LightCluster - Clustering Lights to Accelerate Shadow Computation

Daniel Wiesenhütter Andreas Klein Alfred Nischwitz
Munich University of Applied Sciences
Lothstrasse 64

80335 Munich, Germany
daniel@wiesenhuetter.me, andreas.klein@hm.edu, nischwitz@cs.hm.edu

ABSTRACT

In this paper, we propose a method to reduce the amount of shadow maps required for rendering shadows in scenes with many lights. Our idea is to use the spatial relationship of lights to find clusters and replace the lights of a cluster with a single area light. We use a soft shadow algorithm for area lights to approximate the shadows for the clusters. By carefully placing the cluster centers, we can minimize the errors in the shadows. While the clustering only adds a small overhead in the worst case, it can efficiently reduce the number of shadow maps. Thus, in many cases the resulting error in shadows is acceptable compared to the increase in rendering performance.

Keywords

Light Clustering, Shadow Mapping, Soft Shadows, Shadows for Many Lights.

1 INTRODUCTION

Shadows are an important part of a visualization and give the viewer supplemental details about the appearance of objects. In real-time rendering, shadow mapping [Wil78a] is a popular approach to compute shadows. However, a shadow map must be computed for each light and thus, the memory and the computation time increases with the number of lights.

In this paper, we present an approach, called LightCluster, to automatically select representative light sources and accelerate the computation of direct shadows for scenes with many lights. We carefully select light sources as cluster centers and cluster the remaining lights using a minimum distance metric [Wol57a]. We represent each cluster by an area light source and use a soft shadow algorithm to render shadows for each cluster, such as Percentage Closer Filtering [Ree87a] and Percentage Closer Soft Shadows [Fer05a].

In our implementation, we use omnidirectional point lights. However, the approach can be adapted for other light types, such as directional or spot lights. Figure 1 shows an example of our approach.

The main contributions of our paper are:

- A clustering strategy applied to point lights that selects a variable number of existing light sources as cluster centers.
- A minimum distance metric in order to minimize the amount of shadow maps and to trade off between quality and performance.
- An approximation of point light shadows by using an area light source where the area depends on the minimum distance between the clusters.

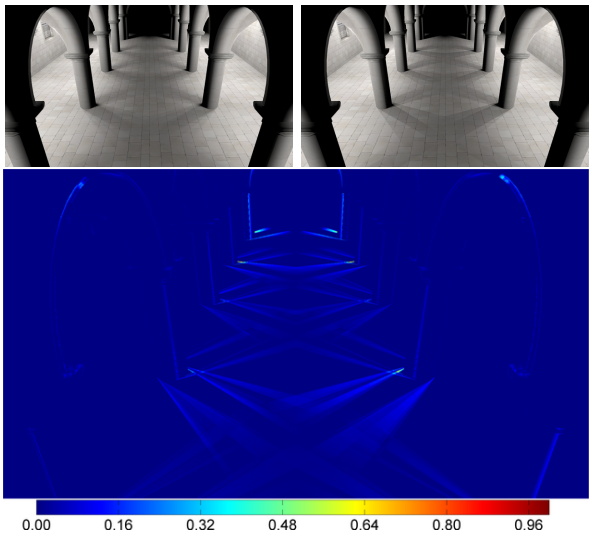


Figure 1: Comparison between a reference solution and our solution for the Dabrovic Sponza scene at 1920x1080 resolution. The top left image shows the reference solution where 80 point light shadows are rendered in 97.8 ms. The top right image demonstrates our solution with 26 cluster shadows using PCSS in 43.1 ms. The bottom image shows the difference between both above images. Note that a dark blue color indicates a low error and red color a high error.

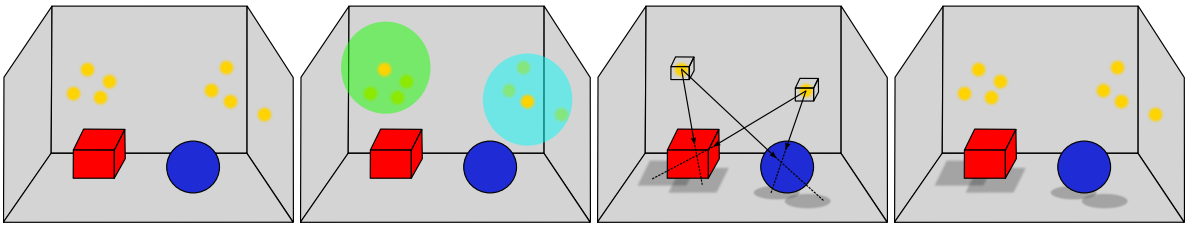


Figure 2: Overview for LightCluster. For a given distribution of point light sources, clusters are selected with the min-distance-cluster algorithm. For each cluster, a cube shadow map is rendered, soft shadows are computed and stored in a visibility texture. This textures are then used during shading to determine the visibility.

2 RELATED WORK

2.1 Real-Time Soft Shadows

The rendering of soft shadows is an active field of research. Therefore, we focus our review on publications closely related to our work. For an exhaustive survey on other methods see [Eis11a].

Shadow Mapping [Wil78a] is a popular method to compute shadows in real-time rendering. The idea is to assume point light sources and to replace the visibility test by comparing depth values from the light's point of view and the observer's point of view. In order to compute shadows for omnidirectional lights, a cube shadow map can be rendered [Ger04a].

Percentage Closer Filtering (PCF) [Ree87a] computes filtered hard shadows by making multiple shadow comparisons within a filter window. This idea is further extended by Fernando [Fer05a] with Percentage Closer Soft Shadows (PCSS) to realize shadows with variable sized penumbras. Instead of using a fixed filter per pixel, the filter window is scaled according to a penumbra size. The penumbra size can be estimated by calculating an average blocker depth and using similar triangles.

2.2 Many-Light Methods

Instant Radiosity [Kel97a] approximates global illumination by distributing virtual point lights (VPLs) and calculating a local illumination model for each VPL. Indirect shadows are realized by rendering a shadow map for each VPL. In order to distribute VPLs by textures, Reflective Shadow Maps (RSMs) [Dac05a] are rendered from the position of light.

In Lightcuts [Wal05a] a binary tree for light sources is built. In each frame the light tree is traversed and a cut is calculated. This light cut determines the relevant lights or representatives with the help of visual criteria, such as geometric properties or material.

Recent approaches try to accelerate the computation of Instant Radiosity by reusing shadow maps for VPLs [Lai07a] or by rendering low resolution shadow maps for a coarse representation of the scene [Rit08a].

Hašan et al. [Haš07a] interpreted the relationship between m surfaces and n lights as a $m \cdot n$ matrix and samples only a small number of rows and columns. In order to minimize flickering in many-light animations, the matrix can be extended to a tensor [Haš08a].

Clustered Visibility [Don09a] accelerates the computation of indirect shadows with RSMs. The method uses k-means clustering on the RSMs to build clusters of VPLs. They interpret each cluster as an area light in order to accelerate the visibility test.

The idea of [Don09a] is closely related to our work. In contrast to Clustered Visibility we focus our work on high frequency shadows for direct lighting. As the lights are not distributed by RSMs, a k-means clustering may lead to errors. A central cluster position can result in a representative light source that is occluded by geometry, i.e. located within walls. Our approach uses an existing light source as a cluster center and clusters the remaining lights with a minimum distance metric.

3 LIGHTCLUSTER

LightCluster reduces the total amount of shadow maps in order to minimize the computation of direct shadows for point lights. To describe our approach in more detail, we separate it into two steps. First, the point lights are clustered and cluster centers are selected. Second, for each cluster a shadow maps is rendered, soft shadows are computed and the result is stored in a set of visibility textures. During shading, we select the representative visibility texture for each point light and use it to determine the visibility factor. Figure 2 illustrates the approach.

3.1 Clustering

In order to reduce the error in shadows, we perform a two pass clustering with different metrics in each pass. Our clustering proceeds as follows. We first select point lights as cluster centers by using the light range and a minimum-distance metric, which is scaled by the camera distance. This allows us to generate smaller clusters and thus, more shadow maps, near the camera position. In the second clustering pass, the remaining point lights are assigned to the nearest cluster centers. Therefore, the error in shadows due to the reduced amount of

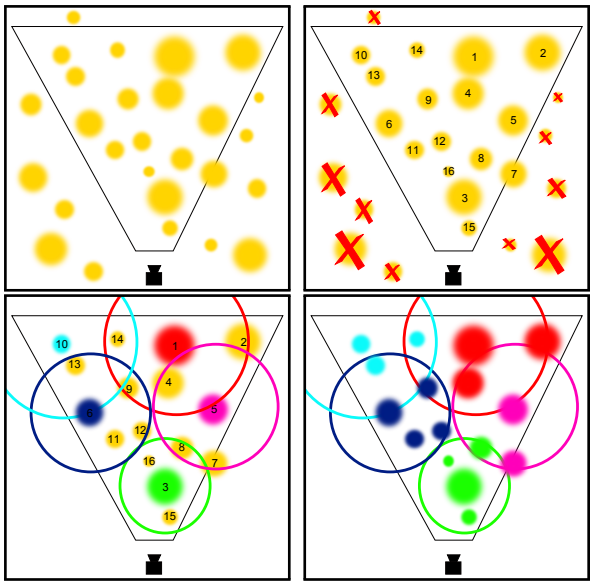


Figure 3: Minimum-distance-cluster algorithm in a 2D example scene. The size of the point lights describes its attenuation. **Top left:** Light distribution in the scene. **Top right:** The point lights are culled against the view frustum and sorted by the attenuation factor. **Bottom left:** Finding cluster centers. Note how the minimum distance d_{min} increases based on the distance to the camera. Due to the sorting, the lights with the highest range are first tested as cluster centers. **Bottom right:** Assigning the remaining point lights to the closest cluster centers. Without this step, lights would assigned to the first cluster within d_{min}.

shadow maps can be reduced. Figure 3 illustrates the clustering for a 2D example scene.

3.1.1 *Minimum-Distance-Cluster Algorithm*

The minimum-distance-cluster algorithm [Wol57a] is a hierarchical clustering algorithm and clusters all objects within a user defined minimum distance. The algorithm can be realized in a top-down strategy and proceeds as follows. First, it is assumed that all objects are within a single cluster. The algorithm iterates over all objects of a cluster and splits them into new clusters, if their distance exceeds the defined minimum distance. This step is repeated until no more clusters can be split.

In contrast to a centroid-based clustering, such as k-means [McQ67a], the number of clusters need not be known in advance. Furthermore, the cluster center is an actual light source and no computed centroid position. Thereby, the position for one point light in a cluster is correct for the shadow computation with PCF or PCSS. The error in the shadows for the remaining lights can be adjusted by the minimum distance parameter.

3.1.2 *Selecting Cluster Centers*

In order to find potential cluster centers, we first search for point light sources that will potentially cast a shadow in the view frustum. As we assume point light sources, we create a bounding sphere for each point light where the radius equals the attenuation range. We then cull the bounding spheres against the view frustum and remove all lights that are classified as outside. In the next step, we sort the visible lights by the attenuation range. As a result, our clustering algorithm will first test the lights with the highest range as cluster centers. This heuristic could be further improved by determining how much of the light source is visible in the view frustum, for example by utilizing occlusion culling or a hierarchical z-buffer [Gre93a].

In the next step, we select the cluster centers. We assume that all lights are within a single cluster and the light with the highest range is the first cluster center. For every point light, we compute the distance to the cluster center and compare it against the minimum distance d_{min}. In order to generate smaller clusters near the camera position, we scale the minimum distance linearly based on the distance to the camera. The minimum distance can be calculated as follows:

$$d_{min} = d_{init} \cdot \left(1 - \frac{|C_{pos}|}{z_{far}}\right)$$

where d_{init} is the initial distance between clusters, C_{pos} the position of the cluster center and z_{far} the maximum view distance.

If the distance to the cluster is greater than the calculated d_{min}, the light is promoted to a new cluster center. The lights that are within d_{min}, are not automatically assigned to the cluster, but marked as unclustered for the next step.

3.1.3 *Assigning the remaining Lights*

After all cluster centers are selected, we assign the lights to the clusters. We iterate over all lights that are marked as unclustered and calculate the distance to all cluster centers. The light is assigned to the cluster with the shortest distance. In addition, the maximum attenuation range C_{att} of the cluster center is adapted as follows:

$$C_{att} = \max\left(C_{att}, d_C + L_{att}\right)$$

where d_C is the distance to the cluster and L_{att} is the attenuation of the light source. We use the maximum attenuation range C_{att} to accelerate the computation of the shadows for a cluster.

3.2 Computing Shadows

Instead of rendering shadows for each point light in a scene, we calculate the shadows only for a smaller set of cluster centers. In order to reduce the error in the resulting shadows, we interpret each cluster center as an area light source and calculate the visibility factor using a soft shadow algorithm.

3.2.1 Direct Lighting Equation with Light Clusters

Following the notation of [Eis11a], the direct-lighting equation for n point lights can be expressed as:

$$L_o(p,\omega) = \sum_{i=0}^{n} L_d(p,\omega,l_i) \cdot L_{c_i} \cdot V(p,l_i)$$

with

$$L_d(p,\omega,l_i) = f_r(p,\omega,p \to l_i) G(p,l_i)$$

where p is a point on a surface, ω a direction, f_r is a Bidirectional Reflectance Distribution Function (BRDF), V the visibility function, L_{c_i} the color of the i-th light and l_i the position of the i-th point light. The notation $p \to q$ is defined as $\frac{q-p}{\|q-p\|}$. The geometric term G is defined as

$$G(p,q) = \frac{cos(n_p, p \to q) cos(n_q, q \to p)}{\|p-q\|^2}$$

with the normal n.

With the clustering, we replace the binary visibility of point lights with the visibility of an area light source A for a cluster c, which yields the following equation:

$$L_o(p,\omega) \approx \sum_{i=0}^{n} L_d(p,\omega,l_i) \cdot L_{c_i} \cdot \frac{1}{A_c} \int_{A_c} V(p,l) dl$$

3.2.2 Rendering Shadows

We render a cube shadow map for each cluster and interpret the cluster as a disc-shaped area light source. The radius of the area light source is given by the minimum distance d_{min} of the cluster. We use this radius to scale the filter window of PCSS [Fer05a] and PCF [Ree87a]. Due to the maximum attenuation range C_{att} of a cluster, we discard a pixel if the distance from the pixel to the cluster center exceeds the attenuation range. The visibility factor is then stored in a texture for each cluster and is accessed during shading. This allows us to calculate the shadows iteratively and reduces the texture memory from a cube shadow map to a screen sized texture per cluster. In this way, we sample the cube shadow maps only once for each cluster and avoid additional PCF or PCSS sampling for each light source during shading.

After all cluster shadows have been computed, we use the set of visibility textures for shading. We shade the scene with each point light source and modulate the resulting color with the visibility value stored in the texture for the point light's cluster.

4 IMPLEMENTATION

We implemented our algorithm using Direct3D 11 and Tiled Deferred Shading [Lau10a]. However, LightCluster can also be implemented with any other method for shading many point lights, such as Tiled Forward Shading [Ols11a]. In each frame, we cluster the point lights, render the G-Buffer, compute the shadows for the clusters and perform the shading.

Our minimum-distance-cluster algorithm is entirely implemented on CPU, as it adds only a small overhead (see section 5 for details). The cluster position, attenuation range and a cluster index per light source is stored in an unordered access view on the GPU and can be accessed in the shaders.

In order to render a cube shadow map in a single render pass, we use a geometry shader to replicate the geometry for each face direction. We store the world-space distance in each face and reconstruct the world-space position from the depth stored in the G-Buffer during the shadow test. The set of visibility textures is realized with a texture array.

5 RESULTS

The following results are rendered on an Intel Xeon E5620 CPU with 2.4 GHz, 8 GB RAM and a NVIDIA GeForce GTX 680 graphics card with 2048 MB memory. The screen resolution for all images is 1920x1080 and the resolution of one face in a cube shadow map is set to 1024x1024. We use a Poisson disk with eight samples for PCF and PCSS.

5.1 Scenes

We present images and measurements for three different scenes. In the Dabrovic Sponza scene 80 lights are placed in circles of eight lights sidewards in the arcade to the right and left (Figure 7a). Within the Cornell Box scene 32 light sources are randomly distributed in the room (Figure 8a). The last scene is a part of a restaurant with two tables and a chandelier. Fourteen lights overall are placed, at the candle stands on the table, at lamps at the wall and at the chandelier (Figure 9a).

5.2 Reference solution

As a reference solution to our approach, we compute shadows with cube shadow mapping for every light source. Since shadows are computed for point light sources, a binary visibility test determines if the pixel is in shadow or not. Consequently, 80 cube shadow maps for Dabrovic Sponza (Figure 7b), 32 cube shadow maps for Cornell Box (Figure 8b) and fourteen cube shadow maps are rendered for the Restaurant scene (Figure 9b).

5.3 Memory

Compared to the reference solution, LightCluster uses additional memory. During the shadow computation, the cluster visibility values are stored in a texture array. Therefore, the additional memory size is cluster count multiplied by the screen resolution. For example, in the Dabrovic Sponza scene with 26 clusters the additional memory size is 205 MB. In contrast, the memory size for 26 cube shadow maps is 624 MB.

Sponza		Time (ms)		**Cornell Box**		Time (ms)		**Restaurant**		Time (ms)	
d_{init}	Clusters	PCF	PCSS	d_{init}	Clusters	PCF	PCSS	d_{init}	Clusters	PCF	PCSS
0.1	80	100.1	115.3	0.3	13	12.9	18.0	0.2	12	36.0	40.1
0.3	44	57.0	67.3	0.4	6	9.8	12.9	0.3	9	28.9	32.9
0.5	26	36.4	43.1	0.5	5	9.1	12.1	0.4	8	26.8	30.5
1.6	10	18.7	20.7	0.6	4	8.6	11.1	0.5	7	24.9	27.8
Ref. with 80 lights		97.8		Ref. with 32 lights		20.0		Ref. with 14 lights		39.3	

Table 1: Duration of LightCluster for the scenes with different d_{init} factors. The higher the factor, the coarser the clustering and lesser the amount of clusters. Thus, the chosen d_{init} factor determines the performance.

5.4 Performance

We compare our performance results against the reference solution. In all our test scenes we use a maximum view distance z_{far} of 30 and vary only the initial distance d_{init} between clusters for the clustering.

As can be seen from Table 1, LightCluster is faster than the reference solution, depending on the chosen d_{init} and the resulting amount of clusters. The higher the d_{init} factor, the coarser the clustering and a lesser amount of cube shadow maps is required. Thus, the chosen d_{init} factor determines the performance of our approach. The worst case scenario for our approach occurs when no lights can be clustered. This can occur when the parameter d_{init} is chosen too small for a given scene. Table 1 shows this case for the Dabrovic Sponza scene with 80 clusters. The performance in this worst case is slightly slower than for the reference solution, because an additional clustering as well as a PCF or PCSS sampling is performed. However, the worst case can be avoided by comparing the amount of lights with the cluster count and choosing the reference solution render path.

Sponza	Time (ms)		Percentage	
Step	PCF	PCSS	PCF	PCSS
Clustering	0.05	0.05	<0.1%	<0.1%
GBuffer	0.53	0.53	0.9%	0.8%
Shadows	49.20	59.42	86.3%	88.4%
Lighting	7.05	7.05	12.3%	10.4%
Compositing	0.20	0.20	0.4%	0.3 %
Total	57.03	67.25	100%	100%

Table 2: Duration of the algorithm steps using the Dabrovic Sponza dataset with 44 clusters.

The duration of the single steps in our algorithm is shown in Table 2 for the Dabrovic Sponza scene with 44 clusters. It can be observed, that the bottleneck in our algorithm is the computation of shadows. The clustering step requires less than 0.1% of the total performance. Thus, a clustering of the light sources can be performed for every frame in order to reduce the amount of cube shadow maps.

Figure 4 shows the performance of the clustering step when using a different amount of lights. The clustering time increases nearly linear with the number of lights.

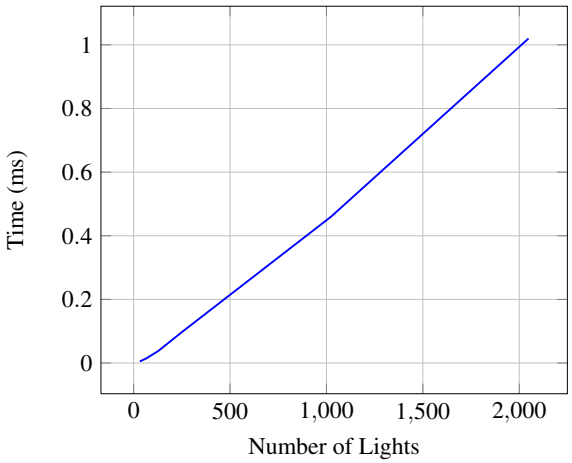


Figure 4: Timings for the min-distance-clustering for a different number of lights.

With 2048 lights, the clustering can be still performed within 1 ms of time.

5.5 Error

In order to compare the error of our approach to the reference solution, we use difference images. The dark blue color of a difference image indicates a low error and a red color indicates a high error.

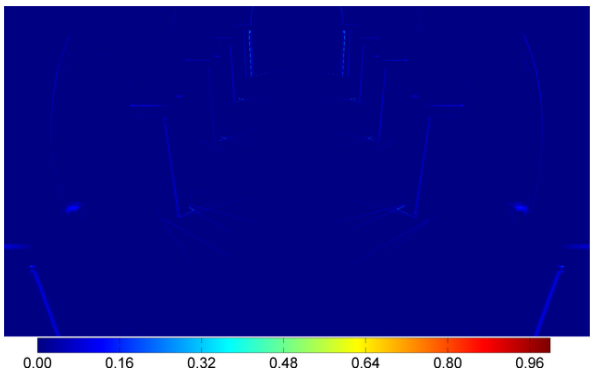


Figure 5: Comparison between the shadows of PCF and PCSS.

Figure 5 shows a difference image between PCSS and PCF shadows. It can be observed, that the PCSS solution generates a contact hardening shadow at the bottom of the columns. Therefore, PCSS provides a more

realistic solution. Nevertheless, PCF can be used to accelerate the computation of shadows further (see Table 1). For the following comparisons we use LightCluster with PCSS.

Figure 7 shows our algorithm with two different cluster settings for the Dabrovic Sponza scene. Figure 7a and 7b shows the light distribution and the reference solution. In the first case, we use an initial distance $d_{init} = 0.5$ which results in 26 clusters. The cluster locations are illustrated in Figure 7c. As can be seen in the difference image, there is a small error at the shadow boundaries and slightly larger errors near the far wall, where the number of clusters is reduced due to the distance scaling. For the second case, we use an initial distance $d_{init} = 1.6$. The clustering step produces ten clusters (Figure 7d). In this case the shadows are rendered for only one out of eight light sources and errors can be clearly seen in the resulting image (Figure 7f) as well as the difference image (Figure 7g). Nevertheless, the general impression of the shadows in the scene is preserved and the rendering performance is increased by a factor of 4.8.

The results in the Cornell Box scene with randomly distributed point lights are presented in Figure 8. The light distribution is shown in Figure 8a and the reference solution in Figure 8b. On the left side, Figure 8c displays the clusters locations with $d_{init} = 0.5$, which results in five clusters. Due to the reduction from 32 point lights to five clusters, the image shows errors in the shadows cast from the left cube and below the right sphere (Figure 8e and 8g). By increasing the initial distance to $d_{init} = 0.6$, 32 point lights are reduced to four clusters. The results in Figure 8f and 8h shows, that the error on the wall behind the sphere increases.

In the Restaurant scene, fourteen lights are distributed (Figure 9a) and Figure 9b shows the reference solution. Due to the large distance between the light sources in this scene, only the lights of the chandelier and the candles are suited for clustering. In Figure 9c the lights of the chandelier are represented by three clusters, which is accomplished by $d_{init} = 0.3$. Figure 9e presents our approximation. The resulting difference image in Figure 9g shows small errors in the shadows cast by the chandelier. By using a initial distance of $d_{init} = 0.4$, the light sources of the chandelier can be represented by two clusters (Figure 9d). The results in Figure 9f and 9h show, that the error in the shadows at the floor is increased.

Figure 6 shows the root mean square error (RMSE) for our approach using different initial cluster distances and PCSS in our test scenes. The error increases when the number of cluster centers and thus the number of shadow maps is adapted, e.g. in the Restaurant scene with a distance of 0.4 and 0.5. On the other hand, the error is nearly constant when the lights are assigned to different clusters according to the second step in the clustering, e.g. in the Cornell scene with a cluster distance between 0.4 and 0.7.

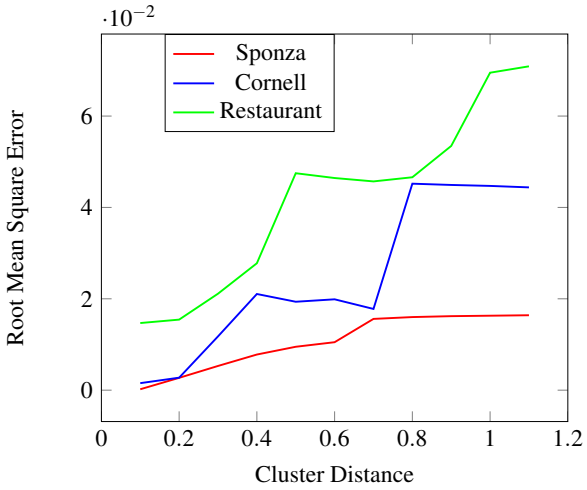


Figure 6: RMSE for different initial cluster distances.

5.6 Discussion

We used a minimum-distance clustering algorithm to cluster the light sources. Compared to a k-means clustering, the minimum-distance clustering has the following advantages in this scenario. First, the number of cluster centers must not be known in advance. This simplifies the algorithm and no additional heuristics for estimating the number of cluster centers in advance are required. Second, we place a cluster center always on an existing light source. Therefore, the light position for shadow mapping is always correct for at least one light source in the cluster. The k-means algorithm on the other hand uses the mean of the light positions within a cluster. As the lights are distributed in space, this could result in cluster centers which are located within geometry, e.g. in a wall. Dong et al. [Don09a] solves this for virtual point lights by using the surface normals as an additional cluster criterion. However, this is not possible in our approach, as the light sources can be freely placed in space. Therefore, an additional visibility test for the cluster centers would be required to ensure that they are not located within geometry.

Our approach can introduce temporal artifacts, such as flickering shadows, when the camera is moved or the light sources are animated. These artifacts can be reduced by disabling the view-adaptive scaling in the clustering step and by creating cluster centers for the animated light sources.

The results showed several insights about LightCluster concerning the scalability, quality and overall performance. By adjusting the clustering parameters, a well defined ratio between quality and performance can be chosen. Due to the approximation of the shadows from many lights with a soft shadow from a representative

light source, errors in the shadow can be reduced. As the clustering step is fast, it can be performed in each frame and only adds a small overhead. If light sources can be merged to clusters, LightCluster can be used to increase the rendering performance while maintaining an acceptable shadow quality in many cases.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we presented an approach, called LightCluster, to find representative light sources for approximating the shadows of many-lights. By focusing on high frequency shadows for direct lighting, we introduced a modified min-distance-cluster algorithm that combines different metrics to find a variable number of clusters in the view frustum. These clusters are then interpreted as an area light source which is scaled according to the cluster distance. In order to reduce the resulting errors in the shadows, we compute the shadows for each cluster with a soft shadow algorithm. Due to an adjustable initial cluster distance, the performance and quality can be trade off. The results showed, that LightCluster can be used to increase the rendering performance while maintaining an acceptable shadow quality in many cases.

In future work, we wish to extend our implementation with different light types, such as spot lights. This will result in a different clustering metric as new properties of the light sources must be incorporated, such as the direction vector. Furthermore, we wish to try other shadow techniques for omnidirectional light sources, such as Paraboloid Shadow Mapping [Bra02a] and Tetrahedron Shadow Mapping [Hun10a].

7 ACKNOWLEDGMENTS

The Dabrovic Sponza and Cornell Box scene is downloaded from [McG13a]. The Restaurant scene is based on the restaurant model from [IDS13a]. The work of A. Klein is funded by MBDA Germany.

8 REFERENCES

[Bra02a] Brabec S., Annen T., Seidel H.-P. Shadow Mapping for Hemispherical and Omnidirectional Light Sources. In Proceedings Computer Graphics International, Springer, 397-408, 2002.

[Dac05a] Dachsbacher C., Stamminger M. Reflective Shadow Maps. In Conf.proc. I3D 2005, 203-231, 2005.

[Don09a] Dong Z., Grosch T., Ritschel T., Kautz J., Seidel H.-P. Real-time indirect illumination with clustered visibility. In Conf.proc. VMV 2009, 187-196, 2009.

[Eis11a] Eisemann E., Schwarz M., Assarsson U., Wimmer M. Real-Time Shadows, Taylor & Francis, 2011.

[Fer05a] Fernando R. Percentage-Closer Soft Shadows. ACM SIGGRAPH 2005 Sketches, 2005.

[Ger04a] Gerasimov P. S. Omnidirectional Shadow Mapping, GPU Gems, Addison-Wesley, 193-203, 2004.

[Gre93a] Greene N., Kass M., Miller G. Hierarchical Z-buffer visibility. In Conf.proc. SIGGRAPH '93. ACM, 231-238, 1993.

[Hašo7a] Hašan M., Pellacini F., Bala K., Matrix row-column sampling for the many-light problem. In Conf.proc. SIGGRAPH '07, ACM, 2007.

[Hašo8a] Hašan M., Velázquez-Armendáriz E., Pellacini F., Bala K., Tensor Clustering for Rendering Many-Light Animations. In Conf.proc. EGSR '08, ACM, 1105-1114, 2008.

[Hun10a] Hung-Chien L., Shadow Mapping for Omnidirectional Light Using Tetrahedron Mapping. GPU Pro, A K Peters, 455-475, 2010.

[IDS13a] IDST Render, accessed March 27, 2013. `http://idst-render.com/scenes.html`.

[Kel97a] Keller A. Instant Radiosity. In Conf.proc. SIGGRAPH '97, ACM, 49-56, 1997.

[Lai07a] Laine S., Lehtinen J., Kontkanen J. Incremental Instant Radiosity for real-time indirect illumination. In Conf.proc. EGSR '07, ACM, 277-286, 2007.

[Lau10a] Lauritzen A. Deferred Rendering for Current and Future Rendering Pipelines. Beyond Programmable Shading, SIGGRAPH 2010, 2010.

[McQ67a] MacQueen J. B. Some methods for classification and analysis of multivariate observations. Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, 281-297, 1967.

[McG13a] McGuire M., Computer Graphics Archive, accessed March 27, 2013. `http://graphics.cs.williams.edu/data`.

[Ols11a] Olsson O., Assarsson U. Tiled Shading. Journal of Graphics, GPU, and Game Tools, 235-251, 2011.

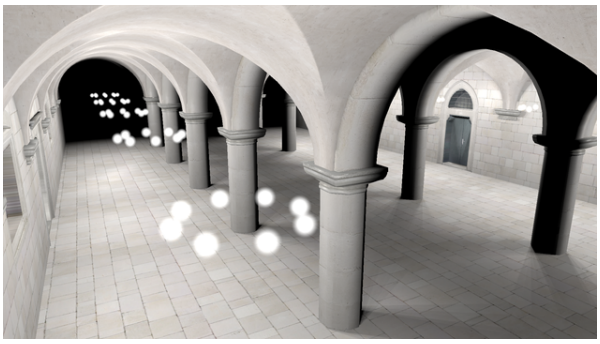
[Ree87a] Reeves W. T., Salesin D. H., Cook R. L. Rendering antialiased shadows with depth maps. In Conf.proc. SIGGRAPH '87, ACM, 283-291, 1987.

[Rit08a] Ritschel T., Grosch T., Kim M., Seidel H.-P., Dachsberger C., Kautz J. Imperfect shadow maps for efficient computation of indirect illumination, In Conf.proc. SIGGRAPH Asia '08, ACM, 2008.

[Wal05a] Walter B., Fernandez S., Arbee A., Bala K., Donikian M., Greenberg D. P. Lightcuts: a scalable approach to illumination, In Conf.proc. SIGGRAPH '05, ACM, 1098-1107, 2005.

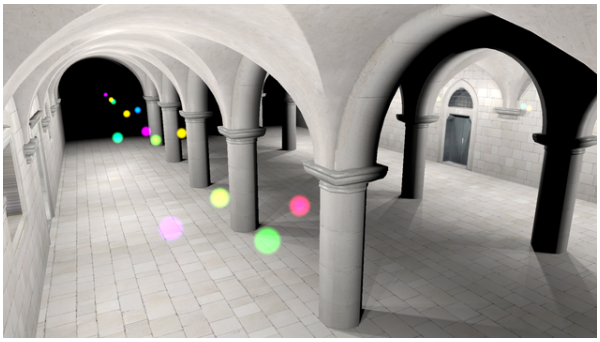
[Wil78a] Williams L. Casting curved shadows on curved surfaces. In Conf.proc. SIGGRAPH '78, ACM, 270-274, 1978.

[Wol57a] Wolfowitz J. The Minimum Distance Method, Annals of Mathematical Statistics, vol. 28, 75-88, 1957

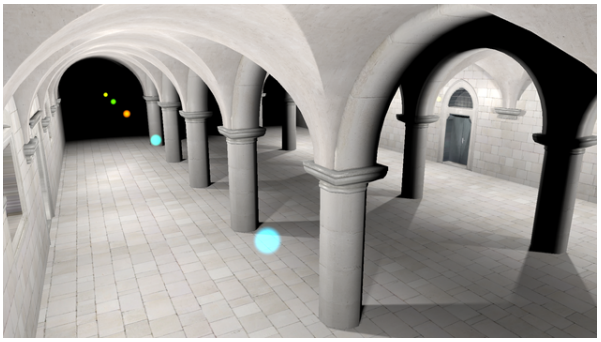


(a) The light distribution of 80 lights with $L_{att} = 7$ each.

(b) The resulting reference solution at 97.8 ms.



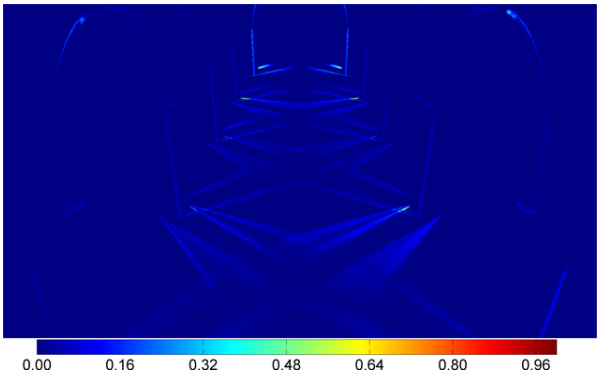
(c) The clustering for $d_{init} = 0.5$ with 26 clusters.



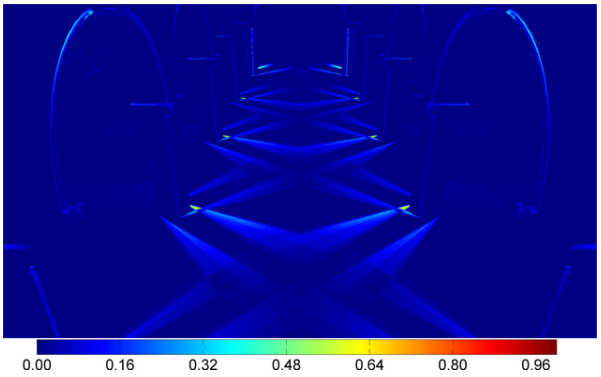
(d) The clustering for $d_{init} = 1.6$ with ten clusters.

(e) The resulting shadows for the cluster distribution at 43.1 ms.

(f) The resulting shadows for the cluster distribution at 20.7 ms.

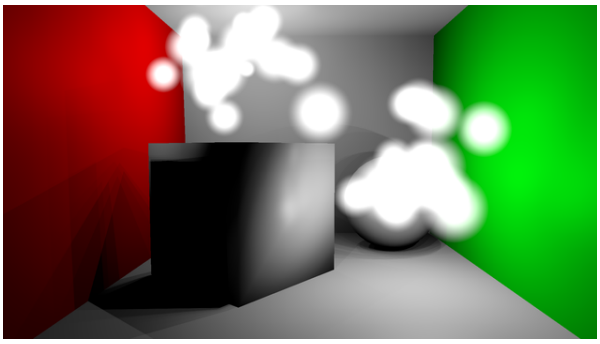


(g) The error for 26 clusters compared to the reference solution.

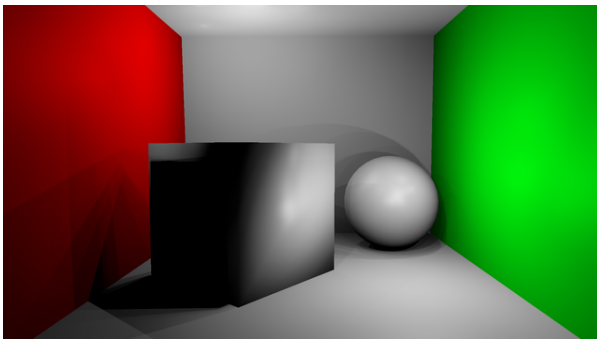


(h) The error for ten clusters compared to the reference solution.

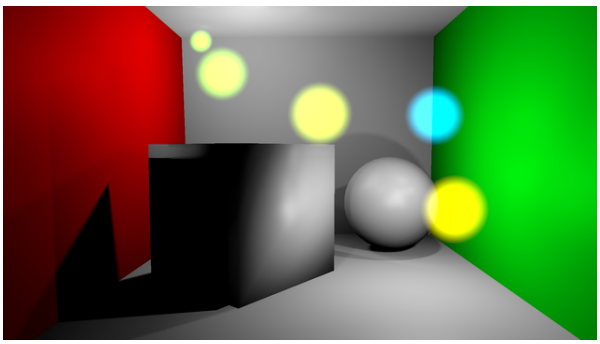
Figure 7: Comparison for the Dabrovic Sponza scene.



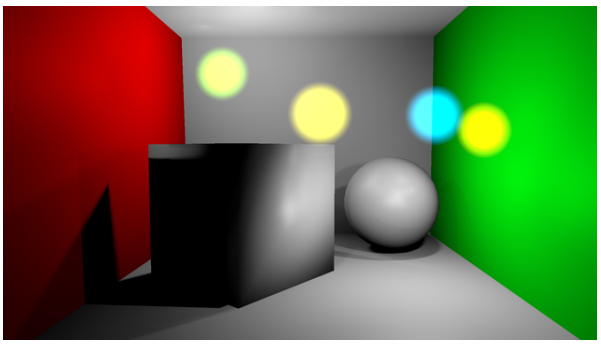
(a) The light distribution of 32 lights with random L_{att}.



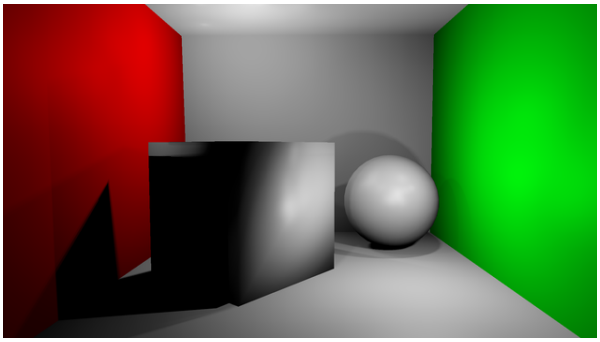
(b) The resulting reference solution at 20.0 ms.



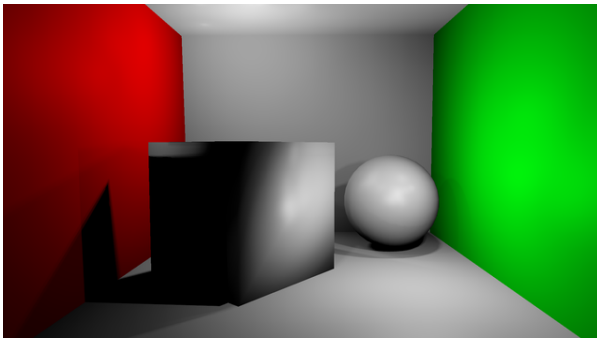
(c) The clustering for $d_{init} = 0.5$ with five clusters.



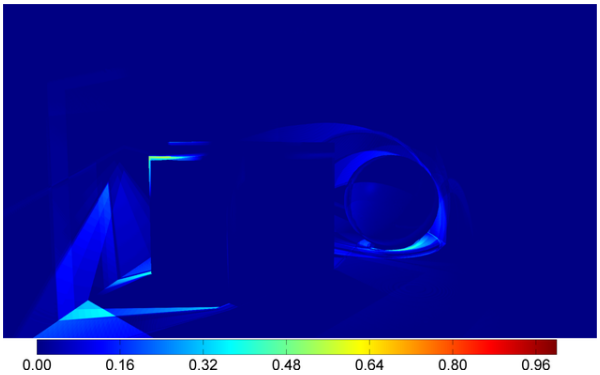
(d) The clustering for $d_{init} = 0.6$ with four clusters.



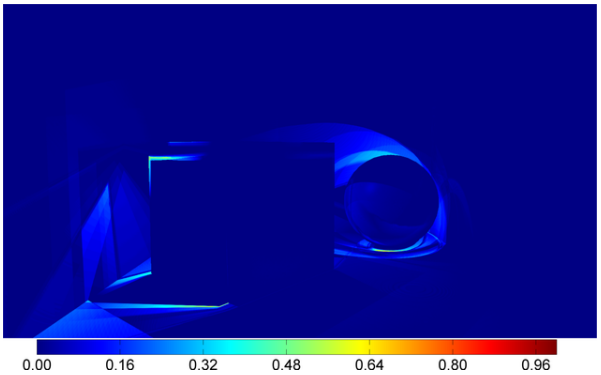
(e) The resulting shadows for the cluster distribution at 12.1 ms.



(f) The resulting shadows for the cluster distribution at 11.1 ms.



(g) The error for five clusters compared to the reference solution.



(h) The error for four clusters compared to the reference solution.

Figure 8: Comparison for the Cornell Box scene.

(a) The light distribution of fourteen lights with $L_{att} = 2$ each.

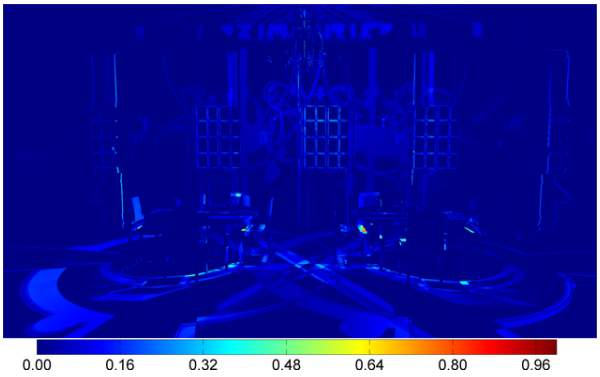
(b) The resulting reference solution at 39.3 ms.

(c) The clustering for $d_{init} = 0.3$ with nine clusters.

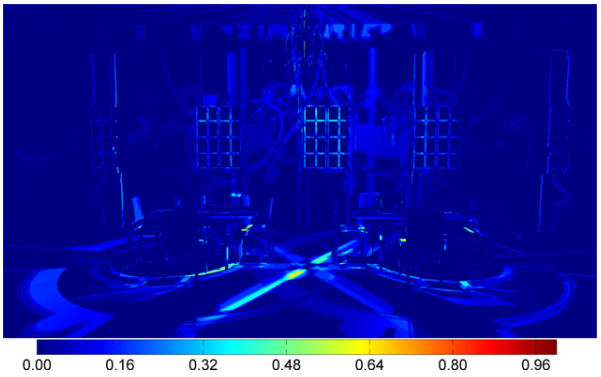
(d) The clustering for $d_{init} = 0.4$ with eight clusters.

(e) The resulting shadows for the cluster distribution at 32.9 ms.

(f) The resulting shadows for the cluster distribution at 30.5 ms.



(g) The error for nine clusters compared to the reference solution.



(h) The error for eight clusters compared to the reference solution.

Figure 9: Comparison for the Restaurant scene.