Efficient VAL-based Real-Time Global Illumination

Christoph Weinzierl-Heigl*
Supervised by: Reinhold Preiner†

Institute of Computer Graphics & Algorithms
Vienna University of Technology
Vienna / Austria

Abstract

Due to recent advances in consumer grade graphics hardware the field of global illumination techniques are not limited to offline rendering anymore. Current real-time techniques to approximate global illumination are mostly based on rendering large amounts of virtual point lights (VPLs), however without considering indirect shadows. One of the popular examples that includes indirect shadows are the imperfect shadow maps (ISMs) which are based on VPLs and point-based shadow maps. In our paper, we reduce the number of required light sources and imperfect shadow maps by using virtual area lights (VALs) instead of VPLs. A PCSS based filtering of the visibility information allows to maintain the quality of the indirect shadows in spite of the much smaller number of ISMs. This way, the computational effort is heavily reduced, resulting in rendering speedups of up to 50%.

Keywords: global illumination, imperfect shadow maps, real time, instant radiosity, virtual area lights

1 Introduction

Global illumination (GI) describes the process of lighting in computer graphics in a precise and complete manner, in which not only the light coming from an emitter (*direct illumination*) is considered, but also the numerous reflections of the light rays between objects within the scene (*indirect illumination*). The problem of global illumination was first formalized by Kajiya [10]. His *rendering equation* (RE) solves the entire image synthesis process in a recursive way. Since both sides of the equation are coupled, the RE is analytically unsolvable. It is however possible to approximate the RE by several approaches (Ray-tracing-[21] or Radiosity-based [7]) from which some even allow rendering GI scenes in real-time.

One efficient group of algorithms are based on *instant radiosity* [11] which employs *virtual point lights* (VPLs) to approximate indirect illumination. To account for fast indirect visibility calculation, Ritschel et al. [19] proposed to use point-based shadow maps, so called *imperfect shadow maps* (ISMs).

The problems with the usage of VPLs and ISMs are:

a) The usage of VPLs can lead to point singularities due to the fact that the lights need to be positioned close to scene geometry. This is often tackled by clamping the geometry term which in fact reduces singularity-artifacts but also leads to energy loss (see Figure 1).

b) In order to achieve soft indirect shadows, the number of VPLs and corresponding ISMs usually needs to be high, which increases the rendering cost for the indirect illumination considerably.

Prutkin et al. [16] eliminates the lighting artifacts in a) by using *virtual area lights* (VALs) for illumination. These area lights are computed by clustering the pixels of a *reflective shadow map* (RSM), wich also reduces the amount of virtual light sources. However for these clustered area light sources, they do not account for indirect visibility at all, which still poses an open problem. Dong et al. [5] proposes *clustered visibility* by using *convolution soft shadow maps* (CSSMs), which provide soft indirect shadows of much higher quality than ISMs. However for indirect illumination, they still use ordinary VPLs, not taking advantage of the clustering at hand.

We propose an integrated real-time global illumination approach, that

a) performs VAL-based indirect illumination, using RSM clustering similar to Prutkin et al. [16], and

b) uses easy-to-compute ISMs for the indirect visibility of these VALs, while still providing smooth soft

Figure 1: Lighting artifacts when shading with VPLs (left), compared to geometry-clamped VPLs (right). Image courtesy of Hasan et al. [9].

*c.weinzierl-heigl@live.com
†rp@cg.tuwien.ac.at

shadows of considerably higher quality by employing a *percentage closer soft shadow* (PCSS) sampling [6] based on the VAL area.

Reducing the number of indirect light sources and circumventing the need for a large amount of shadow maps by instead using ISMs coupled with a sophisticated sampling method, we can achieve similarly looking soft shadows while reducing the computational cost for indirect illumination considerably.

The remaining paper is structured as follows: Section 2 gives an overview of related work done in the field of real-time global illumination. In Sections 3 - 6 we describe our method in detail. Results and performance measurements are presented in Section 7. We will conclude our paper and also give outlooks into possible future work in Section 8.

2 Related Work

The original *reflective shadow maps* (RSMs) [4] method describes a way of indirect illumination where every pixel of the RSM is considered to act as a one-bounce indirect light source. The RSM was invented as an extension to simple shadow mapping which, besides the depth, also stores surface normals and the radiant flux per pixel from the point of view of the light source. We use the term reflective shadow map in our paper as just the means of storing extended information for the direct light emitter that is later used for indirect illumination calculation. Although included in the original paper, we do not use their suggested indirect illumination procedure.

Instead, we build our method on the *instant radiosity* [11] algorithm. In instant radiosity, VPLs are positioned throughout the scene at intersections of light rays emitting from the primary light source and the rendered scene geometry. To apply fast indirect illumination for these VPLs we use *interleaved sampling* [12], which assigns each VPL to a subset of the scene's pixels and only shades those pixels in order to save shading cost. This method is also used in *incremental instant radiosity* [14] who apply a geometry-aware box-filter to merge the interleaved shaded pixels and gain a smooth indirect illumination.

In order to account for fast indirect visibility computation, our approach relies on *imperfect shadow maps* [19]. Instead of using multiple render passes to create shadow maps for each VPL, ISMs allow to use a point-based scene representation which can be rendered into a single large texture atlas containing every ISM using just one render pass.

Another approach to further speed up rendering, is to cluster a bunch of VPLs together and then treat the cluster as just a single area light source. Dong et al. [5] suggest to cluster the VPLs to VALs for indirect shadowing, however still use all VPLs for illumination of the scene. To compute indirect shadows they use a soft shadowing algorithm based on convolution shadow maps which is only calculated for the actual VALs. The work by Prutkin et al. [16] uses a point-to-disk formfactor to approximate a virtual area light but ignores indirect visibility.

3 Algorithm Overview

Figure 2 shows an overview of the main stages of a real time GI pipeline. In each frame, we generate our *reflective shadow map* (RSM) [4] which stores all the relevant information per pixel from the point of view of the light source. We use a spotlight for direct illumination, but the algorithm could easily be extended to work with every other type of illuminant. We perform a similar G-Buffer pass from the eye-point from which we generate a *split/tiled G-Buffer* [14, 13] where each tile only represents a subset of the pixels of the entire G-Buffer.

After these G-Buffer passes, our algorithm initializes seed points for the VALs within the RSM by using a *Halton-sequence* [8]. In the same step we employ *importance warping* [3] to those initial seeds based on the importance-

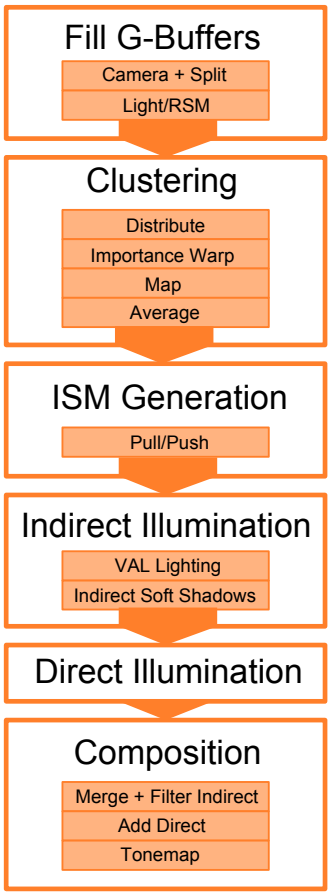
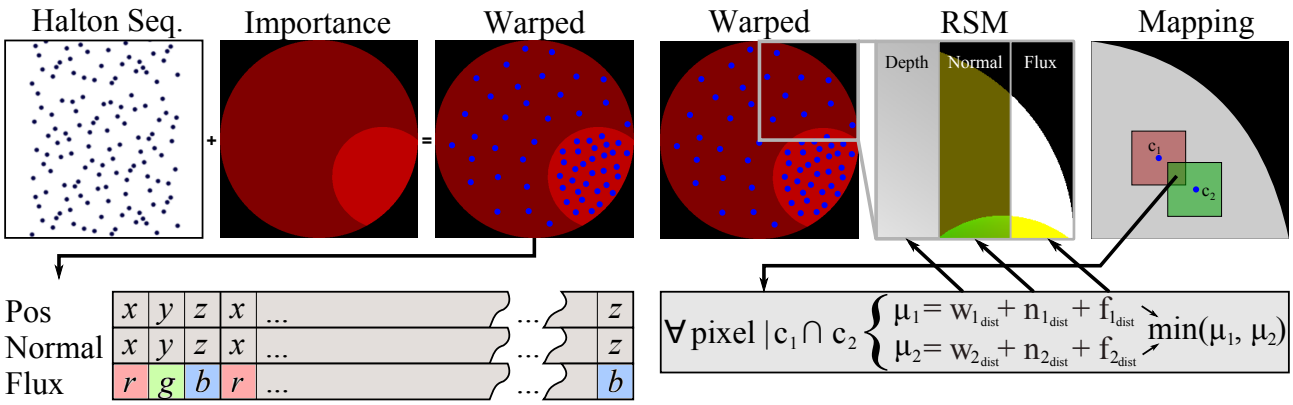


Figure 2: The shading pipeline of our algorithm.



(a) Initial seed points generated from a Halton sequence are combined with an importance map. After performing importance warping the seed points are concentrated at highly specular materials. Clusters are finally initialized with attributes from the RSM at their new, warped position.

(b) Clusters are then rendered as point splats with a custom point size. μ is calculated for each fragment. Overlapping cluster fragments are assigned to the cluster that has the lowest μ-value.

Figure 3: Initialization (a) and mapping (b) behavior of the clustering pipeline.

information stored in the RSM, concentrating the VALs on specular materials. For each seeded VAL/cluster we store its position, normal, flux as well as a count of pixels assigned to it and the cluster's area. The clustering itself is inspired by *k-means clustering* [15] which we adapted to run in a frame-by-frame iterative manner. This way, the clustering algorithm will converge to the final solution over the course of a few frames.

Given the position and direction (normal) of each VAL we generate the *imperfect shadow maps* [19, 18] for them. Since the ISM is a point-based shadow map, we are able to render every single shadow map in just one pass by storing all ISMs within one large texture atlas.

During the final stage of the algorithm the direct and indirect lighting is gathered and composited. Direct lighting for the spotlight happens straight-forward with simple shadow mapping applied. The indirect lighting is calculated using an areal light approximation. During this step we also query the ISMs for indirect visibility information and perform soft shadow mapping [6]. The final image is composited by applying *tone-mapping* [17].

4 Clustering

The clustering of RSM pixels is inspired by *k-Means clustering* [15] which in our case performs an iterative nearest-pixel averaging. After the clustering, each VAL has to store all the information necessary for indirect illumination. For a given number of N clusters, we therefore store the relevant cluster data in six 1D-textures of size $N \times 1$: world-space position, normal, flux, position as texture coordinate, number of RSM-pixels assigned to that cluster as well as the cluster area in world-space. The first three attributes are necessary to generate the light source information (eg. position, direction and color). We also store the position of the cluster as an RSM texture coordinate in order to allow immediate sampling of the cluster's RSM center pixel. This is necessary since a cluster's flux is not averaged by the k-Means method but instead updated from the RSM after the cluster's position and normal has been averaged. The number of RSM-pixels that are assigned to a cluster is required to compute the average during the k-Means step, while the cluster's area is used to perform indirect illumination using a VAL approximation. The clustering itself is actually a three-step algorithm:

Initialization: Here we initialize the cluster's seed position (i.e. its initial position in the RSM) by using a *Halton-sequence* [8] to get a pseudo-random distribution of clusters within the reflective shadow map. Based on an importance value stored in the RSM, we perform *importance-warping* [3] to redistribute the clusters such that they are concentrated at materials with higher specular terms. Once we have those importance-warped seed positions in texture-space, we can sample the RSM at those coordinates and store the first part of the relevant information for each cluster (world-space position, texture-coordinate position, normal and flux). Please note that this step is only required for clusters that need to be initialized, which always happens in the very first frame and can occur in consecutive frames whenever there are no RSM-pixels assigned to a cluster (see Figure 3a).

Mapping: During this step, the algorithm maps RSM-pixels to clusters within its vicinity. To achieve this, we render point-splats at the previously seeded cluster positions with a user-defined point size that can range between $32 - 128$ pixels. In the fragment shader we compute a distance μ based on

world-space distance, normal- and flux-difference (see Equation 1) from the currently shaded pixel to the cluster's center. We use a metric inspired by Prutkin et al. [16], which is given by a convex sum of different distances

$$\mu = d_p \cdot w_p + d_n \cdot w_n + d_f \cdot w_f \quad (1)$$

between a RSM pixel and the cluster. d_p denotes the world-space distance in euclidean space, d_n the distance in normal-space defined by the magnitude of the dot-product of their normals and d_f is their distance in RGB color-space. The convex weights w_p, w_n and w_f ensure a proper weighting of the partial distances in this metric. The resulting μ is then used as a custom-output to the depth-buffer. This way, the depth-buffer automatically assigns the RSM pixel to its nearest cluster with respect to the metric in Equation 1 (see Figure 3b), yielding a so called *cluster map*.

Averaging: In order to perform averaging in the last step of the *k-Means clustering* algorithm [15], we need to compute the total number of pixels assigned to each cluster beforehand. To that end we check each pixel of the *cluster map* for its assigned cluster's index and increment the pixel-count for the respective cluster by utilizing the hardware's additive blend capabilities. From the generated per-cluster pixel-count, we compute the new, averaged cluster positions and normals. Since we do not average the cluster fluxes, these are updated in a separate step in which we simply sample the RSM at the new cluster texture coordinates.

5 Imperfect Shadow Maps

The huge amount of visibility information required for the correct illumination from all virtual light sources is approximated by using *imperfect shadow maps* (ISMs) [19]. The generation of ISMs requires a pre-processing step that generates a point-sampled representation of the scene geometry. Therefore, the algorithm generates random barycentric points within the mesh triangles, which are then used for splatting during ISM generation. The points are stored in the object space of the meshes they were created from to be able to correctly transform this representative points along with their meshes. This allows us to translate/scale/rotate the scene's objects dynamically at runtime but does however not permit deformable meshes.

The ISMs are generated per cluster/VAL and are stored within one large texture atlas. A resolution of 4096×4096 was chosen for the texture atlas and 128×128 for the ISMs contained in it, which allows up to 1024 clusters at once. During the ISM rendering, the point-sampled scene geometry is drawn to the texture atlas and each point is assigned to a different ISM in a round-robin fashion: The current vertex's id v_{id} and the number of clusters N are used to compute $mod(v_{id}, N)$ to assign the point to the respective ISM. Points closer to the cluster position should be rendered larger, while points farther away should be smaller. To this end, the point-size is calculated in the vertex shader based on the distance from the current cluster. Unlike the shadow map for a directional- or spot-light the ISM should actually cover the entire hemisphere of the direction it is looking at. Hence, a parabolic mapping [1] is applied to project the points onto the paraboloid that represents the hemisphere around the cluster.

Figure 4: Example of an ISM with holes (left). The same ISM after 2 iterations of the pull-push algorithm fills small holes (right).

The resulting texture atlas contains all ISMs required to shade the scene. Due to the nature of point sampled geometry, it is likely that the point-based ISMs contain holes simply because the sampling density was too low or the algorithm to produce the points in the first place did not provide evenly distributed samples. This can manifest itself during the indirect shading stage as jagged shadow artifacts and irritating shadow leaks. To alleviate this problem a pull-push algorithm is employed, as suggested by [19], to fill small holes inside the ISMs (see Figure 4).

6 Indirect Lighting & ISM Sampling

Although we have now greatly reduced the number of indirect light sources, the cost of shading every pixel of the G-Buffer with every VAL would still be too expensive. We follow an approach derived from *interleaved sampling* [12], where each pixel is only shaded by a subset of all available VALs. To this end, we generate a *split G-Buffer* [14] from the *initial G-Buffer*, with $n \times m$ tiles. Each tile represents an interleaved set of pixels of the initial G-Buffer: Consider the pixel (x, y) in the tile (i, j), where $0 \leq i < n$ and $0 \leq j < m$. The content of this pixel is read from the initial G-Buffer pixel at position $(xn + i, ym + j)$.

In order to shade the resulting tiles, we build up a *tiled mesh geometry* [13]. This is basically a representation of screen-space positioned quads aligning with the tiles of the split G-Buffer. Each tile of this tiled mesh geometry is assigned either a number of VALs or just one VAL, depending on whether the shading should be done in a single-pass or in a blended multi-pass setting. The allocation of VALs

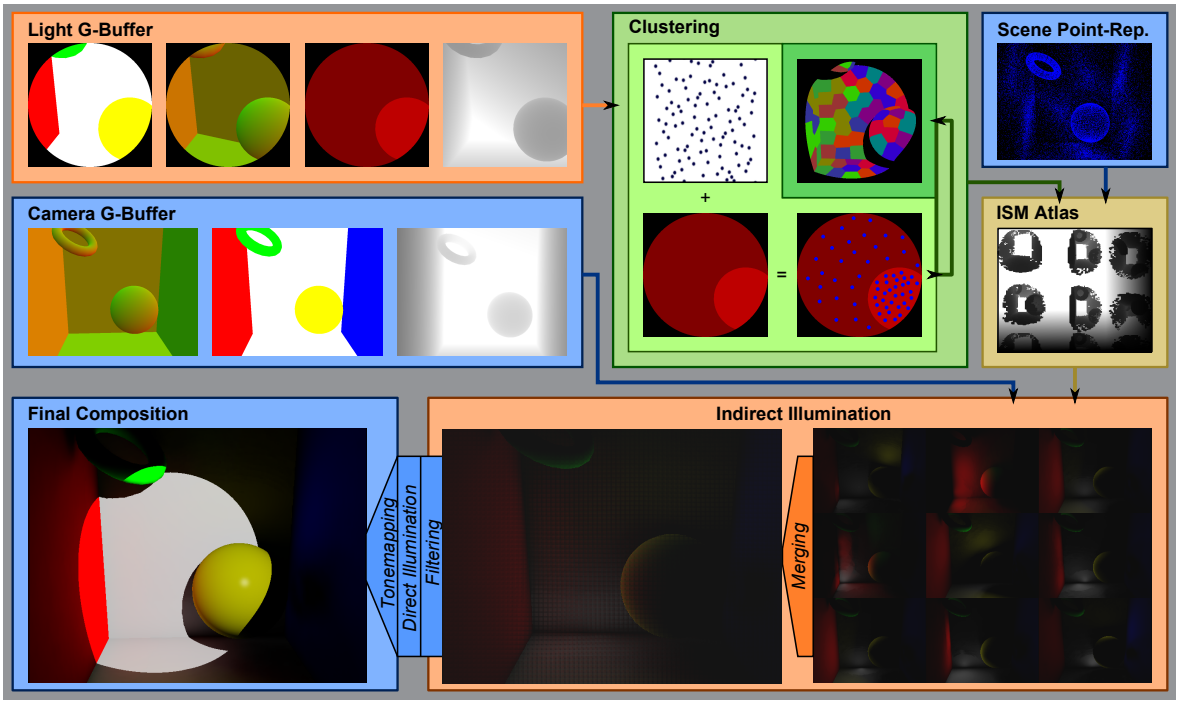


Figure 5: Overview of the complete lighting composition chain including involved buffers.

to tiles is again done in a round-robin fashion: A VAL v is assigned to a tile T by $T_v = mod(v, n \times m)$, where n, m follow the declaration from above. We can then render the tiled mesh geometry and shade each pixel with the assigned VALs.

The actual illumination of a point from a VAL is based on a *point-to-disk formfactor*, similar to Prutkin et al. [16], to approximate a VAL from a disk-shaped region which acts as the area light. Remember that we calculated an approximate area A_c for each cluster in the clustering step, which is used to calculate the formfactor [20]

$$\int_{A_c} \approx \frac{\cos\Theta_x \cdot \cos\Theta_c}{\pi \cdot \|x-c\|^2 + A_c} \tag{2}$$

where Θ_x is the angle between the surface normal at x and the incoming light vector and Θ_c is the angle between the surface normal at c and the outgoing light vector (see Figure 6).

Using this formfactor, we can perform soft indirect illumination without the downsides of regular VPL shading. However, until now we have not considered indirect visibility at all. Soft indirect shadows would normally result automatically by blending a sufficient amount of contributions from multiple ISMs together. As we have however dramatically reduced the number of indirect light sources and corresponding ISMs, the shadow term would suffer from severe artifacts (see Section 7 for comparison screenshots) stemming from the low resolution nature of the ISMs, as well as from the naive shadow lookup involved.

To minimize the artifacts and in order to get similar soft shadows as when using much more indirect light sources, we apply the *percentage closer soft shadows* technique [6] during shadow lookup. Instead of taking just a single sample from the shadow map and applying the result to the lighting term, multiple samples are taken at once in order to calculate a soft shadow term. Three steps are involved in this process:

Blocker search: Searches a finite region of space between the light source and the receiver for occluding geometry whose depth values are closer to the light source than the receiver's depth $z_{receiver}$. The size of the search region depends on the light size (which we calculate from the VALs area) and the receiver's distance from the light source. If the search finds any occluder points we average their depths $z_{blocker}$ and continue to the next step. Otherwise, we can abort the calculation and return full visibility.

Penumbra estimation: The size of the penumbra (soft shadow region) is estimated using similar triangle calculations. We simply assume that the light source, blocker and receiver are arranged on parallel planes. This way we can calculate the penumbra ratio as

$$r_{penumbra} = (z_{receiver} - z_{blocker})/z_{blocker}. \tag{3}$$

Filtering: Finally, a simple PCF computation is applied on the shadow map using *Poisson-disk*-sampled offsets [2] scaled by the estimated penumbra ratio.

Since the indirect illumination shading is done on the tiled

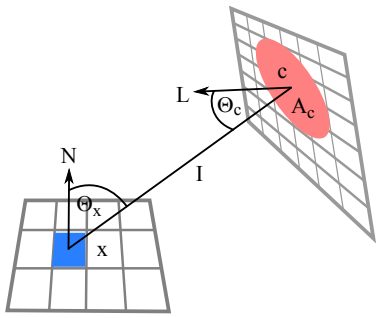


Figure 6: The point-to-disk formfactor for the receiving pixel x is computed from the angle Θ_x between the surface normal N and indirect light direction I as well as the angle Θ_c of emitter c between its surface normal L and I with corresponding area A_c.

mesh geometry, the tiles need to be merged back to a single fullscreen texture. This involves inversing the calculation done when building the split G-Buffer from the initial G-Buffer, as described at the beginning of this Section. The resulting merged picture needs to be filtered using a *geometry-aware filter* [14] of size $n \times m$ to get the final, smooth indirect illumination. Geometry-awareness is achieved by checking the similarities between the kernel's center pixel and the pixel at the current offset. We use two thresholds α, β to vary the allowed similarity differences. α is used as the threshold for maximum world-space difference between two sample points, while β is used as a threshold for surface normal differences. The filtering only occurs for samples that lie below both thresholds.

The last step is to combine direct and indirect lighting contribution and perform tonemapping [17]. Figure 5 depicts an overview of the entire lighting process.

7 Results

This chapter gives a visual and numerical overview of our results. The entire program was written in C++ using OpenGL 3.3. We measure the performance of our new approach and compare the frame timings in different setups. Measurements were achieved on an Intel Core 2 Duo E6600 processor with 2.4 GHz using an AMD Radeon HD 6850 with a frame buffer resolution of 1280×720 pixels.

As we have already outlined in the introduction, our goal was to improve the rendering performance of an instant radiosity-based GI algorithm to gain even higher performance as when using a traditional VPL-based approach. To achieve this, we chose the following two subjects to improve upon:

a) Use VALs instead of VPLs for indirect illumination in order to remove artifacts (mainly singularities) and gain performance (see Section 7.1).

b) Reduce the overall number of indirect light sources without degrading indirect shadow quality (see Section 7.2).

7.1 Indirect Illumination

The usage of a point-to-disk formfactor (refer to Equation 2) allows us to reduce typical artifacts that occur when shading with VPLs. It further allows us to reduce the number of indirect lights used for shading without degrading the quality of the indirect illumination. Simultaneously, this also reduces the shading cost of the indirect illumination since less indirect light contributions need to be computed. On the other hand our VAL-based approach needs to perform a per-frame clustering step which is not necessary for VPL-based illumination.

Hence, we compare achievable indirect illumination results for VPL-shading with our VAL-shading approach. In this arrangement the frame timings for the complete indirect illumination process are kept constant to perform a fair comparison: For VPL-shading we only count the indirect illumination timings, while for our VAL-based approach we also need to consider the time it takes to perform the clustering. Using 256 VPLs and only considering the indirect illumination timings of 13 ms is therefore comparable to our VAL-based approach using 128 VALs considering both, indirect illumination (8 ms) and clustering (5 ms) timings yielding a total of 13 ms. Figure 7 shows that our method produces much less artefacts at the same performance than a VPL-based approach.

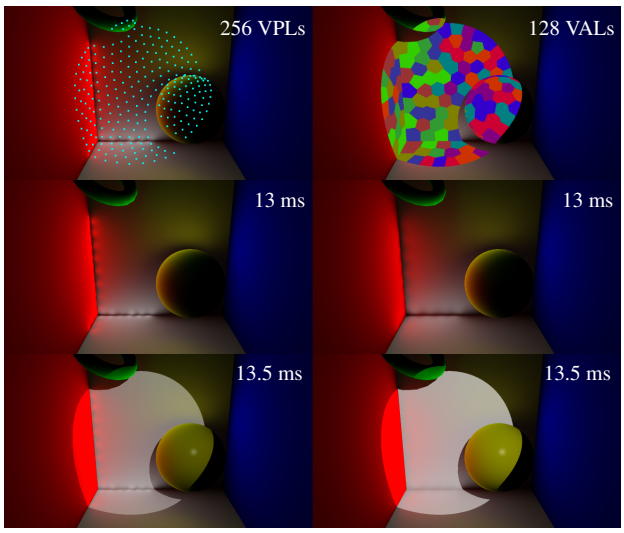


Figure 7: VPL-shading results on the left, VALs on the right. The top row shows VPL positions on one side and VAL clusters on the other side. Second row shows indirect illumination only, while the third row shows combined direct and indirect lighting contributions. VPL singularities are clearly visibile on the left, diminsh on the right.

7.2 Indirect Shadows

Since the reduced number of indirect light sources also results in a reduced number of indirect visibility information stored in less ISMs, we perform a soft shadow lookup that is similar to PCSS in order to achieve similarly looking and visually pleasing indirect shadows as when compared to using much higher amounts of indirect light sources and ISMs using only a simple shadow lookup.

Our tests not only show that we are able to gain perceptually equivalent results with our PCSS-based method (see Figure 8), but also, that it is possible to simultaneously achieve rendering speedups of up to 50%. Table 1 shows timings for the important algorithm stages, comparing 64 PCSS-sampled VALs versus 512 simple-sampled VALs. All measurements were taken in our Cornell Box test scene. Complete frame timings for different VAL amounts, comparing single- vs PCSS-sampled timings are shown in Figure 9.

	64 PCSS VALs	512 simple VALs
Clustering	5 ms	5 ms
Indirect Illum.	11 ms	30 ms
ISM Generation	10 ms	17 ms
Complete	26 ms	52 ms

Table 1: Comparison of 64 PCSS-sampled VALs with 512 simple-sampled VALs. Rendering time is clearly dominated by the generation of the ISMs (incl. 2 iterations of Pull/Push) and the indirect illumination stage. The speedup of 50% considering the complete timings is still significant.

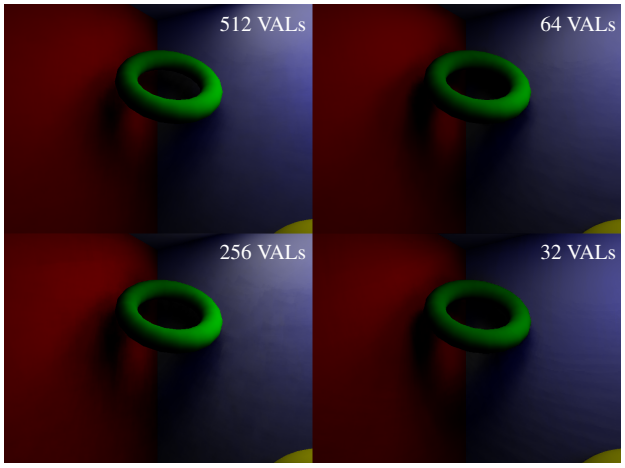


Figure 8: Comparison of indirect shadows using simple shadow lookups (left) and PCSS lookup (right).

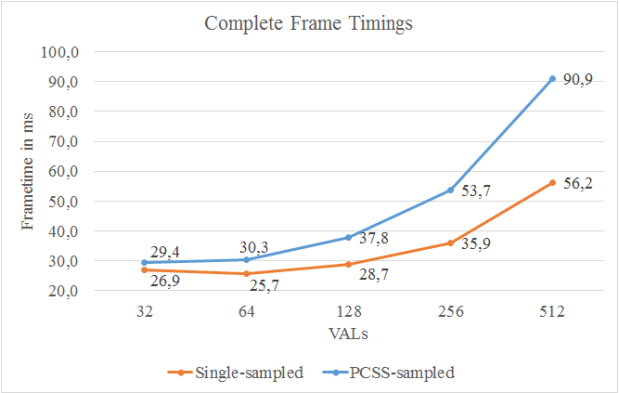


Figure 9: Statistics of complete frame timings for our approach using different VAL counts comparing single-sampled versus PCSS-sampled shadow lookups.

8 Conclusion and Future Work

In this paper we present an efficient, high quality global illumination method based on instant radiosity that is capable of producing real-time frame rates. Our work is comprised of various methods to reduce the required number of indirect light sources, while at the same time producing plausible indirect soft shadows. To that end we integrate a clustering algorithm that groups RSM pixels together in order to form area lights. By computing the cluster areas we can further apply a point-to-disk formfactor to perform indirect illumination using the approximated VALs. A big advantage of clustering the light sources is the reduction of shading costs due to less overdraw and blending.

On the other hand, the reduction of the number of indirect light sources and corresponding shadow maps results in shadowing artifacts that manifest themselves due to the low resultion nature of the ISMs and because too less shadow contributions might be accumulated. Therefore, we propose to use percentage closer soft shadows during shadow map sampling, taking multiple samples per shaded pixel to smooth out the indirect shadow term. Although taking multiple ISM lookups comes with a higher computation cost per indirect light source, this is easily outweighed by the overall reduced number of visibility computations necessary by using VALs.

As our algorithm is currently only capable of producing first-bounce indirect lighting, it would be interesting to expand the approach to incorporate multiple bounces in future work. This could be done by further producing RSMs for the indirect lights and performing a similar illumination for these higher-bounce indirect light sources. Furthermore our method is currently also limited to diffuse indirect bounces. The behavior when performing specular indirect bounces (for instance to produce caustics) would be an interesting topic of future work.

References

[1] S. Brabec, T. Annen, and H.P. Seidel. Shadow mapping for hemispherical and omnidirectional light sources. In *Proc. of Computer Graphics International*, pages 397–408, 2002. 4

[2] R. Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH*, volume 2007, 2007. 5

[3] P. Clarberg, W. Jarosz, T. Akenine-Möller, and H.W. Jensen. Wavelet importance sampling: efficiently evaluating products of complex functions. *ACM Transactions on Graphics (TOG)*, 24(3):1166–1175, 2005. 2, 3

[4] C. Dachsbacher and M. Stamminger. Reflective shadow maps. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 203–231. ACM, 2005. 2

[5] Z. Dong, T. Grosch, T. Ritschel, J. Kautz, and H.P. Seidel. Real-time indirect illumination with clustered visibility. In *Vision, Modeling, and Visualization Workshop 2009*, 2009. 1, 2

[6] R. Fernando. Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches*, page 35. ACM, 2005. 2, 3, 5

[7] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modeling the interaction of light between diffuse surfaces. *SIGGRAPH Comput. Graph.*, 18(3):213–222, January 1984. 1

[8] J. H. Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Commun. ACM*, 7(12):701–702, December 1964. 2, 3

[9] M. Hašan, J. Křivánek, B. Walter, and K. Bala. Virtual spherical lights for many-light rendering of glossy scenes. *ACM Transactions on Graphics (TOG)*, 28(5):143, 2009. 1

[10] J. T. Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA, 1986. ACM. 1

[11] A. Keller. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 49–56. ACM Press/Addison-Wesley Publishing Co., 1997. 1, 2

[12] A. Keller and W. Heidrich. Interleaved sampling. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 269–276. Springer-Verlag, 2001. 2, 4

[13] M. Knecht. Real-time global illumination using temporal coherence. Master's thesis, Vienna University of Technology, Jul 2009. 2, 4

[14] S. Laine, H. Saransaari, J. Kontkanen, J. Lehtinen, and T. Aila. Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering*, pages 277–286, 2007. 2, 4, 6

[15] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. California, USA, 1967. 3, 4

[16] R. Prutkin, A. Kaplanyan, and C. Dachsbacher. Reflective shadow map clustering for real-time global illumination. In *Eurographics 2012-Short Papers*, pages 9–12. The Eurographics Association, 2012. 1, 2, 4, 5

[17] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. *ACM Transactions on Graphics (TOG)*, 21(3):267–276, 2002. 3, 6

[18] T. Ritschel, E. Eisemann, I. Ha, J.D.K. Kim, and H.P. Seidel. Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes. In *Computer Graphics Forum*. Wiley Online Library, 2011. 3

[19] T. Ritschel, T. Grosch, M. H. Kim, H. P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. *ACM Trans. Graph. (Proc. of SIGGRAPH ASIA 2008)*, 27(5), 2008. 1, 2, 3, 4

[20] J. R. Wallace, K. A. Elmquist, and E. A. Haines. A ray tracing algorithm for progressive radiosity. *SIGGRAPH Comput. Graph.*, 23(3):315–324, July 1989. 5

[21] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, 1980. 1