Efficient Screenspace Rendering for Area Lights

Koa, Ming Di; Johan, Henry

2016

Koa, M. D., & Johan, H. (2016). Efficient Screenspace Rendering for Area Lights. Proceedings of the 33rd Computer Graphics International, 29-32.

https://hdl.handle.net/10356/81403

https://doi.org/10.1145/2949035.2949043

Efficient Screenspace Rendering for Area Lights

Ming Di, Koa
School of Computer Science and Engineering,
Nanyang Technological University
N4 Nanyang Avenue
02a-32, Singapore 639798
mdkoa1@e.ntu.edu.sg

Henry Johan
Fraunhofer IDM@NTU, Nanyang Technological University
NS1, 50 Nanyang Avenue
Singapore 639798
henryjohan@ntu.edu.sg

ABSTRACT

Efficient rendering of illumination from area lights in virtual scenes has always proved to be challenging. We extend the work of multi resolution rendering and Light Propagation Volumes (LPV) to simulate direct and indirect illumination from area lights respectively. To compute direct illumination, we create 2D multi resolution fragments to represent the scene on the screenspace, in which higher resolution fragments are created when normal, depth and visibility discontinuity are found. Our subdivision scheme performs a sub-fragment visibility test (SFVT) within each fragment and our proposed gradient aware screenspace subdivision (GASS) algorithm accelerates the refinement by increasing the number of subdivisions based on gradient differences. We also propose a single pass screenspace irradiance upsampling scheme which uses Gaussian radial basis functions (RBF) for interpolating scattered fragments. This reduces artifacts caused by large fragments while also significantly reducing the number of fragments that we require. Our indirect illumination is computed by distributing a set of Poisson sample points in the scene. Each LPV voxel performs a light gathering operation on these samples and deposits them internally. Light intensity in the LPV is propagated simulating indirect illumination from area lights. From experiments, our techniques are able to run at interactive rates.

CCS Concepts

•Computing methodologies → Computer graphics; Rendering;

Keywords

Computer Graphics; Interactive Rendering; Screenspace Rendering;

1. INTRODUCTION

Efficient rendering of illumination from area lights has often been constraint by the integration of the visibility function and radiance over the light surfaces. Direct illumination from area lights produces varying illuminated regions and these are usually visible as soft shadows. Indirect illumination created by area lights tends to produce color bleeding effects. A complex scene with multiple objects of complex geometry usually requires a large amount of visibility samples to produce a smooth image if an area light is present.

In this paper, we aim to handle dynamic lights and viewpoints while efficiently rendering direct and indirect illumination from area lights. Our paper draws inspiration from the multi resolution [5] and Light Propagation Volumes (LPV) [4] rendering. We present three main contributions in this work.

- A screenspace sub-fragment visibility test (SFVT) for checking visibility discontinuities. We also propose a gradient-aware soft shadow (GASS) refinement framework to skip refinement levels compared to former techniques.
- A single pass upsampling method that approximates shadow boundaries with scattered samples by radial basis functions (RBF) interpolation. It is able to negate errors caused by large fragments.
- An area lighting solution for LPV to produce indirect illumination using Poisson samples.

2. RELATED WORK

Multi Resolution Algorithms: Direct illumination from area lights is known to vary smoothly across flat regions. Coarse sampling techniques such as multi resolution splatting by Nichols et al. [5, 6] were devised to take advantage of this property. Multi resolution splatting proposes to split an image into patches known as fragments, where the fragment size depends on the depth, normal and illumination variations within the fragment. As illumination variation decreases, the illumination on a fragment can be represented using information from lower resolution fragments which reduces computation time.

Voxel Based Algorithms: Voxel based techniques use a voxel structure to approximate illumination and scene information. They are also known for rendering indirect illumination. In Crassin et al.'s Gigavoxels [3], a multi level voxel structure is created by voxelizing the scene. During rendering, a voxel cone [3] is used to traverse the voxelized scene. Similarly, voxel cone tracing can also be used for approximating visibility, however their storage costs scale up with accuracy exponentially. In Kaplanyan et al.'s Light

CGI '16, June 28-July 01, 2016, Heraklion, Greece
ACM ISBN -.
DOI: -

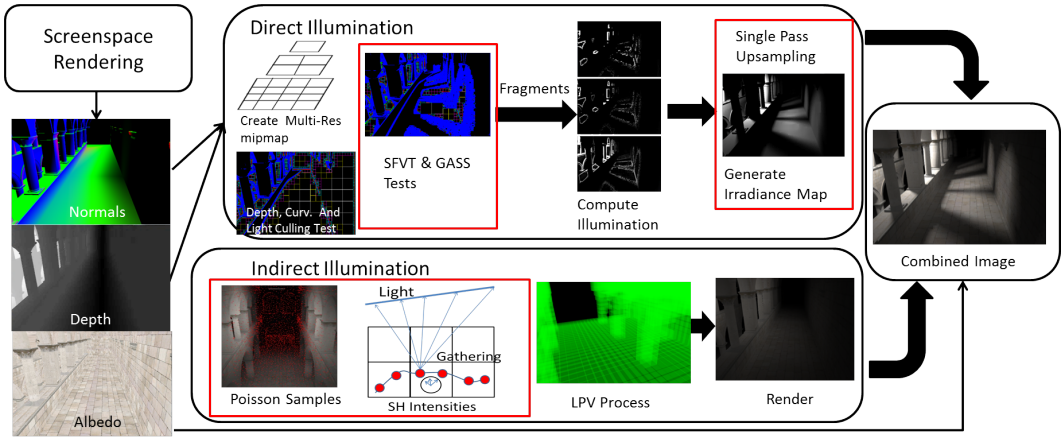


Figure 1. The pipeline of our direct and indirect illumination with area lights. Red boxes indicate our contributions.

Propagation Volumes (LPV) [4], a similar voxel structure is used to store lighting and geometric information. Their voxel resolution is of magnitudes lower than the Gigavoxels and indirect illumination is distributed throughout the voxels using a *propagation* process. In our work, we chose to extend the LPV for indirect illumination because of its low memory footprint and its computation efficiency.

3. SYSTEM DESIGN

Figure 1 describes an overview of our method. Our direct and indirect illumination frameworks run separately. The final render is a combination of direct and indirect illumination, multiplied by the albedo. We explain each stage of our direct and indirect illumination pipeline in this section. The direct illumination pipeline takes in screenspace textures (depth, normal and albedo) as input and outputs an irradiance texture. The irradiance texture will be multiplied by the albedo to obtain direct illumination similar to a deferred shading pipeline. Whereas for indirect illumination, Poisson samples are generated and ray gathering is performed on those samples. The indirect illumination pipeline also generates an output texture with every pixels corresponding to tri-linearly interpolated values from the LPV.

3.1 Direct Illumination - Multi Resolution Refinement for Area Lights

3.1.1 *Geometric Discontinuity and Light Culling*

The first stage of multi resolution refinement receives a depth and normal curvature discontinuity mipmaps as in Nichols et al.'s work [5, 6]. Fragments with high geometric/depth discontinuities are refined. Fragments that passed the geometric discontinuity stage can be further checked if they receive any light through back-face culling. Fragments that fail the light culling stage will have a zero value stored in an illumination texture. The illumination texture stores illumination information of fragments of different mipmap levels.

3.1.2 *Sub-Fragment Visibility Tests (SFVT)*

Nichols et al. [5, 6], performed fragment refinement based on visibility tests. In their work, they ray traced shadow rays to a number of virtual point lights (VPLs) on the light source in fragments within a 3x3 fragment neighbourhood. Next, they used visibility bit comparisons in the 3x3 neighbourhood. If the number of visibility bits differs by a threshold, the fragment would be refined into 4 finer fragments. Instead of performing visibility tests for neighbouring fragments, we perform a subdivision on the current fragment into 4 sub-fragments, which are evenly divided areas within a fragment used for visibility testing. This reduces our bit arrays from 9 to 4 and reduces the total number of visibility rays. In our work, we use 1 bit as our difference threshold. The visibility testing is done using a GPU ray tracer. Although less visibility rays (16 per sub-fragment) as compared to Nichols et al. [6] were used, this SFVT scheme still allows re-using of visibility information for irradiance computation.

3.1.3 *Gradient Aware Soft Shadow Refinement (GASS)*

A standard implementation of fragment refinement would split the fragment into 4 fragments of higher resolution. We note that for cases with high bit discontinuities, this refinement is rarely sufficient to capture the smooth transition in visibility. Eventually, additional stages of refinement and visibility testing would still be required. In our refinement process (Figure 2)), we process a visibility gradient term for each of the 4 sub-fragments, where the gradient term is determined by finding the absolute difference (red arrows) between the previous sub-fragment and current sub-fragment. A high gradient (>4) indicates that both current and previous sub-fragment would need to be subdivided to 4 fragments. Once all needed fragments are generated, we can compute the illumination of each fragment in the illumination texture. A maximum of 16 finer fragments instead of 4 can be generated from this stage. For fragments with zero visibility bits, no refinement is needed and a zero value is written into the illumination texture. This stage of producing fragments of different resolutions can be done using the OpenGL transform feedback shader.

3.1.4 *Additional Samples Generation*

The single sample location in a large fragment center would be insufficient to approximate regions with thin shadows. Hence, additional sample points are added to the refinement. In our case, we generate three finest resolution samples with a certain specific pattern to maximize coverage. These three samples of mipmap level 0 are positioned at the top-center,

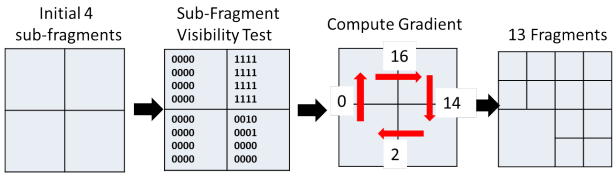
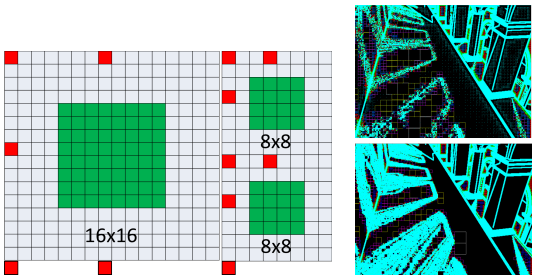


Figure 2. Illustration of how gradients (red arrows) are computed from the visbility information of 4 initial sub-fragments and later refined into 13 finer resolution fragments.

top-left and left-center of the fragment. Additional samples are only generated for fragments of mipmap level greater than two. Figure 3(a) shows the placement of these new samples (in red) and their stable regions (green) as computed during SFVT. These additional samples enable majority of the fragments to have samples to be found along its edges. For example, the 16x16 fragment is connected to two 8x8 fragments on the right and has 4 additional samples along the edge for interpolation. It is also connected to another 16x16 fragment at the bottom edge, which provides it with 2 additional edge samples. This allows lower resolution fragments to be influenced by higher resolution fragments, which are usually more accurate in value. Figure 3(b) shows the additional fragments (in cyan dots) generated in our work compared to Nichols et al. [5]. Only 182,041 fragments were used in our work compared to 340,288 fragments in Nichols et al. [5] for a 1024x768 resolution image.



(a) Additional Samples Placement (b) Comparison with Nichols [5]

Figure 3. (a) Three additional samples (in red) are positioned in the 16x16 and 8x8 sized fragments. Texels in green indicate the stable regions that were computed using the visibility refinement. (b) Comparisons between fragments generated in our work (top) and Nichols et al.[5] (bottom). Each fragment is represented by a colored square tile.

3.1.5 *Screenspace Single Pass Upsampling*

After generating fragments and computing their irradiance information in the illumination texture, the results should be combined into a whole image of finest resolution, known as the irradiance texture. In previous work [6], a multi pass upsampling algorithm performs bilinear interpolation upsampling and addition of illumination information for each mipmap level starting from the coarsest resolution. The multi pass algorithm gives too much influence to fragments from lower resolution which leads to artifacts seen in Figure 4(a) as fragments from lower resolution would have errors propagating to the higher resolution

(a) Multi Pass Upsampling by Nichols et al. [5] (b) Single Pass Upsampling - Ours

Figure 4. Results of irradiance texture from a Sponza scene. (a) Larger fragments are able to influence smaller fragments despite being less accurate. Artifacts appear as small holes near shadow boundaries. (b) Single pass algorithm reduces these artifacts by giving more weights to smaller fragments.

fragments. Although these errors can be removed by over-refining such fragments [6], we demonstrate that these errors can be negated using RBF interpolation while keeping fragment usage low. We refer to Figure 4(b) for our results.

The full irradiance texture is generated by producing a fragment thread for each texel. In this section, we refer the texel of the final irradiance texture as a *target texel*. The values for fragments of high resolution (mipmap level 0 to 2) can be directly copied into their target texel. Subsequently, for target texels of mipmap level 3 and higher, basis functions are created by selecting samples found along the edge (in red) of the fragment shown in Figure 5. Scattered data interpolation techniques [1] with Gaussian RBF are used as they are guaranteed to produce continuous results. Three basis functions, $\mathbf{x_i}$, are selected which overlaps the target texel in Equation 1. The estimated irradiance value $\hat{I}(\mathbf{x})$ at target texel $\mathbf{x}$, can be evaluated by solving the weight w_i for each basis i. This can be done by solving the linear equation in Equation 2a where $\mathbf{w}$ is a vector of weights, w_i, and $\mathbf{\Phi}$ is a correlation/distance matrix consisting of i rows and j columns of Φ. $\mathbf{I}$ is a vector consisting of irradiance values at samples $\mathbf{x_i}$.

$$\hat{I}(\mathbf{x}) = \sum_{i}^{3} w_i \Phi_i(\|\mathbf{x} - \mathbf{x_i}\|), \tag{1}$$

where

$$\mathbf{w} = \mathbf{\Phi}^{-1} * \mathbf{I}, \tag{2a}$$

$$\Phi_{ij} = \exp\left(-d^2/C\right) \tag{2b}$$

In Equation 2b, d is the L2 distance in texels between the chosen sample location and the targeted texel. d is also inversely scaled by the mipmap width of the target texel giving samples from coarser resolution a higher variance. C is a variance scaling factor, which we use C=2. In this work, we use only 3 basis functions so that the inverse could be easily calculated using the *inverse* function in *OpenGL*. For target texels with only 2 basis functions, we only need to do weighted interpolation using the function in Equation 2b between 2 samples, $\mathbf{x_1}$ and $\mathbf{x_2}$ to produce:

$$\hat{I}(\mathbf{x}) = \frac{(\Phi(\|\mathbf{x} - \mathbf{x_1}\|) * I(\mathbf{x_1}) + \Phi(\|\mathbf{x} - \mathbf{x_2}\|) * I(\mathbf{x_2}))}{\Phi(\|\mathbf{x} - \mathbf{x_1}\|) + \Phi(\|\mathbf{x} - \mathbf{x_2}\|)} \tag{3}$$

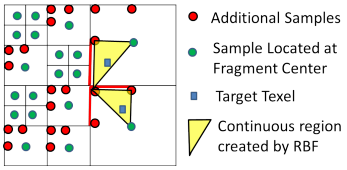


Figure 5. Our upsampling scheme on a target blue texel. The blue texel is computed by a weighted sum of RBF created from its nearest samples. RBF can be chosen from the green circles (fragment centers) and the red samples (additional samples). The yellow region is the continuous region formed by the selected three RBF.

3.2 Indirect Illumination with Poisson Samples

As the usual LPV scheme does not handle area lights, our proposed solution generates a set of Poisson distributed samples [2] around the 3D scene we are rendering. The illumination for each Poisson sample is computed by performing a gathering operation to the light source. Each gathering ray that receives light will deposit a reflected light intensity (represented by spherical harmonics) into a voxel (illustrated in Figure 1). Each voxel will be divided by the number of Poisson samples it received. The remaining processes follows the standard LPV [4].

4. RESULTS AND LIMITATIONS

We show our results rendered in 1024x768 pixels, with 64 samples per fragment in Figure 6. Our images were rendered with a Nvidia GeForce GTX980. The timings required to compute indirect illumination is approximately 10 ms in the rendered scene. The timings in Table 1 indicate the time required for the different stages of our work. Timings for visibility testing for our work includes SFVT and GASS. The large discrepancies in irradiance timings between Figures 6 (a) and (b)[5] compared to the reference are due to lesser fragments and visibility rays being re-used. Figures 6 (d) and (e) show the fragment maps. Further results are available in our supplemental video.

The single pass upsampling algorithm produces certain noisy artifacts. This is mainly due to artifacts caused by extrapolation when using RBF as we cannot guaranteed to find basis functions that overlap the target texel. However, the single pass method uses lesser texture memory due to the removal of intermediate textures that were formerly needed in the multi pass upsampling approach. To improve accuracy of RBF interpolation, it is advisable to use more samples to create basis functions, however for the sake of GPU efficiency, only 3 basis functions were used for this work. Artifacts may also be present due to undersampling in the presence of large area lights, in such cases, it is advisable to use more than 16 visibility rays.

5. CONCLUSION

Our paper outlined a multi resolution approach that is able to render direct illumination efficiently by culling off large portion of unnecessary fragments. A single pass RBF interpolation upsampling approach was proposed to reduce impacts of shadow artifacts that were visible in the previous multi pass upsampling approach. We also proposed a Poisson samples gathering approach to enable the LPV illumination scheme to render indirect illumination created by area lights.

Table 1. Rendering Statistics - Direct Illumination timings and information. VT refers to visibility testing and subdivision, Irr. refers to irradiance computation.

Figure	Fragments	VT(ms)	Irr.(ms)	Total(ms)
6 (a)	186,031	35	32	72
6 (b)	352,085	70	66	144
6 (c)	786,432	-	199	314

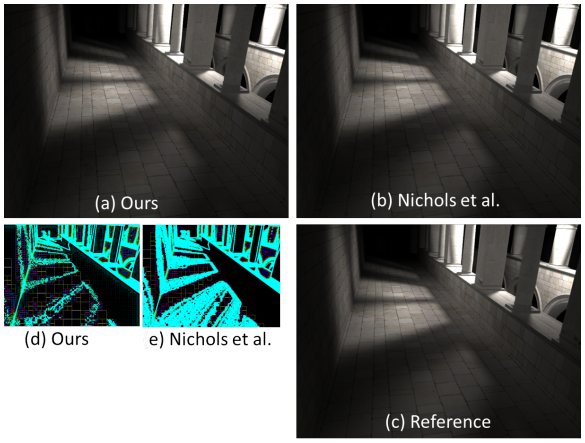


Figure 6. Results of our rendering (a) compared to Nichols et al. [5] in (b) and reference image (c). All images are rendered with 64 samples per fragment. Indirect illumination is overlay-ed on all 3 results. (d) and (e) show the fragment maps.

6. REFERENCES

[1] K. Anjyo, J. P. Lewis, and F. Pighin. Scattered data interpolation for computer graphics. In *ACM SIGGRAPH 2014 Courses*, SIGGRAPH '14, pages 27:1–27:69. ACM, 2014.

[2] J. Bowers, R. Wang, L.-Y. Wei, and D. Maletz. Parallel poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. Graph.*, 29(6):166:1–166:10, Dec. 2010.

[3] C. Crassin, F. Neyret, M. Sainz, S. Green, and E. Eisemann. Interactive indirect illumination using voxel cone tracing: A preview. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 207–207. ACM, 2011.

[4] A. Kaplanyan and C. Dachsbacher. Cascaded light propagation volumes for real-time indirect illumination. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, pages 99–107, 2010.

[5] G. Nichols, R. Penmatsa, and C. Wyman. Interactive, multiresolution image-space rendering for dynamic area lighting. In *Proceedings of the 21st Eurographics Conference on Rendering*, EGSR'10, pages 1279–1288. Eurographics Association, 2010.

[6] G. Nichols and C. Wyman. Interactive indirect illumination using adaptive multiresolution splatting. *IEEE Transactions on Visualization and Computer Graphics*, 16(5):729–741, Sept. 2010.