Real-time Indirect Illumination with Clustered Visibility

Zhao Dong[1] Thorsten Grosch[1] Tobias Ritschel[1] Jan Kautz[2] Hans-Peter Seidel[1]

[1]MPI Informatik, Germany [2]University College London, UK

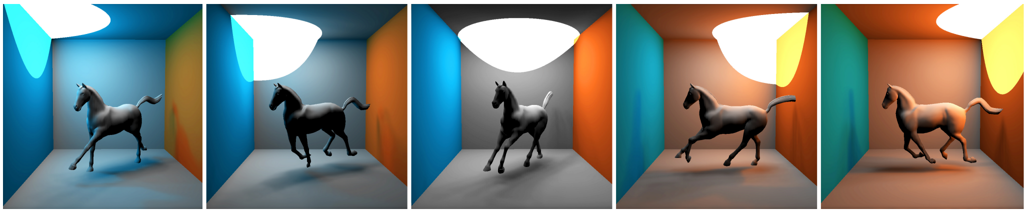

Figure 1: One-bounce diffuse global illumination rendered at 800×800 pixels for a scene with dynamic geometry (17 k faces) and dynamic lighting at 19.7 fps. Our method uses soft shadows from 30 area lights to efficiently compute the indirect visibility.

Abstract

Visibility computation is often the bottleneck when rendering indirect illumination. However, recent methods based on instant radiosity have demonstrated that accurate visibility is not required for indirect illumination. To exploit this insight, we cluster a large number of virtual point lights – which represent the indirect illumination when using instant radiosity – into a small number of virtual area lights. This allows us to compute visibility using recent real-time soft shadow algorithms. Such approximate and fractional from-area visibility is faster to compute and avoids banding when compared to exact binary from-point visibility. Our results show, that the perceptual error of this approximation is negligible and that we achieve real-time frame-rates for large and dynamic scenes.

1 Introduction

Efficient computation of convincing global illumination is a demanding task. To achieve interactive frame-rates most existing methods only consider a substantially restricted sub-problem such as scenes with low-frequency illumination, static geometry or textures. However, there are numerous applications like games, visualization or computer aided design that would benefit from removing such restrictions. Realizing that visibility is often the bottleneck in global illumination methods, fast visibility has raised considerable interest. Instant Radiosity [14] is one option to speed up visibility without imposing many restrictions on the scene. Here, indirect light is approximated with a number of virtual point lights (VPLs). Visibility between these VPLs and the rest of the scene can be efficiently computed using shadow maps and recent graphics hardware (GPUs). However, currently it is not feasible to generate shadow maps for every VPL as required by non-trivial scenes (e.g. in a computer game) at real-time frame-rates. We propose a solution to tackle this problem by introducing *virtual area lights* (VALs). Instead of using a traditional VPL-based instant radiosity algorithm, we cluster the VPLs into a small number of VALs. Visibility between these few VALs and the scene is computed with a very fast soft shadowing technique instead of using hard shadows for a large number of VPLs.

Our contributions include:

- A temporally coherent GPU-based method to cluster a large number of VPLs into a small number of virtual area lights.
- A fast method to render soft shadows from virtual area lights.
- A method to combine illumination from VPLs and visibility from virtual area lights that allows one-bounce global illumination for moderately complex and fully dynamic scenes at interactive to real-time frame-rates.

After reviewing previous work in Section 2, we describe our approach in Section 3. The Instant Radiosity method and its extension to clustered visibility is described in Section 4. Details about the clustering algorithm are given in Section 5, followed by our GPU implementation in Section 6. We show our results in Section 7 before we conclude in Section 8.

2 Previous work

Real-time global illumination is possible with a number of different techniques. One of the early methods is precomputed radiance transfer (PRT) [23]. Most of the PRT variants require static geometry, even though some recent extensions also allow the movement of rigid objects [13]. Often, only low-frequency illumination is supported for dynamic scenes [17]. A different alternative are Instant Radiosity (IR) methods [14]. Here, real-time global illumination is possible for static scenes with a moving light [16] or for dynamic scenes if the visibility is ignored, as shown by Dachsbacher et al. [7]. IR can achieve interactive frame-rates for large scenes by using a crude point-based representation of geometry when computing the shadow map for each VPL [19]. Avoiding visibility tests by sending negative radiance to occluded regions [9] enables global illumination for scenes with limited dynamics. In a similar spirit, visibility can be computed implicitly from a hierarchical radiosity tree [10], but only small scenes yield interactive frame-rates.

Several global illumination methods use the idea of clustered visibility. The lightcuts method [25] imposes a hierarchy over groups of virtual point lights; only a single visibility test is performed for each group. Generating clusters of lights with only a single shadow map for each cluster was used by Hasan et al. [12] for GPU-friendly illumination from many lights. This idea was further extended to visibility cuts [1] and GPU implementations [18] [6]. Kristensen et al. [15] group VPLs into light clouds for real-time relighting of static scenes. In all these approaches, a single binary visibility test for a sender cluster is used. We extend this idea by taking the extent of the sender (cluster of VPLs) into account through the use of a soft shadowing method.

Our algorithm is inspired by the idea of clustering an environment map into a set of area lights for real-time natural illumination [2]. We extend this idea to deal with indirect lighting using a real-time GPU-based clustering technique.

3 Overview

Our goal is to efficiently compute illumination from a large number of virtual points lights (VPLs). Such VPLs are used to simulate global illumination [14] and can be efficiently generated using reflective shadow maps [7]. To compute visibility for every VPL, shadow mapping [26] is popular but has two limitations: the entire scene geometry has to be processed (transformed, clipped, etc.) for every VPL and the total number of depth map pixels is limited. In recent work, visibility was therefore ignored [7,8], approximated [19] or sped up by exploiting temporal coherence [16].

To enable real-time global illumination, we propose to *approximate visibility* by *clustering* the VPLs. Although this significantly reduces the number of required shadow maps, simply drawing a hard shadow for each cluster would result in banding artifacts in penumbra regions. We therefore exploit recent advantages in the computation of real-time soft shadows, i.e. each group of VPLs is treated as a *virtual area light* (VAL) which produces a soft shadow. For the final rendering, we still use all VPLs to illuminate the receiver point, however, visibility is computed from a few VALs only. Fig. 2 shows an overview of our algorithm. Note that we use the VPLs only for indirect illumination in this work. Other possible uses of VPLs, such as for environment map lighting, are not considered here.

4 Instant Radiosity with Clustered Visibility

To compute the global illumination at a point $\mathbf{x}$, instant radiosity approximates the reflected radiance $L(\mathbf{x}, \omega_\mathrm{o})$ in direction ω_o with a set of N VPLs, each carrying a radiant flux Φ_i as

$$L(\mathbf{x}, \omega_\mathrm{o}) = \sum_{i=1}^{N} L_i(\mathbf{x}, \omega_\mathrm{o}) V(\mathbf{p}_i, \mathbf{x}),$$

where

$$L_i(\mathbf{x}, \omega_\mathrm{o}) = f_\mathrm{r}(\mathbf{x}, \omega_i, \omega_\mathrm{o}) \frac{\frac{\Phi_i}{\pi} \cos(\theta_i) \cos(\theta_\mathbf{x})}{d_i^2(\mathbf{x})},$$

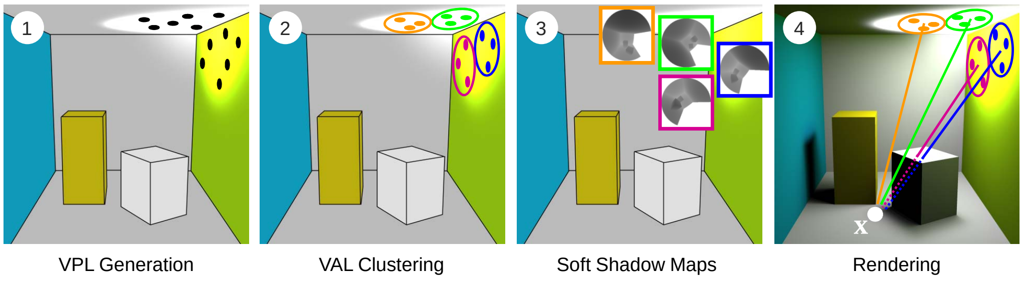

Figure 2: Overview of our algorithm: First, a set of N VPLs is generated to represent the indirect light. In the second step, VALs are generated by grouping the VPLs into M clusters. Next, one (soft) shadow map is rendered for each VAL. The final step is the rendering: The receiver point $\mathbf{x}$ is illuminated by all VPLs. Instead of computing a visibility value for each VPL, only M (fractional) visibility values are computed and shared within each cluster. To avoid banding, each cluster generates a soft shadow, so the penumbra region is composed of soft shadows.

$d_i(\mathbf{x})$ is the distance between VPL i and receiver, θ_i and $\theta_\mathbf{x}$ are the angles between VPL i and receiver normal and the transmission direction. V is the binary visibility term between $\mathbf{x}$ and the VPL position $\mathbf{p}_\mathrm{i}$. $f_\mathrm{r}(\mathbf{x}, \omega_i, \omega_\mathrm{o})$ is the BRDF at position $\mathbf{x}$ from direction ω_i to VPL i in direction ω_o. $\frac{\Phi_i}{\pi}$ is the radiant intensity of VPL i, assuming a Lambertian sender.

Next, the general visibility V (which is more suitable for raytracing [24]) is replaced with visibility $\bar{V}_i$ from VPLs only (which is more suitable for GPUs using shadow maps):

$$L(\mathbf{x},\omega_\mathrm{o}) = \sum_{i=1}^{N} L_i(\mathbf{x}, \omega_\mathrm{o})\bar{V}_i(\mathbf{x}).$$

To accelerate the visibility computation we group the N VPLs into a much lower number of M clusters (VALs) and compute the (now fractional) visibility only for individual VALs:

$$L(\mathbf{x}, \omega_\mathrm{o}) \approx \sum_{i=1}^{N} L_i(\mathbf{x}, \omega_\mathrm{o})\tilde{V}_{\mathcal{C}(i)}(\mathbf{x}).$$

Here we use a mapping function $\mathcal{C} : [1 \ldots N] \rightarrow [1 \ldots M]$ which maps the VPL i to the corresponding virtual area light $\mathcal{C}(i)$. Details on the creation of $\mathcal{C}$ are found in Sec. 5. Instead of computing the visibility $\bar{V}_i(\mathbf{x})$ between VPL i and $\mathbf{x}$, an approximation $\tilde{V}_{\mathcal{C}(i)}(\mathbf{x})$ between the VAL $\mathcal{C}(i)$ and $\mathbf{x}$ is used. This clustering of visibility is based on the insight that indirect light typically contains few high frequencies and estimates can be used without much perceptual difference [19]. Please note that each receiver point is still illuminated from all N VPLs, only the number of visibility computations is reduced to M.

4.1 Convolution Soft Shadow Maps

To approximate the visibility of one of the M virtual area lights, any soft shadow algorithm can be used (see [11] for a recent survey). Due to its high rendering speed, we selected the Convolution Soft Shadows Maps (CSSM) [2] in our implementation. To keep our description self-contained, we briefly review the CSSM method here.

The basic idea is to compute a soft shadow with percentage closer filtering (PCF), where the filter kernel adapts to the shape of the light source. To avoid a large number of depth comparisons inside a possibly large kernel, pre-filtering is applied to the depth map which allows to express the depth comparison over a large region as a Fourier series. So a convolution with a kernel w can be used to compute the clustered visibility:

$$\begin{aligned}\tilde{V}_{\mathcal{C}(i)}(\mathbf{x}) &= [w * f(d_{\mathcal{C}(i)}(\mathbf{x}), z))](\mathbf{p}) \\ &\approx \sum_{j=1}^{K} a_j(d_{\mathcal{C}(i)}(\mathbf{x}))[w * B_j](\mathbf{p})\end{aligned}$$

Here, a step function f is used, which returns 0 or 1, depending on the depth comparison between the distance to the VAL $d_{\mathcal{C}(i)}(\mathbf{x})$ and the depth value z stored in the shadow map at pixel $\mathbf{p}$. B_j are the basis images [3] and a_j are the corresponding coefficients. As shown in [3], a low number of coefficients

K is sufficient for good quality, significantly reducing the number of filter operations. To compute the appropriate filter size for a given area light and a receiver point $\mathbf{x}$, the average blocker depth z_{avg} between the area light and $\mathbf{x}$ must be determined. To this end, an initial search region is estimated by projecting the area light into the near plane of the shadow map. Inside this region, the medium blocker distance is computed by averaging all depth values inside the kernel which are smaller than the current depth value in the center. Again, determining z_{avg} can be efficiently implemented by pre-convolving the depth maps. Now, the penumbra size is back-projected into the shadow map which results in the filter kernel w for PCF. See [2] for details.

4.2 CSSM with parabolic projection

Annen et al. [2] demonstrate that environment map lighting can be efficiently rendered by approximating the map with a number of area lights and then using CSSMs to render each of the area lights. We generalize this approach to *dynamic local* area lights for indirect illumination. Given the M clusters of N VPLs, we place an area light at each cluster center.

Since a diffuse surface reflects towards the whole upper hemisphere, both the perspective and the orthographic projection are not sufficient to compute visibility of an area light representing a cluster of VPLs. Instead, we use a shadow map with a *parabolic* projection, where the sender is oriented around the surface normal [4]. Parabolic convolution soft shadow maps can be realized as follows. First, an initial filter size is estimated from the solid angle of the current sender VAL. While a VPL does not define an area, a VAL allows for such a computation. Then, the average z value z_{avg} is determined in the same way as for a perspective CSSM. The penumbra size p is then estimated from z_{avg} as shown in Fig. 3:

$$p \cdot \cos(\beta) = (d - z_{\text{avg}}) \frac{\Delta}{z_{\text{avg}}},$$

where d is the distance between sender midpoint and receiver point $\mathbf{x}$, Δ is the size of the sender and β is the slope of the receiver surface, viewed from the sender midpoint. Given the penumbra size, the size of the filter kernel in the shadow map is adjusted to the angle α of the penumbra, viewed from the center of the area light:

$$\alpha = \arctan(\frac{p \cdot \cos(\beta)}{d}).$$

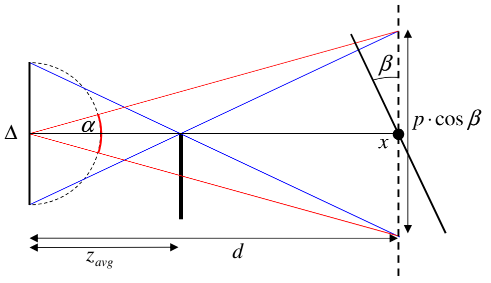

Figure 3: Determining the filter size for a paraboloid map.

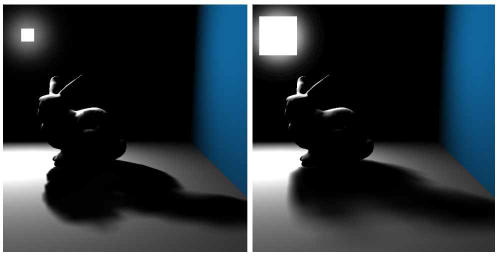

Figure 4: Soft shadows generated with the parabolic CSSM method, rendered with more than 200 fps. The left image shows a small area light, a larger emitter is used in the right image.

Since texture coordinates range from 0 to 1, the size of the filter kernel w can be estimated as $^{\alpha}/_{\pi}$, the fraction between α and the semi-circle π. Fig. 4 shows examples of different soft shadows computed with this approach.

4.2.1 Discussion

Since parabolic maps use a non-linear projection, it is not correct to approximate the projection of the sender with a squared filter region, which is effectively what CSSMs do. We found the resulting visible error to be small, even for difficult cases as shown in Fig. 5. For indirect illumination these errors are acceptable, since many indirect shadows overlap, hiding these artifacts in most cases.

5 Clustering

To accelerate the visibility computation for indirect illumination, we group VPLs with similar normals and similar positions into clusters (VALs), i.e., we

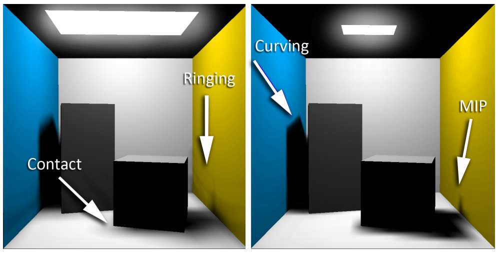

Figure 5: Parabolic CSSM limitations: For very large senders, ringing artifacts can appear (left). Penumbra regions are curved when viewed from a grazing angle of a large sender (right). We also inherit problems at contact shadows (left) and MIP discretization (right) from CSMs. Since the indirect illumination consists of many soft shadows, these artifacts are hidden.

compute the mapping $\mathcal{C}$. We use a variant of the k-means clustering [5] because it is fast and yields good results. After clustering, the position and normal of each VAL are computed by averaging the positions and normals of the contained VPLs. For rendering soft shadows, we additionally compute the area for each VAL (details are described in Sec. 6).

5.1 Clustering criterion

Clustering a set of points with k-means consists of two steps. In a first step, starting from arbitrary cluster centers, each point is assigned to the cluster with the minimum distance to its center . In a second step, each cluster center is recomputed as the average of all point positions assigned to this cluster. These two steps are repeated until convergence.

In our case VPLs must be assigned to VAL clusters. For grouping VPLs into appropriate clusters, *position* and *normal* of the VPL are taken into account. The distance d between a VPL and a cluster center is therefore computed as:

$$d = w_{\mathrm{x}}\Delta x + w_{\alpha}\Delta\alpha.$$

where Δx is the euclidean distance between a VPL and the cluster center and $\Delta\alpha$ is the angle between the VPL normal and the cluster normal. Each term gets a user-defined weight w_{x} and w_{α}. In this way, we create clusters which group nearby VPLs with similar normals. Fig. 6 shows how the different weights affect the clustering. In our examples we use the weights $w_{\mathrm{x}} = 0.7$, $w_{\alpha} = 0.3$.

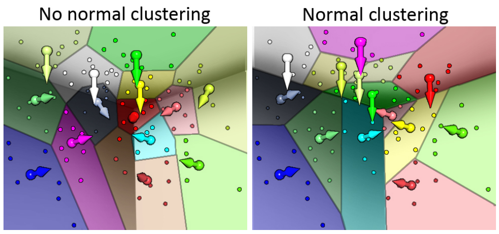

Figure 6: Using k-means clustering simply based on the euclidean distance between the points results in clusters with varying VPL normals, often located at edges (left). When including the angle between the normals in the distance function, planar groups of VPLs can be formed (right).

Including the normals in clustering is important because artifacts in the VAL plane can appear for clusters with different normals. Since the illumination is computed from all VPLs inside the cluster, the illumination may be non-zero at 90 degrees from the cluster normal (see Fig. 7). Because there is no visibility information in the negative halfspace of the VAL, full visibility must be assumed here. If there is a blocker crossing the VAL halfspace, a discontinuity appears, because the blocker is ignored in the negative halfspace of the VAL.

Moreover, the cluster center can be located *inside* the geometry (see Fig. 7). To avoid completely occluded VALs, geometry located near the cluster center has to be ignored, which can result in the loss of some existing shadows. Due to all these problems, groups of VPLs with *similar normals* should be preferred which is achieved by giving them a high weight in the clustering.

5.2 Temporal coherence

To avoid flickering, the clustering between two successive frames should be similar. To achieve this, a simple strategy would be to use the clustering from the *previous* frame as a starting value for the k-means clustering of the *current* frame. In most cases, there are only small changes in light and geometry, so most VPL positions are similar and this quickly converges to a temporally coherent solution.

However, we observed that clusters can be lost, because their center position is in a bad location and all VPLs are assigned to a different cluster center. Fig. 8 (left) shows such a case, here a spot light moves from the wall to the ceiling: Because normals

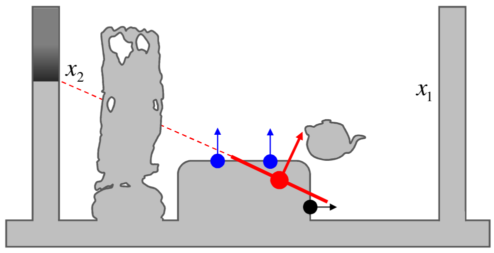

Figure 7: Using VALs with varying VPL normals introduces two problems (in this example, three VPLs are grouped into one VAL cluster): First, the cluster center is located inside the geometry, so nearby geometry must be ignored to avoid incorrect self-shadowing. When introducing such a bias, real occluders like the teapot may be clipped away and the shadow at point x_1 disappears. Secondly, discontinuities in the shadow can appear because the illumination is computed from all VPLs: In the example, point x_2 is in the positive half-space of the blue VPLs and the VAL shadow map correctly detects a shadow above x_2. But there is no occlusion information in the negative half-space of the VAL, so everything below x_2 is assumed to be visible. Consequently, the region below x_2 is incorrectly illuminated by the two blue VPLs.

are taken into account, all VPLs on the ceiling tend to be grouped into only a few clusters on the ceiling. Several other cluster centers are still located on the wall. Due to the different normals, the distance of any VPL to these centers is bigger than the distance to one of the few clusters on the ceiling. This means that several clusters remain *empty*. When moving the light source, more and more cluster centers stay at an old position, without any VPL assigned to it, and the total number of used clusters decreases over time. If the light moves back to and old position, an empty cluster might be reactivated, otherwise it will never be used again.

To overcome this problem, we do not reuse the clusters from the last frame, but restart k-means from an *identical, initial cluster assignment* at each frame. Since our VPLs are generated from a sequence of Quasi-Monte-Carlo random numbers (see Sec. 6), all VPLs are placed to similar positions in each frame in case of small movements of light source or geometry. This means that if we use initial clusters based on the the same VPLs every frame, the k-means algorithm

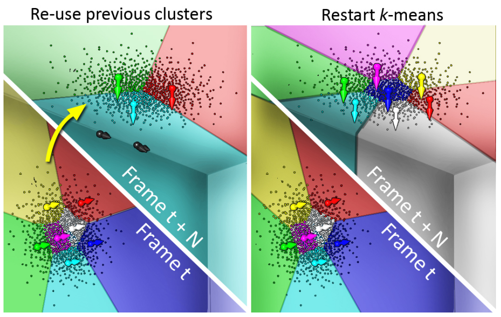

Figure 8: A spotlight is moving (arrow) from the wall (frame t, lower half) to the ceiling (frame $t + N$, upper half). Using k-means clustering with the information from frame t, the number of clusters decreases when moving the spot towards the ceiling, as shown on the left. Since normals are taken into account, the distance of any VPL to such a center is too big, so all VPLs are grouped into a few large clusters on the ceiling. To overcome this problem, k-means is restarted using the same initial VPL to cluster assignments each frame. As shown on the right, the number of clusters stays constant.

will converge to a similar result, as shown in Fig. 8 (right). Although this increases the total number of k-means iterations, the total rendering time is nearly unaffected (see Sec. 7). The accompanying video shows that our clustering strategy leads to virtual area lights that smoothly float over the surface. The clustering always stays temporally coherent, even in case of animated scenes.

6 GPU-Based Rendering from Clustered Visibility

We use a deferred shading renderer, in order to ensure that the expensive indirect illumination is only computed once for every pixel. Geometry is rendered into screen-sized textures for storing position, normal, material and direct illumination, which are then used during the computation of indirect illumination.

VPL Generation We render a reflective shadow map (RSM) from the light's point of view [7] (a cube map is used for point lights and a single texture for projective lights) and sample it using a low-discrepancy sampling pattern (Halton sequence) to convert it into N VPLs (Sec. 4). To this end the RSM is fetched at N Halton-distributed locations using point sampling and the resulting position, normal and color is stored into three N-texel output textures.

Clustering Cluster information is stored in four M-texel textures for position, normal, irradiance and a count of how many VPLs map to a cluster. For each frame, information from the VPL at index ${}^{k \cdot N}/_{M}$ is used as the initial guess for VAL k (i.e., as cluster k's center). As mentioned earlier, this ensures temporal coherence.

In every k-means iteration, we use scattering [21] and blending to update clusters. To this end, for each of the N VPLs a point is drawn using the VPL textures (position, normal, radiance) as input and the four VAL information textures (position, normal, irradiance, count) as output. In a vertex shader, every such point traverses all M clusters, computes the distance, finds the one with the minimum distance and scatters its information to the pixel position of that cluster. We use additive blending and write 1s to the count texture. After every iteration, we draw another full-screen quad, that divides position, normal and radiance by the count resulting in the proper average cluster information.

Note, that in this process, we do not store which VPL maps to which VAL. We create this mapping $\mathcal{C}$ in a final pass and store it as a N-texel texture of pointers into the VAL texture. To this end, we loop over all VPLs, compare them to every VAL and output the pointer to the VAL with minimal distance.

For soft shadow computation we need to know the area of each VAL. We define it as the 2D bounding rectangle of the two-dimensional projection s, t of the VPL position onto the plane perpendicular to the average normal of the cluster it maps to.

In summary, we compute an M-texel texture that stores cluster position, normal and area complemented by an N-texel texture that stores the mapping from each VPL to a VAL, i.e., representing $\mathcal{C}$.

Paraboloid CSSM Instancing is used to draw the scene into a texture array of depth maps with a single pass (the resolution of one paraboloid map is set to 256×256). From this depth map texture array we generate an array of Fourier basis textures (4 term, 8 bit) and Fourier basis-z textures (4 term, 16 bit half float) (cf. Sec. 4.1). Finally, a MIP map is built for both the basis and the basis-z texture array.

Indirect Lighting Indirect lighting is computed using interleaved sampling [22]. We use blocks of $8 \times 8 = 64$ pixels with 1024 VPLs that result in ${}^{1024}/_{64} = 16$ VPLs per pixel. While we use VALs for visibility, we still use the full number of VPLs for lighting. So when shading from VPL i we use the VAL at index $\mathcal{C}(i)$ for visibility, looking up $\mathcal{C}$ in the mapping texture.

We use a geometry aware blur to remove the remaining Monte Carlo noise without blurring over edges. As noted by Laine et al. [16], using $\alpha = 10\%$ of the scene's extend and $\beta = 0.8$ seems to work reasonably well for our results.

7 Results and discussion

In the following, we present results rendered at real-time rates with our technique on a 3 GHz CPU with an NVIDIA GeForce 8800 GTX. All scene components can be fully dynamic (geometry, materials, and lights), as no precomputation is required.

Fig. 9 shows a global illumination solution computed with clustered visibility. To illustrate our approach, we included some individual soft shadow images, generated from selected VAL. To verify the correctness of our approach, we successively increase the number of VALs and compare our result with the ground truth solution from instant radiosity and path tracing. As shown in Fig. 10, the resulting images are similar, even if visibility is computed from a very small number of VALs.

The performance for our test scenes is summarized in Tbl. 1. The rendering time of each individual part of our algorithm is described in Tbl. 2.

Scene	Faces	VPLs	VALs	fps
Cornell Box	18	1024	30	20.4
Cornell Horse	17 k	1024	30	19.7
Sponza	98 k	1024	30	13.4
Metaballs	5 k	1024	30	20.7

Table 1: Frame-rates (800×800 pixels).

As shown in Fig. 11, we can display global illumination in an animated game scenario at interactive fame-rates. Our approach allows for extremely dynamic geometry, such as the iso-surface in Fig. 12.

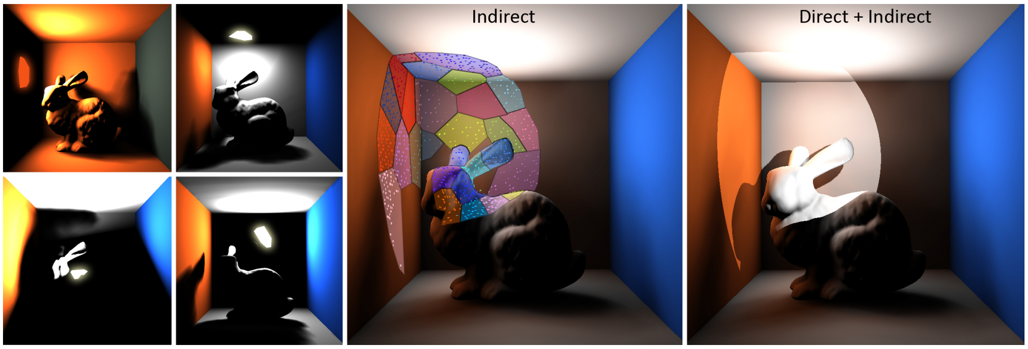

Figure 9: Soft shadows generated for indirect illumination. In this scene a spotlight illuminates the corner of the box, so most of the light is indirect. The images on the left show individual soft shadows from some selected VALs. The complete clustering ($M = 30$ VALs) is shown in the center image. The full global illumination solution is shown on the right.

Step	Time (ms)	Percentage
Deferred rendering	0.4	0.8%
VPL generation	0.1	0.2%
VAL clustering	0.5	1.0%
CSSM	4.2	8.1%
Indirect illumination	35.0	69.0%
Geometry-aware blur	10.7	21.0%

Table 2: Performance breakdown for *Cornell Horse*.

Note, that all pre-computed visibility methods, and even imperfect shadow maps [19], which are restricted to area-preserving deformations, would fail for this scene. We support soft and crisp indirect visibility at the same time, as shown in Fig. 13.

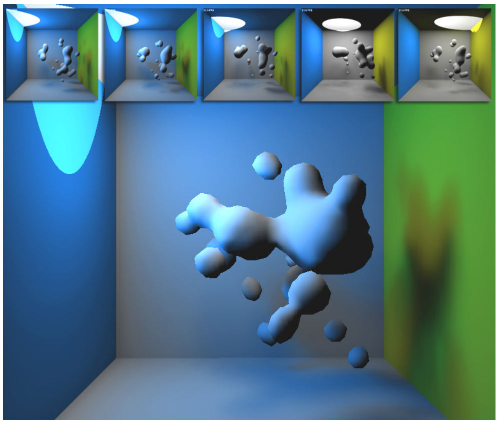

Figure 12: Our method rendering global illumination (20.7 fps) for a scene with dynamic topology (5.1 k faces).

7.1 Discussion

While the use of VALs provides an efficient means to compute indirect illumination, there are some limitations. We currently use reflective shadow maps to generate VPLs [7], restricting us to point and spot lights. The efficiency of the VALs hinges on using rather low-resolution CSSMs, which in turn means that we cannot resolve very thin indirect shadows. Furthermore, we inherit other CSSM limitations, such as difficulties to resolve contact shadows (see [2]). Extending image space shadow bias removal [20] to soft shadows is future work. If an insufficient number of VALs is used, individual shadows from each VAL might be visible, as can be seen in Fig. 10. Using a sufficient number of VALs prevents any artifacts. Our method also depends on the geometric complexity of the scene, since the scene needs to be rendered once for each VAL. However, it might be possible to reduce this dependency with imperfect shadow maps [19].

In contrast to normal instant radiosity, we are less prone to temporal aliasing, since we can start the clustering process with a sufficient number of VPLs yielding good VAL approximations. Furthermore, there is only one major parameter: the number of VALs, which makes our technique more applicable.

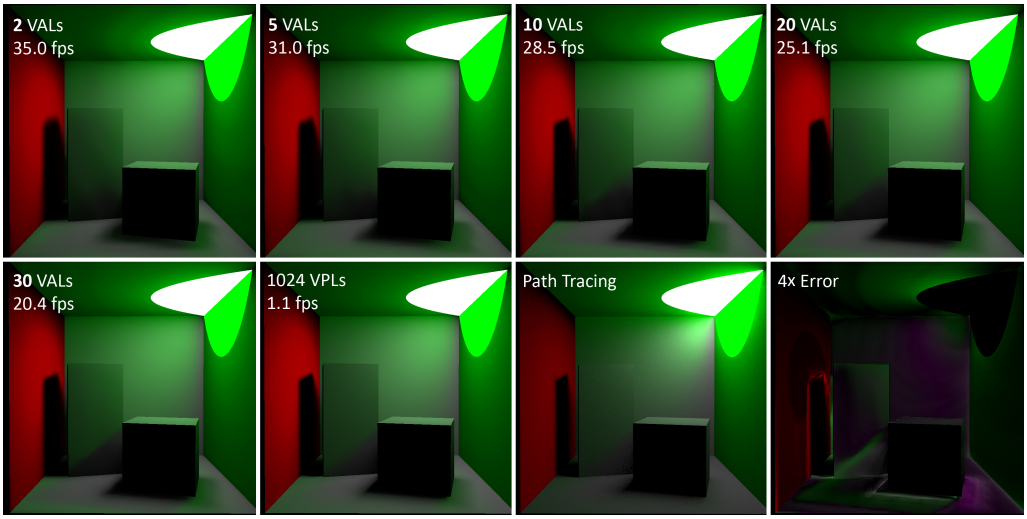

Figure 10: When increasing the number of VALs, the indirect illumination converges to the correct result. The images show (in reading order) 2 to 30 VALs that are used for the indirect visibility. The next two images show an IR solution with a hard shadow for each VPL and a path tracing solution. The difference between our solution (M=30) and the standard IR solution is shown on the bottom right. Note that already a very small number of VALs creates a convincing indirect illumination. An 8×8 G-Buffer was used to reduce the number of VPLs per pixel.

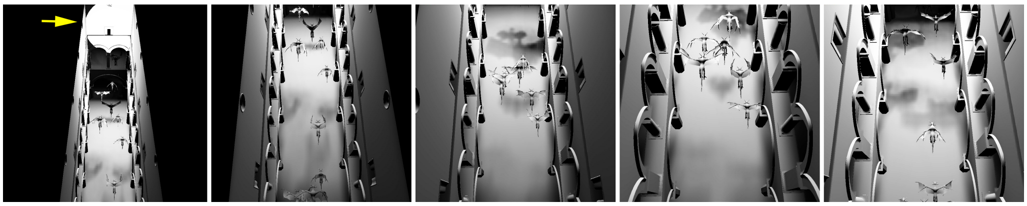

Figure 11: A complex dynamic scene (100 k faces) with multiple animated dragons in *Sponza* (14 fps). Note, how the light bouncing from the back wall dominates (arrow). Please also see the supplemental video.

8 Conclusion and future work

We demonstrated that indirect visibility can be approximated with a small number of area lights in combination with a soft shadowing method. Due to the fast computation time of the soft shadow algorithm we can display approximated indirect illumination at interactive to real time speed without large differences in image quality.

As future work, we will investigate if geometric simplifications can be included on top of the visibility approximations, e.g. if a combination of imperfect shadow maps [19] and coherent soft shadows is possible. Furthermore, we would like to adapt the number of VAL clusters to the illumination complexity, in order to keep the number of clusters at the minimum required number for good visual quality. The extension from one bounce to multiple bounces of light would be an interesting avenue of further research as well as the inclusion of highly glossy materials. Finally, the combination between natural illumination from an environment map and indirect bounces of light should be investigated.

References

[1] Oskar Akerlund, Mattias Unger, and Rui Wang. Precomputed Visibility Cuts for Interactive Relighting

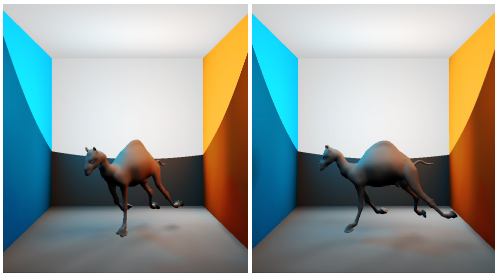

Figure 13: A wide spot casting a soft shadow.

with Dynamic BRDFs. In *Proc. Pacific Graphics*, 161–170, 2007.

[2] Thomas Annen, Zhao Dong, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. Real-Time, All-Frequency Shadows in Dynamic Scenes. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 27(3), 2008.

[3] Thomas Annen, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. Convolution Shadow Maps. In Jan Kautz and Sumanta Pattanaik, editors, *Proc. EGSR*, 51–60, 2007.

[4] Stefan Brabec, Thomas Annen, and Hans-Peter Seidel. Shadow Mapping for Hemispherical and Omnidirectional Light Sources. In *Proc. Computer Graphics International*, 397–408, 2002.

[5] Nathan Carr, Jesse Hall, and John Hart. GPU Algorithms for Radiosity and Subsurface Scattering. In *Proc. Graphics Hardware*, 51–59, 2003.

[6] Ewen Cheslack-Postava, Rui Wang, Oskar Akerlund, and Fabio Pellacini. Fast, Realistic Lighting and Material Design using Nonlinear Cut Approximation. *ACM Trans. Graph. (Proc. SIGGRAPH ASIA 2008)*, 27(5), 2008.

[7] Carsten Dachsbacher and Marc Stamminger. Reflective Shadow Maps. In *Proc. I3D*, 203–213, 2005.

[8] Carsten Dachsbacher and Marc Stamminger. Splatting Indirect Illumination. In *Proc. I3D*, 93–100, 2006.

[9] Carsten Dachsbacher, Marc Stamminger, George Drettakis, and Fredo Durand. Implicit Visibility and Antiradiance for Interactive Global Illumination. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 26(3), 2007.

[10] Zhao Dong, Jan Kautz, Christian Theobalt, and Hans-Peter Seidel. Interactive Global Illumination Using Implicit Visibility. In *Proc. Pacific Graphics*, 77–86, 2007.

[11] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch, and François Sillion. A Survey of Real-Time Soft Shadows Algorithms. *Computer Graphics Forum*, 22(4):753–774, 2003.

[12] Miloš Hašan, Fabio Pellacini, and Kavita Bala. Matrix Row-Column Sampling for the Many-Light Problem. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 26(3):26, 2007.

[13] Kei Iwasaki, Yoshinori Dobashi, Fujiichi Yoshimoto, and Tomoyuki Nishita. Precomputed Radiance Transfer for Dynamic Scenes Taking into Account Light Interreflection. In *Proc. EGSR*, 35–44, 2007.

[14] Alexander Keller. Instant Radiosity. In *SIGGRAPH '97*, 49–56, 1997.

[15] Anders Wang Kristensen, Tomas Akenine-Möller, and Henrik Wann Jensen. Precomputed Local Radiance Transfer for Real-time Lighting Design. In *ACM trans. Graph. (Proc. SIGGRAPH)*, 1208–1215. ACM, 2005.

[16] Samuli Laine, Hannu Saransaari, Janne Kontkanen, Jaakko Lehtinen, and Timo Aila. Incremental Instant Radiosity for Real-Time Indirect Illumination. In *Proc. EGSR*, 277–286, 2007.

[17] Zhong Ren, Rui Wang, John Snyder, Kun Zhou, Xinguo Liu, Bo Sun, Peter-Pike Sloan, Hujun Bao, Qunsheng Peng, and Baining Guo. Real-Time Soft Shadows in Dynamic Scenes using Spherical Harmonic Exponentiation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 25(3):977–986, 2006.

[18] Tobias Ritschel, Thorsten Grosch, Jan Kautz, and Hans-Peter Seidel. Interactive Global Illumination Based on Coherent Surface Shadow Maps. In *Proc. Graphics Interface*, 185–192, 2008.

[19] Tobias Ritschel, Thorsten Grosch, Min H. Kim, Hans-Peter Seidel, Carsten Dachsbacher, and Jan Kautz. Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. *ACM Trans. Graph. (Proc. SIGGRAPH ASIA 2008)*, 27(5), 2008.

[20] Tobias Ritschel, Thorsten Grosch, and Hans-Peter Seidel. Approximating dynamic global illumination in image space. In *Proc. I3D*, 75–82, 2009.

[21] Thorsten Scheuermann and Justin Hensley. Efficient histogram generation using scattering on GPUs. In *Proc. I3D*, 33–37, 2007.

[22] Benjamin Segovia, Jean-Claude Iehl, Richard Mitanchey, and Bernard Péroche. Non-interleaved Deferred Shading of Interleaved Sample Patterns. In *Proc. Graphics Hardware*, 53–60, 2006.

[23] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 21(3):527–536, 2002.

[24] Ingo Wald, Carsten Benthin, and Philipp Slusallek. Interactive Global Illumination in Complex and Highly Occluded Environments. In *Proc. EGSR*, 74–81, 2003.

[25] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: A Scalable Approach to Illumination. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 24(3):1098–1107, 2005.

[26] Lance Williams. Casting Curved Shadows on Curved Surfaces. In *Computer Graphics (Proc. SIGGRAPH)*, 270–274, 1978.