Eurographics Symposium on Rendering - Experimental Ideas & Implementations (2016)
E. Eisemann and E. Fiume (Editors)

Fast Shadow Map Rendering for Many-Lights Settings

K. Selgrad, J. Müller, C. Reintges, M. Stamminger

Computer Graphics Group, University of Erlangen-Nuremberg, Germany

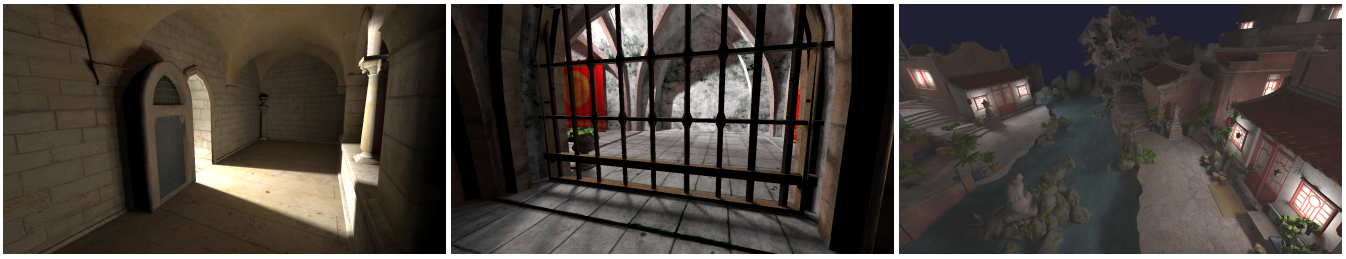


Figure 1: *With our culling approach, shadow map based visibility tests for many-lights rendering can be sped up considerably for a wide range of shadow map resolutions. The left image,* DOOR*, shows the upper level of the Sponza scene illuminated by VPLs distributed from a light source inside the wine cellar. Note the indirect shadow of the door and the shadows behind the column in the corner. In the center image,* GRILLE*, a fine structure casts indirect soft shadows from light reflected by the opposite wall. The right image,* VILLAGE*, shows diffuse lights cast from the windows in a village. Rendering time in comparison to light and camera frustum culling is reduced to, from left to right, 47%, 87% and 74% when using 512×512 parabolic shadow maps, and to 26%, 74% and 46% when using cube maps of the same resolution (per-face).*

Abstract

In this paper we present a method to efficiently cull large parts of a scene prior to shadow map computations for many-lights settings. Our method is agnostic to how the light sources are generated and thus works with any method of light distribution. Our approach is based on previous work in culling for ray traversal to speed up area light sampling. Applied to shadow mapping our method works for high- and low-resolution shadow maps and, in contrast to previous work on many-lights rendering, does neither entail scene approximations nor imposes limits on light range, while still providing significant gains in performance. In contrast to standard culling methods shadow map rendering itself is sped up by a factor of 1.5 to 8.6 while the speedup of shadow map rendering, lookup and shading together ranges from 1.1 to 4.2.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Picture/Image Generation—Color, shading, shadowing, and texture I.3.7 [Computer Graphics]: Picture/Image Generation—Visible line/surface algorithms

1. Introduction

Rendering scenes with hundreds of light sources allows smooth and detailed shading, producing much more pleasant and plausible results when compared to only using a small number of light sources. A larger number of light sources can be applied to provide particle lighting [OA11, HMY12], to support many manually placed lights (e.g. to set a certain mood or be placed on a map), or to distribute them via physically based light tracing [Kel97]. Until recently, only limited, local shading has been possible for shading many lights at interactive rates. Shadows were not included [DWS*88, OA11, DS06], or only computed from very approximate structures [RGK*08, HREB11].

We propose to tackle the challenge of shading many lights with shadows by efficiently culling large, non-convex parts of the scene prior to shadow map computation. Olsson et al. [OBS*15] propose a method that also relies on culling and achieve very good results for limited light sizes. Our method is not as fast, however, we remove the restriction of having only lights with a small radius of influence. Even though we lift this restriction our scheme still benefits from small lights and we believe that our approach might combine well with Olsson et al.'s work.

Contribution. We show how voxel-based culling can be used to remove large parts of the scene during shadow map rendering. Our method works over a wide range of shadow map resolutions and

clearly outperforms standard approaches. Our method is not approximate (aside from using shadow maps) and thus produces accurate results such as shadows from fine geometry that are not available in undersampled scene representations.

The remainder of this paper is structured as follows: In Section 2 we provide a short overview of general shadowing techniques and list more recent advances in computing shadows for many-lights settings. In Section 3 we review the non-convex voxel-based culling that our work builds upon, and in Section 4 we give a detailed description of how we apply this culling scheme to reduce the cost of computing shadows for many lights. We present our results in Section 5 and conclude with Section 6.

2. Related Work

Shadows are one of the major features that facilitate understanding of rendered 3D scenes, and in accordance with their fundamental importance there is a vast body of research on accurate and efficient rendering of shadows with varying feature sets. For interactive settings shadow volumes [Cro77] and even more so, shadow maps [Wil78], are the standard methods. With ever increasing ray traversal performance [ALK12, Gut14] and fast hierarchy construction [LGS*09, PL10, Kar12] ray tracing is becoming an interesting option for interactive settings [Sto15], especially when taking multiple samples [SMS15]. Recent alternatives to these established methods provide very high performance for compressed, precomputed shadows via voxel-representations [SKOA14, KSA15] and shadow map-based multi-resolution hierarchies [SBE16].

Apart from (point light) hard shadows there is also an enormous body of research on rendering soft shadows, that is shadows from light sources with spatial extent. The traditional solutions to this problem are simple accumulation of multiple samples via an accumulation buffer [HA90] or distributed ray tracing [CPC84]. There has been much work on soft shadowing methods based on shadow maps for which we refer to two recent books [ESAW11, WP12].

The use of many lights helps to create more interesting scenes with moody lighting, many separately moving light sources (e.g. rows of torches, particles) or to integrate the results of global illumination methods [Kel97, DKH*14]. Disregarding shadowing, shading with many lights can be computed in forward-rendering using tiled shading [OA11, OBA12] or when using deferred shading [DWS*88, ST90] via splatting [DS06, ML09, NW09]. However, especially for light sources not carefully placed by hand, such as when using global illumination methods in interactive contexts, this can lead to serious artefacts such as light bleeding [DS06, ML09, NW09]. To this end Laine et al. [LSK*07] present a method to cache a limited number of shadow maps and only incrementally rebuild them according to a predetermined budget. Ritschel et al. [RGK*08] propose to instead reduce the resolution of both the shadow maps as well as the scene representation by extremely downsampling both. Holländer et al. [HREB11] present a method to compute a level of detail structure for more efficient shadows in many-lights settings. In contrast to these approximations, Harada et al. [HMY12] compute shadows from many lights via ray tracing. To this end they use lights with limited radius of influence and apply culling in a tiled shading [OA11] fashion. More recently, Olsson et al. [OBS*15] presented a method for evaluating shadows of many lights with limited radius of influence. They further exploit virtual shadow maps and determine the resolution of each light's shadow map dynamically.

3. Voxel-based pre-BVH Culling for Casting Shadow Rays

Our work is based on a recent method to improve BVH construction and ray traversal performance when casting shadow rays to area light sources [SMS15]. We will shortly review this pre-BVH culling scheme as it will be our acceleration structure for shadow map creation.

The basic idea of pre-BVH culling is to use a coarse grid as a fast and lightweight classification structure. The input primitives of the scene are then spatially classified and assigned to the cells they overlap. Based on this mapping from grid cells to overlapping geometry, a simple process to select the part of the scene relevant for casting shadow rays to an area light is as follows: The grid is seeded with the cells visible to the observer tagged as 'source' cells, and the cells overlapping the light source as 'target' cells. With this setup a conservative 3D line is rasterized [AW87] from each source to each destination cell and all encountered cells are tagged as 'relevant' (naturally, only non-empty cells are tagged). Finally, the BVH is built using only geometry from those cells.

The major limitation of this approach is that a separate BVH for tracing shadow rays has to be constructed. Even though the BVH can be constructed much faster on the culled data, the gains for ray traversal are not equally strong (due to its logarithmic complexity). Therefore, pre-BVH culling only pays off in a certain band where enough shadow samples are taken to amortize the construction of the culled BVH and the cost for computing a very high quality BVH does not.

4. Voxel-based Culling for Many-Lights Rendering

In this section we describe how we apply the culling scheme reviewed in Section 3 to compute shadow maps for many-lights rendering. Figure 2 illustrates our culling process for sampling a single light source and Figure 3 outlines the algorithm itself: From the scene representation used for rendering we compute for each triangle which cells it overlaps and register it with all these cells. This structure is not view-dependent as it encompasses the entire scene. We compute the visible part of the scene in a (full resolution deferred or early-z) pass and find the voxels that the sampled positions map to. These are tagged as start voxels and used throughout the entire frame. We then iterate over all lights and for each light mark the voxel containing it as destination voxel. We traverse a conservative 3D line from each source voxel to this destination voxel, thereby tagging all encountered voxels as relevant for shadow map construction as they contain potential occluder geometry. In contrast to Selgrad et al. [SMS15] there is always a single destination voxel as (interactive) many-lights rendering is commonly based on point lights, yielding an increased culling ratio. The result of this culling is a much reduced subset of the scene which is then used to render to a shadow map for that light. This process is then repeated for each light source and the shading accumulated appropriately.

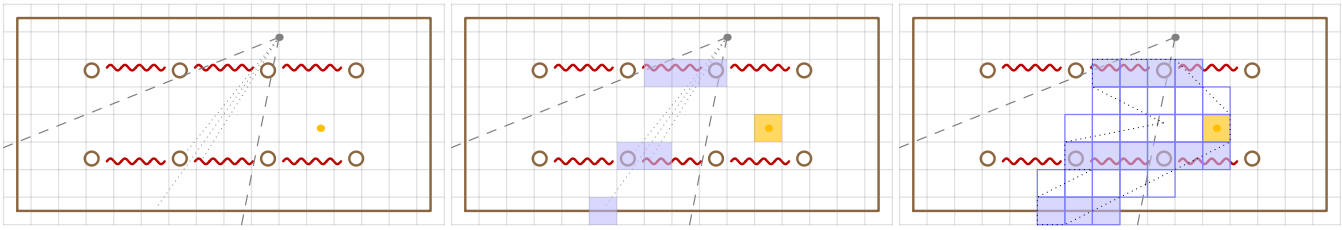


Figure 2: *Stages of culling for a single light, illustrated by a top-down view of a* SPONZA*-like scene with columns and curtains. Left: The camera frustum (gray) looks at the curtain and a small part of the scene behind it is visible (smaller, dotted frustum). Geometry relevant for shadows of the single point light will be determined. Center: The voxels containing visible geometry (blue) and the light (orange) are tagged. Right: The (non-empty) cells between the starting voxels and the light voxel are tagged (right); empty voxels encountered during line traversal are indicated with blue borders, only. Furthermore note the non-convex shape resulting from line traversal (dotted gray outline).*

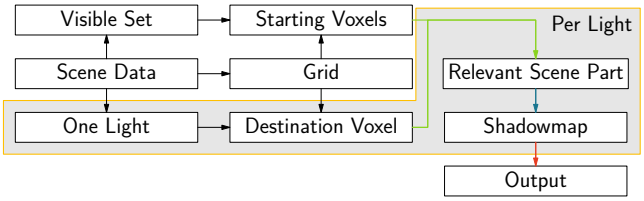


Figure 3: *Stages of our algorithm for fast shadows from many lights. Starting from scene data, the visible set is computed by rendering and starting voxels are tagged. For each light the voxel it contains is marked, and all starting voxels are connected with this voxel, tagging encountered cells as relevant during this process. Based on the thusly selected subset of the scene we generate the shadow map and accumulate shading in the final output buffer. The nodes in the shaded region are iterated for each light (cluster), colored edges map to the performance breakdown shown in Figure 5.*

Figure 2 shows this evaluation for one light source from two different camera positions and shows tagged and traversed voxels with the resulting geometry used for shadow computation. Note how the start voxels in the second row are scattered through the scene and the voxels traversed form a non-convex region of the scene. In the following we describe key aspects of our implementation of culling for many-lights rendering.

Culling for Many Lights vs Frustum Culling. The benefit of using this voxel-based culling in a rasterization context is that the number of primitives submitted affects the render time in a more linear fashion. Thus, culling efficiency translates directly to more efficient shadow map rendering, as our results in Section 5 show. In contrast to frustum culling our method incorporates information about the visible part of the scene, thereby also providing a form of occlusion culling as scene parts behind larger occluders (i.e. spanning an entire voxel) will be omitted.

Grid Resolution and Clipping. Our grid structure is very coarse and thus very fast to traverse and cheap to store in memory. We evaluated different sizes (see Section 5) and, per default, use a resolution of $32 \times 32 \times 32$. Note that the grid resolution also determines the number of batches that are submitted for rasterization when rendering the shadow maps.

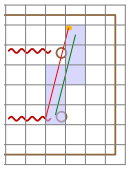
To ensure that no occluders are missed regardless of which combination of cells are selected, primitives have to be inserted into all voxels they overlap. However, this causes multiple rasterization of the same primitives if more than one voxel containing them are selected, which is especially costly for scenes containing very large triangles (such as Sponza).

We have evaluated two options to prevent this scenario: Clipping the geometry to each cell's extend and inserting the resulting triangles into the voxel's primitive list, or simple duplication where excessively large triangles are inserted into a separate list that is exempt from culling. With the latter method we sort triangles to this no-cull list as soon as they overlap more than two voxels. Note, however, that for faster rendering via parabolic shadow maps [BAS02] clipping triangles may be unavoidable to prevent rendering artifacts.

Clustering. As we employ conservative line rasterization to mark relevant voxels, mutual voxel-to-voxel visibility produces consistent results, regardless of the actual light or geometry positions involved. It follows that for lights falling into the same voxel, culling should only occur once, and not be repeated for each of these lights. As culling itself is a very fast process this is, however, only a minor optimization, see Section 5.

An interesting direction for future work would be to cluster lights more aggressively, for example by no longer restricting the culling process to $n : 1$, but to allow $n : m$ line traversals for m nearby light voxels. Culling efficiency will then drop with increasing m (by the spatial extend of the voxel cluster), however, more shadow maps could be rendered on the same data. Since culling is very fast it is not clear if the resulting performance loss from being overly conservative per-light will outweigh the reduced culling overhead.

Warp-parallel DDA. In previous work line rasterization is implemented by a simple DDA kernel that traverses the grid by iterative stepping, tagging the current voxel in each iteration. To ensure conservative culling Selgrad et al. [SMS15] also tag the one-ring around each voxel. This ensures that occluders between positions on the edges of voxels are also incorporated properly (as illustrated in the inset image showing the right part of the SPONZA-like scene). We improved this step twofold. We note that, as only a small subset of all voxels will be visible, there is only a limited number of starting voxels and thus the GPU will be underutilized during DDA traversal.



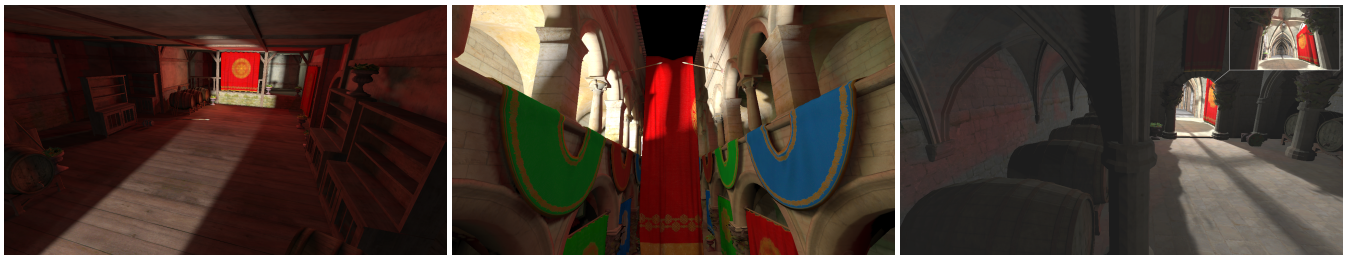


Figure 4: *Further test scenes that we use to evaluate the performance of our method:* ROOM *(left),* FLAG *(center) and* CELLAR *(right). In comparison to frustum culling render time is reduced to (f.l.t.r.)* 47%*,* 75% *and* 43% *for* 512×512 *parabolic shadow maps, and to* 29%*,* 68% *and* 23% *for cube shadow maps of the same resolution (per-face).*

We improve GPU utilization by employing a warp-parallel version of the line traversal. To this end we first select the longest axis of traversal and assign each step along this direction to a separate thread of the same warp. The entire line is then traversed in parallel where each thread only takes care of a slice orthogonal to the major traversal axis. This way we prevent tagging the one-ring around starting and end positions, which is clearly overly conservative, without introducing unnecessary checks in the inner traversal loop. Furthermore, checking if adjacent voxels should be tagged as well is also simplified. Note that this scheme maps particularly well to our default grid dimension described above. In our test scenes this optimization reduced the culling overhead by close to 50%, more importantly however, the number of triangles left after culling is reduced by 20%, resulting in a decrease in rendering time of 10% (see Section 5).

Parabolic vs Cube Maps and Tiling. Our evaluation shows results for both standard cube maps [Gre86] and parabolic shadow maps [BAS02]. In contrast to frustum culling, our approach is agnostic as to which shadow map projection is used.

Similarly to the encoding of imperfect shadow maps [RGK*08], the shadow maps computed by our method can also be evaluated in batches. To this end (when using parabolic maps) we store a number of (separately rendered) shadow maps in a tiled fashion, even when using higher resolution shadow maps. In our experiments we found that a 4×4 tiling works best during shadow map lookup over a wide range of resolutions. We have not looked into using virtual textures and adaptive shadow map resolution [OBS*15], but are confident that these approaches can be used to further increase the utility of our approach.

5. Results

In this section we compare rendering times of our method to using no culling at all and to dual (light and camera) frustum culling, for cube and paraboloid shadow maps from virtual point lights (note that light frustum culling is not very effective for paraboloid projections). We analyze our algorithm for the following test scenes:

DOOR The upper level of Sponza with light shining out of a door and bouncing on the balustrade, generating indirect shadows (see Figure 1, left, 190 VPLs).

GRILLE A grille in the wine cellar, light diffusely reflects off the wall and the thin bars cast soft, indirect shadows (see Figure 1, center, 96 VPLs).

VILLAGE A small village at night time, where the windows are illuminated by area lights shining directly at them, scattering light diffusely in the scene (see Figure 1, right, 548 VPLs).

ROOM A messy room in the wine cellar, illuminated by light reflecting off the carpet opposite the door (see Figure 4, left, 192 VPLs).

FLAG The Sponza scene, indirectly illuminated by a light source at the floor of the second level (see Figure 4, center, 384 VPLs).

CELLAR The wine cellar, illuminated by light from the lamp in the anteroom. The lamp's shadow is also generated by a number of VPLs sampled around its center (see Figure 4, right, 96 VPLs).

Figure 5 shows a breakdown of the average render time for shading a single light (note that a full tabulation of these and further times can be found in our supplemental material). It can be seen that using no culling at all is clearly inferior in all cases and that the rendering time of our method is consistently below the time required when using dual frustum culling. The figure shows that culling and draw-call generation (preparing the appropriate structure for `glMultiDrawElementsIndirect`) are negligible and that our warp-parallel approach is in fact faster than (also parallelized) frustum culling. This is due to the fact that, using our approach, only a subset of the voxels are even considered to be checked for relevance. Shading using the computed shadow maps is a constant offset as we use the same procedure and in most cases generating the shadow map is clearly more expensive.

The figure also shows the percentage of scene triangles left after culling (orange lines). When no culling is used all triangles are submitted for rasterization (for the left bar the line is always at 100%). This overlay illustrates that shadow map rendering performance is linearly dependent on the number of primitives rendered.

The difference of using our method in comparison to frustum culling is scene dependent, as Figure 5 shows. This is due to the fact that in certain cases the culled results are similar. In these cases (such as with GRILLE) the camera is aimed at the outer regions of the scene for which frustum culling is as efficient as our culling. Our culling is more efficient when there are further objects hidden behind occluding geometry. In this common scenario the number of voxels tagged is limited by the position of the starting voxels and the light's location, which can lead to large parts of the scene not even being considered (see Figure 2). This property can be interpreted as a limited form of integrated occlusion culling.

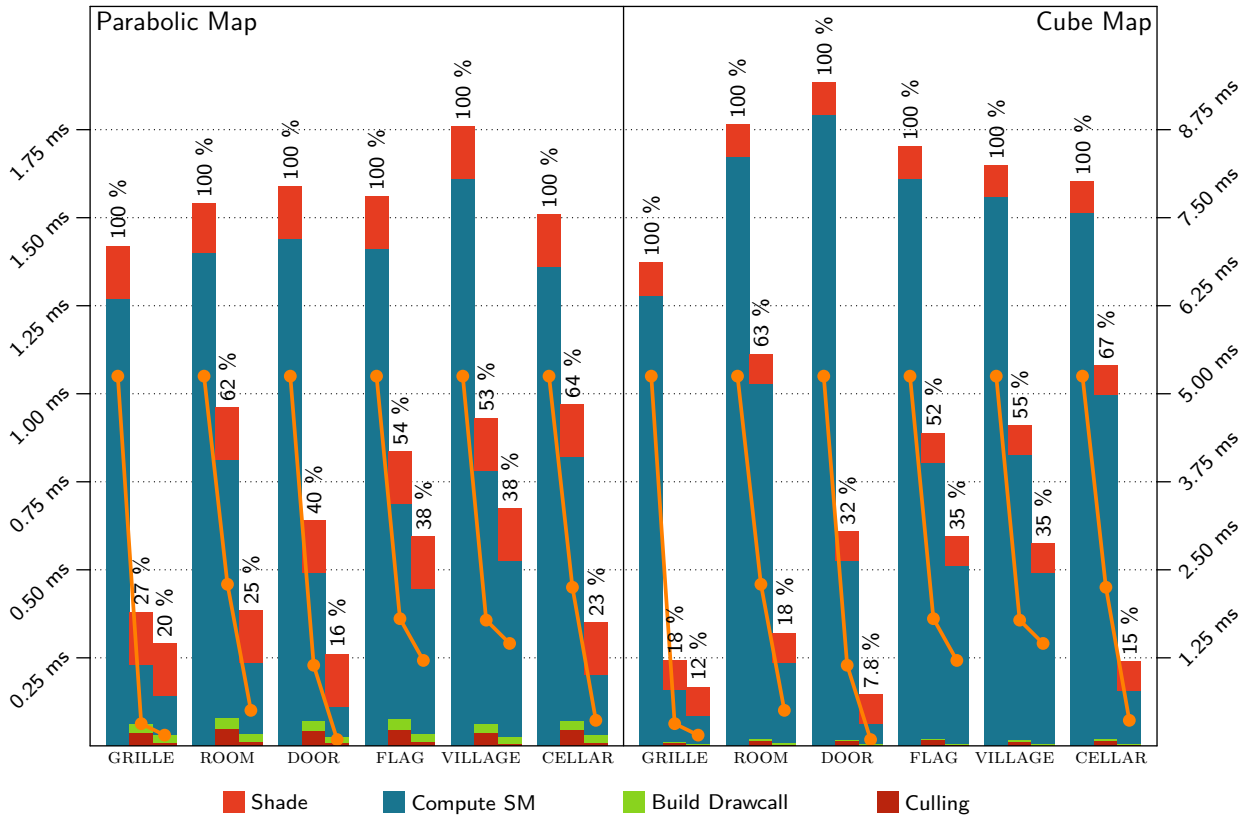


Figure 5: *Average time (in ms) to compute the different steps for shadows (at* 512×512*) from a single light in our test scenes. For each scene we break down the times (from left to right) for simple rendering, rending with dual-frustum culling and with voxel culling. The left part shows results using parabolic shadow maps, the right part shows cube map results (with* 512×512 *per face). Relative performance to not using culling is noted above each bar. The orange line further shows the relative number of primitives submitted for rasterization after culling (starting with* 100% *for no culling on the left bar).*

The total post-processing render times illustrated in Figure 6 show that our method achieves a considerable speed-up in most cases, even when accounting for the work common to the methods (e.g. framebuffer creation, shading per light, final composition). The figure does not include, however, the time to map triangles to voxels as this part of our method is, in its current form, a sequential routine executed on the CPU. Thus, the render times listed can only be achieved for static scenes, but with dynamic lights where shadows are computed on a per-frame basis. As described in Section 4, our method can also be run with a simpler grid construction process where very large primitives are kept in a separate list that is exempt from culling. For a limited number of dynamic objects (such as a few characters) this method can be employed to remove that limitation. When this version is used the construction overhead in our test scenes is between 40ms and 50ms, which, as can be seen in Table 6 moves our method closer to frustum culling, but does not cause or method to be slower in most cases. However, we believe that fully dynamic clipping and mapping of the input primitives can be implemented to be very fast and we consider this one of the most important directions for future work. Furthermore, even the faster, non-clipping binning approach is still a sequential CPU-bound process and will be subjected to optimization in future work. The difference in rendering time between those two approaches to binning is insubstantial, suggesting that the easier way of keeping excessive triangles separately might lead to much improved results quickly (see tabulation in our supplemental material).

Of the smaller-scale optimizations (see Section 4) using our warp-parallel line rasterization results in a decrease in rendering time of 10%, and clustering yields a speed-up from 4.5% to 9% We have evaluated our method on shadow map resolutions from 256×256 up to 2048×2048 for all our test scenes for both parabolic and cube shadow maps (see our supplemental material) and found that our method performs very similar to the results presented in Figures 5 and 6 when compared to frustum culling.

6. Conclusion

In this paper we have shown how we can apply voxel-based culling to speed up shadow map rendering for many-lights settings. We have shown that culling efficiency directly translates to rendering efficiency and that the overhead of using our method does not exceed that of frustum culling. We have further shown that our approach is consistently faster than frustum culling that considers both camera and light setup.

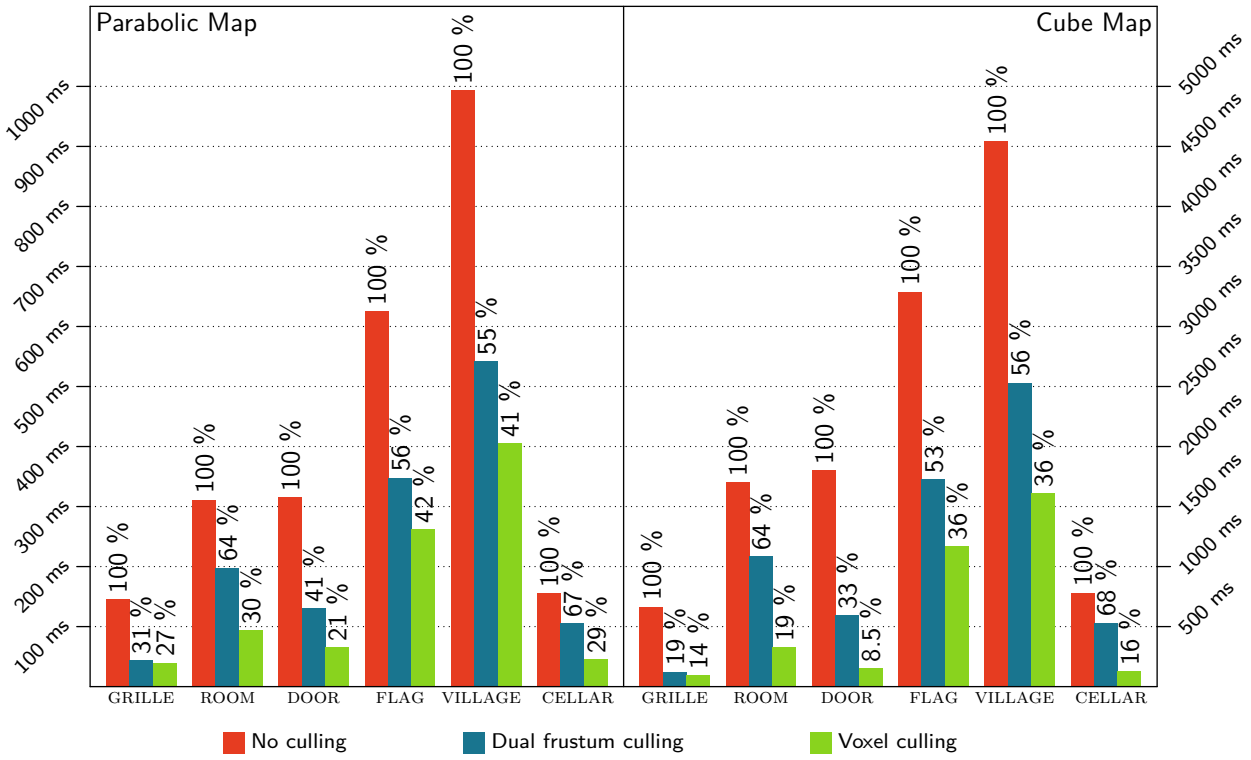


Figure 6: *Rendering time (including shadow map generation, shadow lookups and shading) for one frame of the scenes described at the outset of Section 5 (see also Figures 1 and 4). Note that even when incorporating common factors to produce the image, our method consistently achieves shorter rendering times.*

Future Work. Our work provides many options for future work. The most important direction will be to provide a fast binning approach to construct the initial grid to allow for fully dynamic, not only partially dynamic, scenes. We believe that this will not be an obstacle as the process should map well to highly parallel GPU construction. Further directions will be to evaluate if more approximate structures can benefit from voxel-based culling. One example would be allowing for higher quality approximations with imperfect shadow maps [RGK*08] (sampling more points, and culling them before rendering), or speeding up Many-Lods [HREB11]. We would also like to investigate multi-resolution representations of our grid to better adapt to highly sparse regions. Another promising direction could be to investigate more aggressive (and view-dependent) clustering, potentially along the lines of lightcuts [WFA*05]. We also believe that adapting virtual textures and adaptive shadow map resolution from Olsson et al.'s work [OBS*15] will further improve our method.

Acknowledgments

The authors gratefully acknowledge the generous funding by the German Research Foundation (GRK 1773).

References

[ALK12] AILA T., LAINE S., KARRAS T.: *Understanding the Efficiency of Ray Traversal on GPUs – Kepler and Fermi Addendum*. NVIDIA Technical Report NVR-2012-02, NVIDIA Corporation, June 2012. 2

[AW87] AMANATIDES J., WOO A.: A Fast Voxel Traversal Algorithm for Ray Tracing. In *In Eurographics '87* (1987), pp. 3–10. 2

[BAS02] BRABEC S., ANNEN T., SEIDEL H.-P.: Shadow Mapping for Hemispherical and Omnidirectional Light Sources. In *In Proc. of Computer Graphics International* (2002), pp. 397–408. 3, 4

[CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed ray tracing. In *Proceedings SIGGRAPH '84* (1984), ACM, pp. 137–145. 2

[Cro77] CROW F. C.: Shadow algorithms for computer graphics. *Computer Graphics (Proc. of SIGGRAPH) 11*, 2 (1977), 242–248. 2

[DKH*14] DACHSBACHER C., KŘIVÁNEK J., HAŠAN M., ARBREE A., WALTER B., NOVÁK J.: Scalable Realistic Rendering with Many-Light Methods. *Computer Graphics Forum 33*, 1 (2014), 88–104. 2

[DS06] DACHSBACHER C., STAMMINGER M.: Splatting indirect illumination. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), ACM, pp. 93–100. 1, 2

[DWS*88] DEERING M., WINNER S., SCHEDIWY B., DUFFY C., HUNT N.: The Triangle Processor and Normal Vector Shader: A VLSI System for High Performance Graphics. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1988), SIGGRAPH '88, ACM, pp. 21–30. 1, 2

[ESAW11] EISEMANN E., SCHWARZ M., ASSARSSON U., WIMMER M.: *Real-Time Shadows*. A.K. Peters, 2011. 2

[Gre86] GREENE N.: Environment mapping and other applications of

world projections. *IEEE Comput. Graph. Appl. 6*, 11 (Nov. 1986), 21–29. 4

[Gut14] GUTHE M.: Latency Considerations of Depth-first GPU Ray Tracing. In *Eurographics 2014 - Short Papers* (2014), Galin E., Wand M., (Eds.), The Eurographics Association. 2

[HA90] HAEBERLI P., AKELEY K.: The accumulation buffer: hardware support for high-quality rendering. In *Proceedings SIGGRAPH 1990* (1990), ACM, pp. 309–318. 2

[HMY12] HARADA T., MCKEE J., YANG J. C.: Forward+: Bringing Deferred Lighting to the Next Level. In *Eurographics 2012 - Short Papers Proceedings, Cagliari, Italy, May 13-18, 2012* (2012), pp. 5–8. 1, 2

[HREB11] HOLLÄNDER M., RITSCHEL T., EISEMANN E., BOUBEKEUR T.: ManyLoDs: Parallel Many-View Level-of-Detail Selection for Real-Time Global Illumination. *Computer Graphics Forum (Proc. EGSR 2011)* (2011). 1, 2, 6

[Kar12] KARRAS T.: Maximizing parallelism in the construction of bvhs, octrees, and k-d trees. In *Proceedings of the EUROGRAPHICS Conference on High Performance Graphics 2012, Paris, France, June 25-27, 2012* (2012), pp. 33–37. 2

[Kel97] KELLER A.: Instant Radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 49–56. 1, 2

[KSA15] KÄMPE V., SINTORN E., ASSARSSON U.: Fast, Memory-Efficient Construction of Voxelized Shadows. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2015), ACM. 2

[LGS*09] LAUTERBACH C., GARLAND M., SENGUPTA S., LUEBKE D., MANOCHA D.: Fast BVH Construction on GPUs. *Computer Graphics Forum* (2009). 2

[LSK*07] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental Instant Radiosity for Real-Time Indirect Illumination. In *Proceedings of Eurographics Symposium on Rendering 2007* (2007), Eurographics Association, pp. 277–286. 2

[ML09] MCGUIRE M., LUEBKE D.: Hardware-accelerated global illumination by image space photon mapping. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), ACM, pp. 77–89. 2

[NW09] NICHOLS G., WYMAN C.: Multiresolution Splatting for Indirect Illumination. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2009), I3D '09, ACM, pp. 83–90. 2

[OA11] OLSSON O., ASSARSSON U.: Tiled Shading. *Journal of Graphics, GPU, and Game Tools 15*, 4 (2011), 235–251. 1, 2

[OBA12] OLSSON O., BILLETER M., ASSARSSON U.: Tiled and Clustered Forward Shading. In *SIGGRAPH '12: ACM SIGGRAPH 2012 Talks* (New York, NY, USA, 2012), ACM. 2

[OBS*15] OLSSON O., BILLETER M., SINTORN E., KÄMPE V., ASSARSSON U.: More Efficient Virtual Shadow Maps for Many Lights. *Visualization and Computer Graphics, IEEE Transactions on 21*, 6 (June 2015), 701–713. 1, 2, 4, 6

[PL10] PANTALEONI J., LUEBKE D.: HLBVH: Hierarchical LBVH Construction for Real-time Ray Tracing of Dynamic Geometry. In *Proceedings of the Conference on High Performance Graphics* (Aire-la-Ville, Switzerland, Switzerland, 2010), HPG '10, Eurographics Association, pp. 87–95. 2

[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. 27*, 5 (Dec. 2008), 129:1–129:8. 1, 2, 4, 6

[SBE16] SCANDOLO L., BAUSZAT P., EISEMANN E.: Compressed Multiresolution Hierarchies for High-Quality Precomputed Shadows. *Computer Graphics Forum (Proc. Eurographics) 35*, 2 (May 2016). 2

[SKOA14] SINTORN E., KÄMPE V., OLSSON O., ASSARSSON U.: Compact Precomputed Voxelized Shadows. *ACM Transactions on Graphics 33*, 4 (2014). 2

[SMS15] SELGRAD K., MÜLLER J., STAMMINGER M.: Faster Ray-Traced Shadows for Hybrid Rendering of Fully Dynamic Scenes by Pre-BVH Culling. In *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference* (2015), Giachetti A., Biasotti S., Tarini M., (Eds.), The Eurographics Association. 2, 3

[ST90] SAITO T., TAKAHASHI T.: Comprehensible Rendering of 3-D Shapes. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1990), SIGGRAPH '90, ACM, pp. 197–206. 2

[Sto15] STORY J.: Hybrid Ray-Traced Shadows, 2015. GDC'15. 2

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: A scalable approach to illumination. *ACM Trans. Graph. 24*, 3 (July 2005), 1098–1107. 6

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. *Computer Graphics (Proc. of SIGGRAPH) 12*, 3 (1978), 270–274. 2

[WP12] WOO A., POULIN P.: *Shadow Algorithms Data Miner*. A K Peter/CRC Press, 2012. 2