Eurographics Symposium on Rendering (2004)
H. W. Jensen, A. Keller (Editors)

Alias-Free Shadow Maps

Timo Aila and Samuli Laine

Helsinki University of Technology/TML and Hybrid Graphics Ltd.

Abstract

In this paper we abandon the regular structure of shadow maps. Instead, we transform the visible pixels $P(x,y,z)$ *from screen space to the image plane of a light source* $P'(x',y',z')$*. The* (x',y') *are then used as sampling points when the geometry is rasterized into the shadow map. This eliminates the resolution issues that have plagued shadow maps for decades, e.g., jagged shadow boundaries. Incorrect self-shadowing is also greatly reduced, and semi-transparent shadow casters and receivers can be supported. A hierarchical software implementation is outlined.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Color, shading, shadowing, and texture I.3.3 [Computer Graphics]: Picture/Image Generation – Bitmap and framebuffer operations

1. Introduction

The presence of shadows in computer-generated images is valuable because they both reveal information about the spatial relationships of objects and increase the level of realism. Shadow mapping [Wil78] is a simple and widely used method that supports a large number of rendering primitives.

Shadow mapping generates a depth buffer from the point of view of a light source. This depth buffer, called a *shadow map*, is a discretized representation of the scene geometry as seen by the light source. A pixel is in shadow if its depth value, transformed into the light-space, is greater than the corresponding depth value in the shadow map.

The regularly spaced sampling points (i.e., pixels) of the shadow map do not correspond to pixels in screen space. This, along with the discrete resolution of a shadow map is a serious source of artifacts. False self-shadowing arises when a surface incorrectly occludes itself. This problem is primarily due to a transformed screen-space pixel landing somewhere between the sampling points of the shadow map. There is no guarantee that all features, such as discontinuities can be faithfully reconstructed from the sampling points. A number of authors have embedded additional data to the sampling points in order to capture the scene more accurately, e.g., plane equation [Gra92] or multiple surfaces per sampling point [Woo92, Gra92]. The second, much lesser source of artifacts is the numerical error due to the transformation between coordinate systems. This error is independent of the scene, and can thus be fixed with a constant bias.

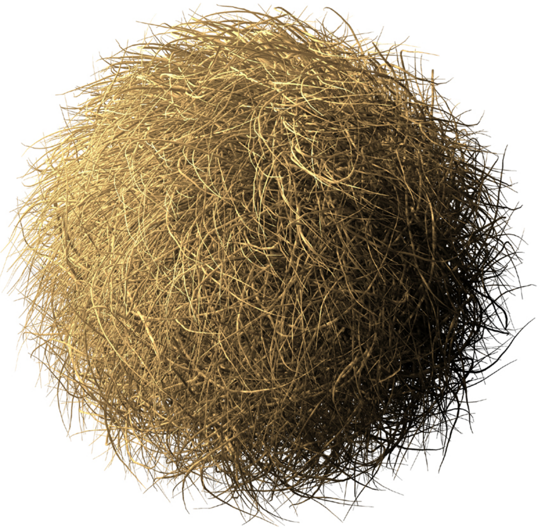

Figure 1: *The self-shadowing hair ball consists of 5000 hairs, each made of 576 triangles. The hairs act as semi-transparent shadow casters with opacity of 10%. In total there are 2.9M triangles, and the rendering took 12.6 seconds with a software prototype running on a 1.6GHz Mobile Pentium 4 at* 1536 × 1088 *resolution.*

A local resolution mismatch between the shadow map and the output image often causes jagged shadows, even when the shadow map has a resolution higher than the screen-space image. Partial solutions exist for this problem [RSC87, LV00, FFBG01, SD02, SCH03].

Overview Our idea is straightforward. Clearly, the shadow terms are needed only for the visible pixels of the screen-space image. We compute exactly that by transforming the visible pixels to the image plane of the light source, and do *not* discretize the projected (x,y)-coordinates. The projected points are then used as sampling points when the geometry is rasterized from the light source. This eliminates the concept of shadow map resolution. The results correspond exactly to hard shadows computed using shadow rays [Whi80]. Resolution mismatch problems are completely eliminated, and incorrect self-shadowing is greatly reduced. The new representation can also support semi-transparent shadow receivers and casters.

Efficient rasterization is more challenging using the new representation due to lost coherence. We outline a hierarchical method that has a simple software implementation and acceptable performance characteristics.

2. Related Work

This section concentrates on the problems of shadow maps. A vast amount of literature exist about shadow algorithms in general, and several surveys are available [WPF90, HM01, HLHS03].

Shadow mapping [Wil78] is based on generating a z-buffer [Cat74] from the light source. In practice, up to six shadow maps are required for an omni-directional light source that is inside the view frustum.

Resolution mismatch The resolution mismatch between the output image and the shadow map has been studied by several authors. Fernando et al. [FFBG01] organize the shadow map into a hierarchical grid. The resolution of the shadow map is then locally increased in places where depth discontinuities occur or the resolution of the screen-space image is greater than the resolution of the shadow map.

Stamminger and Drettakis [SD02] propose generating the shadow map in post-perspective space, i.e., after the projection to the view frustum. This significantly improves the shadow quality in many scenes. Special treatment is needed to ensure that all shadow casting objects are taken into account.

Brabec and Seidel [BS02a] shrink the frustum of the light source to tightly enclose the visible part of the view frustum. Sen et al. [SCH03] augment the shadow map by storing the location of a vertex inside a shadow map pixel. This significantly reduces jagged shadow boundaries, but only one vertex is stored per shadow map pixel, and thus high quality results are obtained only in a subset of scenes.

Bias problems False self-shadowing (aka bias problem) is a severe problem caused primarily by the limited precision of the discrete shadow map. It occurs when an illuminated surface appears to reside behind its own shadow map footprint. Williams [Wil78] proposed solving the problem by adding a constant bias value to the transformed depth value in order to weaken the coincidence test. Unfortunately, a constant bias factor cannot handle all situations satisfactorily. The artifacts can be reduced by comparing the transformed depth values to the second closest surface [WM94], or against an intermediate surface, which lies between the two surfaces that are closest to the light source at that pixel [Woo92, Gra92].

Grant [Gra92] proposes adding a plane equation to each sampling point of a shadow map. This helps in certain cases, but fails when more than one polygon should be represented inside a single shadow map pixel. Hourcade and Nicolas [HN85] assign each object or polygon a unique ID, and augment the depth values with the corresponding IDs. Incorrect self-shadowing is avoided by comparing the IDs instead of depth values.

Generalizations of shadow maps Deep shadow maps by Lokovic and Veach [LV00] generalize shadow maps to semi-transparent surfaces while also significantly reducing the aliasing artifacts in off-line rendering. Dachsbacher and Stamminger [DS03] augment shadow maps with irradiance values and normal vectors in order to approximate subsurface scattering. Percentage closer filtering [RSC87] creates approximate soft shadows by blurring the shadow boundaries.

Ray Tracing Ray tracing [Whi80] produces high-quality shadows by testing if the line-of-sight between the point to be shaded and a light source is blocked. Shadow maps with our correct sampling points give the same answer as tracing the shadow rays.

Concurrent Work In a parallel work, Johnson et al. [JMB04] propose the same core idea as this paper. Their paper is inclined towards a possible hardware implementation, and includes an accuracy analysis of earlier shadow mapping methods. Semi-transparent shadow casters or receivers are not discussed.

3. Projection of Sampling Points

The general idea of our algorithm is illustrated in Figure 2. First, a depth buffer is computed from the point of view of the camera. The screen-space (x,y)-coordinates of the pixels along with the corresponding depth buffer values, are the visible samples $P(x,y,z)$ (Figure 2a). Then, the $P(x,y,z)$ are transformed into the image plane of the light source (Figure 2b), producing sampling points (x',y') and the corresponding light-space depth values z'. The (x',y',z') are stored into a separate buffer. The (x',y') that land inside the

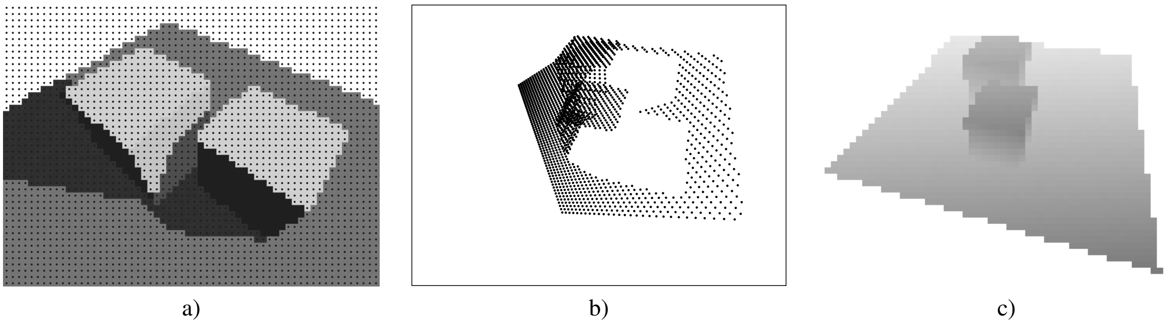

Figure 2: *a) A simple test scene with shadows, as seen from the camera. The black dots are the pixel centers. b) The visible pixels of (a) transformed into the image plane of the light source. The dots are used as sampling points when the scene is rasterized to the shadow map. The large empty areas correspond to regions that are not visible from the camera, and thus need no shadow information. c) The corresponding traditional shadow map is shown for comparison purposes only. In a traditional shadow map algorithm, the regularly sampled map (c) would be tested exactly at the sampling points shown in (b). Clearly, the regular structure of (c) is not suitable for accurately answering the queries.*

shadow map are the optimal sampling points, and exactly correspond to the intersections of shadow rays and the image plane of the light source.

Note that the distribution of sampling points in Figure 2b is highly irregular. The large empty areas correspond to regions that are not visible to the camera. The lack of sampling points in such regions means that no shadow computations are performed for redundant areas.

The (x', y') are used as sampling points when the scene is rasterized from the light source. Traditionally the sampling points have been at the center of each pixel, but that is not a requirement for rasterization. A slow brute force implementation would test all sampling points when rasterizing a geometric primitive. A more efficient hierarchical variant is discussed in Section 5. If a sampling point is covered by an input primitive, the depth value of the primitive is computed at the sampling point and compared to the light-space z' of the sample. If the comparison indicates that the sample is in shadow, the shadow term is written directly to the corresponding screen-space sample, for example to a stencil buffer or an additional color buffer. As a result, separate shadow map queries are not needed. This is different from traditional shadow maps, and possible only because we are working directly with the visible samples of the output image.

The numerical error caused by the floating point transformation between the coordinate systems is the only reason why the resulting shadows might differ from the results of tracing a shadow ray from each visible sample. This is different than the bias value [Wil78] because the numerical value of the error depends only on the particular implementation of matrix functions, i.e., it does *not* depend on the scene.

The physical dimensions of the viewport on the image plane of the light source are identical to the traditional shadow maps. However, the concept of shadow map resolution is eliminated because our novel shadow map formulation does not require light-space discretization. The shadow boundaries stay sharp no matter how close the camera is, as illustrated in Figure 3. When supersampling or multisampling is used for antialiasing, the shadow term needs to be determined separately for each sample. This is accomplished by transforming all visible samples to the image plane of the light source. As with traditional shadow maps, multiple maps are generally needed for modeling omni-directional light sources.

4. Semi-Transparent Surfaces

Similarly to traditional shadow maps, semi-transparent shadow receivers can be supported by storing all visible surfaces to each screen-space pixel. The visible samples are then transformed to the image plane of the light source in order to get sampling points for the shadow map rasterization. Finally, the output image is obtained by compositing the correctly shadowed semi-transparent surfaces to the opaque background.

Semi-transparent shadow casters that modulate the percentage of light that passes through the surface can also be supported. Each sampling point of a shadow map is assigned an RGB color, which is initially white. A semi-transparent surface that is rasterized to the front of the light-space depth of a sampling point modulates the corresponding RGB color. Opaque surfaces set the color to black. After all surfaces have been rasterized to the shadow map, the RGB values indicate the color of the shadow. Figure 1 shows an example of semi-transparent shadow casters. Deep shadow maps [LV00] store the light attenuation as functions to each shadow map pixel. We do not need such functions to get the same result, but it deserves to be pointed out that deep shadow maps have a very broad range of applications. For example, efficient rendering of volumetric effects certainly benefits from the attenuation functions, and ideally deep shadow maps would be combined with our sampling points. Also, deep shadow maps support high quality filtering. Section 6 discusses this and other important extensions.

5. Hierarchical Implementation

Ideally, only the sampling points that are covered by a geometric primitive would be tested during rasterization. With regularly spaced sampling points this is achievable, but an irregular structure poses challenges. A brute force implementation would test all sampling points for every primitive, resulting in tremendous performance penalty. We present a hierarchical method that finds the necessary sampling points in logarithmic time. As a comparison, this is a constant-time operation in traditional rasterization.

5.1. Light-Space Hierarchy

After the sampling points have been transformed to the image plane of the light source, they are organized into an axis-aligned 2D BSP tree (Figure 4). A quadtree or any other subdivision algorithm could also be used, but an axis-aligned BSP tree adapts to irregular points very well, and has an easy and fast implementation. We use alternating splits along the x- and y-axes so that the set of sampling points is always divided in half. The quickselect algorithm [Hoa61] finds the position of each splitting plane in expected linear time. The nodes are subdivided until each leaf node contains less than a predetermined number of sampling points; we currently use 64.

Once the hierarchy has been constructed, the shadow casting primitives are rasterized using the subdivision. The traversal starts from the root node, and proceeds to subnodes that are at least partially covered by the input primitive. The sampling points in the leaf nodes are then tested individually. Finally, the depth value is computed for each covered sampling point, and compared to the stored light-space depth value.

We have additionally incorporated occlusion culling into our hierarchical rasterization. Once all the sampling points belonging to a node have been determined to be in shadow, the consecutive rasterization to the node is skipped for all subsequent primitives. Additionally, each node stores the maximum light-space depth value. If a primitive is farther than the maximum depth value, the primitive cannot cast shadows to the output image and can thus be skipped.

The performance and scalability results of our prototype software implementation are shown in Table 1. The tests were run on a 1.6GHz Mobile Pentium 4 with 512MB of memory. The scenes are particularly difficult for shadow algorithms, as can be observed from Figure 3. The sizes of the triangles were adjusted so that their total area is approximately constant in all the scenes, thus eliminating the effect of occlusion culling. As can be seen, the prototype scales well with scene complexity. The cost of BSP construction depends only on the screen resolution (1024 × 768). In the simplest scene the cost of BSP construction dominates, whereas in the more complex scenes its cost becomes negligible. Increasing the average size of the triangles creates more occlusion, and the execution time drops significantly. As a conclusion, the performance numbers are promising when compared to the quality of the resulting shadows.

#Triangles	1K	10K	100K	1M
#2D BSP nodes	16K	16K	16K	16K
#2D BSP nodes visited	91K	471K	3.2M	23M
#nodes occlusion culled	5.3K	5.4K	5.8K	6.8K
#sampling points tested	458K	513K	506K	501K
Transformation of samples	130			
BSP construction	400	402	391	408
Rasterization of blockers	145	381	1568	8102
Total time (ms)	675	913	2089	8640

Table 1: *Scalability statistics from four scenes with 1K-1M random triangles. The resolution was* 1024×768*, and the viewpoint corresponded to the middle figure on the top row of Figure 3. All timings are in milliseconds.*

6. Open Issues and Future Work

The new irregular representation of shadow maps has highly desirable properties, but there are several open issues and possible extensions that present interesting avenues for future work.

It seems obvious that the simplicity of filtering the shadow boundaries [RSC87, BS02b] or shadow map extensions that are more physically-based [LV00, CD03, WH03] are at least partially lost. However, this does not mean that something similar would not be feasible in conjunction with the new representation.

Interactive and real-time use of shadow maps requires hardware support. It seems that the algorithm cannot be efficiently implemented on currently available hardware. The minimal amount of necessary modifications is an important question, and calls for a careful analysis of the entire spectrum of possible implementations with respect to their limitations. A dialogue with hardware vendors should be fruitful in formulating the hierarchies and traversal in a maximally hardware-friendly way.

7. Conclusions

We have introduced a new representation of shadow maps that completely abandons the previously used regular structures. The resulting shadow quality equals that of shadow rays, and thus the resolution problems of shadow maps are solved. We believe that the results presented in this study open new interesting possibilities for developing aliasing-free shadow map-related algorithms. We have outlined and tested a hierarchical implementation. Our test results indicate that the algorithm performs very well with fine geometric detail that would otherwise need excessively high shadow map resolutions.

Acknowledgements

We would like to thank Jukka Arvo, Jaakko Lehtinen, Ville Miettinen and Lauri Savioja for their helpful comments. This work has been partially funded by the National Technology Agency of Finland, Bitboys, Hybrid Graphics, Nokia and Remedy Entertainment.

References

[BS02a] BRABEC S., SEIDEL H.-P.: Practical shadow mapping. *Journal of Graphics Tools 7*, 4 (2002), 9–18. 2

[BS02b] BRABEC S., SEIDEL H.-P.: Single sample soft shadows using depth maps. In *Proceedings of Graphics Interface* (2002), pp. 219–228. 4

[Cat74] CATMULL E.: *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, University of Utah, 1974. 2

[CD03] CHAN E., DURAND F.: Rendering fake soft shadows with smoothies. In *Proceedings of the Eurographics Symposium on Rendering* (2003), Eurographics Association, pp. 208–218. 4

[DS03] DACHSBACHER C., STAMMINGER M.: Translucent shadow maps. In *Proceedings of the 14th Eurographics workshop on Rendering* (2003), Eurographics Association, pp. 197–201. 2

[FFBG01] FERNANDO R., FERNANDEZ S., BALA K., GREENBERG D. P.: Adaptive shadow maps. In *Proceedings of ACM SIGGRAPH 2001* (2001), ACM Press, pp. 387–390. 2

[Gra92] GRANT C.: *Visibility Algorithms in Image Synthesis*. PhD thesis, University of California, 1992. 1, 2

[HLHS03] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F.: A Survey of Real-Time Soft Shadows Algorithms. *Computer Graphics Forum, 22*, 4 (2003). State-of-the-Art Report. 2

[HM01] HAINES E., MÖLLER T.: Real-Time Shadows. In *Proceeding of Game Developers Conference* (March 2001), pp. 335–352. 2

[HN85] HOURCADE J. C., NICOLAS A.: Algorithms for antialiased cast shadows. *Computer Graphics 9*, 3 (1985), 259–265. 2

[Hoa61] HOARE C. A. R.: Algorithm 65: Find. *Communications of the ACM 4*, 7 (1961), 321–322. 4

[JMB04] JOHNSON G. S., MARK W. R., BURNS C. A.: *The Irregular Z-Buffer and its Application to Shadow Mapping*. Tech. rep., The University of Texas at Austin, Department of Computer Sciences, April 2004. 2

[LV00] LOKOVIC T., VEACH E.: Deep shadow maps. In *Proceedings of ACM SIGGRAPH 2000* (2000), ACM Press, pp. 385–392. 2, 3, 4

[RSC87] REEVES W. T., SALESIN D. H., COOK R. L.: Rendering antialiased shadows with depth maps. In *Computer Graphics (Proceedings of ACM SIGGRAPH 87)* (1987), pp. 283–291. 2, 4

[SCH03] SEN P., CAMMARANO M., HANRAHAN P.: Shadow Silhouette Maps. *ACM Transactions on Graphics, 22*, 3 (2003), 521–526. 2

[SD02] STAMMINGER M., DRETTAKIS G.: Perspective shadow maps. In *Proceedings of ACM SIGGRAPH 2002* (2002), ACM Press, pp. 557–562. 2

[WH03] WYMAN C., HANSEN C.: Penumbra maps: Approximate soft shadows in real-time. In *Proceedings of the Eurographics Symposium on Rendering* (2003), Eurographics Association, pp. 202–207. 4

[Whi80] WHITTED T.: An improved illumination model for shaded display. *Communications of the ACM 23*, 6 (1980), 343–349. 2

[Wil78] WILLIAMS L.: Casting Curved Shadows on Curved Surfaces. In *Computer Graphics (Proceedings of ACM SIGGRAPH 78)* (1978), ACM, pp. 270–274. 1, 2, 3

[WM94] WANG Y., MOLNAR S.: *Second-Depth Shadow Mapping*. Tech. rep., The University of North Carolina at Chapel Hill, 1994. 2

[Woo92] WOO A.: The shadow depth map revisited. *Graphics Gems III* (1992), 338–342. 1, 2

[WPF90] WOO A., POULIN P., FOURNIER A.: A Survey of Shadow Algorithms. *IEEE Computer Graphics and Applications 10*, 6 (1990), 13–32. 2

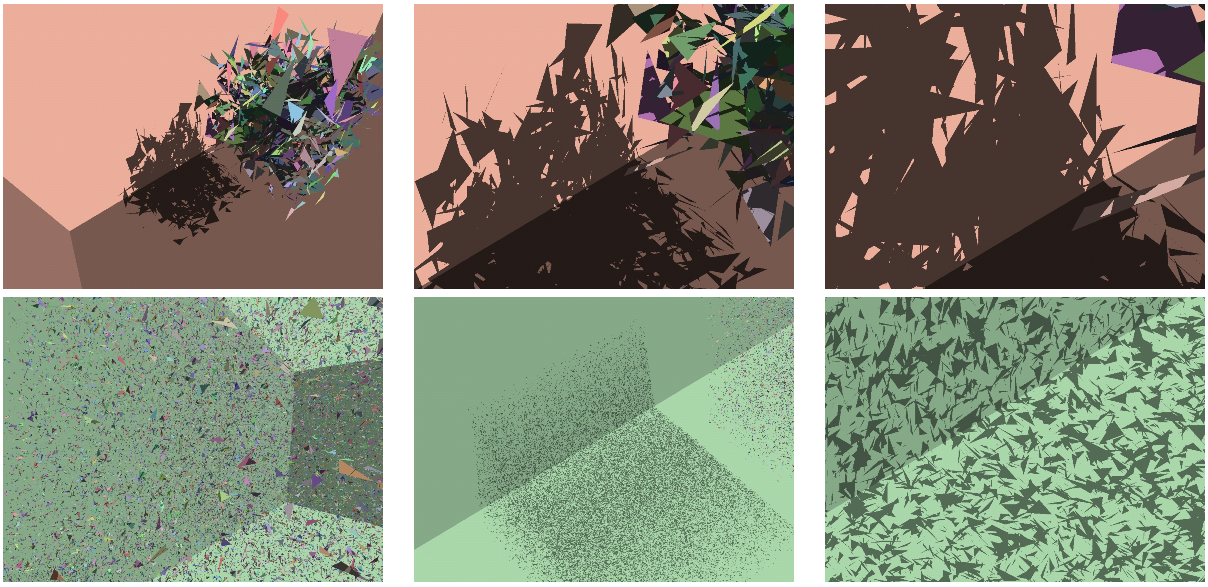

Figure 3: Top row: *The zoom sequence demonstrates that jagged shadows are completely avoided because the concept of shadow map resolution has been eliminated. While crisp shadow boundaries can be obtained by using traditional, adaptive or perspective shadow maps, the resolution requirements are completely dependent on the light source position, viewing parameters, and especially the scene geometry. On the contrary, our representation always gives accurate results in all scenes due to its one-to-one correspondence with shadow rays. The number of sampling points equals the screen resolution,* 1024×768 *in this example.* Bottom row: *A particularly challenging scene with 100K small random triangles. The two-frame zoom sequence again demonstrates the shadow quality resulting from our method. The shadow computation time using our prototype implementation was* $1.8-2.8$ *seconds per frame, depending on the viewpoint. The resolution was* 1024×768.

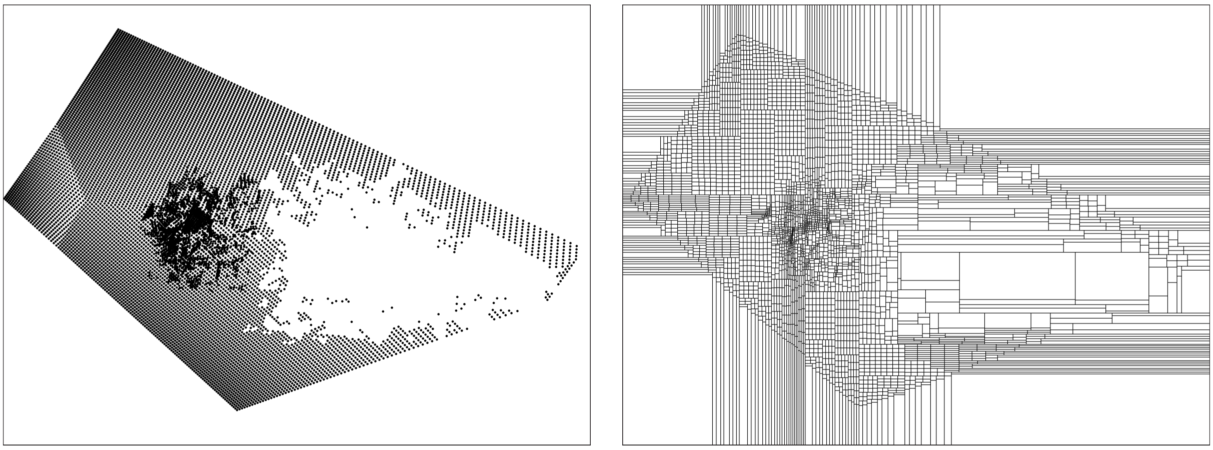

Figure 4: *Left: the pixels of leftmost scene in Figure 3 (top row) transformed to the image plane of the light source. For illustration purposes, only* $\frac{1}{64}$*th of the sampling points are shown. Right: The axis-aligned BSP built from the sampling points. In this visualization, every leaf in the BSP tree contains at most 256 sampling points.*