from the virtual objects being rendered.

IBL has traditionally been limited to scene lighting captured at a single point in space and at a single instant in time. The main reason for this is that the HDR image capture, as introduced in the computer graphics community by Debevec and Malik [DM97], has been carried out by combining a series of differently exposed low dynamic range images into an HDR image covering the full dynamic range of the scene. This technique, often referred to as exposure bracketing, requires that several images are captured and can thus not handle dynamic scenes or moving cameras. For an overview of the background of HDR imaging see Reinhard et al. [RWPD06].

Moving beyond conventional cameras, a number of approaches and hardware setups for HDR imaging and even video have been proposed in the literature. In order to minimize the temporal artifacts in the exposure bracketing algorithm, Unger et al. [UG07] presented an HDR video camera, where they programmed a SMART CMOS sensor from SICK-IVP AB to capture the exposures back to back on a per pixel basis in rolling shutter fashion, and do the HDR assembly directly on the sensor chip itself. Another option to the time-multiplexing is to trade spatial resolution for dynamic range. Nayar and Mitsunaga [NM00] placed spatially varying exposure filter array in front of the sensor. This approach was extended to rgb-color image capture [NN05] and even capture of multi-spectral HDR-images [YMIN10]. Currently the best performing approach for single shot and HDR-video capture is based on the idea of internally, inside the camera system, splitting the optical path onto multiple synchronized sensors [AA04, WRA05, MH07, TKTS11, KUG12]. By placing Natural Density (ND) filters with varying density in front of the sensors, e.g. [AA04], or more sophisticated beam splitter optics, e.g. [TKTS11], the different low dynamic range exposures can be captured with full resolution at the same instant in time and with the same integration time. This prevents ghosting artifacts from scene motion and ensures correct motion blur for all sensors. These systems, now, also enable high quality HDR-video capture.

The benefit and impressive rendering results from using IBL has motivated research and development of techniques for real-time rendering. These techniques use a single light probe image captured at a single instant in time, and generally focus on describing $L(\mathbf{x},\vec{\omega}_i)$ and ρ in the rendering equation, Eq. 1, as well as local visibility information using approximations that make it possible to solve the rendering equation in real-time for complex scenes. Ramamoorthi and Hanrahan [RH01] projected the captured illumination onto *Spherical Harmonics* (SH) basis functions, and showed that diffuse materials could be accurately simulated by representing the environment illumination with 9 SH coefficients. Sloan et al. [KSS02, SKS02, SLS05] later extended this technique and introduced *Precomputed Radiance Transfer* (PRT) of glossy materials, performing expensive computations as a pre-computation step. An in-depth overview of PRT-methods is presented in Ramamoorthi [Ram09].

For dynamic environment maps methods like sequential Monte Carlo sampling [GDH06] have also been considered however these methods do not run in real-time and therefore not suitable for our purpose.

Building on this body of work, our method extends IBL and PRT to include also the temporal domain, i.e. in our framework $L(\mathbf{x},\vec{\omega}_i)$ in Eq. 1 becomes $L(\mathbf{x},\vec{\omega}_i,t)$. We also demonstrate how the projection of the captured light probes onto a spherical harmonics basis can be parallelized and computed in real-time for each frame in the input HDR video sequences. This means that, under the assumption of low angular frequency in the illumination, our processing and rendering framework supports dynamic HDR environment maps.

3. PRT for Realtime Rendering with HDR Video Sequences

Considering the illumination as HDR light probes, the underlying assumptions in this paper are that each pixel I_j in a light probe image can be thought of as a radiance contribution from a corresponding incident direction $\vec{\omega}_i$, and that the captured real environment is far enough so that the radiance contribution I_j is parallel over the entire scene.

Rendering objects that are illuminated by such distant environmental lighting requires solving the rendering integral, Eq. 1. The integrand describes the product between the incident radiance distribution and the surface material that includes the cosine falloff factor. To take into account shadowing, we also include a visibility factor $V(\mathbf{x},\vec{\omega}_i)$ that describes the self occlusion at a surface point $\mathbf{x}$ such that: $V(\mathbf{x},\vec{\omega}_i)=1$ if the distant environment is visible in direction $\vec{\omega}_i$, and $V(\mathbf{x},\vec{\omega}_i)=0$ if geometry occludes the environment in the direction $\vec{\omega}_i$. Assuming that the scene is static and only contains Lambertian material, the material ρ will be independent of the viewing angle $\vec{\omega}_o$ and only depend on the diffuse albedo $\rho_A(\mathbf{x})/\pi$ and the cosine between $\vec{\omega}_i$ and the surface normal $\mathbf{N}_x$ at $\mathbf{x}$. Consequently, Eq.1 (with the temporal domain included) is simplified to the following for Lambertian surfaces:

$$L(x,\vec{\omega}_o,t)=\frac{\rho_A(\mathbf{x})}{\pi}\int_{\Omega}L(\vec{\omega}_i,t)V(\mathbf{x},\vec{\omega}_i)(\mathbf{N}_x\cdot\vec{\omega}_i)d\vec{\omega}_i \quad (2)$$

where Ω is the hemisphere centered at the normal $\mathbf{N}_x$. We now define what is referred to as the transfer function T, that includes the visibility and cosine falloff :

$$T(\mathbf{x},\vec{\omega}_i)=V(\mathbf{x},\vec{\omega}_i)(\mathbf{N}_x\cdot\vec{\omega}_i) \quad (3)$$

and the rendering integral becomes:

$$L(\mathbf{x},\vec{\omega}_o,t)=\frac{\rho_A(\mathbf{x})}{\pi}\int_{\Omega}L(\vec{\omega}_i,t)T(\mathbf{x},\vec{\omega}_i)d\vec{\omega}_i \quad (4)$$

Our task is now to compute the rendering integral at high frame rate, while supporting HDR light probe sequences as input. Our algorithm is based on pre-computed radiance transfer [SKS02] and extends these ideas to the temporal domain. We approximate the spherical lighting measurements I_j and local transfer function T using spherical harmonics and utilize the orthogonality of this basis to efficiently solve the rendering integral Eq. 4 as a dot product between SH coefficients. For the sake of the presentation, here we use static scenes and Lambertian materials. It should, however, be noted that our technique applies similarly to previously presented methods in PRT that support both dynamic scenes as well as glossy materials, e.g. [SLS05].

3.1. Spherical Harmonics Representation

Definition Spherical harmonics basis functions, denoted as $y_l^m(\theta,\phi)$ are frequency space basis functions which are defined over the unit sphere. The order, or the number of bands, of the SH functions is indicated by l, and m is the degree where $l>0$ and $-l<m<l$. Using real-valued spherical basis functions, we can define them as follows:

$$y_l^m(\theta,\phi)=\begin{cases}K_l^m P_l^{|m|}(\cos(\theta))cos(|m|\phi) & \text{if } m\geq 0\\ K_l^m P_l^{|m|}(\cos(\theta))sin(|m|\phi) & \text{if } m<0\end{cases} \quad (5)$$

where K_l^m are normalization constants, and $P_l^{|m|}$ are Legendre polynomials. Each band is equivalent to polynomials of the same degree. Since the SH functions are orthonormal, the projection of a spherical function $f(\vec{\omega})$ defined over a domain s onto the SH basis functions is as follows:

$$f_l^m=\int_s f(s)y_l^m(s)ds \quad (6)$$

An n^{th}-order SH basis function is described by n^2 coefficients. It is common practice to truncated the order, n, to a finite number, in the PRT case typically $3,4$ or 5 . An approximation of the projected function $f(\vec{\omega})$ can then be reconstructed from the truncated SH basis functions according to:

$$f(\vec{\omega})\approx\sum_{l=0}^{n-1}\sum_{m=-l}^{l}y_l^m(\vec{\omega})f_l^m \quad (7)$$

Spherical harmonics are rotation invariant and orthonormal. The orthonormality makes it possible to compute the

convolution of two function defined on a spherical domain as a dot, or inner, product of the SH coefficients in the projective space. In the PRT case the transport function $T(\mathbf{x}, \vec{\omega}_i)$ and the spherical radiance distribution $L(\vec{\omega}_i, t)$ are projected onto an SH basis, and the dot product between the SH coefficients corresponds to solving the integral in Eq. 4. For more details about the SH basis functions and their use in PRT, we refer to [RH04], [Ram05] and [Gre03].

3.2. Algorithm Overview

We use an experimental setup consisting of an HDR video camera with a resolution of 2400×1700 pixels capturing the full dynamic range of the environment through the reflection in a mirror sphere (see Section 4). The camera is running at 25 or 30 fps and the processing and rendering is performed in real-time. Our algorithm can be outlined in an off-line pre-processing step, an on-line processing and rendering step:

Off-line pre-processing of the transfer function:
At each vertex on each object in the virtual scene, the transfer function, T, described in Eq. 3,is calculated and projected onto an SH basis. This information is stored on disk and uploaded onto the GPU at runtime.

On-line processing:
For each HDR light probe image streamed to the GPU, the real-time algorithm can be outlined as:

1. *Light probe image processing* - Down-sampling, filtering and processing of the HDR light probe image prior to SH projection.
2. *Lighting SH projection* - The down-sampled image, I_j, is projected onto a SH basis of order $3, 4$ or 5 according to Eq. 6 above. A CUDA kernel is used to accelerate the process and enables real-time performance.
3. *Temporal filtering* - Since we only use a single light probe image at each time step t, the entire virtual scene will be affected by lighting variations that in reality only would affect a small part of it. This may introduce flickering artifacts introduced by strong spatial variations in light probe images. To avoid this, we (optionally) perform filtering of the projected lighting SH coefficients in the time domain.
4. *Lighting reconstruction and rendering* - The SH representation of the pre-processed transfer function, T, of each vertex in the scene and the SH projection of the incident illumination $L(\vec{\omega}_i, t)$ are used to efficiently solve the rendering integral, Eq. 4, at each fragment being rendered. Finally, the full resolution image is projected onto a quad oriented perpendicular to the virtual camera and used as backdrop onto which the rendered objects are composited.

Below, we describe each step in our algorithm in detail, and present an overview of the real-time processing, SH-projections, filtering and reconstruction.

3.3. Off-line Pre-processing of the Transfer Function

The local visibility, V, in Eq. 2 is computationally expensive to compute, and cannot be determined on-line. Therefore the transfer function T in Eq.3, is calculated in a pre-processing step. For each vertex on each surface, the local visibility is sampled on a spherical domain using ray-casting. The dot product between the sample direction and the surface normal is calculated, and the transfer function, T is computed and projected onto SH basis functions as described in Eq.6. The transfer function coefficients , which we denote as c^T_{lm}, are stored to be used during on-line image synthesis. Using Monte-Carlo integration, the integral of Eq.6 can be numerically evaluated as:

$$c^T_{lm} = \int_\Omega T(x, \vec{\omega}_i) {Y_l}^m(\omega_i) d\omega_i \approx \sum_{i=1}^{N} w(\vec{\omega}_i) T(x, \vec{\omega}_i) {y_l}^m(\vec{\omega}_i) \tag{8}$$

where $w(\vec{\omega}_i)$ is a weighting function and N is the number of sample directions. Using stratified sampling over a unit sphere leads to a constant weight function $w(\vec{\omega}_i) = 4\pi/N$. Note that at each shading point we are sampling a spherical domain instead of hemisphere. This is to avoid transformation of global coordinate to local coordinate.

3.4. Light Probe Image Processing

The HDR light probe image available for time step t, is streamed to the GPU. We first iteratively down-sample the image to a lower resolution version for the SH projection. This is reasonable as the order of the SH projection is generally low and thus low pass filter the image. We use a Gaussian blur filter for down-sampling. The result of down-sampling is passed to the projection kernel.

3.5. Lighting SH Projection

The environment lighting is considered to be dynamic and based on HDR video streams. This means that the incident lighting is changing in each frame, and the SH approximation of the radiance distribution needs to be re-computed for each frame. Traditional methods of projecting the environment lighting onto SH basis functions require a considerable processing time not suitable for real-time frame rates and temporal coherency. In order to accelerate this process, we use GPGPU programming to perform these operations in real time.

During the SH projection, we loop over each pixel in the input image, i.e. we perform uniform sampling in image space, and that each pixel corresponds to a solid angle in a certain direction. The light probe images are captured in real-time and mapped with latitude longitude mapping. The image domain is parametrized with $u, v \in [0, 1]$. Thus

the mapping from image coordinates to directions in world coordinates can be described as:

$$(\theta, \phi) = (\pi v, 2\pi u) \tag{9}$$

$$(D_x, D_y, D_z) = (rsin\theta cos\phi, rsin\theta sin\phi, rcos\theta) \tag{10}$$

The incident lighting corresponding to each pixel is then projected onto SH basis functions. As described in Section 3.3, projecting a function, f onto an SH basis requires the integral of the products of f, and the SH basis functions over the domain of f to be computed. The incident lighting, denoted $L(\vec{\omega}_i)$, is projected onto SH basis functions according to:

$$c_{lm}^L = \int_{S^2} L(\vec{\omega}_i) {Y_l}^m(\vec{\omega}_i) d\vec{\omega}_i \tag{11}$$

Using a Riemann sum over the light probe image, Eq.11 can be estimated numerically as follows:

$$c_{lm}^L \approx \frac{1}{N} \sum_{i=1}^{N} w(\theta, \phi) L(\theta, \phi) {Y_l}^m(\theta, \phi) \tag{12}$$

where $N = n_u n_v$, n_u and n_v determines the number of samples on v and u. $w(\theta, \phi) = sin\theta \times 2\pi^2$ is the area measure transformation of sampling from (u, v) space to the direction on the space on the sphere. In order to enable real-time processing we perform these computations using a CUDA kernel.

In order to make use of all available cores on the GPU and to minimize memory latency, CUDA programming requires launching a large number of threads and using fine grained parallelism.

We have designed a CUDA kernel assigning one thread to each pixel in the light probe image. During execution, the threads are grouped together into blocks [NVI11]. The threads are distributed over a set of 2-dimensional blocks. Each block has access to a limited fast shared memory. In the process 16×16 pixels of light probe image are passed to each block where n^2 (for n-band SH basis) coefficients are computed. Each block calculates a partial sum of Eq.12 that is stored in the shared memory. Shared memory is chosen for achieving better performance, since it is the fastest way for the threads within a block to communicate with each other. The result of one block is shown in Fig. 1.

The actual coefficients are then calculated by summing up the projection results from each thread and block. In order to calculate the first coefficient, we need to add up the projection results of N light directions onto the first SH basis function. Therefore, the first element of each thread is added together. This process is applied for the other elements in each block. We use a parallel reduction technique for computing the partial sum in each block, shown in Fig.2. At each step, the addition is reduced in half and this process is repeated until all threads are processed. The result is written into global memory and further reduction is performed between blocks until all partial sums of Eq.12 are calculated. Using reduction is the best way to massively parallelize the computation. In the kernel, samples are converted from uv-coordinate to spherical coordinate using Eq.10 to compute the n^2 SH coefficients.

Spherical harmonics basis functions are well-suited for reconstructing low frequency lighting and shadows. However, when the signal is clamped, this reconstruction is accompanied with a ringing effect known as Gibbs phenomena [HH79]. We use a Hanning window to minimize this problem, see [Slo08] for more details.

3.6. Temporal Filtering

The PRT approximation of distant environments means that we use only a single light probe image per time step. Since a light probe image measures only angular variations in the radiance distribution, it is not possible to capture the spatial variation that would occur in the real world if an object was gradually moving from e.g. strong illumination into shadow. Instead the entire rendered scene will be affected at once. In lighting environments with high spatial variations this may lead to flickering. To avoid such temporal artifacts, we (optionally) perform filtering of the lighting SH coefficients in the time domain. Using the lighting coefficients of the previous frame , $c_{lm}^L(t-1)$, and the current frame, $c_{lm}^L(t)$, the final lighting coefficient is calculated by a linear interpolation as follows:

$$c_{lm}^L(t) = \frac{c_{lm}^L(t) + \alpha c_{lm}^L(t-1)}{\alpha + 1} \tag{13}$$

where α is a non-negative constant. Note that since each two sequential coefficients are interpolated, the coefficients of the previous frames will have a small affection on the current frame's coefficients. This leads to a smooth transition in illumination changing.

3.7. Lighting Reconstruction and Rendering

To relight the scene, according to Eq.2, the projected functions need to be reconstructed. Using orthonormality of SH basis functions, the projections of two functions, T and L over the unit sphere, satisfy:

$$L(\vec{\omega}_i, t) \approx \sum_{k=1}^{n} c_k^L Y_k(\vec{\omega}_i)$$

$$T(x, \vec{\omega}_i) \approx \sum_{k=1}^{n} c_k^T Y_k(\vec{\omega}_i)$$

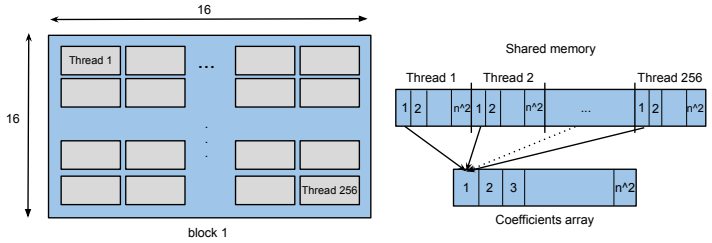

Figure 1: *The arrangements of threads in each block and the shared memory used for storing the result.*

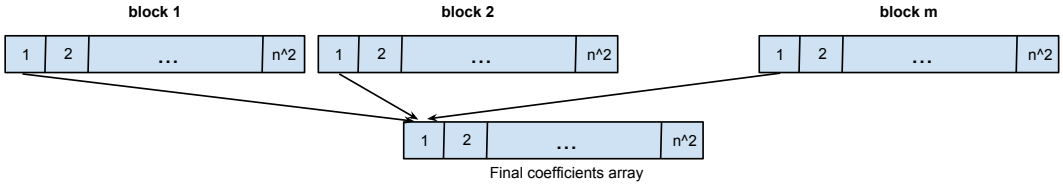

Figure 2: *The result of partial summations in each block is added to the other results using reduction. m is dependent on the width and height of the input light probe image.*

$$L(x,\vec{\omega}_o,t) = \int T(x,\vec{\omega}_i)L(\vec{\omega}_i,t)d\vec{\omega}_i \approx \sum_{i=1}^{n^2} c_i^L c_i^T. \qquad (14)$$

Thus, the rendering integral, Eq.2, is reduced to a scalar product of the projected coefficients.

4. Results and Discussions

In this section, we give an overview of the experimental setup used to test our approach in practice, present a set of example renderings, as well as an evaluation of the algorithm in terms of performance.

4.1. HDR Video Setup

The dynamic HDR light probe input data used for the experiments presented in this paper were captured using an HDR video camera prototype developed in a collaboration between Linköping University, Sweden and the German camera manufacturer SpheronVR AG. The camera system captures images with a resolution of 2400×1700 pixels with a dynamic range in the order of $10.000.000 : 1$ at up to 30 frames per second. For the experiments presented here, the camera is mounted in a real-time light probe setup depicted in Figure 3. In this setup the HDRv camera uses a Zeiss Makro-Planar T* 100mm f/2 ZF.2 lens and images the scene through the reflection in a mirror sphere placed at a distance of 1250 mm from the center of projection along the optical axis. For each captured frame, this yields a close to 360° panoramic HDR image that covers the full dynamic range of the scene. HDR video sequences are stored on disk as individual OpenEXR frames.

Figure 3: *The HDRv camera captures HDR images of* 2400×1700 *pixels exhibiting a dynamic range in the order of* $10.000.000 : 1$ *at up to* 30 *frames per second. Here the camera is displayed in a real-time light probe setup and images the scene through the reflection in a mirror sphere.*

4.2. Results

The test results were acquired on a 3.2 GHz Xeon computer running Linux Ubuntu 11.04 with 23.6 GiB memory and an NVIDIA GeForce GTX580 graphics card. All stages of our method were implemented using C++, CUDA, OpenGL and GLSL respectively. For reducing memory bandwidth cost for data transfer between the GPU and CPU, we are using the OpenGL vertex buffer objects (VBO) extension. We are using diffuse reflecting surfaces and our models are standard models from Stanford university database.

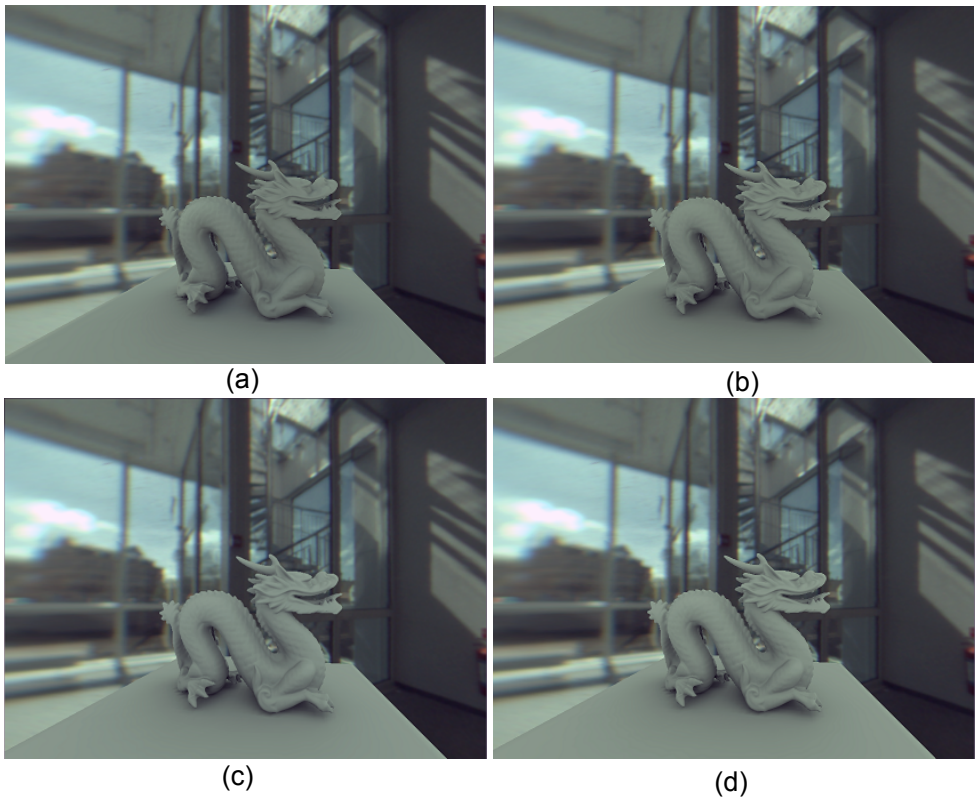

Figure 4: *Testing the shading of the scene with different scaling of the light probe image resolution of a single frame of the environment map. (a)* 1628×814*, (b)* 814×407 *(c)* 203×101 *(d)* 50×25

The CUDA kernel which calculates light's coefficients, utilizes overall $W \times H$ threads, where W is the width and H is the height of the image that is captured in real-time. For each HDR-video frame, the image is uploaded as an OpenGL texture to the graphics memory where it is used as input to the CUDA PRT kernel, as well as for rendering the environment background. For achieving better performance, the environment map input image is down-sampled before SH projection. The actual resolution can be changed in real-time.

Figure 4 displays the dragon model rendered with different resolution of the environment map. As shown in Fig. 4-b, the first iteration of the down-sampling does not have a visible impact on the output shading. However as the procedure continues with higher ratio, the shadows become softer and starting to fade away. Table 2 shows how the performance measured in fps scales according to light probe image resolution.

Table 1 shows timings of the pre-processing phase with visibility on and off. The visibility calculations are multi-threaded. For a test scene containing the Stanford bunny and a ground plane with 7455 vertices, and 14868 triangles, the processing time for computing the transfer function coefficients is 7.17 min with visibility testing and 0.1 sec without it. In Fig. 5 the dragon model is rendered with visibility testing on and off. Despite the time consuming pre-process step that is required, as Fig. 5-b indicates, shadows add to the realism of the results. The number of sample directions per shading point, are also important in calculating the transfer function for shadow testing. Using more samples will result in more accurate shadows and less ringing. This is shown in Fig. 8, where the model is rendered with 100, 900 and 2500 sample directions per vertex. The result shows that 2500 samples are enough for our application.

Using OpenGL vertex buffer objects, the transfer function coefficients are transferred as vertex attributes to the graphics memory. The final reconstruction and rendering is then performed in a GLSL shader. To do so for an object with 6384 vertices and with spherical harmonics with 4 bands, we need a texture of 6348×16 pixels stored in GPU memory. In order to perform the final multiplications according to Eq.14, the coefficients are transferred to the GLSL shader memory. We are using an RGB color model, and for each channel there will be 16 coefficients based on order-4 SH functions. The method was tested for 5^{th} order SH functions, and we found that for lighting a Lambertian surface order-

Table 1: *Pre-process time required for our test scenes with 2500 sample directions, (the plane is included in the calculations)*

Shape	*Vertices*	*Faces*	Time(Visibility off)	Time(Visibility on)
Sphere	6384	12760	0.1 sec	5.19 min
Bunny	7455	14868	0.1 sec	7.17 min
Buddha	54938	109900	0.9 sec	330 min
Dragon	54952	109900	0.9 sec	376 min

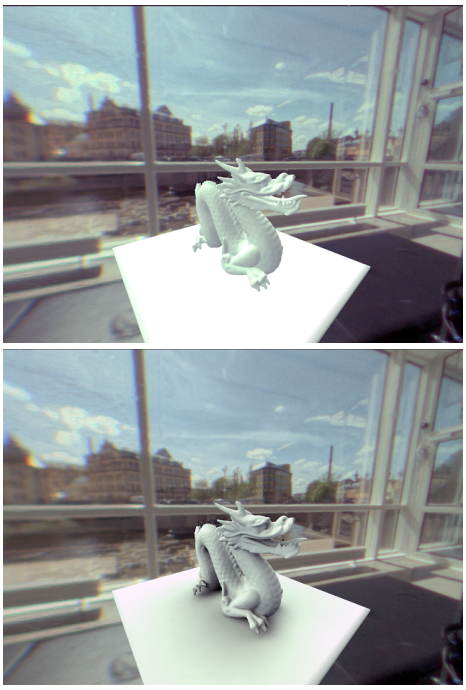

Figure 5: *The Dragon model rendered in the scene (a) without and (b)with visibility testing.*

4 SH functions suffice. However, in order to reduce ringing artifacts, higher order SH bases can be used. Ringing effects can also be minimized using spatial filtering as described.

Finally, Figure 7 shows renderings using input HDR-video sequences from three different scenes, captured both indoors and outdoors. The figure shows the result from the processing, reconstruction and rendering steps in the real-time algorithm described in Section 3.2.

The effect of temporal filtering is to reduce flickering to some extend and by experience we observed that adding more weights to the coefficients of the previous frame can result in a smoother change of illumination in the rendered objects. The result of temporal filtering is shown in the supplementary video attached to this paper.

To simulate specular reflection of the environment in the synthetic object we are using reflection mapping of the environment and the following equation is used for finding the final color of each pixel:

Figure 6: *Specular reflection using environment map and our method.*

$$Color_{final} = C_{diffuse} + c_k * C_{reflection} \tag{15}$$

where $c_k \in [0,1]$ denotes specular reflectivity in the material of the object. The result of this reflection is shown in Fig. 6

5. Conclusion and Future Works

This paper presented a method for real-time rendering of synthetic objects illuminated by real world lighting captured as HDR-video streams. The method is based on pre-computed radiance transfer, where the rendering integral is efficiently computed by projecting the reflectance distribution and spherical lighting environment onto spherical harmonics basis functions. The algorithm was tested using input data from a high resolution HDR-video camera.

In future work, we would like to investigate other basis functions as well as explore augmented reality as an application where real and virtual objects are mixed. We will also further investigate temporal filtering and, based on the coherency between consecutive frames in the input sequences,

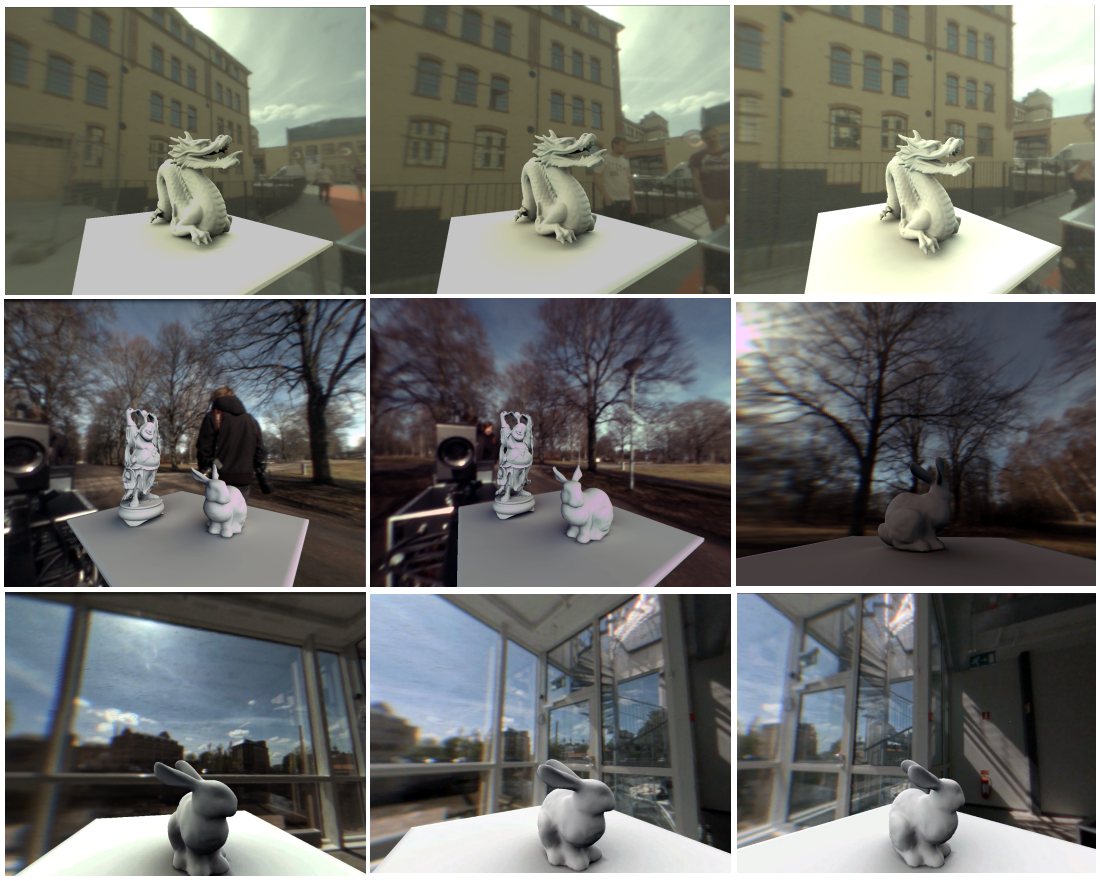

Figure 7: *Synthetic scenes rendered in three different environments.*

techniques for reusing already computed information in the basis projection.

6. Acknowledgement

We would like to thank Per Larsson for assisting with the hardware setups and organizing the lab environment. This work was funded through the Swedish Foundation for Strategic Research, the Linnaeus Environment CADICS, and the Center for Industrial Information Technology CENIIT.

Table 2: *Change of FPS according to light probe image resolution*

Resolution	*FPS*
1628×814	9
814×407	24
407×203	40
203×101	44
101×50	55
50×25	64

References

[AA04] AGGARWAL M., AHUJA N.: Split aperture imaging for high dynamic range. *International Journal of Computer Vision 58*, 1 (2004), 7–17. 50

[AB91] ADELSON E. H., BERGEN J. R.: *Computational Models of Visual Processing*. MIT Press, Cambridge, Mass., 1991, ch. 1. The Plenoptic Function and the Elements of Early Vision. 50

[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 the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 189–198. 49, 50

[DM97] DEBEVEC P. E., MALIK J.: Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH 97* (August 1997), pp. 369–378. 50

[GDH06] GHOSH A., DOUCET A., HEIDRICH W.: Sequential sampling for dynamic environment maps. In *ACM SIGGRAPH 2006 Sketches* (New York, NY, USA, 2006), SIGGRAPH '06, ACM. 51

[Gre03] GREEN R.: Spherical harmonic lighting: The gritty details. *Sony Computer Entertainment America* (2003). 52

[HH79] HEWITT E., HEWITT R. E.: The gibbs- wilbraham phe-

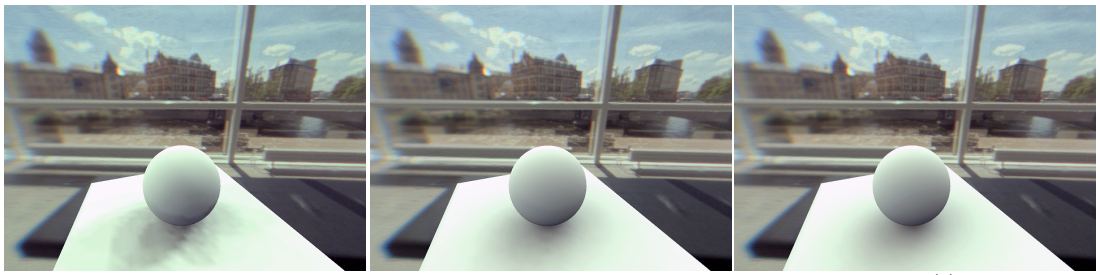

Figure 8: *The effect of under-sampling in calculating transfer function: a) Number of sample directions = 100, b) 900 and c)2500.*

nomenon: An episode in fourier analysis. *Arch. Hist. Exact Sci. 21* (1979), 129–160. 53

[Kaj86] KAJIYA J.: The rendering equation. In *SIGGRAPH 86* (1986), pp. 143–150. 50

[KSS02] KAUTZ J., SLOAN P.-P., SNYDER J.: Fast, arbitrary brdf shading for low-frequency lighting using spherical harmonics. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2002), Eurographics Association, pp. 291–296. 50

[KUG12] KRONANDER J., UNGER J., GUSTAVSON S.: Real-time hdr video reconstruction for multi-sensor systems. *Siggraph 2012 Posters* (August 2012). 50

[MH07] MCGUIRE M., HUGHES J. F.: Optical Splitting Trees for High-Precision Monocular Imaging. *IEEE Computer Graphics And Applications 27*, April (2007), 32–42. 50

[NM00] NAYAR S., MITSUNAGA T.: High dynamic range imaging: Spatially varying pixel exposures. In *Proc. of CVPR* (2000), pp. 472 – 479. 50

[NN05] NARASIMHAN S. G., NAYAR S. K.: Enhancing Resolution Along Multiple Imaging Dimensions Using Assorted Pixels. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 4 (2005), 518–530. 50

[NVI11] NVIDIA CORPORATION: NVIDIA CUDA C programming guide, 2011. Version 4.2. 53

[Ram05] RAMAMOORTHI R.: Modeling illumination variation with spherical harmonics. In *In Face Processing: Advanced Modeling and Methods* (2005), Academic Press. 52

[Ram09] RAMAMOORTHI R.: Precomputation-based rendering. *Found. Trends. Comput. Graph. Vis. 3*, 4 (Apr. 2009), 281–369. 49, 50

[RH01] RAMAMOORTHI R., HANRAHAN P.: An efficient representation for irradiance environment maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 497–500. 50

[RH04] RAMAMOORTHI R., HANRAHAN P.: A signal-processing framework for reflection. *ACM Trans. Graph. 23*, 4 (Oct. 2004), 1004–1042. 52

[RWPD06] REINHARD E., WARD G., PATTANAIK S., DEBEVEC P.: *High Dynamic Range Imaging – Acquisition, Display and Image-Based Lighting*. Morgan Kaufmann, San Francisco, CA, 2006. 50

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM, pp. 527–536. 50, 51

[Slo08] SLOAN. P.-P.: Stupid spherical harmonics (sh). *Microsoft Corporation.* (2008). 53

[SLS05] SLOAN P.-P., LUNA B., SNYDER J.: Local, deformable precomputed radiance transfer. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 1216–1224. 50, 51

[TKTS11] TOCCI M. D., KISER C., TOCCI N., SEN P.: A versatile hdr video production system. *ACM Trans. Graph. 30*, 4 (July 2011), 41:1–41:10. 50

[UG07] UNGER J., GUSTAVSON S.: High-dynamic-range video for photometric measurement of illumination. In *Proceedings of Sensors, Cameras and Systems for Scientific/Industrial Applications X, IS&T/SPIE 19th International Symposium on Electronic Imaging* (Feb 2007), vol. 6501. 50

[WRA05] WANG H., RASKAR R., AHUJA N.: High dynamic range video using split aperture camera. In *Proc. of OMNIVIS* (2005). 50

[YMIN10] YASUMA F., MITSUNAGA T., ISO D., NAYAR S. K.: Generalized Assorted Pixel Camera : Postcapture Control of Resolution , Dynamic Range , and Spectrum. *IEEE Transactions on Image Processing 19*, 9 (2010), 2241–2253. 50