Screen-Space Bias Compensation for Interactive High-Quality Global Illumination with Virtual Point Lights

Jan Novák Thomas Engelhardt Carsten Dachsbacher *
Computer Graphics Group / Karlsruhe Institute of Technology

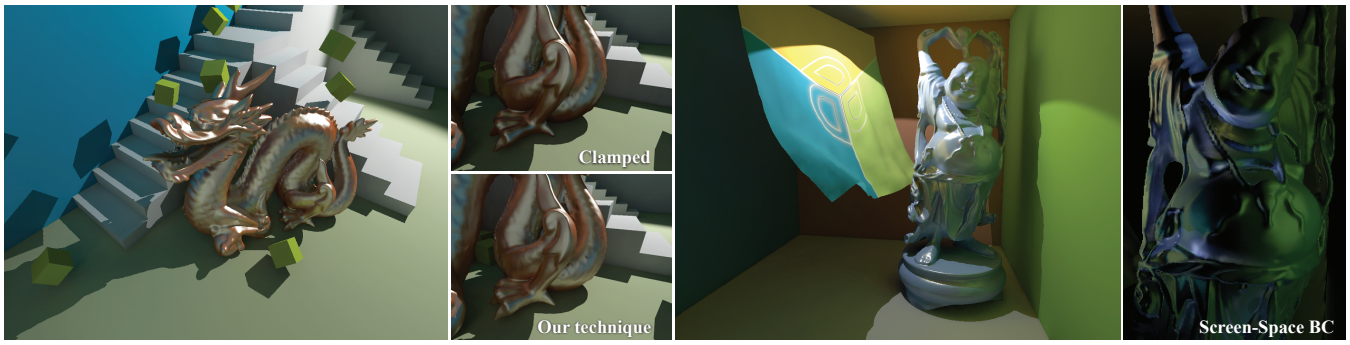


Figure 1: *Our algorithm enables high quality rendering for VPL based methods by restoring the short distance illumination that is usually clamped to avoid artifacts. By employing our hierarchical integration scheme in screen space, which takes about 27 ms for these images, we can efficiently compute the missing energy (right image), and obtain results that are comparable to offline renderers at interactive frame rates (10 fps for the Dragon, and 7.2 fps for the Happy Buddha scene).*

Abstract

In this paper we present a method that targets high-quality global illumination at interactive frame rates. As many techniques in this context, our method is based on instant radiosity, which represents the indirect illumination in a scene with a set of virtual point lights, and therefore enables efficient GPU rendering. Instant radiosity captures light transport over larger distances well, but it requires clamping of the point lights' contribution to avoid bright splotches on nearby surfaces. By bounding the short distance light transport, the algorithm removes some energy and thus introduces bias, that is visible as incorrect darkening near edges and corners. Our method improves the quality and correctness of the rendered images by removing bias using a hierarchical screen space approach. We show that the bias compensation can be formulated as a post-processing step and demonstrate renderings comparable to results from offline algorithms at interactive speed.

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

Keywords: global illumination, instant radiosity, screen space, bias compensation

1 Introduction

Global illumination algorithms simulate light transport in virtual scenes to render photorealistic images. However, due to the inherent complexity of light transport, causing effects such as interreflections, caustics, and subsurface scattering, many global illumination algorithms build on ray tracing and stochastic sampling, and are only feasible for offline rendering. Instant Radiosity [Keller 1997], in contrast, is a GPU-friendly method that casts the global illumination problem onto computing direct illumination from virtual point lights (VPL), and is thus very well suited for interactive applications. The steadily increasing computational power of GPUs and improvements of the original instant radiosity method, such as imperfect shadow maps [Ritschel et al. 2008], allow us to render thousands of VPLs and achieve images of good quality at interactive speed. However, lighting from VPLs exhibits singularities if the VPLs are close to a surface. These singularities cause high intensity peaks that appear in the image as distracting bright splotches. We can remove these artifacts by bounding the contribution of a VPL to surfaces in its proximity. This, unfortunately, removes energy from the light transport near edges, corners, and creases, leading to incorrect darkening in these regions.

Kollig et al. [2004] compute a correction term that compensates for this energy loss. Unfortunately, their method requires tracing many additional rays through the scene, which is not feasible without spatial index structures and thus difficult in the context of interactive rendering. The rendering times of their compensation, compared to that of the bounded solution, are orders of magnitude longer, as the compensation can essentially degenerate to path tracing that samples all VPLs at each vertex.

We propose to compute the bias compensation in a GPU-friendly manner in screen space: as the compensation essentially recomputes the missing light transport over short distances, we observe that most of the required information about nearby surfaces can be acquired from screen space representation. Our approach is based on a solid theory obtained from a reformulation of the rendering equation (Section 3.2). To further accelerate the compensation, we use a hierarchical approach that reduces the number of arithmetic and memory operations, and also discuss strategies to suppress artifacts due to sparse or missing sampling of surfaces in screen space. We show several examples demonstrating that our method achieves comparable results to offline rendering algorithms while achieving interactive speed on contemporary GPUs.

*e-mail: {jan.novak,thomas.engelhardt,dachsbacher}@kit.edu

2 Previous Work

A large body of research in global illumination (GI) exists and even a brief overview would exceed the scope of this paper. Therefore, we focus on the most related work here, and we refer the reader to Dutré et al. [2006] for a comprehensive description of global illumination algorithms, and to Dachsbacher and Kautz [2009] for an overview on GI techniques for real-time rendering.

Instant Radiosity Instant radiosity (IR) [Keller 1997] is a widely used algorithm for approximating GI in interactive rendering. IR is a bidirectional method that first creates a set of virtual point lights (VPLs) using random walks that trace light paths starting from the primary light sources. Direct contribution of these VPLs then approximates the entire multi-bounce light transport in the scene. The conceptually simple idea of breaking down GI to point light sources, combined with shadow mapping (the original work used shadow volumes), makes this method GPU-friendly, and it has received a lot of attention in recent years. The shadow computation has significant impact on the rendering performance and thus previous work tried to lower this cost, e.g. by incrementally updating and reusing shadow maps [Laine et al. 2007], using imperfect shadow maps [Ritschel et al. 2008], neglecting visibility for indirect light [Dachsbacher and Stamminger 2006], or ray casting accelerated with grids [Grün 2010]. Wald et al. [2003] also used an IR-based approach in the context of offline rendering.

The main limitation of IR is that it typically can only be used with diffuse or moderately glossy surfaces [Křivánek et al. 2010], the reason being that only relatively few paths (several hundreds) are created and thus the set of all possible light paths is only sparsely sampled. Increasing the numbers of paths captures more complex light transport phenomena, but a high number of VPLs requires different (non-real-time) rendering techniques using hierarchies of VPLs [Walter et al. 2005].

Screen Space Algorithms Another popular class of rendering techniques is GI approximation in screen space. These GPU-friendly techniques are typically used when high performance needs outweigh quality. Many of them are based on reflective shadow maps (RSMs) [Dachsbacher and Stamminger 2005]. Similar to a standard shadow map, an RSM captures directly lit surfaces, but stores additional information, such as position, normal, and material properties, that are required to compute one-bounce indirect illumination. Multi-resolution splatting [Nichols and Wyman 2009] is also based on RSMs computing indirect lighting (without visibility) at lower resolution for smooth surfaces, and more accurately at detailed parts of the image. Image-space radiosity [Nichols et al. 2009] uses a hierarchy for both RSM and image space. Computing ambient occlusion in image-space (e.g. see [Bavoil et al. 2008]) is widely used nowadays, and has been extended by Ritschel et al. [2009b] to account for directional lighting with colored shadows and indirect illumination from nearby surfaces. Micro-rendering [Ritschel et al. 2009a] renders thousands of tiny frame buffers to compute indirect illumination using a point representation of the scene, however, targeting high-quality rendering and interactive speed.

3 Global Illumination with Instant Radiosity

The rendering equation [Kajiya 1986] describes the light transport in a scene, where the light leaving the surface at point $\mathbf{y}$ towards point $\mathbf{x}$ is the sum of emitted and reflected light:

$$L(\mathbf{x}\leftarrow\mathbf{y}) = L_e(\mathbf{x}\leftarrow\mathbf{y}) + \int_A f_r(\mathbf{x}\leftarrow\mathbf{y}\leftarrow\mathbf{z}) G(\mathbf{y}\leftrightarrow\mathbf{z}) V(\mathbf{y}\leftrightarrow\mathbf{z}) L(\mathbf{y}\leftarrow\mathbf{z}) dA.$$

The reflected light is computed by the reflection integral that convolves the incoming light from all surfaces, $\mathbf{z}$, towards $\mathbf{y}$ with the BRDF $f_r(\mathbf{x}\leftarrow\mathbf{y}\leftarrow\mathbf{z})$, the geometry term $G(\mathbf{y}\leftrightarrow\mathbf{z}) = \cos^+\theta_\mathbf{y} \cos^+\theta_\mathbf{z} / \|\mathbf{y}-\mathbf{z}\|^2$, and the binary visibility function $V(\mathbf{y}\leftrightarrow\mathbf{z})$, which is one for mutually visible points and zero otherwise.

In the following, we will use the convenient operator notation of Arvo et al. [1994] to derive our method. The *transport operator* $\mathbf{T}$ represents the reflection integral:

$$(\mathbf{T}L)(\mathbf{x}\leftarrow\mathbf{y}) = \int_A f_r(\mathbf{x}\leftarrow\mathbf{y}\leftarrow\mathbf{z}) G(\mathbf{y}\leftrightarrow\mathbf{z}) V(\mathbf{y}\leftrightarrow\mathbf{z}) L(\mathbf{y}\leftarrow\mathbf{z}) dA,$$

thus the rendering equation can be written as: $L = L_e + \mathbf{T}L$. Recursively expanding this leads to the Neumann series, which formulates the light incident at a surface as the sum over light that has been reflected n-times:

$$L = \sum_{n=0}^{\infty} \mathbf{T}^n L_e$$

3.1 Instant Radiosity and Bias Compensation

Computing a solution to the rendering equation, e.g. using path tracing, is typically costly due to the inherent recursive nature of the light transport. In order to accelerate the computation, instant radiosity [Keller 1997] approximates global illumination using a sparse set of precomputed VPLs. The algorithm starts by computing (quasi-)random paths from primary light sources and creating a VPL at every surface location visited by a path. The direct lighting due to all VPLs then accounts for the indirect illumination in the scene. In terms of the operator notation, instant radiosity represents all indirect illumination with VPLs ($\hat{L}$) and replaces all but the first two terms of the Neumann series:

$$L = \underbrace{L_e}_{\text{emission}} + \underbrace{\mathbf{T}L_e}_{\text{direct illum.}} + \underbrace{\mathbf{T}\hat{L}}_{\text{indirect illum.}}. \quad (1)$$

One of the main problems of this approach is that lighting from VPLs exhibits a singularity in the geometry term $G(\mathbf{x}\leftrightarrow\mathbf{y})$ resulting in high intensity peaks when a shading point is close to a VPL. This leads to distracting artifacts usually seen as bright splotches in the rendered image. They are typically avoided by bounding the geometry term to an arbitrary bound b: $G_b(\mathbf{x}\leftrightarrow\mathbf{y}) = \min(G(\mathbf{x}\leftrightarrow\mathbf{y}), b)$, resulting in a *bounded transport operator* $\mathbf{T}_b$. Bounding, however, removes energy from the light transport and thus introduces bias.

Kollig et al. [2004] describe a bias compensation technique within the context of ray tracing. For each shading point, they compute a correction term by performing a localized final gathering to compensate for the energy loss due to clamping the VPL's contribution. Even though they search only within a close proximity of the shading point (the domain where clamping occurs), they need to recompute the incident light at nearby surfaces to illuminate the current shading point. As clamping can potentially occur also during the compensation step, the algorithm is recursive and quickly degenerates to distributed ray tracing. A more efficient technique, which creates additional light sources within the bounding region of the shading point, has been recently presented in [Davidovič et al. 2010, to appear], however, their method is still too costly to be applicable in interactive rendering.

3.2 Efficient Compensation using Residual Transport

In the following, we will reformulate the bias compensation in such a way that it does not require recomputing the incident illumination for compensation. To this end, we express the energy difference between the unbounded and bounded transport operators using a new *residual operator* $\mathbf{T}_r = \mathbf{T} - \mathbf{T}_b$.

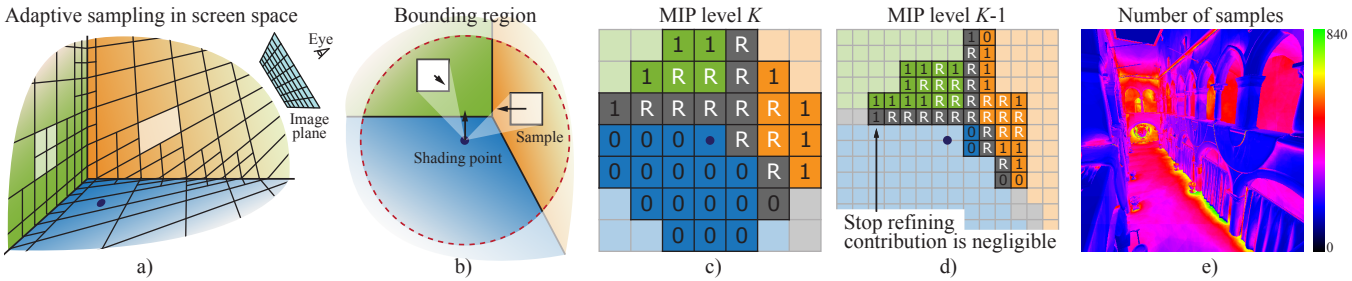


Figure 2: *We compute the bias compensation by adaptively applying the residual operator in screen space and projecting individual samples back into world space (a). We first estimate a bounding region (b) that contains all surfaces with possible non-zero contribution, and then, using a hierarchical traversal, evaluate the contribution of individual neighboring samples (c) and (d). Samples spanning discontinuities or subtending a large projected solid angle are refined (R), until their contribution can be estimated accurately (1), or drops to zero (0). The total number of processed samples for each pixel is for the Crytek Sponza scene shown in (e).*

Similarly to $\mathbf{T}_b$, the residual operator $\mathbf{T}_r$ differs from $\mathbf{T}$ only in the geometry term: $G_r(\mathbf{x}\leftrightarrow\mathbf{y}) = \max(G(\mathbf{x}\leftrightarrow\mathbf{y}) - b, 0)$. From the previous equation, it is obvious that the unbiased transport can be expressed as $\mathbf{T} = \mathbf{T}_r + \mathbf{T}_b$, allowing us to reformulate the indirect illumination term in Equation 1 using the new operators:

$$L = L_e + \mathbf{T}L_e + \mathbf{T}_b\hat{L} + \mathbf{T}_r\hat{L}$$

We observe that $\mathbf{T}_r$ also suffers from singularities when being applied to indirect illumination represented by VPLs ($\hat{L}$). To derive our new bias compensation technique, we remove the source of singularities ($\hat{L}$) and replace it by a general reflected illumination ($L - L_e$) that is not approximated by point light sources. Conceptually, the equation remains the same, the only difference is that $\mathbf{T}_r$ is now integrating reflected light instead of using VPLs:

$$L = L_e + \mathbf{T}L_e + \mathbf{T}_b\hat{L} + \mathbf{T}_r(L - L_e) \tag{2}$$

By recursively expanding Equation 2 we obtain our novel unbiased formulation of the rendering equation for IR:

$$L = L_e + \sum_{i=0}^{\infty} \mathbf{T}_r^i(\mathbf{T}L_e + \mathbf{T}_b\hat{L}) \tag{3}$$

This means that we can compute the unbiased solution as an infinite sum of residual operators that are recursively applied to direct illumination and clamped indirect illumination. That is, we can approximate GI using VPLs with clamping, and afterwards compensate the bias using only this information. The first term of the sum represents the direct and the clamped indirect illumination. Every further summand represents the (clamped) compensation for the clamped contribution of the previous step. As clamping can occur at each iteration, the results will be unbiased only for infinite sums.

In practice, we will only evaluate a finite number of iterations. The error ϵ^N introduced by considering only the first N iterations can be expressed as a sum over all omitted higher order terms, or even more concisely as $\epsilon^N = \mathbf{T}_r^{N+1}(\mathbf{T}L_e + \mathbf{T}\hat{L})$. Notice that the error measure uses the complete transport operator $\mathbf{T}$ for the VPL lighting, therefore no further compensation is necessary. An important observation is that the energy gain due to the compensation is $(N+1)$-times convolved with the BRDF, and therefore dropping exponentially with increasing N. For practical applications, this translates into choosing N such that the *visible* bias is removed. In our scenes we used 1 to 3 iterations, which yielded nearly indistinguishable results from unbiased reference solutions.

Using our reformulation, we derive a new rendering algorithm with bias compensation that consists of two major steps:

1. We first compute direct and clamped indirect illumination of each shading point by applying $\mathbf{T}$ to light from primary light sources and $\mathbf{T}_b$ to illumination from VPLs, respectively.
2. Next, we apply the residual operator iteratively N-times to compensate for the clamped contribution.

Instead of storing shading points over all surfaces, we will use a screen space approach: we compute illumination for visible surfaces only, and also use exclusively these surfaces to compensate for the bias. We detail the use of our compensation in screen space in the next section.

4 Screen Space Bias Compensation

In this section we describe how to transform our novel formulation of bias compensation into an efficient screen space method, which we denote as screen space bias compensation (SSBC). This is necessary, as the computational expense of compensating using Kollig et al.'s method [Kollig and Keller 2004] typically exceeds the rendering time of the clamped solution by orders of magnitude and is thus not feasible for interactive rendering. Bias compensation in screen space has two major advantages: we can exploit the observations from Section 3.2 and apply the residual transport operator as a post-process that operates on the illumination sampled in screen space. Furthermore, we can easily find all surfaces that potentially contribute energy to the compensation, as these have to be nearby in world space, and thus also in screen space.

4.1 Integration in Screen Space

The transport operators, including $\mathbf{T}_r$, can be formulated as integrals over surfaces. Computing an approximation to $\mathbf{T}_r$ in screen space means that we replace the integral by a sum over a finite number of pixels (we discuss how to handle the inherent limitations of screen space approaches in Section 4.3). For every pixel we obtain its position $\mathbf{z}_i$ and normal $\mathbf{n}_i$ from a G-buffer of the rendered image, and compute the surface area A_i, that the pixel represents in world space, using screen-space derivatives. For a shading point $\mathbf{y}$ we sum over M pixels and get:

$$\mathbf{T}_r\hat{L} \approx \sum_{i=1}^{M} f_r(\mathbf{x}\leftarrow\mathbf{y}\leftarrow\mathbf{z}_i)G_r(\mathbf{y}\leftrightarrow\mathbf{z}_i)V(\mathbf{y}\leftrightarrow\mathbf{z}_i)L(\mathbf{y}\leftarrow\mathbf{z}_i)A_i.$$

As previously mentioned, only surfaces nearby $\mathbf{y}$ contribute to the compensation – otherwise the geometry term $G_r(\mathbf{y}\leftrightarrow\mathbf{z}_i)$ becomes zero. This has two consequences: first, we can estimate the radius of this region in screen space and restrict the sum to pixels therein. Second, we observe that nearby surfaces are rarely occluded and can thus omit the visibility function, as also proposed by Arikan et al. [2005].

To restrict the sum to nearby surfaces, we first estimate the spherical bounding region in world space that contains surfaces contribut-

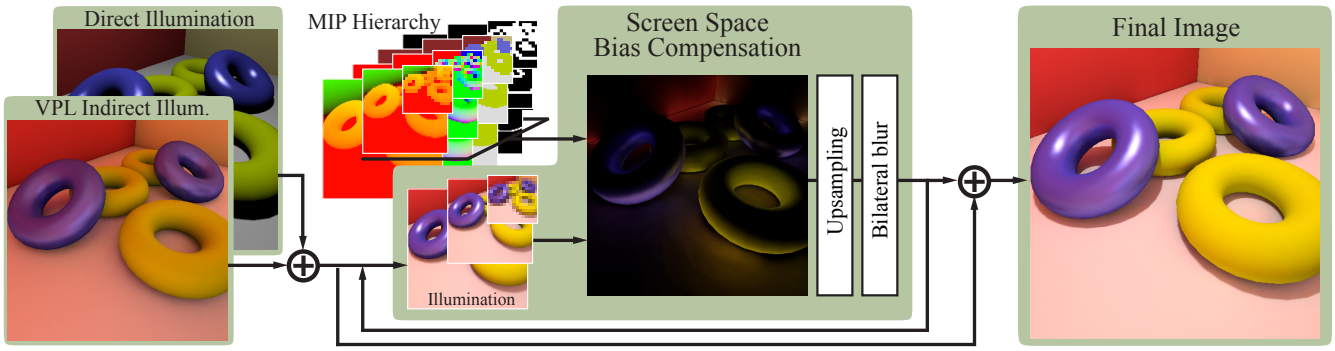


Figure 3: *Our SSBC takes a multi-resolution image of direct and clamped indirect illumination, and the hierarchical G-buffer as input. The compensation result is upsampled and optionally blurred, and then added to the direct and clamped illumination yielding the final image.*

ing to the compensation. A conservative estimation, which does not rely on geometric information in advance, defines the bounding region only based on the distance (assuming the two cosine terms in G_b to be 1) and the bound b of G_b. The radius of the bounding sphere in world space is then $r = 1/\sqrt{b}$, and can be transformed to screen space, e.g. for a perspective projection as $r_s = r/(\tan(\delta/2)\,\|\mathbf{x}-\mathbf{y}\|)$, where δ is the field of view. Note that r_s depends on the distance from the camera and for closer $\mathbf{y}$ the screen space radius becomes bigger spanning more pixels. Since the number of pixels lying inside the bounding bounding region can exceed several thousands (see Figure 4.b), we derive a hierarchical integration scheme that greatly helps to achieve interactive performance.

4.2 Hierarchical Integration

The goal of the hierarchical integration in screen space is to compute the contribution from smooth or more distant (yet still in the bounding region) surfaces using less pixel samples. Figure 2 demonstrates the general idea of adaptively sampling the integration domain and refining where the information is inaccurate. For this we compute a mip-map chain for the G-buffer, which contains averaged positions, normals, and material properties, summed pixel areas, and illumination computed from the bounded light transport, i.e. the color of the pixels. We also compute a discontinuity buffer (and a mip-map chain thereof), similarly to Nichols et al. [2009], to avoid integrating over sharp edges and depth discontinuities.

When integrating the surfaces' contributions, we start on the coarsest mip level (typically a resolution of 64^2 for an image resolution of 1024^2), and determine whether the computation using this sample is accurate enough, or if we have to refine and proceed to the next mip level. In order to avoid spatial and temporal artifacts in dynamic scenes, we refine the integration if:

1. The projected solid angle of a surface corresponding to a sample exceeds a given threshold (typically 0.08 steradians) because it is too large or too close to the shading point $\mathbf{y}$.
2. The current sample spans a discontinuity. Then the position, normal, and area at the coarser mip level do not represent the original geometry correctly.

If at least one of the criteria is met, we omit the sample and refine by recursively integrating over the four corresponding pixels at the next finer level. The second criterion can cause a lot of refinement in areas with negligible contribution, e.g. distant surfaces. Therefore, we only refine at discontinuities when the projected solid angle is higher than a user specified threshold (≈ 0.04). This significantly improves the performance without introducing noticeable artifacts, as it is only effective for samples with low contribution.

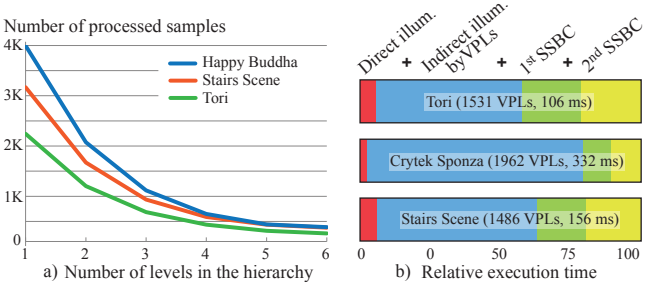


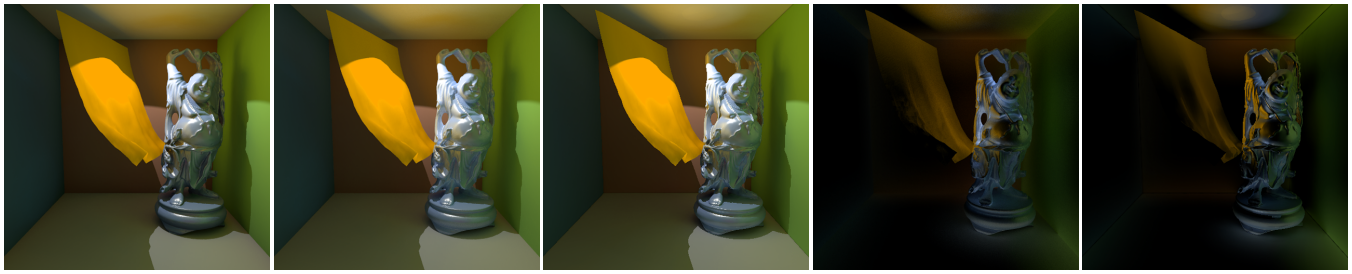
Figure 4: *a) The average number of samples per pixel used for bias compensation is highly dependent on the number of mip levels used for the hierarchical G-buffer. b) Relative time spent on the different tasks: the cost of 2 compensation steps is moderate compared to that of computing illumination from VPLs. The number of VPLs and the total rendering time is given in parentheses.*

4.3 Avoiding Artifacts in Screen-Space Integration

There are inherent problems that are common to all screen space approaches. Information about surfaces can either be missing completely, or sampled too sparsely when surfaces are seen from grazing angles. Hidden surfaces can be revealed using depth peeling or multi-fragment rendering, but trying to acquire and use all this information makes a screen space approach more complicated and less elegant. As our goal is a simple-to-apply algorithm, we focus on the case where we have information on the frontmost surfaces only. In screen space, the information is rather inaccurate for surfaces that are nearly perpendicular to the view direction. Pixels where these surfaces are seen represent a large area in world space with just a few samples in the G-buffer, even on the finest mip level. This can result in overestimated contributions to the bias compensation, and thus to brighter areas in the image (please see the accompanying video). This brightening would sometimes be more distracting than the non-clamped contribution of a nearby VPL, therefore, we decay the contribution of pixels that have the angle between the surface normal and the viewing direction greater than 80 degrees by a quadratic falloff.

5 Implementation Details

We implemented our technique in DirectX 11 and all steps of our bias compensation are executed on the GPU. The only exception are the random walks distributing the VPLs, which are computed on the CPU. Note that this is never the bottleneck, as we only generate up to a few thousands of VPLs.



a) Rendering without BC | b) with BC akin [Kollig and Keller 2004] | c) with 3 SSBC steps | d) BC akin [Kollig and Keller 2004] | e) 3 SSBC steps

Figure 5: *Comparison of our SSBC to a ground-truth solution: (a) offline rendering with clamped indirect illumination without bias compensation (20 min), and (b) with bias compensation [Kollig and Keller 2004] (additional 10.9 hours). (c) shows the result with three SSBC steps (i.e. three iterations in Equation 3). The energy recovered from both compensation methods is shown in (d) and (e), respectively.*

During rendering the illumination from the VPLs is computed using imperfect shadow maps [Ritschel et al. 2008]. The bias compensation is implemented in a compute shader, which enables more efficient handling of the hierarchical refinement via a stack placed in the group-shared memory. The compute shader takes a mip map chain of the G-buffer storing position, normal, area, BRDF, and illumination (direct and clamped VPL contribution) as input. Optionally, we apply an edge-preserving bilateral gaussian blur to the compensation result to avoid blocky artifacts that can appear when using the hierarchical approach with very few samples. When using multiple compensation steps, we always have to use the illumination buffer from the previous step and thus have to update its mip maps as well. Figure 3 shows the workflow of our method.

As the indirect illumination usually exhibits smooth gradations only, we can increase the performance – without noticeable impact on visual quality – by computing the indirect illumination and bias compensation at half resolution. We then use bilateral upsampling as in [Ritschel et al. 2009a] and compute these two components only at full resolution where the upsampling failed.

6 Results

In this section we compare the quality of our results to an implementation of the unbiased instant radiosity algorithm [Kollig and Keller 2004], analyze the rendering performance, and discuss different settings. All renderings have been computed using an ATI Radeon HD 5870 running on an Intel Core i7 860 with 2.8 GHz with 8 GB of RAM. Figure 4a shows the dependency of the number of screen space samples required for the compensation on the number of mip levels of the G-buffer. Without the hierarchical approach, the algorithm gathers the illumination from up to four thousand samples for every pixel. This number quickly decreases as we use more levels of the hierarchy. Figure 4b reports the relative time spent on the individual parts of the pipeline. Here we always used hierarchies with 6 levels computing the compensation for 1024×768 images and performing two compensation steps (green and yellow).

In Figure 5 we compare the quality of the SSBC. For a faithful comparison we replaced the shadow maps in the GPU renderer with ray traced shadows in this figure. The ground-truth rendering took about 11.2 hours to produce results with an acceptable noise level. Our bias compensation obtains very similar results at interactive frame rates (3 SSBC steps at highest quality settings take 550 ms).

Figure 6 demonstrates SSBC for three scenes and illustrates the differences between one and two bias compensation steps, and provides a comparison to clamped and reference solutions. We observe that one compensation step is often sufficient for diffuse surfaces, while a second step still contributes noticeably on glossy surfaces.

7 Conclusions

In this paper we presented a novel method for interactive rendering of high-quality global illumination based on virtual point light methods. We show that bias compensation can be formulated as a post-processing step allowing for efficient screen space approximations. Our method improves the quality and correctness of VPL-based methods by recovering the clamped energy using a hierarchical screen space approach, whereas previous approaches, operating in world space, require ray casting and are orders of magnitude slower. The relative time required for our compensation is only a fraction of the time spent on illuminating the scene with VPLs.

In order to overcome the artifacts stemming from the incomplete information in screen space, our method could be combined with any technique revealing information on hidden surfaces, such as depth peeling, scene voxelization, or surfel injection.These improvements trade speed for quality and might even be suitable for accelerating high-quality bias compensation in offline renderers.

References

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2005. Fast and detailed approximate global illumination by irradiance decomposition. *ACM Trans. on Graphics (Proc. of SIGGRAPH) 24*, 3, 1108–1114.

ARVO, J., TORRANCE, K., AND SMITS, B. 1994. A framework for the analysis of error in global illumination algorithms. In *SIGGRAPH '94*, 75–84.

BAVOIL, L., SAINZ, M., AND DIMITROV, R., 2008. Image-space horizon-based ambient occlusion. ACM SIGGRAPH 2008 talks.

DACHSBACHER, C., AND KAUTZ, J. 2009. Real-time global illumination for dynamic scenes. In *SIGGRAPH '09: ACM SIGGRAPH 2009 Courses*, 1–217.

DACHSBACHER, C., AND STAMMINGER, M. 2005. Reflective shadow maps. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 203–213.

DACHSBACHER, C., AND STAMMINGER, M. 2006. Splatting indirect illumination. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 93–100.

DAVIDOVIČ, T., KŘIVÁNEK, J., HAŠAN, M., SLUSALLEK, P., AND BALA, K. 2010, to appear. Combining global and local lights for high-rank illumination effects. In *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia)*.

DUTRÉ, P., BALA, K., AND BEKAERT, P. 2006. *Advanced Global Illumination*. AK Peters.

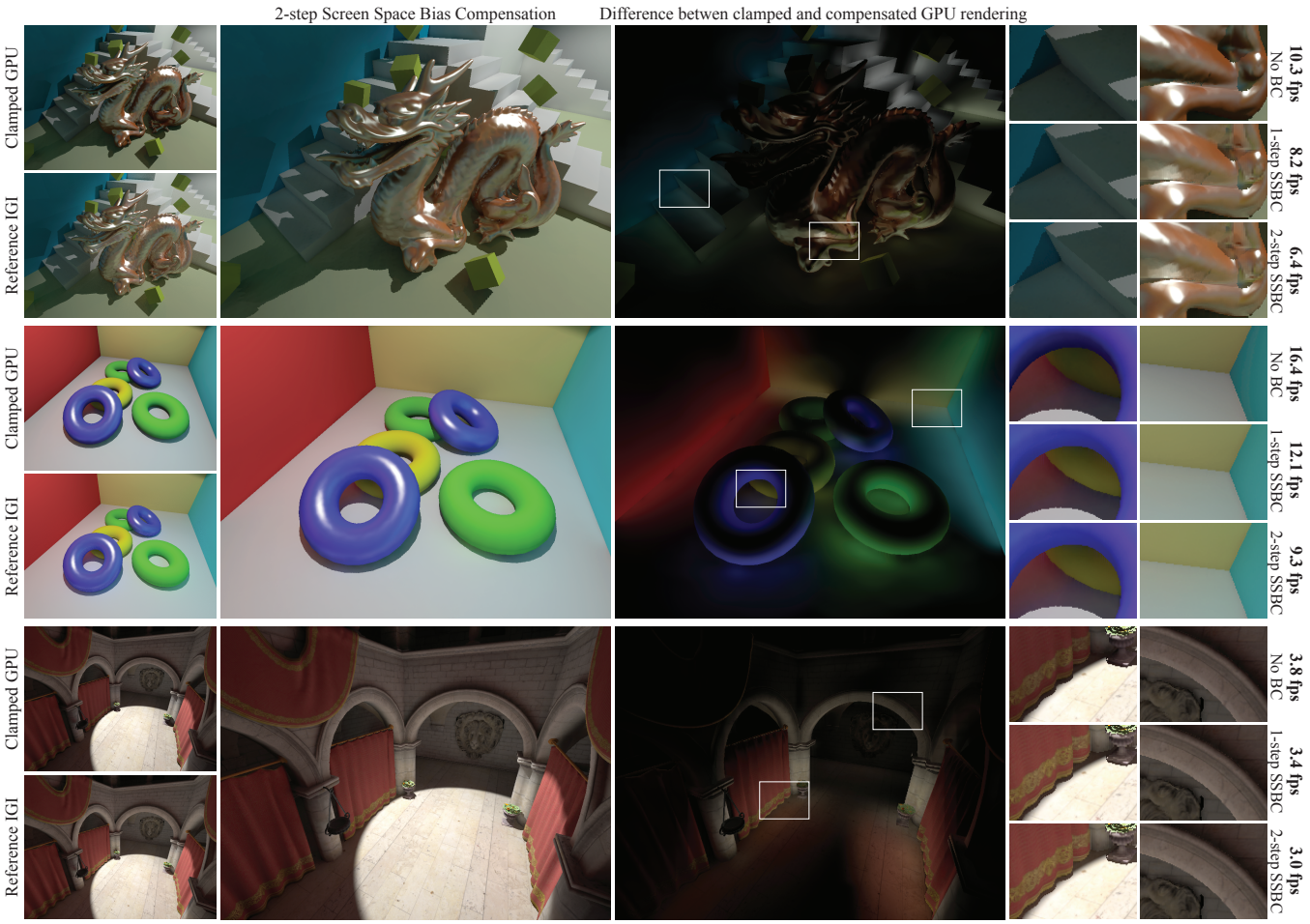


Figure 6: *Three of our test scenes for which we show a clamped GPU and a reference CPU solution (left column), and our technique with 2 compensation steps (middle columns). The insets on the right detail parts, where the compensation contributes significantly. Note that a single compensation step is typically sufficient for diffuse surfaces, while glossy surfaces (e.g. the dragon) benefit from more steps. Each compensation step takes approximately 27 ms for an image resolution of* 1024×768.

GRÜN, H., 2010. Direct3d 11 indirect illumination. Game Developers Conference.

KAJIYA, J. 1986. The rendering equation. In *Computer Graphics (Proc. of SIGGRAPH '86)*, 143–150.

KELLER, A. 1997. Instant radiosity. In *SIGGRAPH '97*, 49–56.

KOLLIG, T., AND KELLER, A. 2004. Illumination in the presence of weak singularities. *Monte Carlo and Quasi-Monte Carlo methods*.

KŘIVÁNEK, J., FERWERDA, J. A., AND BALA, K. 2010. Effects of global illumination approximations on material appearance. *ACM Trans. on Graphics (Proc. of SIGGRAPH) 29*, 4, 1–10.

LAINE, S., SARANSAARI, H., KONTKANEN, J., LEHTINEN, J., AND AILA, T. 2007. Incremental instant radiosity for real-time indirect illumination. In *Proc. of the Eurographics Symposium on Rendering*, 277–286.

NICHOLS, G., AND WYMAN, C. 2009. Multiresolution splatting for indirect illumination. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 83–90.

NICHOLS, G., SHOPF, J., AND WYMAN, C. 2009. Hierarchical image-space radiosity for interactive global illumination. *Computer Graphics Forum 28*, 4, 1141–1149.

RITSCHEL, T., GROSCH, T., KIM, M. H., SEIDEL, H.-P., DACHSBACHER, C., AND KAUTZ, J. 2008. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. on Graphics (Proc. of SIGGRAPH Asia) 27*, 5.

RITSCHEL, T., ENGELHARDT, T., GROSCH, T., SEIDEL, H.-P., KAUTZ, J., AND DACHSBACHER, C. 2009. Micro-rendering for scalable, parallel final gathering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 2009) 28*, 5.

RITSCHEL, T., GROSCH, T., AND SEIDEL, H.-P. 2009. Approximating dynamic global illumination in image space. In *Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 75–82.

WALD, I., BENTHIN, C., AND SLUSALLEK, P. 2003. Interactive Global Illumination in Complex and Highly Occluded Environments. In *Proc. of Eurographics Symposium on Rendering*, 74–81.

WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. 2005. Lightcuts: A scalable approach to illumination. *ACM Trans. on Graphics (Proc. of SIGGRAPH) 24*, 3, 1098–1107.