Reflective Shadow Map Clustering for Real-Time Global Illumination

Roman Prutkin[†], Anton S. Kaplanyan[‡] and Carsten Dachsbacher[§]

Karlsruhe Institute of Technology, Germany

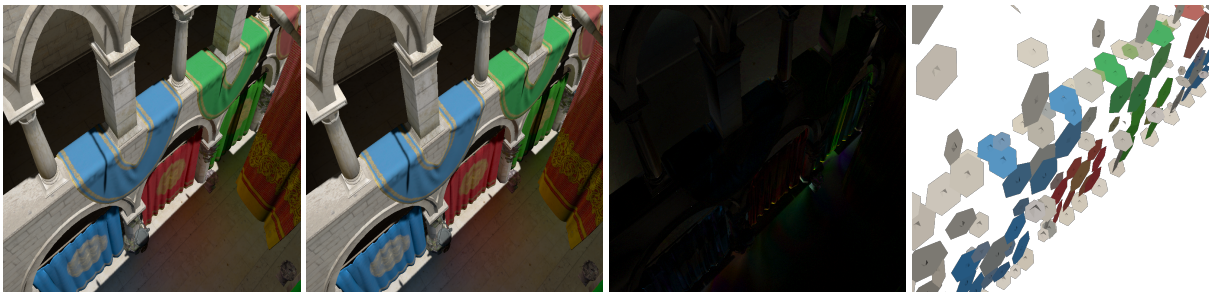

Figure 1: CRYTEK SPONZA *scene rendered with indirect illumination. From left to right: (1) indirect illumination using 256.000 VPLs generated by RSM; (2) our approach: using 960 area light sources (disk-shaped); (3) pixel-wise difference, 8 times amplified; (4) the disk lights (depicted as hexagons) obtained from a clustering of a reflective shadow map (RSM). The RSM is clustered with budget of 1024 clusters, exploits temporal coherence and uses importance sampling.*

Abstract

We present a method for real-time clustering of VPLs obtained from Reflective Shadow Maps (RSM). The clusters are treated as area light sources and used to approximate indirect illumination. The spatial extent of a cluster is used to deduce the shape and size of the respective area light source. Our method is fully GPU-based and avoids flickering by temporally coherent reinitialization of the clustering. It also incorporates importance sampling for view-dependent clustering. We show visually indistinguishable results of indirect illumination with only a fraction of secondary light sources.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Shading

1. Introduction

Due to its global nature, dynamic indirect illumination in real-time remains a challenging task even with current graphics hardware. However a visually plausible approximation to indirect illumination is sufficient in many real-time scenarios. Existing techniques typically provide approximations or imply restrictions such as long precomputation time, low-frequency indirect lighting or static geometry. Methods based on Instant Radiosity [Kel97] approximate indirect illumination by creating light paths from the primary light sources and place *Virtual Point Lights* (VPLs) at ray intersections with the scene geometry. In our method, we use *Reflective Shadow Maps* [DS05] which is a GPU-friendly method for creating VPLs for single-bounce indirect illumination. To this end, the scene is rendered from the perspective of the primary light source, and each pixel of the resulting image can be interpreted as a VPL for indirect lighting calculations. In this paper we focus on the clustering and thus reduction of these VPLs. We show results to unoccluded single-bounce indirect illumination for diffuse surfaces. Note that handling visibility for many point lights or area lights is an orthogonal problem, where fast solutions exist [RGK*08, ADM*08].

2. Related Work

Splatting Indirect Illumination [DS06] is based on RSMs and replaces the gathering of the VPL contributions by splatting each VPL onto the image; the authors also adapted the importance sampling technique from Clarberg et al. [CJAMJ05] to select important VPLs. The method runs entirely on the GPU in real-time, handles non-diffuse materials, and can render plausibly looking caustics.

† roman.prutkin@student.kit.edu

‡ anton.kaplanyan@kit.edu

§ dachsbacher@kit.edu

The light cuts method [WFA*05] uses a hierarchical collection of point lights. For shading a surface point, a cut through the hierarchy is selected, and the contributions of all clustered point lights are summed up, while visibility is resolved by ray-casting toward a representative for each cluster. In the same spirit, Dong et al. [DGR*09] proposed to share the visibility queries for groups of VPLs obtained by clustering. They partition RSMs into a small number of clusters, which are treated as area lights when testing visibility for indirect lighting. However, indirect lighting is still computed using non-clustered VPLs.

Inspired by these approaches, we cluster the RSM into area light sources, which are then used for computing indirect illumination. Our method runs entirely on GPU and provides stable and temporally coherent results, even with low numbers of clusters. Similarly to Georgiev and Slusallek [GS10], we augment our clustering algorithm with a bidirectional scheme to provide view-dependent clustering in highly occluded scenes. Note that our work focuses on reducing the number of VPLs, not rendering many VPLs or computing shadowing efficiently.

3. Algorithm Overview

For each frame, we first render the G-Buffer for the camera view, and the RSM typically at 256×256 resolution. For shadows of the direct light, we use high resolution cascaded shadow maps. Then we apply a clustering technique inspired by k-means clustering [Mac67] combined with bidirectional importance maps [REH*11] to the obtained RSM. Subsequently, we convert the resulting clusters into area light sources that approximate the indirect illumination. For lighting with area lights we then use point-to-disk or point-to-polygon form factors. We compare these form factors in context of indirect illumination in Section 5.

4. Clustering

For reducing the number of VPLs obtained with the RSM, we use a GPU-friendly clustering technique similar to the k-means algorithm. K-means clustering iteratively partitions n points into k clusters, where each point is assigned to the cluster whose centroid is nearest. We compute a bidirectional importance map [REH*11] using the surfaces visible in the camera view to form more clusters, i.e. area lights, in the areas with higher contribution to the final image. Before the actual iteration, we have to initialize the algorithm with seeds for the cluster centroids. In the first frame, we directly use the bidirectional importance map and use importance warping to distribute seed points for the clusters accordingly. For subsequent frames, we initialize cluster seeds using the clusters from the previous frame to improve temporal coherency. For every frame, we perform the following two steps once: (1) assign each RSM pixel to a cluster with the most similar properties (see next paragraph); (2) update the cluster centers according to the current pixel-to-cluster mapping. This can be seen as one iteration of k-means clustering, while – in contrast to the original algorithm – we spread iterations across frames. For lighting computation, we compute the total flux of a cluster as the sum of its VPLs. Please note that it might happen that no pixels are assigned to a cluster. In this case, we do not create any area light source for the cluster, and create a new seed point for it in the next frame prior to clustering.

Algorithm 1 RSM clustering

```
 1: seeds ← HaltonSequence2D(k)
{Build importance map:}
 2: for all pixel in RSM do
 3:     importanceMap[pixel] ← bidirImportance[pixel]
 4: end for
{Importance Warping:}
 5: for all seed in seeds do
 6:     seed ← warp(seed, importanceMap)
 7: end for
{Clustering initialization:}
 8: for all i 1 to k do
 9:     if is_empty(cluster[i]) then
10:         cluster[i].{center, normal, flux} ← RSM[seed[i]]
11:     end if
12: end for
{Clustering:}
13: for all pixel in RSM do
14:     clusterMap[pixel] ← argmin_{cluster.id} (μ(pixel, cluster))
15: end for
16: clusterMap ← sort(clusterMap)
17: RSM ← reorder(RSM, clusterMap)
18: for all cluster in clusters do
19:     cluster.center ← average_{p∈RSM, clusterMap[p]=cluster.Id} (p.position)
20:     cluster.normal ← average_{p∈RSM, clusterMap[p]=cluster.Id} (p.normal)
21:     cluster.flux ← Σ_{p∈RSM, clusterMap[p]=center.Id} (p.flux)
22: end for
```

All steps of the algorithm are executed on the GPU. At line 2 of Algorithm 1 the bidirectional importance of an RSM pixel is calculated in the function *bidirImportance*. Starting with an initially uniform distribution of the seeds, we obtain the importance warped locations as described in [DS06] (line 5). Then, for each pixel of the RSM in parallel, we find the cluster with the lowest difference according to the metric μ defined below (line 13) and save its *id* into the cluster map. To update the cluster centers, for each center we iterate in parallel over the cluster map and calculate average positions and normals and the total flux of the pixels assigned to it, see line 18.

Difference Metric. For clustering we define a custom distance metric between a cluster c and an RSM pixel p, which consists of a weighted sum of differences between normals, positions, and colors:

$$\mu(p,c) = w_d \|x_p - x_c\|^2 + w_n(1 - n_p \cdot n_c) + w_\Phi \left(\frac{\Phi_p}{|\Phi_p|} - \frac{\Phi_c}{|\Phi_c|} \right)^2. \quad (1)$$

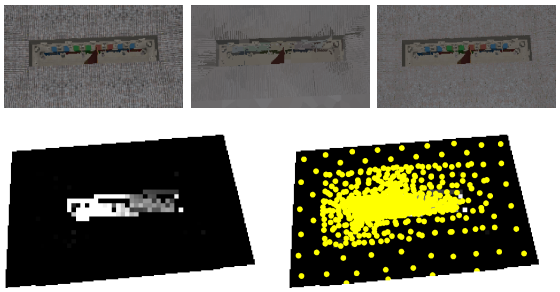

Figure 2: CRYTEK SPONZA *scene, light shines from above. Top: Flux from original RSM (left); clustered with* $w_\Phi = 0$*, finer details like flags disappear due to averaging (middle); Top, right: choosing a high value for* w_Φ *allows to preserve fine details in color (right). Bottom: bidirectional importance map (left) and cluster centers (right).*

This metric is inspired by the existing work (e.g. [DGR*09] and [WRC88]). Here, w_d, w_n, w_Φ are the weights for distance, normal deviation, and the VPLs' flux terms respectively; x_p and x_c are the centers of the pixel and the cluster; $\|\cdot\|$ is the Euclidean distance metric; n_p and n_c are the normals of the RSM pixel and the cluster. The center and the normal of a cluster are computed as a sum of position and normal of all VPLs in the cluster, weighted by their respective flux and normalized by the total flux of all VPLs. For the color, we calculate the squared distance between the normalized (R,G,B)-triples of flux values to account for both saturation and hue variations. For faithful clustering of RSM pixels it is important to consider their normals: If the weight w_n is too small, clusters often cross edges of objects causing artifacts in the indirect lighting. Secondary light sources might end up hanging in the air or inside objects whenever their respective clusters comprise RSM pixels belonging to differently oriented faces.

Improving temporal stability. For small changes in scene geometry and light sources, or smooth camera movements, the clustering changes only for a few pixels of the RSM. In case of significant changes, Dong et al. [DGR*09] increase temporal stability of the clustering by initializing the seeds using the same pseudo-random sequence. Instead, we use the result from the previous frame to initialize the cluster centroids. In contrast to [DGR*09], if the number of active VPLs in a cluster shrinks to zero, we reinitialize such clusters from the RSM according to the importance map, see line 8 of Algorithm 1.

View-Dependent Importance Sampling. For more complex scenes with significant occlusion, like CRYTEK SPONZA in Fig. 2 for example, we strive to create more clusters for RSM regions which contribute more to the image. For this purpose we use importance warping [CJAMJ05, DS06] to distribute the seed points according to the importance map computed as described by Ritschel et al. [REH*11]. In the function *bidirImportance* at line 2 of Algorithm 1 the bidirectional importance of an RSM pixel

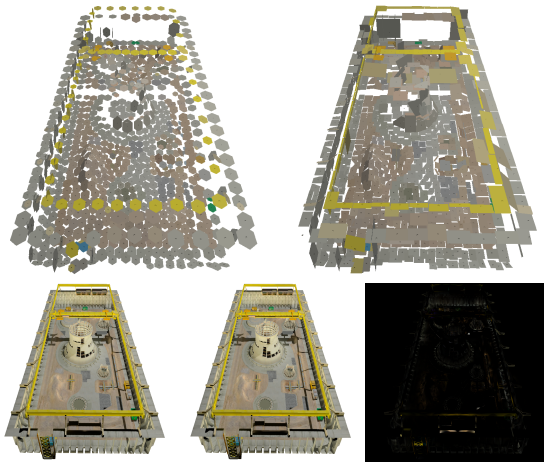

Figure 3: POWER PLANT *scene. Top: (left) disk approximation; (right) polygon approximation. Bottom: (left) rendered images for the disk; (middle) polygon approximation; (right) the* $4\times$ *difference between them.*

is calculated by accumulating the geometry terms from 256 Halton-distributed G-buffer pixels.

GPU Implementation. We use a regular grid of size $\sqrt{k}\times\sqrt{k}$ over the RSM to accelerate the clustering. The positions, normals and colors of all clusters are stored in a GPU array. As we search for the nearest cluster for each RSM pixel, we only consider centroids in grid cells which lie at most r cells apart in the square neighborhood. We typically set $r = 2$. By this, the number of potential pixels to be assigned to a cluster is reduced from n to $\frac{n}{k}(2r+1)^2$, yielding a significant speedup for the clustering. This also allows to avoid highly stretched cluster shapes and disconnected regions inside one cluster. The *cluster map* (of the same resolution as the RSM) stores in every pixel the index of the cluster it is assigned to. Between the assignment and the update steps of a clustering iteration, we sort the pixels of the RSM on the GPU with bitonic sort. As a result all pixels belonging to the same cluster are stored contiguously in a list. Similarly to Bai et al. [tBlHtO*09], we sort only the cluster *id*'s. Then we reorder the RSM accordingly, so that the clusters are stored contiguously. When updating cluster centers, we use a binary search in the sorted cluster map to find the beginning of a cluster. Sorting and reordering provides at least a triple speedup in our tests, as we avoid iterating over the entire cluster map.

5. Approximating Cluster Geometry

A common problem of all VPL-based methods are artifacts due to the singularities caused by lighting with point lights. These artifacts are often solved by clamping the geometry term. However, the energy lost during clamping requires compensation (see Novak et al. [NED11] for a recent real-time solution). Our approach to avoid these VPL-typical artifacts is to exploit the spatial extent of the clusters and approximate them as polygons or disks (Figure 3, top). This

allows us to treat clusters of VPLs as area light sources and use form factors such as the analytical point-polygon form factor, or the point-disk approximation [WEH89]:

$$\int_{A_{\text{disk}}} \frac{G(x,z)}{\pi} \mathrm{d}A_z \approx \frac{A_{\text{disk}} \cdot \cos\theta_x \cdot \cos\theta_{x_c}}{A_{\text{disk}} + \pi \cdot \|x - x_c\|^2}. \qquad (2)$$

In our examples, we use a disk approximation to the clusters, since it delivers the same visual quality (see Figure 3, bottom) and, unless the point-polygon form factor, is computationally as cheap as the computations for one point light.

6. Results

Table 1 shows execution times for the proposed RSM clustering algorithm including importance mapping, as in Alg. 1. The tests were performed on a PC equipped with an Intel Core i7-2600 CPU and Nvidia 560 GTX Ti GPU. We use Microsoft DirectX 10 API with DirectCompute for rendering and GPGPU computations. The total frame time is around 140-250 ms for our test scenes and is mostly dominated by the rendering of secondary lights. We use the disk-shaped light sources with the form-factor from Eq. 2 and notice no performance difference comparing to rendering the same number of VPLs. We use different weights for Eq. 1 depending on the scene and the intended clustering. For example, we use $w_d = 1$, $w_n = 10$, $w_\Phi = 0$ for the CORNELL BOX scene due to the major dependence on the normals. However, for the POWER PLANT and the CRYTEK SPONZA scenes we chose $w_d = w_n = 1$, $w_\Phi = 10$ to achieve more visually pleasing color bleeding by preserving finer color details. In the latter scene we also had to increase the RSM resolution for that.

Scene	Active/total clusters	Time, ms	RSM
CORNELL BOX	75 / 256	0.92	128^2
POWER PLANT	986 / 1024	3.2	128^2
CRYTEK SPONZA	960 / 1024	9.5	256^2

Table 1: *Performance results for the clustering algorithm. The timings are provided for a single frame and include the clustering and the importance map construction.*

7. Conclusion and Future Work

In this work, we introduced a real-time method for clustering Reflective Shadow Maps based on k-means clustering. Compared to selecting subsets of RSM pixels as VPLs our method has several benefits, such as approximating the geometry with better-than-VPL primitives, e.g. disks or polygons. Consequently, it allows for significantly better results with the same budget of secondary lights at little additional cost per light. In addition to that, our method has good temporal stability, behaves well even under sudden changes of lighting. In addition, the view-dependent importance map provides a reliable clustering heuristic for complex scenes. The computational cost of the GPU-based clustering is low and makes our approach well-suited for real-time applications. It can further be used with any RSM-based technique to improve the selection of virtual lights. Our approach currently ignores indirect visibility, however, clustered visibility [DGR*09] can be used to augment our approach to address this problem. Another promising direction is to extend the rendering of clustered lights to non-diffuse materials.

References

[ADM*08] ANNEN T., DONG Z., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Real-time, all-frequency shadows in dynamic scenes. *ACM Transactions on Graphics 27*, 3 (2008), 1–8. 1

[CJAMJ05] CLARBERG P., JAROSZ W., AKENINE-MÖLLER T., JENSEN H. W.: Wavelet importance sampling: efficiently evaluating products of complex functions. *ACM Trans. Graph. 24*, 3 (2005), 1166–1175. 1, 3

[DGR*09] DONG Z., GROSCH T., RITSCHEL T., KAUTZ J., SEIDEL H.-P.: Real-time indirect illumination with clustered visibility. In *VMV* (2009), DNB, pp. 187–196. 2, 3, 4

[DS05] DACHSBACHER C., STAMMINGER M.: Reflective shadow maps. In *Symposium on Interactive 3D Graphics* (2005), pp. 203–231. 1

[DS06] DACHSBACHER C., STAMMINGER M.: Splatting indirect illumination. In *Symposium on Interactive 3D Graphics* (2006), ACM, pp. 93–100. 1, 2, 3

[GS10] GEORGIEV I., SLUSALLEK P.: Simple and Robust Iterative Importance Sampling of Virtual Point Lights. In *Proceedings of Eurographics 2010* (May 2010). 2

[Kel97] KELLER A.: Instant radiosity. In *SIGGRAPH* (1997), pp. 49–56. 1

[Mac67] MACQUEEN J. B.: Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967), vol. 1, University of California Press, pp. 281–297. 2

[NED11] NOVAK J., ENGELHARDT T., DACHSBACHER C.: Screen-Space Bias Compensation for Interactive High-Quality Global Illumination with Virtual Point Lights. In *In Symposium on Interactive 3D Graphics and Games (I3D '11)* (2011), pp. 119–124. 3

[REH*11] RITSCHEL T., EISEMANN E., HA I., KIM J. D. K., SEIDEL H.-P.: Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes. *Computer Graphics Forum 30*, 8 (2011), 2258–2269. 2, 3

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H.-P., DACHSBACHER C., KAUTZ J.: Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. *ACM Trans. Graph. (Proc. of SIGGRAPH ASIA 2008) 27*, 5 (2008). 1

[tBlHtO*09] TAO BAI H., LI HE L., TONG OUYANG D., SHAN LI Z., LI H.: K-means on commodity gpus with cuda. In *CSIE (3)* (2009), Burgin M., Chowdhury M. H., Ham C. H., Ludwig S. A., Su W., Yenduri S., (Eds.), IEEE Computer Society, pp. 651–655. 3

[WEH89] WALLACE J. R., ELMQUIST K. A., HAINES E. A.: A ray tracing algorithm for progressive radiosity. In *SIGGRAPH* (1989), ACM, pp. 315–324. 4

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: a Scalable Approach to Illumination. In *ACM SIGGRAPH* (2005). 2

[WRC88] WARD G. J., RUBINSTEIN F. M., CLEAR R. D.: A ray tracing solution for diffuse interreflection. In *SIGGRAPH* (1988), ACM, pp. 85–92. 3