Reflective Shadow Maps

Carsten Dachsbacher*
University of Erlangen-Nuremberg

Marc Stamminger†
University of Erlangen-Nuremberg

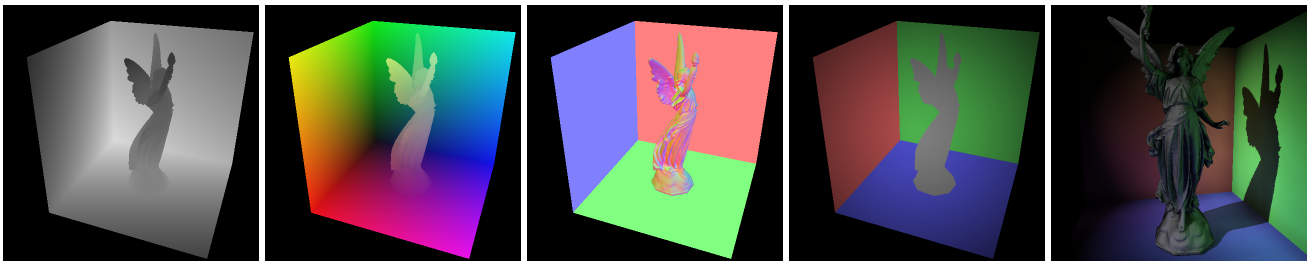

Figure 1: This figure shows the components of the reflective shadow map (depth, world space coordinates, normal, flux) and the resulting image rendered with indirect illumination from the RSM. Note that the angular decrease of flux is shown exaggerated for visualization.

Abstract

In this paper we present "reflective shadow maps", an algorithm for interactive rendering of plausible indirect illumination. A reflective shadow map is an extension to a standard shadow map, where every pixel is considered as an indirect light source. The illumination due to these indirect lights is evaluated on-the-fly using adaptive sampling in a fragment shader. By using screen-space interpolation of the indirect lighting, we achieve interactive rates, even for complex scenes. Since we mainly work in screen space, the additional effort is largely independent of scene complexity. The resulting indirect light is approximate, but leads to plausible results and is suited for dynamic scenes. We describe an implementation on current graphics hardware and show results achieved with our approach.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and textureI.3.3 [Computer Graphics]: Hardware Architecture—Graphics processors

Keywords: indirect illumination, hardware-assisted rendering

1 Introduction

Interactive computer graphics has developed enormously over the last years, mainly driven by the advance of graphics acceleration hardware. Scenes of millions of polygons can be rendered in real-time on consumer-level PC cards nowadays. Programmability allows the inclusion of sophisticated lighting effects. However, these effects are only simple subcases of global illumination, e.g. reflections of distant objects or shadows of point lights. Real global illumination, however, generates subtle, but also important effects that are mandatory to achieve realism.

Unfortunately, due to their global nature, full global illumination and interactivity are usually incompatible. Ray Tracing and Radiosity—just to mention the two main classes of global illumination algorithms—require minutes or hours to generate a single image with full global illumination. Recently, there has been remarkable effort to make ray tracing interactive (e.g. [Wald et al. 2003]). Compute clusters are necessary to achieve interactivity at good image resolution and dynamic scenes are difficult to handle, because they require to update the ray casting acceleration structures for every frame. Radiosity computation times are even further from interactive. Anyhow, a once computed radiosity solution can be rendered from arbitrary view points quickly, but, as soons as objects move, the update of the solution becomes very expensive again.

It has been observed that for many purposes, global illumination solutions do not need to be precise, but only plausible. In this paper, we describe a method to compute a rough approximation for the one-bounce indirect light in a scene. Our method is based on the idea of the shadow map. In a first pass, we render the scene from the view of the light source (for now, we assume that we have only one spot or parallel light source in our scene). The resulting depth buffer is called *shadow map*, and can be used to generate shadows. In a *reflective shadow map*, with every pixel, we additionally store the light reflected off the hit surface. We interpret each of the pixels as a small area light source that illuminates the scene. In this paper, we describe how the illumination due to this large set of light sources can be computed efficiently and coherently, resulting in approximate, yet plausible and coherent indirect light.

2 Previous Work

Shadow maps [Williams 1978; Reeves et al. 1987] and shadow volumes [Crow 1977] are the standard shadowing algorithms for interactive applications. Recently, there have been extensions of both approaches to area lights [Assarsson and Akenine-Möller 2003; Chan and Durand 2003; Wyman and Hansen 2003]. Sometimes, such soft shadows are already referred to as 'global illumination'. In this paper, we concentrate on indirect illumination from point lights, but our approach can easily be combined with any of these soft shadow techniques.

*e-mail: dachsbacher@cs.fau.de
†e-mail: stamminger@cs.fau.de

Others generate global illumination images at interactive rates, but they rely on costly precomputations and are thus not suited for dynamic scenes [Walter et al. 1997; Sloan et al. 2002; Bala et al. 2003]. In Instant Radiosity [Keller 1997], the indirect light is represented by a set of point lights, the contributions of which are gathered using many rendering passes. With Instant Radiosity, dynamic objects and lights are possible, but many rendering passes are required.

Interactive ray tracing can generate global illumination effects at interactive frame rates [Wald et al. 2001b; Wald et al. 2001a; Wald et al. 2002; Wald et al. 2003]. These approaches still require clusters with several PCs to achieve interactivity at high resolution.

Our approach combines ideas from two previous publications. In [Tabellion and Lamorlette 2004], a global illumination method for offline film production rendering is presented. The authors demonstrate that one-bounce indirect illumination is sufficient in many cases. They generate a texture atlas containing the direct light, and then gather the first bounce indirect light from this texture. The second method we build upon are Translucent Shadow Maps [Dachsbacher and Stamminger 2003]. In this paper, a shadow map is extended such that all pixels in a shadow map are considered as subsurface light sources. By gathering their contribution, translucent lighting can be approximated.

3 Reflective Shadow Maps

Reflective Shadow Maps (RSMs) combine the ideas of [Tabellion and Lamorlette 2004] and [Dachsbacher and Stamminger 2003]. The idea is that we consider all pixels of a shadow map as indirect light sources that generate the one-bounce indirect illumination in a scene. This idea is based on the observation, that if we have a single point light source, all one-bounce indirect illumination is caused by surfaces visible in its shadow map. So in this case, the shadow map contains all information about the indirect lighting, and no radiosity texture atlas as in [Tabellion and Lamorlette 2004] is needed. Thus Reflective Shadow Maps are more similar to Translucent Shadow Maps [Dachsbacher and Stamminger 2003], where the pixels of a shadow map are also considered as point lights.

In the following, we describe what we exactly store in a reflective shadow map (Sect. 3.1), how a reflective shadow map is generated (Sect. 3.2), and how the indirect illumination can be evaluated from it (Sect. 3.3). Since the indirect light evaluation is expensive, we introduce in Sect. 4 a screen-space interpolation method that reduces the number of evaluations and leads to interactive display rates.

3.1 Data

We assume that all surfaces in the scene are diffuse reflectors. An RSM stores with every pixel p the depth value d_p, the world space position x_p, the normal n_p, and the reflected radiant flux Φ_p of the visible surface point (see Fig.1). Every pixel is interpreted as a *pixel light* that illuminates the scene indirectly. The world space position could be recomputed from the pixel coordinates and the depth value, however we can save valuable pixel shader instructions by having the world space positions directly available. The flux Φ_p defines its brightness, and the normal n_p its spatial emission characteristics (see Fig. 2). If we assume that the light source is infinitely small, we can describe the radiant intensity emitted into direction ω as

$$I_p(\omega) = \Phi_p \max\{0, \langle n_p | \omega \rangle\}, \quad \text{where } \langle | \rangle \text{ is the dot product.}$$

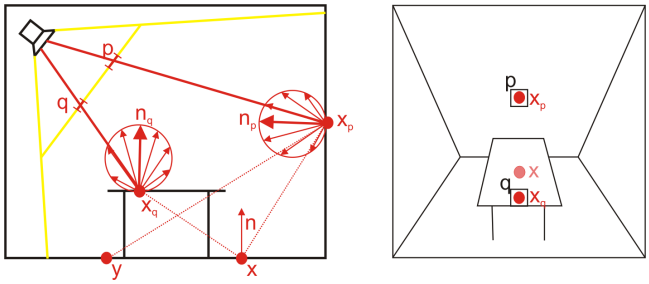

Figure 2: Two indirect pixel lights x_p and x_q corresponding to two RSM pixels p and q

The irradiance at a surface point x with normal n due to pixel light p is thus:

$$E_p(x,n) = \Phi_p \frac{\max\{0, \langle n_p | x - x_p \rangle\} \max\{0, \langle n | x_p - x \rangle\}}{||x - x_p||^4}. \quad (1)$$

Note that we intentionally decided to store radiant flux instead of radiosity or radiance. By this, we don't have to care about the representative area of the light, which makes the generation and the evaluation simpler.

3.2 Generation

An RSM is generated just like a standard shadow map, but with multiple render targets: additionally to the depth buffer storing the depth values d_p, we generate a normal buffer and a world space position buffer with the n_p and x_p, and a flux buffer storing Φ_p. Since Φ_p stores the reflected flux, its computation is simple. First, we have to compute the flux emitted through every pixel. For a uniform parallel light, this is a constant value. For a uniform spot light, this flux decreases with the cosine to the spot direction due to the decreasing solid angle. The reflected flux is then the flux through the pixel times the reflection coefficient of the surface. No distance attenuation or receiver cosine must be computed. As a result, the flux buffer looks like an unshaded image (compare Fig. 1, 4th column).

As it is typical for many global illumination algorithms, problems appear along the common boundary of two walls. In this case, the illumination integral has a singularity, which is difficult to integrate numerically. We found that these problems can be largely reduced, if we move the pixel lights in negative normal direction by some constant offset. This is possible, because we do not consider occlusion for indirect illumination (see below).

3.3 Evaluation

The indirect irradiance at a surface point x with normal n can be approximated by summing up the illumination due to all pixel lights:

$$E(x,n) = \sum_{\text{pixels}p} E_p(x,n) \quad (2)$$

Consider as example Fig. 2. A spot light illuminates a room with a table from the upper left. For a particular light view pixel p, we have a pixel light source at position x_p, illuminating the scene

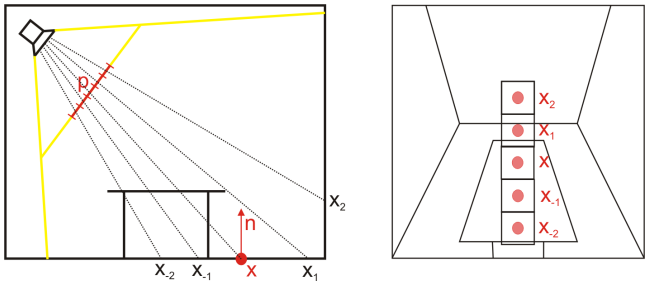

Figure 3: RSM sampling

along its normal n_p. The point x on the floor receives light from x_p according to Equ. 2. For pixel q, the pixel light lies on the table and thus does not illuminate x.

Note that we do not consider occlusion for the indirect light sources, so that in Fig. 2 y is indirectly illuminated by x_p, although x_p is not visible from y. This is a severe approximation, and can lead to very wrong results. However, in many cases it suffices to generate the subtle indirect lighting effects; unprecise results are often acceptable, as long as the indirect lighting effect is visible.

For a typical shadow map, the number of pixels is large (512x512), so the evaluation of the above sum is very expensive and not practical in a realtime context. Instead, we have to reduce the sum to a restricted number of light sources, e.g. 400. We do this using an importance-driven approach, where we try to concentrate the sampling to the relevant pixel lights. The idea can be best described for the example in Fig. 3. x is not directly illuminated, so it is not visible in the shadow map. If we project x into the shadow map, the pixel lights that are closest in world space are also close in the shadow map. x_{-2} and x_{-1} are relatively close, but since their normal points away from x, they do not contribute indirect illumination. x_1 is very close, but lies on the same plane (floor) as x, so it also does not contribute. The most relevant pixel light is x_2.

In general, we can say that the distance between x and a pixel light x_p in the shadow map is a reasonable approximation for their distance in world space. If the depth values with respect to the light source differ significantly, the world space distance is much bigger and we thus overestimate the influence. However, the important indirect lights will always be close, and these must also be close in the shadow map.

So we decided to obtain the pixel light samples as follows: first, we project x into the shadow map ($\rightarrow (s,t)$). We then select pixel lights around (s,t), where the sample density decrease with the squared distance to (s,t). This can easily be achieved by selecting the samples in polar coordinates relative to (s,t), i.e. if ξ_1 and ξ_2 are uniformly distributed random numbers, we select the pixel light at position

$$(s + r_{\max}\xi_1 \sin(2\pi\xi_2), t + r_{\max}\xi_1 \cos(2\pi\xi_2)). \quad (3)$$

We then have to compensate the varying sampling density by weighting the achieved samples with ξ_1^2 (and a final normalization). An example sampling pattern is shown in Fig. 4.

In our implementation, we precompute such a sampling pattern and reuse it for all indirect light computations, which gives us coherency. This temporal coherency reduces flickering in dynamic scenes, however the spatial coherence can result in banding artifacts if the number of samples is not large enough. In our example scenes, 400 samples were sufficient. Because we use a once computed sampling pattern, we could use Poisson sampling to obtain a more even sample distribution.

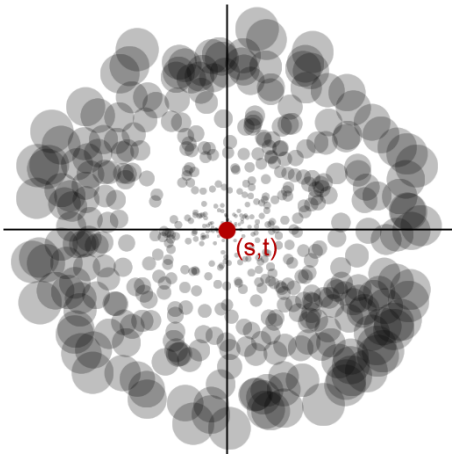

Figure 4: Sampling pattern example. The sample density decreases and the sample weights (visualized by the disk radius) increases with the distance to the center.

4 Screen-Space Interpolation

We compute the direct illumination using per-pixel lighting, where the RSM is bound as standard shadow map. The indirect lighting computation as described above is still too expensive to be performed for every pixel in an interactive application. However, with a simple interpolation scheme we can drastically reduce the number of evaluations and use cheap interpolation for the majority of the pixels.

In a first pass, we compute the indirect illumination for a low-resolution image of the camera view. We then render the full resolution camera view and check for every pixel, whether the indirect light can be interpolated from the four surrounding low-res samples. Such a low-res sample is regarded as suitable for interpolation if the sample's normal is similar to the pixel's normal, and if its world space location is close to the pixel's location. Each sample's contribution is weighted by the factors used for bi-linear interpolation, including a normalization if not all four samples are used. If three or four samples are considered as suitable, we interpolate the indirect illumination. Otherwise, we discard the pixel in this render pass and rather compute the indirect illumination with the complete gathering step in a final pass.

Fig. 5 shows the effectiveness of our solution. Only for the red pixels, interpolation was not sufficient to compute the indirect light. Of course, this effectiveness depends on the scene. On smooth surfaces, interpolation works very well, whereas on unconnected, complex geometry, such as a tree, the interpolation scheme will not be applicable, so that the method falls back to full evaluation.

5 Implementation

We implemented our approach using Direct3D9 with Microsoft's High Level Shader Language and contemporary graphics hardware supporting floating point render targets and programmable vertex and fragment processing. GPUs providing Pixel Shaders 2.0 are sufficient for an implementation of our method, although we used Pixel Shader 3.0 hardware (supporting floating point blending natively) for measuring our results.

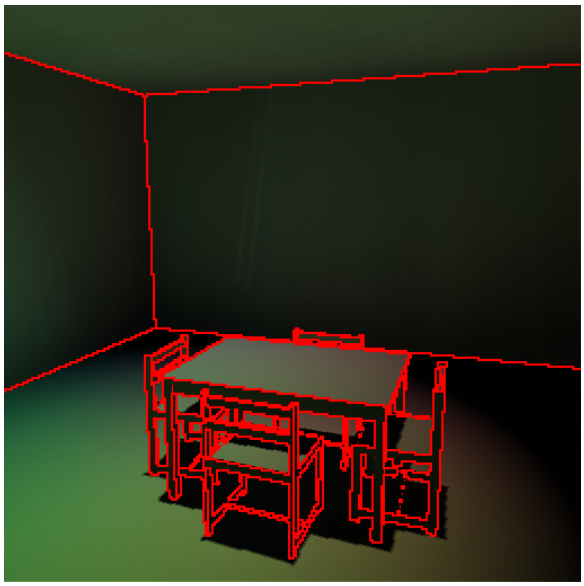

Figure 5: Efficiency of screen space interpolation: only for the red pixels no interpolation was reasonable.

The generation of the final image consists of multiple render passes, most of which are required for the indirect lighting computation. In an initial pass, the scene is rendered from the light's view. But instead of storing only the depth value to the first visible surface through each pixel—like normal shadow maps—we also store the surface normals, its world space location and the energy flux. Although the world space location could be recomputed from the depth value, we chose to store it in the render targets (three in total) to save per-pixel instructions in subsequent render passes.

In order to decouple scene complexity from rendering performance as far as possible and to prevent the execution of complex shaders for hidden surfaces, we apply the direct and indirect illumination as a deferred shading process. We generate textures storing the required information per pixel, namely world space location and normal, its corresponding coordinates in light space and the material parameters. These textures are view-dependent and need to be updated when the camera moves.

The next step is the gathering of the indirect illumination. This requires sampling of the RSM at many different locations. For each sample location the illumination is computed as in Equ. 1 and the contributions are accumulated. When using Pixel Shader 2.0 GPUs, instructions and texture lookups per shader are limited and the computation has to be split up to distinct render passes and intermediate results need to be accumulated. In this case, the sampling positions may be provided by the main application via shader constants. Applying GPUs supporting PS3.0, all sampling can be done in a single loop, where the sample positions are provided in a lookup texture. Both options achieve comparable results, but the latter is faster.

We also implemented the screen-space interpolation scheme as described above. First, a low-resolution image is computed with full evaluation of the indirect lighting. Then, a full resolution image is rendered, where a pixel shader evaluates the number of low-res-samples suitable for interpolation and performs the interpolation if three or four samples are suitable. After this, our image is largely finished, but contains some pixels for which we still have to compute the indirect illumination. This is done with the same Pixel Shaders as used for the low-res image. To avoid the recomputation of all pixels, early z-culling techniques could be used where available. We decided to render the two final render passes as a grid of quadrilaterals on the screen and to use an occlusion query for each quad. This tells us, which quadrilaterals are complete reconstructed (all pixels rendered) in the first step and which contain discarded pixels which need accurate computation of the indirect illumination.

6 Results

We implemented our approach using an ATI Radeon 9700 card for testing the Pixel Shader 2.0 version and a GeForce Quadro FX4000 for the PS3.0 code. Table 1 shows the result of our implementation of the Reflective Shadow Map method using a resolution of 512x512 for the reflective shadow map and the camera view. We measured the difference in performance for various screen subsamplings and number of sample taps (the amount is determined empirically). As it can be taken from the table, the performance does not only depend on the total number of computed indirect illumination values, but also suffers from pipeline stalls through the occlusion query. All timings were collected on a Pentium 4 processor with 2.4GHz and the before mentioned GeForce card. Note that we recompute the light and camera view for each frame, that is, the light source and camera can be moved freely and interactively by the user. The impact on the image quality for various settings is shown in Figure 7. Figure 8 shows the contribution of one-bounce indirect illumination to the table test scene.

scene	fps	samples	indirect computed	low-res first pass	screen quads
table	24.2	112	63299	32×32	32×32
	14.3	224	63299	32×32	32×32
	22.0	112	66120	64×64	32×32
	13.0	224	66120	64×64	32×32
	27.5	112	92745	32×32	16×16
	15.9	224	92745	32×32	16×16
lucy	19.3	112	77613	32×32	32×32
	11.4	224	77613	32×32	32×32
	15.8	112	82031	64×64	32×32
	9.1	224	82031	64×64	32×32
	13.3	112	116320	64×64	16×16
droids	15.1	112	98048	64×64	32×32
	7.9	224	98048	64×64	32×32
	4.2	448	98048	64×64	32×32
	16.4	112	95744	32×32	32×32
	9.2	224	95744	32×32	32×32
	4.8	448	95744	32×32	32×32
	18.2	112	134144	32×32	16×16
	10.3	224	134144	32×32	16×16
	5.5	448	134144	32×32	16×16

Table 1: This table lists the achieved frame rate at a resolution of 512×512, depending on the number of RSM samples, the number of indirect illumination evaluations, the first pass image resolution, and the number of screen space quads. All timings were captured using a GeForce Quadro FX4000.

The application of deferred shading buffers does not completely decouple scene complexity from rendering time. The adaptive refinement of the initial sub-sampling depends on the variation of depth and normal values and thus from scene complexity.

The computation of indirect illumination only considers a few hundred samples. Thus, in the RSM textured surfaces should be ren-

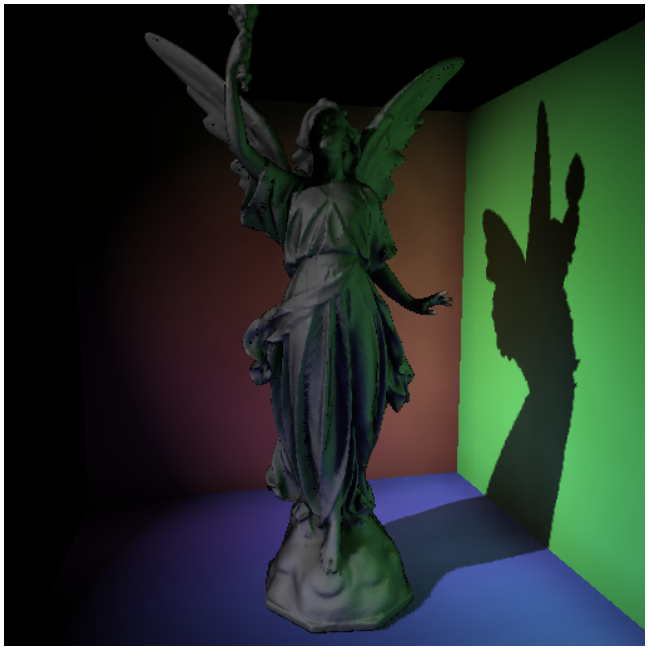

Figure 6: Ambient occlusion is used to simulate the self-shadowing for indirect illumination.

dered using filtered versions of the texture map, otherwise the strongly varying pixel light brightness leads to flickering.

7 Conclusions and Future Work

Our approach extends the concept of shadow maps as done before with the Translucent Shadow Maps by storing additional data, mainly normal and flux information, per texel. It offers the possibility of approximating indirect illumination in dynamic scenes completely on the GPU. It is perfectly suitable to be combined with scenes whose static lighting conditions are pre-computed and stored as lightmaps or vertex-colors. The contribution of dynamic lights and the resulting indirect illumination from them can be simply blended additively to the pre-lit scene.

Assuming sufficient or adaptive tessellation of the scene, it is possible to compute the low-frequency indirect illumination per vertex instead of a per-pixel computation or interpolation. Modern graphics hardware is able to sample textures during execution of vertex shaders, which enables us to render these color values into textures which are then used for final rendering.

Since our method does not handle self-shadowing for the indirect light, we propose to apply ambient occlusion techniques [Landis 2002] to the rendered models, as shown in Figure 6.

Reflective Shadow Maps can be combined with any soft-shadow algorithm for the direct illumination. When using area light sources, the flux buffer—rendered for a point light source—cannot capture penumbra regions. But considering the used approximations, this point has only small impact.

In principle, RSMs can be extended to non-diffuse reflectors. The flux buffer would become a material ID buffer. Whenever the flux of a pixel light is to be computed, the material BRDF must be evaluated to compute the flux reflected towards the current receiver. However, as also noticed in [Tabellion and Lamorlette 2004], this would require significantly higher sample numbers to avoid strong artifacts. Although it would be possible to update the sampling pattern during runtime, since it is stored in shader constants or textures, dynamic importance sampling would be too expensive for the interactive GPU implementation.

In the future we would like to implement our method for other types of light sources. Shadow maps for omni-directional lights can be stored as cube maps. For these, our technique can work, too, when using adapted sampling patterns.

References

ASSARSSON, U., AND AKENINE-MÖLLER, T. 2003. A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Transactions on Graphics 22*, 3 (July), 511–520.

BALA, K., WALTER, B. J., AND GREENBERG, D. P. 2003. Combining edges and points for interactive high-quality rendering. *ACM Transactions on Graphics 22*, 3 (July), 631–640.

CHAN, E., AND DURAND, F. 2003. Rendering fake soft shadows with smoothies. In *Eurographics Symposium on Rendering: 14th Eurographics Workshop on Rendering*, 208–218.

CROW, F. C. 1977. Shadow algorithms for computer graphics. vol. 11, 242–248.

DACHSBACHER, C., AND STAMMINGER, M. 2003. Translucent shadow maps. In *Eurographics Symposium on Rendering: 14th Eurographics Workshop on Rendering*, 197–201.

KELLER, A. 1997. Instant radiosity. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 49–56.

LANDIS, H. 2002. Production-ready global illumination. *Siggraph Course Notes #16, 2002*.

REEVES, W. T., SALESIN, D. H., AND COOK, R. L. 1987. Rendering antialiased shadows with depth maps. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, vol. 21, 283–291.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics 21*, 3 (July), 527–536.

TABELLION, E., AND LAMORLETTE, A. 2004. An approximate global illumination system for computer generated films. *ACM Trans. Graph. 23*, 3, 469–476.

WALD, I., SLUSALLEK, P., AND BENTHIN, C. 2001. Interactive distributed ray tracing of highly complex models. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering*, 277–288.

WALD, I., SLUSALLEK, P., BENTHIN, C., AND WAGNER, M. 2001. Interactive rendering with coherent ray tracing. *Computer Graphics Forum 20*, 3, 153–164.

WALD, I., KOLLIG, T., BENTHIN, C., KELLER, A., AND SLUSALLEK, P. 2002. Interactive global illumination using fast ray tracing. In *Rendering Techniques 2002: 13th Eurographics Workshop on Rendering*, 15–24.

WALD, I., BENTHIN, C., AND SLUSALLEK, P. 2003. Interactive global illumination in complex and highly occluded environments. In *Eurographics Symposium on Rendering: 14th Eurographics Workshop on Rendering*, 74–81.

WALTER, B., ALPPAY, G., LAFORTUNE, E. P. F., FERNANDEZ, S., AND GREENBERG, D. P. 1997. Fitting virtual lights for non-diffuse walkthroughs. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, 45–48.

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 78)*, vol. 12, 270–274.

WYMAN, C., AND HANSEN, C. 2003. Penumbra maps: Approximate soft shadows in real-time. In *Eurographics Symposium on Rendering: 14th Eurographics Workshop on Rendering*, 202–207.

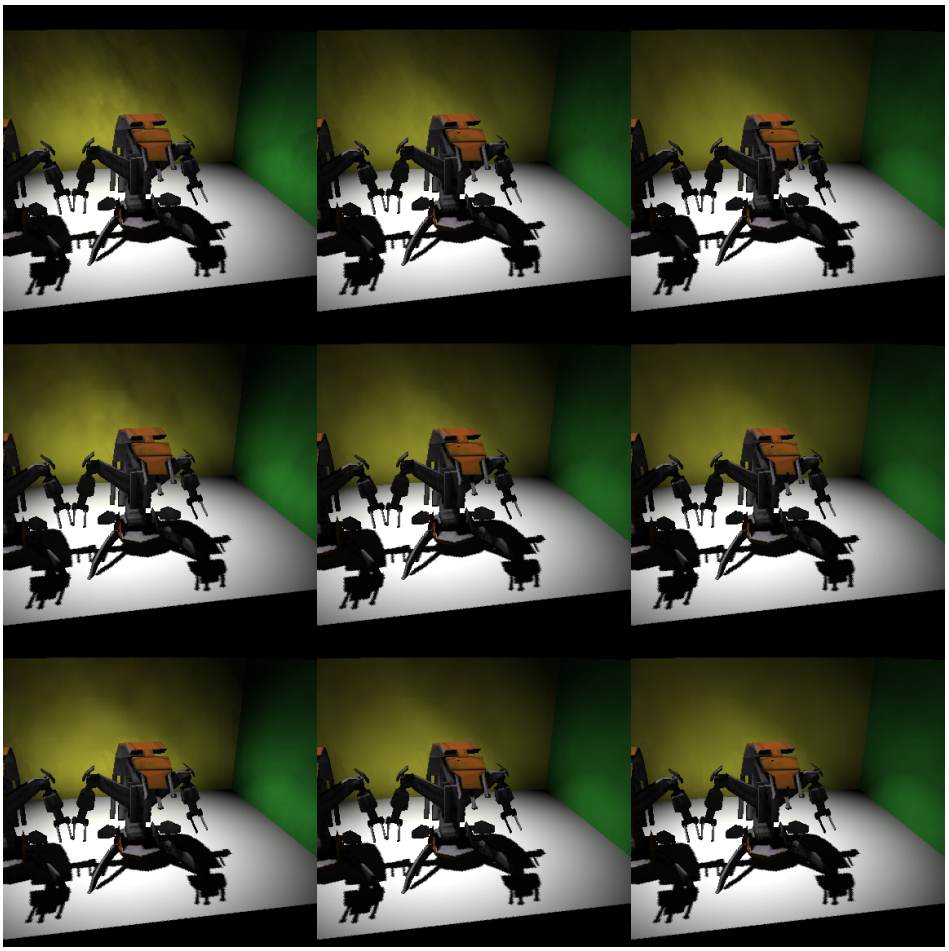

Figure 7: This figure illustrates the impact of screen sub-sampling and filter samples. The number of samples are 112, 224 and 448 for the left, middle and right column. The screen sub-sampling of 128×128 for the top row, 64×64 for the center and 32×32 for bottom.

Figure 8: On the left, local illumination and shadow mapping is shown. The indirect illumination (center image) is approximated from the RSM and combined to obtain the final image (right).