Vis Comput (2018) 34:1415–1426
https://doi.org/10.1007/s00371-017-1449-7

ORIGINAL ARTICLE

Intermediate shadow maps for interactive many-light rendering

Lili Wang[1] · Wenhao Zhang[1] · Nian Li[1] · Boning Zhang[1] · Voicu Popescu[2]

Published online: 8 November 2017

Abstract We present an efficient method for computing shadows for many light sources (e.g., 1024). Our work is based on the observation that conventional shadow mapping becomes redundant as the number of lights increases. First, we sample the scene with a constant number of depth images (e.g., 10), which we call intermediate shadow maps. Then the shadow map for each light is approximated by rendering triangles reconstructed from the intermediate shadow maps. The cost of rendering these triangles is much smaller than rendering the original geometry of a complex scene. The algorithm supports fully dynamic scenes. Our results show that our method can produce soft shadows comparable to those obtained by conventional shadow mapping for each light source or by ray tracing, but at a higher frame rate.

Keywords Many lights · Visibility · Shadow mapping

1 Introduction

Improving the quality of computer rendered imagery demands not only increasing the geometric detail of the scene, but also increasing the detail with which scene lighting is modeled. A frequently used light modeling primitive is the point light source, which is simple and versatile, supporting the modeling of complex area light sources and indirect illumination. However, complex lighting requires a large number of point light sources. Rendering complex scenes with many point light sources at interactive rates remains an open research problem. The core challenge is to determine visibility between the scene geometry and the light sources. When the scene consists of millions triangles and when lighting is modeled with hundreds or even thousands of point light sources, determining visibility can be very time-consuming, precluding rendering at interactive rates. The conventional approach for rendering shadows in interactive graphics applications is shadow mapping, which does not scale with scene complexity and with the number of lights, as it requires rendering the scene once for each light.

In this paper, we present an efficient method for computing shadows for many point light sources (e.g., 1024). Our method is based on the observation that conventional shadow mapping becomes redundant as the number of lights increases. Given a shadow map SM_i rendered for a light source L_i, SM_i contains a significant part of the visibility information needed to compute shadows for a different light L_j. Given a set of k intermediate shadow maps, the set contains almost all of the visibility information needed to compute shadows for any number of additional lights. Rendering shadow maps for the additional lights is redundant. Instead, our method approximates the shadow map of a light from the set of intermediate shadow maps.

Our method has two steps. First, we sample the scene with a constant number of depth images (e.g., 10), which we call intermediate shadow maps (INSMs). Each INSM is then triangulated by connecting four adjacent samples with two triangles. Finally, the shadow map for each light L is approximated by rendering the triangles of all INSMs from the viewpoint L. Since the number of INSMs is con-

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s00371-017-1449-7) contains supplementary material, which is available to authorized users.

✉ Lili Wang
wanglily@buaa.edu.cn

[1] State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing, China

[2] Purdue University, West Lafayette, IN, USA

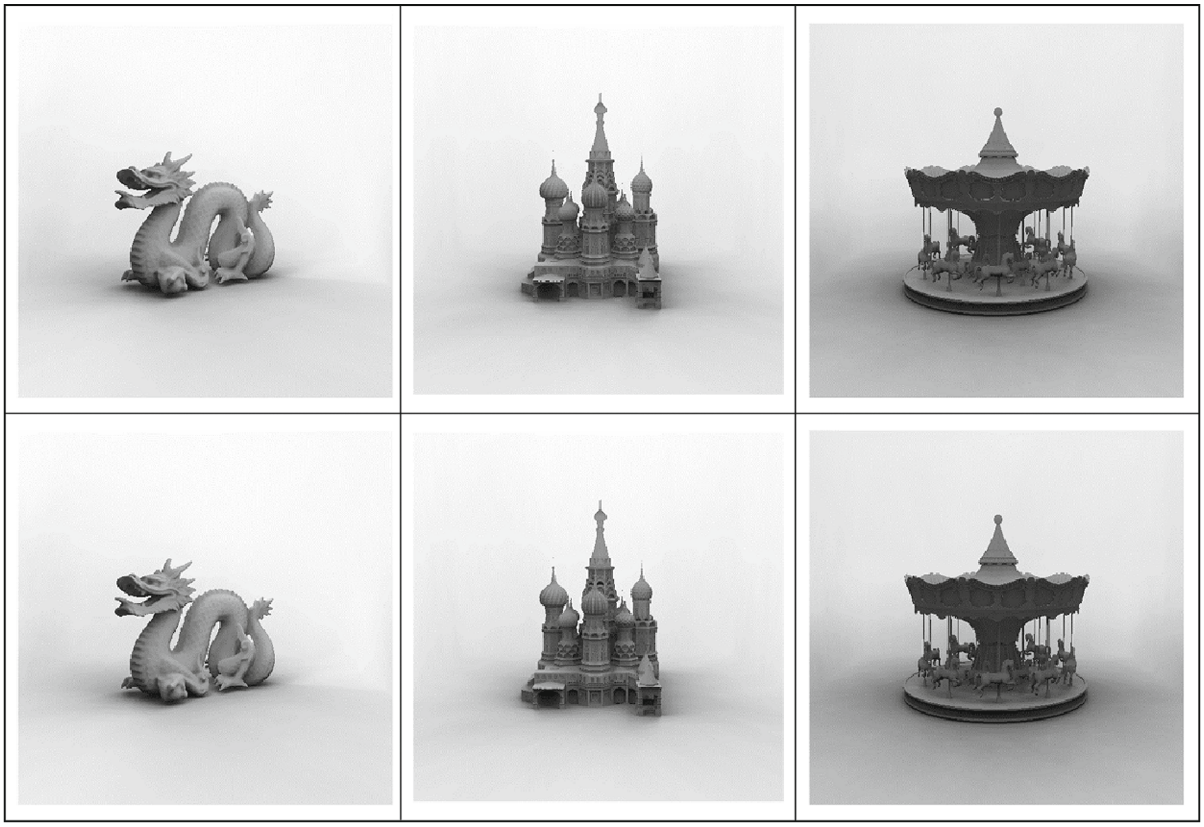


Fig. 1 Scenes with 1024 lights rendered with our method (row 1) and with ray tracing (row 2). Compared to ray tracing, the average per pixel intensity errors for our method are 0.1, 0.7 and 1.1%, and the speedups are 12×, 25× and 17×. Compared to conventional shadow mapping, our method brings speedups of 5×, 18× and 10×

stant, the cost of rendering the triangles reconstructed from them is also constant, and, for complex scenes, this cost is significantly smaller than rendering the entire scene. We tested our approach on several complex scenes where we obtained high-quality shadows and good performance (Fig. 1). We also refer the reader to the accompanying video.

2 Previous work

The classical methods for shadow computation are shadow mapping, shadow volumes and ray tracing. However, these methods are slow for a large number of lights. In order to accelerate visibility computation in the context of shadow rendering, researchers pursued two kinds of approximations: scene geometry approximation, which implies replacing the original scene geometry with a lower cost representation, and lighting approximation, which implies reducing the number of lights by clustering. In addition to our overview of prior work below, we also refer the reader to a recent survey of many-light techniques [5].

2.1 Ray tracing-based methods

Approximating the scene geometry has been pursued in the context of ray tracing. One method is micro-rendering [19], which uses a point tree hierarchy to approximate the geometry of objects in the scenes. The exact visibility of the leaf nodes is determined by ray casting after the cut of tree is computed. Another method is based on incremental voxel-space visibility computations [13], which uses a screen-space voxelization to discretize the scene geometry, and introduces an efficient incremental query to determine the visibility from output samples to light sources.

Several ray tracing-based methods focus on simplifying the lights, which brings more time performance advantage for both illumination and visibility, but at the cost of a quality decrease. One method groups lights using an octree [17] and resorts to volumetric visibility approximation. Another method partitions the geometry of the scene into zones and clusters the lights into sets of similar lights per zone, which results in an unstructured light cloud [12]. Lightcut [23] is a popular scalable solution to the many-light problem. A binary light tree is built by clustering the original lights, and cuts through the tree are selected for output samples. The method uses the trivial upper bound of one for the visibility term (i.e., all lights are potentially visible). Many researchers improved visibility computation accuracy based on the original lightcut method [3,4,24].

There are some ray tracing-based method that approximate both lights and scene geometry. Such as VisibilityClusters in [26]. The method constructs VisibilityClusters with high visibility coherence and estimates average visibility by exploiting the sparse structure of the matrix and shooting only few shadow rays between clusters.

2.2 Shadow mapping-based methods

Compared to ray tracing, shadow mapping-based methods are faster, but still not fast enough for interactive performance in the context of a large number of lights and of a complex scene. Shadow mapping acceleration was pursued by scene geometry approximation.

One straight forward method available to practitioners is to simplify the scene geometry with off-the shelf LOD tools such as Simplygon [2] or 3Ds Max [1]. Compared to geometry simplification, our method is robust and it works for any scene, whereas geometry simplification is complex and it requires tuning scene-specific parameter values. Our method does not preprocess geometry, so it is suitable to geometry that becomes available in real time, such as geometry acquired with real-time depth cameras. Moreover, simplified geometry will cast acceptable shadows for area light sources that generate soft shadows, but any hard shadow will reveal the coarseness of the underlying geometric model.

If geometry simplification is to take into account the current positions of the lights in order to avoid oversimplifying blockers that cast hard shadows, it can only do so by running for every frame, as the lights are dynamic and the hardness of a shadow cast by one light changes from frame to frame. Furthermore, geometry simplification typically takes into account a single viewpoint, i.e., the eye of the camera that renders the output image. It is difficult to meet the constraints of thousands of viewpoints, each creating a silhouette. For example, the ManyLoDs method [10] uses a bounding volume hierarchy (BVH) to approximate the scene geometry and to compute LoDs efficiently in parallel by balancing the workload within and among the LoDs. For dynamic scenes, the BVH has to be updated for each frame based on scene graph cuts defined by the thousands of lights. In order to complete these steps in real time, the method has to find some high-level cuts, which correspond to a coarse approximation of scene geometry, which might acceptable for faint shadows but not for shadows with higher definition.

Ritschel et al. [21] propose coherent shadow maps (CSM). The precomputed and compressed depth maps allow visibility tests between moving objects and a high number of lights outside their convex hulls using simple shadow mapping, but the method is unsuitable for virtual point light sources placed on an objects surface, as needed for indirect lighting in global illumination. In order to solve this problem, coherent surface shadow map (CSSM) was proposed, which is a more accurate technique that approximates visibility at scene points using local cube maps [22]. CSM and CSSM precompute thousands of shadow maps for each scene object, which is prohibitively slow in the context of fully dynamic scenes, where geometry becomes available in real time, or where objects deform, which makes their shadow maps obsolete.

The virtual area lights (VALs) method [7] computes directly the soft shadows cast by the smaller number of VALs using convolution soft shadow mapping with parabolic projection, which has a smaller overall cost than computing hard shadows for all the point light sources. The VAL method is fast, and it can handle fully dynamic scenes, but the shadows are approximate. VAL relies on a last "shadow blurring" step where high-frequency artifacts are filtered out. Our method renders the intermediate shadow maps in real time without fudging the shadows in a postprocess.

Imperfect shadow map (ISM) [20] is a more general method for many-light visibility determination in fully dynamic scenes. A low-resolution shadow map is rendered for each light from a coarse point-based approximation of scene geometry by splatting followed by pull–push reconstruction. ISM is a popular method for interactive rendering with many lights, so we compare our method to ISM in detail in the Results section. The virtual shadow maps technique [15,16] subdivides view volume into clusters which contain visible geometry and then creates a list of lights influencing each cluster of scene geometry. It also proposes an adaptive method for selecting the proper resolution of shadow maps, which are used to determine visibility. For the scenes with hundreds of light sources, this method achieves high-quality shadows in real time and within a bounded memory footprint.

In matrix row–column sampling [9], the columns of a matrix represent all output pixel samples lit by an individual light, and the rows represent an individual sample lit by all lights. A set of representative rows and columns of the matrix are computed first using conventional shadow mapping. A row is computed by rendering the scene from the viewpoint of the sample of the row, and a column is computed by rendering the scene from the light of the column. Then the other matrix elements are approximated by interpolation. The visibility clustering method [6] clusters lights, renders a representative shadow map to approximate the visibility in each cluster and combines the approximate visibility with accurate per light shading. Visibility clustering requires rendering fewer shadow maps than standard matrix row–column sampling. Matrix row–column sampling was extended to rendering massive scenes with out-of-core geometry and complex lighting [25]. Another extension uses a new matrix sampling-and-recovery scheme to gather illuminations efficiently by only sampling visibility for a small number of representative lights and surface points [11]. The clustering step on which all of these methods rely is too time-consuming for interactive rendering, and these methods are reserved for providing previews in animation systems, at the cost of several to hundreds of seconds per frame.

Our method falls into the category of shadow mapping-based methods. Like in visibility clustering, our method computes a set of intermediate shadow maps, but then the intermediate shadow maps are reprojected to the viewpoint of each light source, which results in a higher-quality approximation of visibility than simply using the representative shadow map for all the lights in the cluster.

3 Intermediate shadow maps

Our method avoids the redundancy of rendering hundreds of shadow maps. A small number of intermediate shadow maps are used to approximate the shadow map of each of the many lights. Using the visibility information contained in an intermediate shadow map $INSM_j$ for a light L_i can be done in many ways.

One approach is to leverage epipolar geometry. Given an output image sample S_{uv}, defined by a color value and a depth value which allows unprojecting the sample to a 3D point in scene space, the intersection between the light ray L_i S_{uv} and $INSM_j$ can be computed by projecting L_i S_{uv} onto $INSM_j$ and tracing the projection in search of an intersection. This approach was introduced in inverse 3D image warping [27] and then later used in relief texture mapping [14] and in specular reflection rendering [8]. The advantage is reducing the cost of intersecting a depth image with a ray from $2D$ to $1D$.

However, unlike in the case of inverse 3D image warping, relief texture mapping and specular reflections where there is a single ray per output image pixel, in our context there are n rays per pixel, where n is the number of lights, which could be in the hundreds or even the thousands. Fortunately, the large set of rays that arises in the context of many-light rendering is coherent, as the light rays can be grouped in concurrent bundles, with one bundle per light. This enables a second, more efficient approach for using the visibility information of the intermediate shadow map $INSM_j$ for approximating the shadow map of L_i. The second approach, which we adopt, is to transform $INSM_j$ into a triangle mesh and to render the triangle mesh from L_i. This approach leverages the GPU strength of rendering triangles by projection followed by rasterization.

3.1 Algorithm overview

Algorithm 1 outlines the main steps of our approach.

Algorithm 1 Many-Light Rendering with Intermediate Shadow Maps

```
Input: scene $S$ with $N$ triangles and $n$ light sources $L_i$, $k$ reference viewpoints $V_j$ and output view $V$.
Output: Image $I$ that shows $S$ rendered from $V$ with shadows cast by $L_i$.
1: for $j = 1$ to $k$ do
2:     $INSM_j$ = Render($S$, $V_j$)
3:     $TM_j$ = Triangulate($INSM_j$)
4: $I$ = Render($S$, $V$)
5: for $i = 1$ to $n$ do
6:     $SM_i = \emptyset$
7:     for $j = 1$ to $k$ do
8:         $SM_i$=Render($TM_j$, $L_i$)
9:     Shadow mapping $I$ from $L_i$ using $SM_i$
10: return $I$
```

In steps 1–2 the intermediate shadow maps are rendered conventionally from the reference viewpoints V_j that are designed to sample the scene S uniformly and comprehensively. We place the intermediate shadow map reference viewpoints at the midpoints of the eight edges and the centers of the left and right planes in the axis aligned bounding box of S (Fig. 2).

Then each intermediate shadow map $INSM_j$ is converted to a triangle mesh TM_j by defining two triangles for each neighborhood of 2×2 $INSM_j$ samples. No triangles are generated across depth discontinuities, as such triangles will incorrectly connect the foreground surface to the background surface, casting incorrect shadows. Depth discontinuities are computed by thresholding the second-order difference in the depth map. The second-order difference is surface orientation independent, i.e., it is exactly 0 for any plane, no matter its orientation. Step 3 is described in detail in Sect. 3.2.

Step 4 renders the scene without shadows to image I, which defines the samples for which shadows have to be computed, in deferred shading fashion. Steps 5–9 compute the number of visible lights for each pixel in I. For each light L_i, an approximate cube shadow map SM_i is constructed first by rendering all intermediate shadow map triangle meshes TM_j from L_i (steps 6–8). Then SM_i is used to compute visibility from L_i, for every pixel in I.

3.2 Intermediate shadow map triangulation

We triangulate the intermediate shadow maps on the GPU, by processing neighborhoods of 2×2 intermediate shadow maps samples in parallel, as described in Algorithm 2.

Sample (u, v) is the top left sample of the 2×2 neighborhood. The four samples are connected with two triangles. A triangle is kept if its vertices pass the connectivity test. The triangle connectivity test IsConnected (a, b, c) checks connectivity for each of the three triangle edges (a, b), (b, c), and (c, a) by thresholding a second-order depth difference along

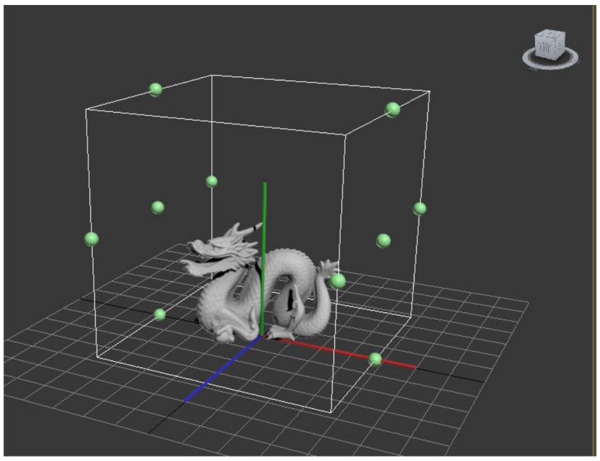


Fig. 2 Reference viewpoint placement for the intermediate shadow maps. There are 10 reference viewpoints (dots) from where 14 intermediate shadow maps (arrows) are rendered

Algorithm 2 Intermediate Shadow Map Triangulation

```
Input: $INSM_j$
Output: triangle mesh $TM_j$ obtained from $INSM_j$
1: for each $INSM_j$ sample $(u, v)$ do
2:     if IsConnected($(u, v), (u, v+1), (u+1, v+1)$) then
3:         $TM_j$ += $[(u, v), (u, v+1), (u+1, v+1)]$
4:     if IsConnected($(u+1, v+1), (u+1, v), (u, v)$) then
5:         $TM_j$ += $[(u+1, v+1), (u+1, v), (u, v)]$
6: return $TM_j$
```

the edge [18]. For example, edge $((u, v), (u, v+1))$ passes the connectivity test if both of the two conditions below are met, where $z(u, v)$ is the depth value of sample (u, v), and ε is a threshold that depends on the scene.

$$|z(u, v-1) + z(u, v+1) - 2z(u, v)| < \varepsilon$$
$$|z(u, v) + z(u, v+2) - 2z(u, v+1)| < \varepsilon$$

This second-order depth difference is exactly zero for a planar surface, regardless of the orientation of the plane. In other words, our connectivity computation will never disconnect a planar surface in the scene. Figure 3 shows triangles that are kept and triangles that are discarded, including invalid triangles that involve null (i.e., background) samples.

4 Results and discussion

In this section, we discuss the quality of the shadows rendered by our method, we report frame rate measurements, and we discuss limitations.

We tested our algorithm with several scenes: *Dragon*, *Church*, *Carousel*, *City*, *Planes*, *Grass* and *Sponza*. Table 1 gives the number of triangles for each scene, as well as the number of triangles obtained after triangulating the intermediate shadow maps. All performance measurements reported

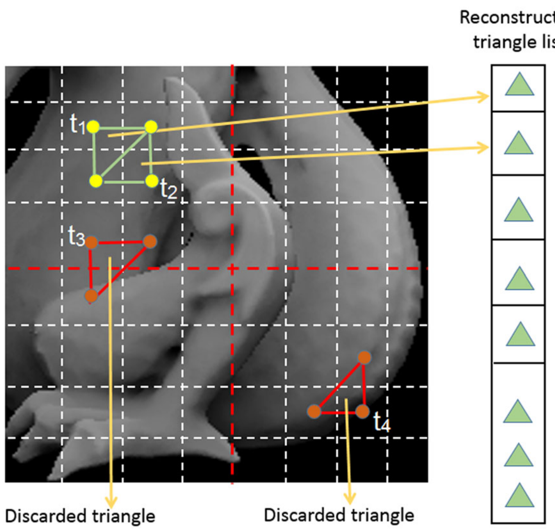


Fig. 3 Triangle mesh reconstruction from intermediate shadow maps. Four neighboring samples in the intermediate map are connected with two triangles. Triangle t_1 and t_2 are added to the mesh. Triangle t_3 and t_4 are discarded since they span a depth discontinuity or involve an invalid background sample

in this paper were recorded on a 3.8 GHz Intel(R) Core(TM) i7-2600K CPU PC with 12 GB of RAM and an NVIDIA GeForce GTX 660 graphics card. We use NVIDIA's Optix ray tracer.

4.1 Quality

We compare the shadows rendered with our method to shadows rendered using imperfect shadow maps (ISM) and to ground truth shadows rendered with ray tracing. The error metric used is the average shadow intensity error per valid image pixel.

$$Avg\,.\,pixel\ error = \left(\sum_{1}^{n} (|C(u, v) - C_r(u, v)|)\right) /n/255$$

n is the number of non-background pixels, and $C(u, v)$ and $C_r(u, v)$ are the shadow intensity values at a pixel with our method and with ray tracing. We normalize the error by dividing by the maximum shadow intensity value 255 and we provide the error as a percentage. Although it is a relaxed error measure for visibility computation, as positive and negative errors from different lights may cancel out at a given pixel, the average intensity error is directly related to the quality of the output image.

Our method renders high-quality shadows, comparable to shadows rendered by ray tracing, for complex scenes with 1024 lights (see Figs. 1, 4, as well as the accompanying video). In all our experiments, the default resolution of the intermediate shadow maps is 128 × 128, the default resolu-

Table 1 Number of triangles in the original scenes and in the INSMs

Scene	*Dragon*	*Church*	*Carousel*	*City*	*Planes*	*Grass*	*Sponza*
Tris [k]	871	1868	1336	1117	1000	1198	1063
INSM Tris [k]	150	121	159	167	65	71	263

tion of the shadow maps computed for each light is 512×512, the default number of intermediate shadow maps is 10, and the default number of light sources is 1024. The output image resolution is 512×512. The third and the fifth columns of Fig. 4 visualize the error for the six images shown in Figs. 1 and 4.

Our method typically underestimates blockers, by only considering what was captured in the intermediate shadow maps and by eroding surface edges a half-pixel during reconstruction, which results in light leaking. Consequently, the small approximation errors in our images are typically due to pixels that are brighter than they should be. The occasional "too dark" approximation errors are due to incorrect depth discontinuity detection which generates superfluous blocker surfaces. Consider the case of a foreground surface that is incorrectly connected to a background surface due to failing to detect the depth discontinuity between them. The connective surface does not exist in reality, and it casts a shadow that makes the output pixel too dark. The *Planes* and *Grass* scenes are the most challenging scenes for our method due to the high depth complexity, which is challenging for the small number of intermediate shadow maps we use, and the minute detail, which is challenging for the reconstruction of the blocker surfaces from the sample base representation brought by the intermediate shadow maps. Even for these challenging scenes, the approximation errors introduced by our method are small.

Our intermediate shadow map triangulation algorithm always connects the top right sample to the bottom left sample. Better results could be obtained by choosing the best way of triangulating a group of neighboring four samples, based on actual scene geometry. For example, when the geometry is concave, choosing the wrong diagonal could provide a convex approximation of geometry, and vice versa. We ran an experiment where the triangulation used selectively the top right/bottom left or the top left/bottom right diagonals based on the best approximation of depth at the center of the square defined by the four samples. The error improved from 0.12 to 0.11% for the *Dragon* scene, but this small improvement came at a substantial cost in triangulation time, which went up from 65 to 155 ms per frame. For most scenes and applications, such an adaptive triangulation is probably not warranted.

ISM uses points sampling to approximate the geometry of the scene, which is rendered by splatting and pull–push operations to generate a low-resolution shadow map for every light source. In our experiments, we use about 12,000 point samples to render each of the 1024 128×128 ISMs, which yields a frame rate comparable to that of our method. Table 2 shows that our method has a smaller average shadow intensity error for our scenes.

Tables 3, 4 and 5 show the dependency of the approximation error on the resolution of the intermediate shadow maps, on the resolution of the approximate shadow maps computed for each light and on the number of intermediate shadow maps, for the *Dragon* scene. As expected, the error decreases as the three parameters increase. For this scene, the values of the three parameters after which returns diminish are 128×128, 512×512 and 10.

The viewpoints of our intermediate shadow maps are distributed evenly in the scene to achieve a good sampling of geometry. The same reference viewpoints are used for all frames, which brings simplicity and good temporal coherence. In the scenes described so far, the desired viewpoint is outside, looking in at the scene. In such an outside-looking-in case, we place the intermediate shadow map viewpoints outside the scene, and the view directions are aimed at the scene. We have also tested our method with an inside-looking-out scene *Sponza*. In this case, we set the viewpoints inside the scene, distributed uniformly, and with view directions aimed at the center of the scene. Figure 5 gives a top view illustration of the reference camera placement. Figure 6 shows our results compared to ground truth, as well as shadow error images. The errors are 5.5 and 4.0%, and our method is 3 times faster than ray tracing.

Our method is intended for soft shadows resulting from hundreds of light sources whose shadows interfere. However, our method also supports hard shadows, as shown in the accompanying video and in Fig. 7. Quality hard shadows are obtained because scene geometry is approximated in detail with hundreds of thousand of triangles, which are used to compute high-resolution shadow maps for each light. This is a particular strength of our method compared to other methods such as ISM, which do not demonstrate hard shadows.

4.2 Performance

We have compared the performance of our algorithm to that of NVIDIA's Optix ray tracer and to conventional shadow mapping that renders a shadow map for each light from the original scene geometry. Table 6 provides the frame rendering times for all three methods, broken down into INSM rendering and triangulation (T), and shadow computation

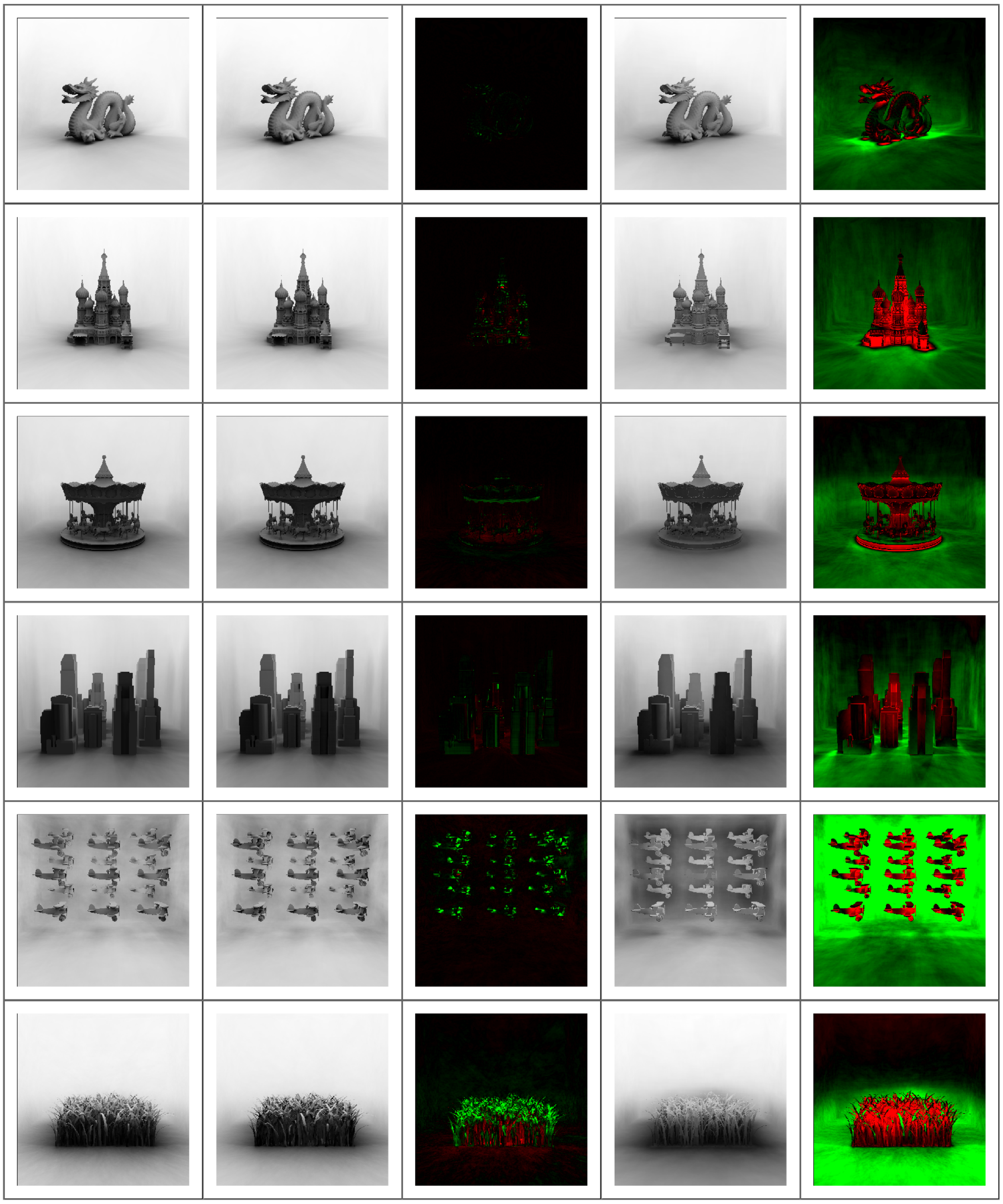


Fig. 4 Comparison between ray tracing (first column), our method (second column) and imperfect shadow maps (fourth column). The approximation introduced by ISM translates into noticeable shadow errors. The third and fifth columns visualize the approximation errors of our method and of ISM, respectively. The error is scaled by a factor of 5 for illustration purposes. Red/green highlights pixels that are too bright/dark

using the INSMs (S). The time needed to render and triangulate the INSMs is negligible compared to the time needed to compute the shadows using the INSMs. Our method is between 11.8× and 24.9× faster than ray tracing and between 5.0× and 17.5× faster than conventional shadow mapping. The speedup comes from replacing the original scene geometry with the triangle meshes reconstructed from the intermediate shadow maps, when computing the individ-

Table 2 Average pixel shadow errors and root mean squared error (RMSE) for our method and for the prior imperfect shadow Maps method

Scene	*Dragon*	*Church*	*Carousel*	*City*	*Plane*	*Grass*
Ours (%)	0.1	0.7	1.1	0.8	3.4	3.9
ISM (%)	3.9	5.8	4.7	4.6	10	9.0
Ours RMSE	0.94	1.7	2.4	2.4	3.3	4.5
ISM RMSE	8.6	10	8.6	8.6	11	8.6

Table 3 Approximation error as a function of intermediate shadow map resolution

Resolution of INSM	64×64	128×128	256×256	512×512
Avg. pixel error (%)	0.56	0.12	0.09	0.08

Table 4 Approximation error as a function of individual light shadow map resolution

Resolution of INSM	256×256	512×512	1024×1024	2048×2048
Avg. pixel error (%)	0.18	0.12	0.10	0.09

Table 5 Approximation error as a function of the number of intermediate shadow maps

Number of INSMs	6	8	10	12
Avg. pixel error (%)	0.30	0.16	0.12	0.11

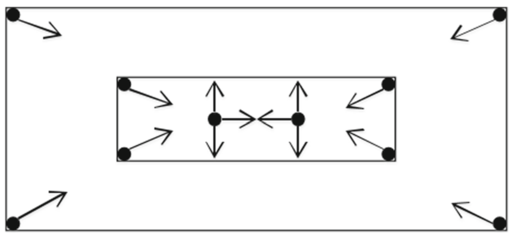


Fig. 5 Reference viewpoint placement for the intermediate shadow maps for the *Sponza* scene. The cameras are shown by the dots, and their view directions are shown the arrows

ual light shadow maps. The numbers of triangles in these meshes are given in Table 1 for the *Dragon*, *Planes*, *City*, *Grass*, *Carousel* and *Church* scenes, which are considerably less than the numbers of triangles in the scene models. For a scene with N triangles, for k intermediate shadow maps of resolution $w \times w$ and for n lights, the number of triangles rendered by our method is at most $kN + 2nkw^2$, where we counted two reconstructed triangles per intermediate shadow map sample. Conventional shadow mapping renders nN triangles, so our method scales much better with scene geometric and lighting complexity.

In our experiments, we tried to perform an equal quality comparison to ISM by increasing the number of scene point samples used in ISM. No matter how much we increased the number of point samples, ISM quality remained inferior to the quality of our method. Once the number of point samples increases above what can be handled by the GPU in a single pass, the additional rendering pass made performance slower than that of ray tracing.

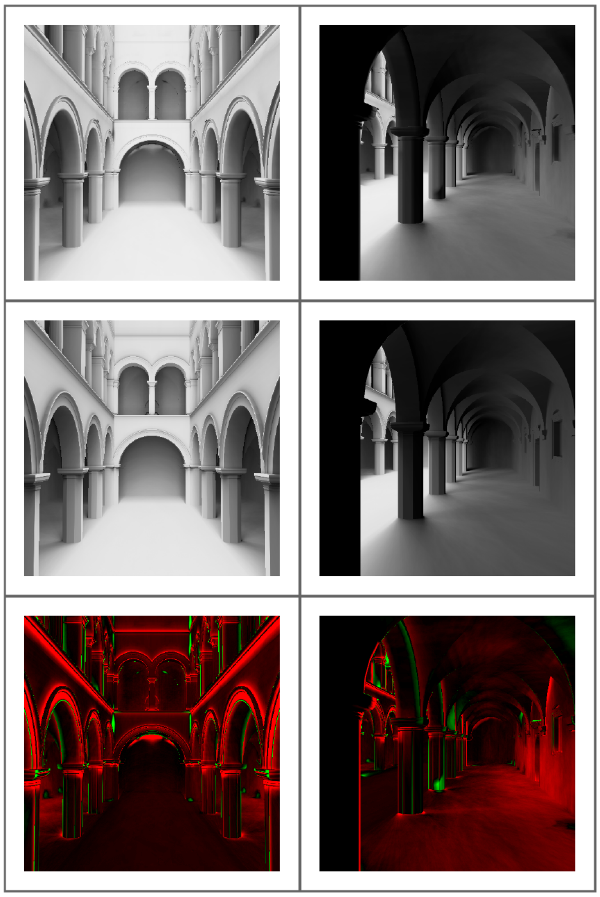


Fig. 6 *Sponza* inside-looking-out scene with 1024 lights rendered with our method (top) and with ray tracing (middle). The approximation errors of our method are 5.5% (left) and 4.0% (right). We visualize the approximation errors scaled up by a factor of 5, with red/green highlighting pixels where the images are too bright/too dark

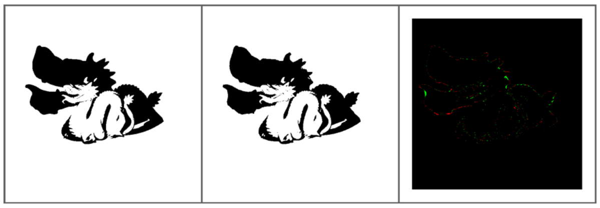


Fig. 7 *Dragon* scene with 1 light rendered with ray tracing (left) and our method (middle). The approximation errors of our method are 0.4% (right). Pixels where the images are too bright/too dark are shown with red/green

Table 6 Frame rendering times in milliseconds for our method (INSM), for conventional shadow mapping (SM), and for ray tracing (RT)

Scene	INSM		SM	*SM/INSM*	RT	*RT/INSM*
	T	S				
Dragon	65	780	4,200	5.0×	10,000	11.8×
Church	59	685	13,000	17.5×	18,500	24.9×
Carousel	73	762	8300	9.9×	14300	17.1×
City	86	874	7300	7.6×	12,500	15.2×
Planes	58	435	4,100	8.3×	10,500	21.3×
Grass	76	604	9,400	13.8×	16,000	23.5×

For INSM, the frame time is broken down into the time needed for INSM rendering and triangulation (T) and the time needed to compute shadows from using the INSMs (S)

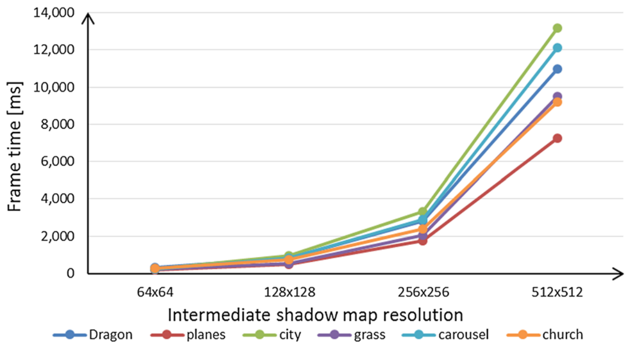


Fig. 8 Frame rendering time as a function of the linear resolution of the intermediate shadow maps

The graphs in Figs. 8, 9 and 10 confirm the quadratic dependence of performance on the linear resolution w of the intermediate shadow maps and the linear dependence on the number of intermediate shadow maps and on the number of lights.

The graph in Fig. 11 confirms that the resolution of the shadow maps computed for individual lights does not affect performance much, which indicates that rendering the individual light shadow maps is geometry and not fill-rate bound.

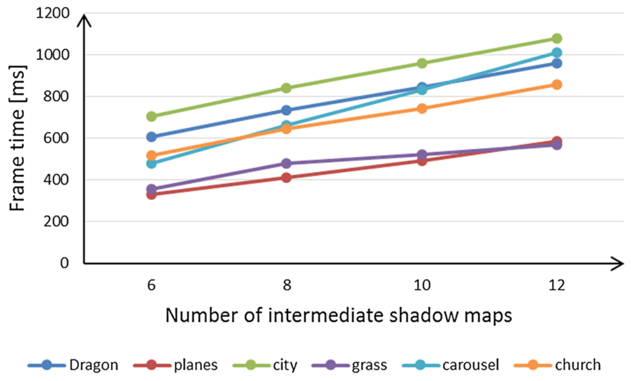


Fig. 9 Frame rendering time as a function of the number of intermediate shadow maps

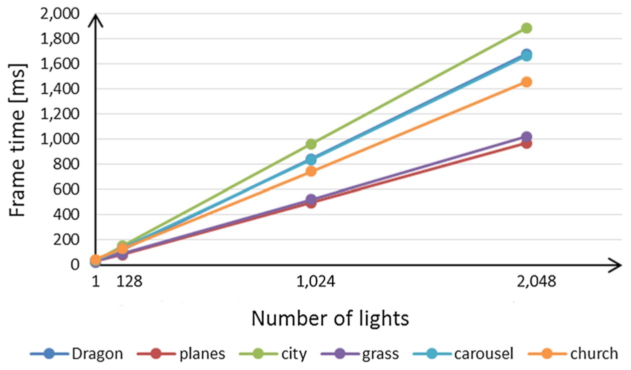


Fig. 10 Frame rendering time as a function of the number of lights in the scene

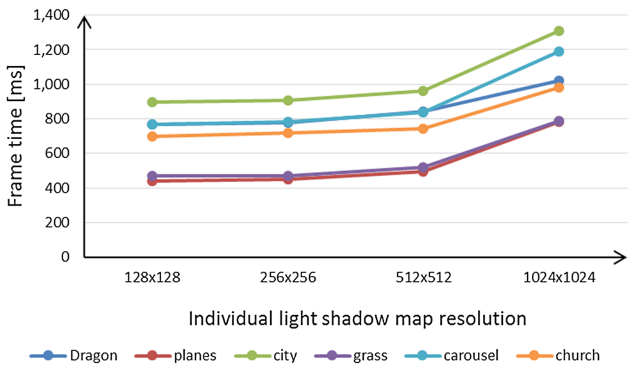


Fig. 11 Frame rendering time as a function of the resolution of the individual light shadow maps

4.3 Limitations

As discussed, our method approximates blocker geometry with intermediate shadow maps, which can result in light leaks when the blocker geometry is not sampled well enough. Insufficient sampling can be caused by high depth complexity, i.e., many occluding layers, or by high surface complexity, i.e., minute details. Our method provides a straightforward approach for mitigating these challenges:

increasing the number of intermediate shadow maps and increasing the resolution of intermediate shadow maps. Adequate values for these two essential parameters should be determined based on the scene and based on the application.

In this work, we have chosen to place the intermediate shadow maps uniformly, according to the inside-looking-out and the outside-looking-in cases, as described in Sect. 4.1. This simple placement of the intermediate shadow maps produces good results for a variety scenes, with a small number of intermediate shadow maps. Future work could consider optimizing the placement of intermediate shadow maps, in order to maximize coverage for a given number of intermediate shadow maps, borrowing from the large body of literature on the "next best view" problem. Our uniform placement has the advantage of supporting dynamic scenes without the cost of recomputing the intermediate shadow map placement for every frame.

Another limitation of our approach is that, in order to surpass the performance of conventional shadow mapping, the scene has to be sufficiently complex such that the triangle meshes reconstructed from the intermediate shadow maps be less expensive than the original scene model, and the number of lights should be sufficiently large such that these per light gains accumulate to overtake the initial start-up cost of rendering and triangulating k intermediate shadow maps.

5 Conclusions and future work

We have presented a general and efficient method for rendering shadows for many light sources. The method handles robustly fully dynamic scenes with millions of triangles and a thousand light sources and renders high-quality soft shadows. Our method decreases the redundancy of conventional shadow mapping a large number of lights by only rendering the scene geometry a small number of times to generate intermediate shadow maps, which are then used to approximate visibility from the individual light sources. The intermediate shadow maps are rendered for every frame, from the same set of uniform reference sampling locations. The intermediate shadow maps contain much of the visibility information needed for the many light sources. We extract this information carefully by reprojecting the intermediate shadow maps to the viewpoints of the individual lights. We do *not* approximate visibility by interpolation, since visibility is notoriously nonlinear. We do *not* cluster lights, and we truly estimate visibility for each one of the many individual lights. Our lights are free to change from a uniform distribution to a clustered distribution or even to converge to a single point, and our method produces quality shadows, gradually changing from soft to harder and then to hard shadows, without temporal artifacts.

We compared our results to ground truth obtained by ray tracing and to conventional shadow mapping over a variety of scenes, and we showed that our method brings a substantial performance gain at the cost of only small approximation errors.

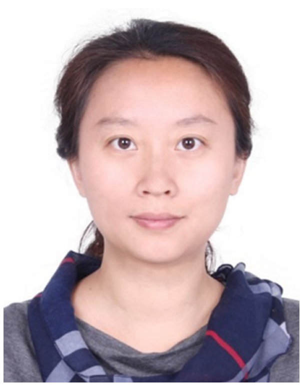
Our method makes progress in the direction of substantially increasing the number of lights that are available to interactive graphics applications. An important direction of future work is to provide algorithmic support for lighting design by automatically placing and calibrating the individual light sources. Another direction of future work is to investigate the extension of our method to global illumination where surface samples become virtual point light sources from where secondary light rays originate.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China through Projects 61272349, 61190121 and 61190125 and by the National High Technology Research and Development Program of China through 863 Program No.2013AA01A604.

References

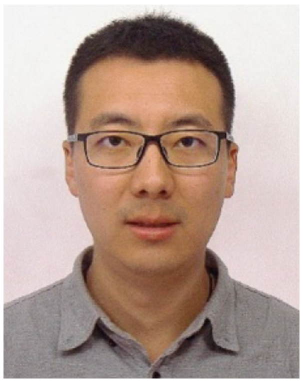
1. 3ds max. http://www.autodesk.com/products/3ds-max/overview (2016)
2. Simplygon. https://www.simplygon.com/ (2016)
3. Akerlund, O., Unger, M., Wang, R.: Precomputed visibility cuts for interactive relighting with dynamic brdfs. In: Conference on Computer Graphics and Applications, pp. 161–170 (2007)
4. Cheslack-Postava, E., Wang, R., Akerlund, O., Pellacini, F.: Fast, realistic lighting and material design using nonlinear cut approximation. ACM Trans. Graph. **27**(5), 32–39 (2008)
5. Dachsbacher, C., Křivánek, J., Hašan, M., Arbree, A., Walter, B., Novák, J.: Scalable realistic rendering with many-light methods. In: Sbert, M., Szirmay-Kalos, L. (eds.) Computer Graphics Forum, vol. 33, pp. 88–104. Wiley Online Library (2014)
6. Davidovič, T., Křivánek, J., Hašan, M., Slusallek, P., Bala, K.: Combining global and local virtual lights for detailed glossy illumination. ACM Trans. Graph. **29**(6), 143:1–143:8 (2010). https://doi.org/10.1145/1882261.1866169
7. Dong, Z., Grosch, T., Ritschel, T., Kautz, J., Seidel, H.P.: Real-time indirect illumination with clustered visibility. In: Proceedings of the Vision, Modeling, and Visualization Workshop 2009, November 16–18, 2009, Braunschweig, Germany, pp. 187–196 (2009)
8. Feris, R., Raskar, R., Tan, K.H., Turk, M.: Specular reflection reduction with multi-flash imaging. In: Proceedings of the Computer Graphics and Image Processing, XVII Brazilian Symposium, SIBGRAPI '04, pp. 316–321, Washington, DC, IEEE Computer Society (2004)
9. Hašan, M., Pellacini, F., Bala, K.: Matrix row-column sampling for the many-light problem. ACM Trans. Graph. **26**(3), 26 (2007). https://doi.org/10.1145/1276377.1276410
10. Holländer, M., Ritschel, T., Eisemann, E., Boubekeur, T.: ManyLoDs: parallel many-view level-of-detail selection for real-time global illumination. Comput. Graph. Forum **30**(4), 1233–1240 (2011). https://doi.org/10.1111/j.1467-8659.2011.01982.x
11. Huo, Y., Wang, R., Jin, S., Liu, X., Bao, H.: A matrix sampling-and-recovery approach for many-lights rendering. ACM Trans. Graph. (TOG) **34**(6), 210 (2015)

12. Kristensen, A.W., Akenine-Möller, T., Jensen, H.W.: Precomputed local radiance transfer for real-time lighting design. ACM Trans. Graph. **24**(3), 1208–1215 (2005)
13. Nichols, G., Penmatsa, R., Wyman, C.: Interactive, multiresolution image-space rendering for dynamic area lighting. In: Lawrence, J., Stamminger, M. (eds.) Computer Graphics Forum, vol. 29, pp. 1279–1288. Wiley Online Library (2010)
14. Oliveira, M.M., Bishop, G., McAllister, D.: Relief texture mapping. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '00, pp. 359–368. ACM Press, New York (2000)
15. Olsson, O., Billeter, M., Sintorn, E., Kampe, V., Assarsson, U.: More efficient virtual shadow maps for many lights . IEEE Trans. Vis. Comput. Graph. **21**(6), 701–713 (2015). https://doi.org/10.1109/TVCG.2015.2418772
16. Olsson, O., Sintorn, E., Kämpe, V., Billeter, M., Assarsson, U.: Efficient virtual shadow maps for many lights. In: Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, pp. 87–96. ACM (2014)
17. Paquette, E., Poulin, P., Drettakis, G.: A light hierarchy for fast rendering of scenes with many lights. In: Ferreira, N., Göbel, M. (eds.) Computer Graphics Forum, vol. 17, pp. 63–74. Wiley Online Library (1998)
18. Popescu, V., Eyles, J., Lastra, A., Steinhurst, J., England, N., Nyland, L.: The warpengine: an architecture for the post-polygonal age. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2000, New Orleans, 23–28 July 2000, pp. 433–442 (2000)
19. Ritschel, T., Engelhardt, T., Grosch, T., Seidel, H.P., Kautz, J., Dachsbacher, C.: Micro-rendering for scalable, parallel final gathering. ACM Trans. Graph. **28**(5), 89–97 (2009)
20. 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), 32–39 (2008)
21. Ritschel, T., Grosch, T., Kautz, J., Eller, S.: Interactive illumination with coherent shadow maps. In Proceedings of the EGSR 2007, pp. 61–72 (2007)
22. Ritschel, T., Grosch, T., Kautz, J., Seidel, H.P.: Interactive global illumination based on coherent surface shadow maps. In: Proceedings of Graphics Interface 2008 (2008)
23. Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M., Greenberg, D.P.: Lightcuts: a scalable approach to illumination. ACM Trans. Graph. **24**(3), 1098–1107 (2005)
24. Walter, B., Khungurn, P., Bala, K.: Bidirectional lightcuts. ACM Trans. Graph. **31**(4), 13–15 (2012)
25. Wang, R., Huo, Y., Yuan, Y., Zhou, K., Hua, W., Bao, H.: Gpu-based out-of-core many-lights rendering. ACM Trans. Graph. (TOG) **32**(6), 210 (2013)
26. Wu, Y.T., Chuang, Y.Y.: Visibilitycluster: average directional visibility for many-light rendering. IEEE Trans. Vis. Comput. Graph. **19**(9), 1566–1578 (2013)
27. Yang, T., Hui-zhong, W., Fu, X., Liang, X.: Inverse image warping without searching. In: International Conference on Control, Automation, Robotics and Vision, pp. 386–390 (2004)



Lili Wang received the Ph.D. degree from the Beihang University, Beijing, China. She is a professor with the School of Computer Science and Engineering of Beihang University, and a researcher with the State Key Laboratory of Virtual Reality Technology and Systems. Her interests include real-time rendering, realistic rendering, global illumination, soft shadow and texture synthesis.

Wenhao Zhang received his B.S. degree in computer science in China University of Geosciences in 2016. He is currently working toward his Master degree in the State Key Laboratory of Virtual Reality Technology and Systems at Beihang University. His research interests include global illumination and real-time rendering.



Nian Li received his B.S. degree at Beihang University in 2012 and received M.S degree in computer science in the State Key Laboratory of Virtual Reality Technology and Systems at Beihang University in 2016.

Boning Zhang received his B.S. degree in Mathematics in Northwestern Polytechnical University in 2013. He earned his M.S degree in Computer Science in the State Key Laboratory of Virtual Reality Technology and Systems at Beihang University in 2016, and is working in Perfect World Company now.

Voicu Popescu received a B.S. degree in computer science from the Technical University of Cluj-Napoca, Romania in 1995, and a Ph.D. degree in computer science from the University of North Carolina at Chapel Hill, USA, in 2001. He is an associate professor with the Computer Science Department of Purdue University. His research interests lie in the areas of computer graphics, computer vision, and visualization. His current projects include camera model design, visibility, augmented reality for surgery telementoring, and the use of computer graphics to advance education.