Soft Irregular Shadow Mapping: Fast, High-Quality, and Robust Soft Shadows

Gregory S. Johnson [1,2] †
Allen Hux [2] †
Christopher A. Burns [1,2] †
Warren A. Hunt [1,2] ‡
William R. Mark [1,2] ‡
Stephen Junkins [2] ‡

University of Texas at Austin [1]
Intel Corporation [2]

Abstract

We introduce a straightforward, robust, and efficient algorithm for rendering high-quality soft shadows in dynamic scenes. Each frame, points in the scene visible from the eye are inserted into a spatial acceleration structure. Shadow umbrae are computed by sampling the scene from the light at the image plane coordinates given by the stored points. Penumbrae are computed at the same set of points, per silhouette edge, in two steps. First, the set of points affected by a given edge is estimated from the expected light-view screen-space bounds of the corresponding penumbra. Second, the actual overlap between these points and the penumbra is computed analytically directly from the occluding geometry. The umbral and penumbral sources of occlusion are then combined to determine the degree of shadow at the eye-view pixel corresponding to each sample point. An implementation of this algorithm for the Larrabee architecture yields from 27 to 33 frames per second in simulation for scenes from a modern game, and produces significantly higher image quality than other recent methods in the real-time domain.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Visible Line / Surface Algorithms

Keywords: real-time, soft shadows, shadow mapping, Larrabee

1 Introduction

Conceptually, the soft shadow computation consists of determining the degree of illumination arriving at a point in the scene from an area light. This value depends primarily on the spatial relationship of the light and scene geometry, and properties of the light itself. Specifically, the irradiance E from a diffuse area light incident on a point with normal $\vec{n}$ (Figure 2a) can be expressed as an integral over the area A of the light. In Equation 1, the term $\Phi(\mathbf{x})$ is the intensity of the light at point $\mathbf{x}$. The shape of the light determines both $\hat{n}_l(\mathbf{x})$ and the domain of integration A. Vector $L(\mathbf{x})$ extends from the receiver point to $\mathbf{x}$.

$$E = \int_{\mathbf{x}\in A} (\hat{n} \bullet \hat{L}(\mathbf{x}))(\hat{n}_l(\mathbf{x}) \bullet -\hat{L}(\mathbf{x})) \frac{\Phi(\mathbf{x})}{|L(\mathbf{x})|^2} V(\mathbf{x}) d\mathbf{x} \qquad (1)$$

Many common light sources have nearly constant values for $\hat{n}_l$ as well as Φ. Furthermore, if the light is small (also common) then L is nearly constant, and both of the dot products as well as the distance term $1/|L(\mathbf{x})|^2$ can be moved outside the integral (Equation 2), leaving only the visibility term $V(\mathbf{x})$. This term is largely responsible for the visual appearance of shadow penumbrae, but is also the most difficult to compute, requiring a search over the scene geometry to identify occluders.

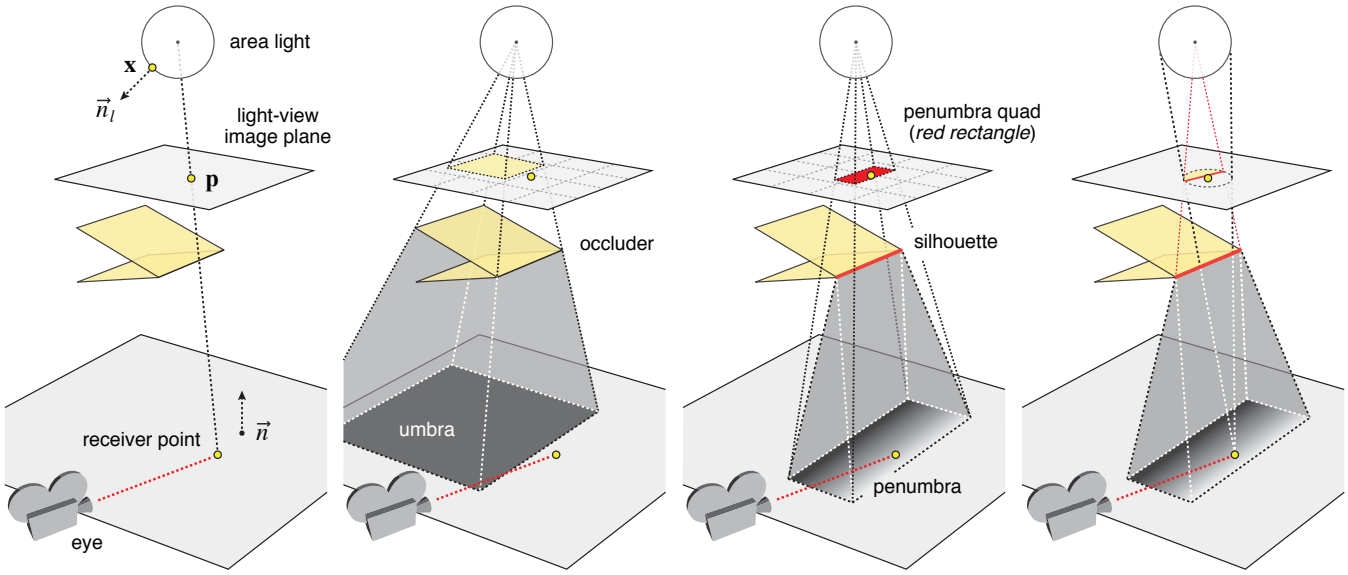
Figure 1: *Our soft shadow algorithm applied to a saloon scene from Call of Juarez by Techland. To emphasize the quality of the shadow penumbrae, no antialiasing, texture mapping, or complex shading is performed. Scene is used with permission.*

$$E \approx (\hat{n} \bullet \hat{L})(\hat{n}_l \bullet -\hat{L}) \frac{\Phi}{|L|^2} \int_{\mathbf{x}\in A} V(\mathbf{x}) d\mathbf{x} \qquad (2)$$

The primary contribution of this paper is a new method for computing the visibility term of Equation 2 that is high quality, efficient, robust, and straightforward to implement. Our algorithm is based on two principles. First, accuracy and efficiency favor computing light-view visibility *only at, and exactly at, points in the scene visible from the eye*. Second, this occlusion computation can be split into umbral (i.e. a point is fully occluded) and penumbral (a point is partially occluded) parts. This allows us to restrict the relatively costly penumbral computation to points near silhouettes.

A key property of our approach is that points in the scene visible from the eye are stored in an irregular spatial acceleration structure in light space. The storage order of these points is determined at run time based on their relative positions. High performance per-frame construction and traversal of such data structures requires efficient scatter / gather memory operations, global atomic operations, and synchronization. The Larrabee architecture due in 2009 or 2010 [Intel Corporation 2008] is one example of a processor with these features. Through simulation on this architecture, we find that the performance of our algorithm is comparable to the best performing existing methods, but produces substantially higher image quality.

†e-mail: {gregory.s.johnson, allen.hux, christopher.a.burns}@intel.com
‡e-mail: {warren.hunt, william.r.mark, stephen.junkins}@intel.com



(a) Data structure construction. (b) Computation of umbrae. (c) Coarse penumbra computation. (d) Final penumbra computation.

Figure 2: *Soft irregular shadow mapping occurs in four steps. The scene is rendered from the eye and the visible points (i.e. receiver points) are inserted into a spatial acceleration structure* (a)*. Umbrae are determined by projecting scene primitives into the light-view image plane and testing for overlap with the image plane coordinates* ($\mathbf{p}$) *of the points in this data structure* (b)*. Penumbrae are computed in two steps. The first estimates the light-view screen-space bounds of the penumbra cast by a silhouette edge* (c)*. For each point located inside these bounds, a more accurate visibility test is performed. Here, the occluding surface is clipped to the bounds of the area light as seen from the receiver point, and the fractional area of occlusion is measured* (d)*. The umbral and penumbral sources of occlusion are combined to determine the amount of shadow at the eye-view pixel corresponding to each receiver point.*

2 Algorithm

Conceptually, the computation of penumbral occlusion consists of determining the area of the light occluded by geometry as seen from a receiver point. For a light of uniform intensity, occlusion can be found in 2D by projecting the light and occluder into the light-view image plane and measuring the area of overlap. Our algorithm does exactly this, and combines the result with umbral occlusion computed via hard irregular shadow mapping [Johnson et al. 2004; Aila and Laine 2004]. As with all other soft shadow algorithms in the real-time performance regime ours is approximate. However, the errors introduced are commonly imperceptible.

The complete algorithm is illustrated in Figure 2. First, receiver points are identified by rendering scene geometry from the eye (a). The points are sorted by their light-space coordinates, and inserted into the spatial acceleration structure shown in Figure 3. Second, umbral occlusion is computed by projecting scene geometry into the light-view image plane (b). Each primitive is tested for overlap against the receiver points stored in the data structure. Third, a coarse penumbral occlusion computation is performed (c). Here, the set of receiver points affected by a given silhouette edge is estimated by testing the points for overlap against a "penumbra quad". The quad represents the expected light-view screen-space bounds of the penumbra cast by the edge. Fourth, a more accurate penumbral occlusion computation is performed for each point covered by a quad (d). The area of overlap between the silhouette geometry and the light-view image plane projection of the light (as seen from the point) is measured. The silhouette contributes positive occlusion in the case of an outer penumbra, and negative occlusion in the case of an inner penumbra. Finally, the umbral and penumbral occlusion accumulated at a receiver point is used to modulate the intensity of the corresponding eye-view pixel.

2.1 Silhouette Edge Detection

The identification of silhouette edges on modern graphics hardware is straightforward. For example, the DX10 API [Blythe 2006] exposes vertex adjacency information which can be used within a geometry shader to identify silhouettes. Though identification of silhouettes improves the performance of our algorithm, doing so *is not necessary for correctness*. In the absence of adjacency information we assume all edges are silhouettes. The composition of umbral and penumbral occlusion discussed in Subsection 2.4 ensures correctness is preserved.

2.2 Silhouette Edge Representation

Note that the penumbra cast by a silhouette edge forms a wedge as seen in Figure 2c. The projection of this wedge into the light-view image plane can be (conservatively) represented as a rectangle. As a result, the "penumbra quads" used during the coarse visibility test in our algorithm are rectangular and coplanar to the light-view image plane. The width of a quad depends on the light radius, the minimum depth of the silhouette edge, and the maximum depth of any receiver point occluded by the surface adjacent to the edge [Parker et al. 1998]. Unfortunately, the maximum receiver depth cannot be accurately determined prior to computing a complete visibility solution. Therefore, this value is initially set to the maximum light-view depth of any receiver point in the scene and later refined (Subsection 2.5).

2.3 Penumbral Occlusion

To review, occlusion from a spherical light of uniform intensity can be determined by projecting the light and occluding silhouette edge

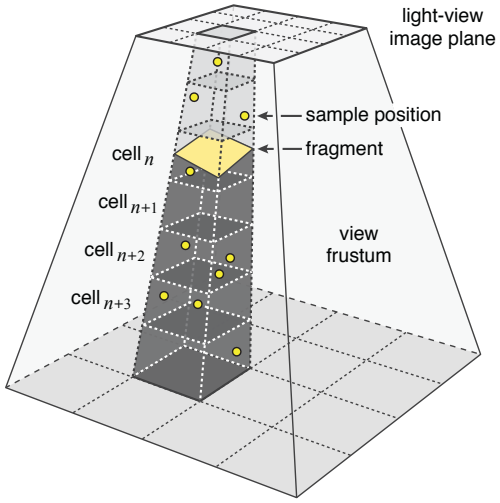


Figure 3: *Conceptually, our light-view spatial acceleration structure is a 3D perspective grid where each cell stores a list of the samples within the cell bounds. This data structure is irregular in the number of samples per cell. Range queries can be performed by rasterizing scene primitives or penumbra quads to the frontmost face (i.e. image plane). In memory, samples are stored contiguously in a separate 1D array. The storage order of samples in this array is such that samples from* cell_{n+3} *immediately follow those from* cell_{n+2} *in memory, which in turn follow those from* cell_n *in memory (since* cell_{n+1} *contains no samples). This design exposes spatial reuse of samples on the same cache line, and maximizes the efficiency of vector memory operations during computation of occlusion.*

into the light-view image plane, and measuring the area of the light footprint clipped by the edge. This clipping strategy (Figure 4) avoids penumbral aliasing often found in methods which measure occlusion at discrete points on the light surface. The radius (R) of the light footprint is estimated from the radius of the actual light, the depth of the receiver point ($\mathbf{p}$), and the depth of a point ($\mathbf{q}$) on the silhouette edge nearest $\mathbf{p}$. A negative value indicates the silhouette geometry is further from the light than the receiver point and no occlusion is possible. For positive values of R, the normalized area of the minor circular segment clipped by the silhouette edge (shaded area) is computed from Equation 3. The sign of the dot product of the edge equation and $\mathbf{p}$ determines if the result is added (i.e. $\mathbf{p}$ is outside the edge) or subtracted ($\mathbf{p}$ is inside the edge) from the total occlusion accumulated at the receiver point.

$$\begin{aligned} \theta &= cos^{-1}\left(\frac{d^2 - w_0 w_1}{\sqrt{(d^2 + w_0^2)\,(d^2 + w_1^2)}}\right) \\ \tilde{V}'' &= \frac{\pm\,\theta - d\,(w_0 + w_1)}{2\pi} \end{aligned} \tag{3}$$

This strategy of determining penumbral occlusion by measuring the area of overlap between a light and silhouette geometry is not new. For example, Assarsson et al. [Assarsson et al. 2003] describe an occlusion kernel for spherical lights similar to ours, though the method of computation is different. In our algorithm, an inverse cosine function call is used in place of a table of inverse tangent values. The latency of the function call can be higher than that for the table lookups if the entries are in cache, but the increased accuracy inhibits penumbral banding. Further, in our algorithm the clipping operation is performed in 2D and so avoids evaluating the quadratic equation used in intersecting a line with a cone in 3D.

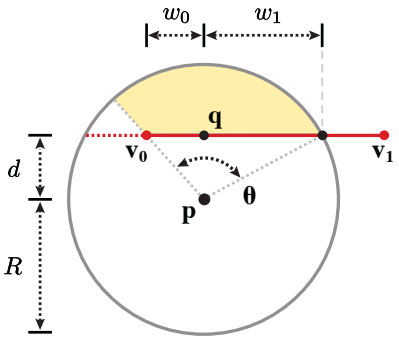


Figure 4: *A silhouette edge (red) and adjacent surface are clipped to the bounds of a 2D projection of the light (enclosing circle). The radius R is computed from the light radius and the light-view depth to the point* ($\mathbf{q}$) *nearest* $\mathbf{p}$ *on the silhouette edge. The signed and normalized area of the minor circular segment defined by the edge (yellow) is computed with Equation 3. This method is accurate even when one or both edge vertices lie inside the sample bounds.*

The computation of penumbral occlusion using a spherical light is motivated by the relative simplicity of the calculation which results from the radial symmetry of this shape. However, the overall algorithm is *not restricted to spherical lights*. For example, Assarsson et al. describe an occlusion kernel for rectangular lights [Assarsson et al. 2003], which can be used with our algorithm. In general, any function which returns the amount of occlusion given a silhouette edge and receiver point can be used, though the performance of the overall algorithm is dominated by this kernel.

2.4 Composition of Occlusion

The visibility function at a given receiver point cannot be accurately reconstructed from penumbral occlusion alone [Laine et al. 2005], since this computation is only performed for silhouette edges that pass within a distance of R of the point. Occlusion due to surfaces which fully cover the light as seen from the receiver point (i.e. umbral occlusion) is unaccounted for. Since umbral occlusion is constant for all points on the light it can be measured from a single point anywhere on the light surface using a hard shadow algorithm. We do so via hard irregular shadow mapping [Johnson et al. 2005]. However, this algorithm is modified slightly to measure the total depth complexity between a receiver point ($\mathbf{p}$) and the light. Combined with the penumbral occlusion as in Equation 4, the result forms a complete solution to the visibility term of Equation 2.

$$\int_{\mathbf{x}\in A} V(\mathbf{x})d\mathbf{x} \approx 1 - \Bigg(\underbrace{\sum_{i=1}^{N} \tilde{V}'(object_i, \mathbf{p})}_{\text{umbra}} + \underbrace{\sum_{j=1}^{E} \tilde{V}''(edge_j, \mathbf{p})}_{\text{penumbra}}\Bigg)_{[0,1]} \tag{4}$$

2.5 Optimizations

A source of inefficiency in many rasterization-based soft shadow algorithms is overdraw. Here, *overdraw* refers to the unnecessary computation of penumbral occlusion at receiver points, and occurs when an edge is tested for occlusion against a point but is found not to occlude the point. Overdraw is difficult to address since it stems from a circular dependence in the penumbral occlusion calculation. This calculation is expensive and should be performed only where needed, *but the set of receiver points affected by a given silhouette*

edge cannot be accurately determined without performing this calculation. In our algorithm, a coarse visibility test reduces overdraw by estimating the spatial relationship between silhouette edges and receiver points. During this test, the expected light-view screen-space bounds of the penumbra cast by a silhouette edge is represented by a quad. Each quad must be wide enough to occlude all receiver points potentially affected by the respective edge, but overestimation of the width can result in significant overdraw. Unfortunately, this width cannot be accurately pinpointed until a complete visibility solution is computed, as it is proportional to the maximum depth of any object occluded by the silhouette. The width is bounded by the maximum light-view depth (Z) of any receiver point, but this bound is insufficient to avoid substantial overdraw.

We address this problem with two simple optimizations. One uses hierarchical depth information accessed through the Larrabee rasterization pipeline to reduce overdraw in the light-view image plane. The Larrabee rasterizer is a "sort-middle" design [Seiler, Carmean, et al. 2008]. Geometry is sorted into screen-space bins, and rasterization proceeds bin by bin. The width of a penumbra quad is initially computed from Z, and is recomputed during sorting using the maximum receiver depth in each bin (measured during data structure construction). If the quad no longer overlaps the bin it is not included in the bin geometry list. The second optimization reduces overdraw in the image plane and in depth, during rasterization. Observe that the penumbra cast by a silhouette edge forms a wedge (Figure 2c). Samples outside of this wedge are not in the penumbra. Therefore, for each fragment from a penumbra quad, we compute the intersection between the wedge due to the penumbra represented by the quad, and the column of grid cells under the fragment. Final visibility (Equation 3) is only computed for samples at this intersection point and deeper in the grid.

2.6 Asymptotic Behavior

Table 1 illustrates the sensitivity of our algorithm to changes in eye-view image resolution, number and coverage of scene primitives, number of silhouette edges, and area light radius.

Image Resolution

Recall that there exists a single light-view sample per eye-view pixel. Soft irregular shadow mapping is linear in the number of eye-view pixels and hence in the number of light-view samples. In principle, construction of a spatial acceleration structure requires sorting the coordinates of the elements to be stored (in this case samples). In practice, our data structure consists of a grid in which the samples within a given cell are unordered. This partial sort can be performed in $O(n)$ time rather than the $O(n \log n)$ time required for a full sort. Similarly, the hard and soft shadow kernels consist of point sampling or area sampling geometric primitives. This operation is constant per sample per primitive, and thus occurs in linear time overall.

n	Data Structure Construction	Hard Shadow Kernel	Soft Shadow Kernel
image resolution	$O(n)$	$O(n)$	$O(n)$
scene geometry	--	$O(n)$	--
silhouette edges	--	--	$O(n)$
area light radius	--	--	$O(n^2)$

Table 1: *The asymptotic performance of our algorithm is shown in relation to several scene-specific properties. Recall that the number of light-view samples is equal to the number of eye-view pixels, and the scene primitives and silhouette edges are used only in the computation of umbral and penumbral occlusion respectively.*

Scene Geometry

The performance of our algorithm depends in part on the number and light-view image plane extents of primitives composing the scene. This geometry is used only in the computation of umbral occlusion, and does not play a role in data structure construction or in the calculation of penumbral occlusion. During the computation of umbral occlusion, the geometry is projected into the light-view image plane and tested for overlap against the sample coordinates stored in the data structure. This point-in-primitive test is linear in the average depth complexity of the scene.

Silhouette Edges

Soft irregular shadow mapping generates a penumbra quad per silhouette edge, representing the expected light-view screen-space bounds of the penumbra cast by the edge. The computation of penumbral occlusion consists of testing these quads for overlap against the sample coordinates stored in the data structure, and calculating an area sample (Subsection 2.3) at each overlapped point. This computation is constant per sample per quad, and thus the total work is linear in the number of silhouette edges.

Area Light Radius

Though the performance of the penumbral occlusion computation is linear in the number of penumbra quads, the light-view screen-space extents of these quads are quadratic in the radius of the area light. A light with twice the radius of another, produces penumbra quads that occupy four times the area in the light-view image plane. This quadratic can be problematic in scenes with very large light sources and is an issue common to soft shadow algorithms based on shadow geometry or shadow mapping. However, the size of a penumbra (and thus extents of the penumbra quad) also depends on the distance of the occluding object from the light. Many common light sources are either large and distant (e.g. sun) or small and comparatively near occluding geometry (light fixture in a room).

3 Architectural Support

Our algorithm utilizes a non-uniform spatial acceleration structure in which the storage order of member elements is determined at run time. Specifically, the light-space coordinates of receiver points are sorted such that spatial locality is maximized (Figure 3). However, efficient construction and traversal of this structure requires high-performance scatter / gather memory operations, global atomic operations, and global synchronization. The Larrabee architecture due in 2009 or 2010 [Intel Corporation 2008] is one example of a processor with these features. We briefly review this design here.

3.1 The Larrabee Architecture

The Larrabee architecture is illustrated in Figure 5. Shown are a small number of fixed-function units for specialized operations like texture filtering, memory controllers, a large on-chip L2 cache, and multiple programmable cores, joined by a ring-based interconnect. The programmable core is derived from the Intel Pentium® line of CPUs. As such, it is multithreaded and supports 32 and 64-bit integer and floating scalar point arithmetic as well as the Intel Pentium x86 instruction set, but is different from a modern CPU in its short in-order instruction pipeline. Die area normally occupied by out-of-order control logic is instead devoted to 16-wide SIMD units. These units support 32-bit integer and 32 and 64-bit floating

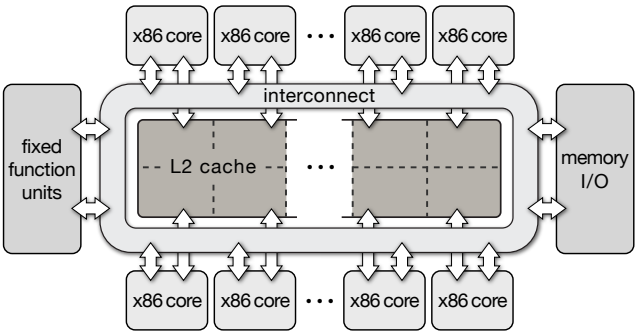


Figure 5: *The Larrabee architecture is a scalable, multi-core design. A large coherent cache structure and flexible interconnect support efficient global atomics and scatter memory operations. These in turn enable efficient construction and traversal of irregular data structures in which the storage order of member elements is determined at run time, as required by our algorithm.*

point arithmetic, conditional execution via vector element masking, simple vector load / store memory operations, and scatter / gather memory operations. The scatter / gather operations load or store up to 16 data values from non-contiguous addresses specified by a secondary source vector operand.

The Larrabee memory hierarchy is fully cache coherent and consists of a per-core L1, large L2 partitioned across cores, and off-chip memory accessible through on-chip controllers distributed around the ring interconnect. L1 is shared among threads of the same core. Data sharing between cores is enabled by hardware-assisted communication across L2 partitions. All vector memory instructions operate through cache.

4 Performance Evaluation

Comparing the performance of our algorithm with that of existing methods on other architectures is challenging. As of this writing, porting other algorithms to Larrabee is difficult due to the prototype nature of the software tool chain. Alternatively, published results for other algorithms could be normalized against Larrabee based on peak FLOPs or another metric. However, such scaling is dubious as Larrabee is a radically different architecture in the design and performance of its fixed-function units, programmable cores, ISA, interconnect, and memory hierarchy. Instead, we report competing results as published. As guidance on interpreting these results we note that the peak FLOPs of the Larrabee configuration simulated is within a factor of 2 of recent GPUs. Further, any bias in favor of Larrabee is substantially reduced by our use of a prototype compiler and software rasterization pipeline. Neither is fully optimized.

4.1 Simulation Environment

Our simulation infrastructure has three parts. The first is a reference implementation of hard and soft irregular shadow mapping written in C/C++. In addition to the routines specific to our algorithm, the code contains a functionally complete (but unoptimized) Z-buffer graphics pipeline capable of rendering a colored, shaded, and shadowed image from a scene specification. This code permits rapid evaluation of image quality and algorithmic correctness, and coarse performance estimation (e.g. operation counting, raw memory bandwidth measurements). Additionally, the data structure construction and hard and soft shadow kernels from this code have been hand vectorized and multithreaded using Larrabee intrinsics. The results presented below derive from the execution of this code on the Larrabee hardware simulator.

The second simulation component is the Larrabee software rasterization pipeline. Its key features include: multithreading with minimal locking, vectorization, and a "sort-middle" pipeline structure [Seiler, Carmean, et al. 2008]. When run on the Larrabee hardware simulator, this code provides insight into the performance of classical rasterization from the light, as required for the coarse visibility solution used by our algorithm. Since this code remains under development at Intel, no effort has been made to integrate it with the reference implementation of our algorithm.

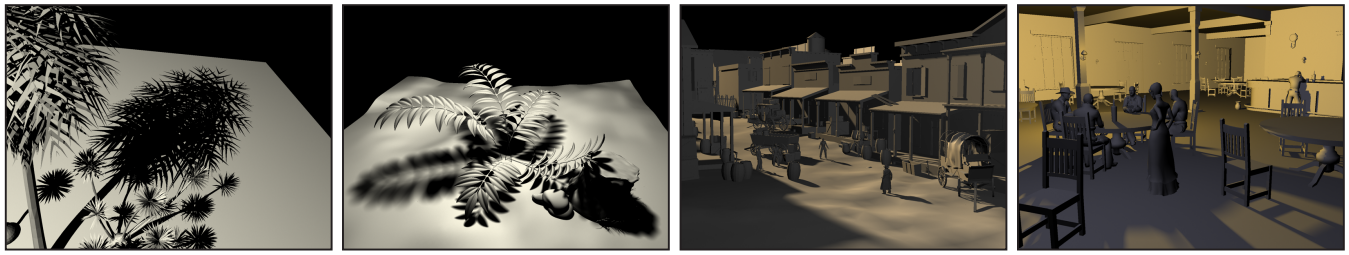
The third simulation part is the Larrabee hardware simulator itself. It descends from validated, cycle-accurate simulators used in designing Intel multi-core CPUs. Different chip configurations can be modeled. The number of cores, threads per core, and clock rate are adjustable, as are properties of the memory hierarchy. The core count and clock frequency of the actual Larrabee hardware have not yet been announced. Therefore, we simulate a conservative chip configuration with 24 cores at 1 GHz.

4.2 Results

Table 2 illustrates the hard and soft shadow performance of our algorithm for the scenes in Figure 6. The *palm* and *fern* scenes provide a basis for image quality and performance comparisons with several existing hard [Lefohn et al. 2007] and soft [Annen et al. 2008; Fernando 2005; Guennebaud et al. 2006; Schwarz and Stamminger 2007] shadow algorithms. These algorithms are state-of-the-art in image quality and performance, and are widely cited in the literature. Our test suite also includes scenes from a modern game. These scenes represent two environments with very different light-geometry relationships. The *street* scene is an exterior environment lit with a single light source positioned far from the geometry composing the scene, and is challenging for our algorithm due to the large number of silhouette edges. The *saloon* scene is an interior environment lit with a single light that sits within the eye-view frustum and is comparatively near the geometry composing the scene. This scene is a challenging case due to the size of the light relative to its average distance from the geometry, resulting in wide penumbrae as seen from the eye.

The times reported for our algorithm do not include the cost of eye-view color, depth, or shading. These costs are well understood and are expected to be comparatively small. Similarly, the cost of two geometry shaders and screen-space binning [Seiler, Carmean, et al. 2008] used in light-view rasterization are not included due to a dependence on features not present in the prototype software graphics pipeline used. The geometry shaders identify silhouette edges and generate a penumbra quad for each. The cost of both is expected to be small. For example, silhouette edge detection requires at most one cross product and one dot product per edge using adjacency information provided by DX10 [Blythe 2006], and only for triangles which pass face culling and frustum clipping.

In the hard shadow case, our algorithm achieves image quality comparable to adaptive shadow mapping [Lefohn et al. 2007], at real-time frame rates, using only a single light-view sample per eye-view pixel. In the soft shadow case, the performance of our algorithm is interactive and is within a factor of two of all but backprojection soft shadows [Guennebaud et al. 2006]. However, the image quality of our method is significantly higher than that of backprojection, percentage closer [Fernando 2005], bitmask [Schwarz and Stamminger 2007], and convolution [Annen et al. 2008] soft shadows, and is comparable to distribution ray tracing (Figure 7). Our algorithm computes shadow penumbrae directly from silhouette geometry at exactly the points in the scene visible from the eye. As a result, the images are free from artifacts due to undersampling a discretized representation of the scene (i.e. shadow



(a) Palm [Lefohn et al. 2007]. (b) Fern [Annen et al. 2008]. (c) Street (Call of Juarez). (d) Saloon (Call of Juarez).

Figure 6: *The four scenes evaluated in our performance analysis. The image quality and run time of our algorithm is compared against those of existing state-of-the-art approaches, using the first two scenes. The third and fourth are used to illustrate the performance of our algorithm in scenes from an actual game. These scenes are from Call of Juarez and are used with permission from Techland.*

Scene	Triangle Count	Silhouette Edges	Image Size	Algorithm	Data Structure Construction	Hard Shadow Kernel	Soft Shadow Kernel	Total Frame Rate
Palm	44K	--	1024 x 1024	[Lefohn et al. 2007]	--	--	--	40 fps
				[Johnson et al. 2009]	11.4 ms (69%)	3.8 ms (23%)	--	60 fps
Fern	212K	13K	800 x 600	[Annen et al. 2008]	--	--	--	23 fps
				Fernando. 2005]	--	--	--	18 fps
				[Guennebaud et al. 2006]	--	--	--	41 fps
				[Schwarz et al. 2007]	--	--	--	19 fps
				[Johnson et al. 2009]	6.4 ms (9%)	3.6 ms (5%)	55.0 ms (80%)	15 fps
Street	155K	29K	1600 x 1200	[Johnson et al. 2009]	14.2 ms (38%)	9.8 ms (26%)	11.6 ms (31%)	27 fps
Saloon	60K	8K	1600 x 1200	[Johnson et al. 2009]	11.9 ms (40%)	3.6 ms (12%)	13.6 ms (45%)	33 fps

Table 2: *Simulated performance for our algorithm (denoted in gray) on the Larrabee architecture with 24 cores at 1 GHz, for the scenes from Figure 6. Times for the major phases of the algorithm are reported individually, along with the percentage of the total render time represented by each. These percentages do not sum to 100% due to the additional cost of classical rasterization from the light. Total frame rates are also shown for our algorithm. Not included in these results are the cost of eye-view color, depth, or shading. Finally, published data for five other methods measured on a NVIDIA 8800 GTX part are shown for comparison. Though our soft shadow results are preliminary, they are within a factor of 2 of all but one other method. Moreover, our algorithm yields significantly higher image quality (Figure 7) than any other method.*

map). For example, convolution-based methods can omit umbrae in cases when an occluder is near a receiver (Figure 7c). This is due to the use of an average occluder depth at each receiver point, and the loss of frequency content resulting from the use of a low-precision, pre-filtered shadow map. In contrast, our algorithm simultaneously captures high-frequency shadow details and smooth, low-frequency penumbrae. Our method is approximate (Section 5), but the errors introduced are not widespread and are commonly imperceptible.

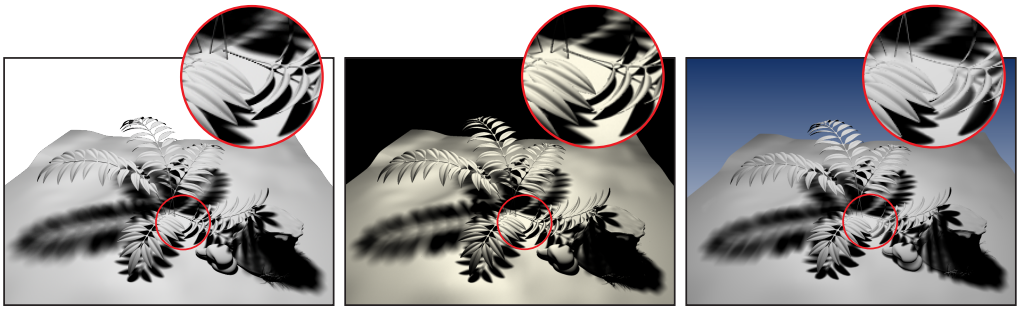
5 Approximations

As of this writing it is impractical to solve the visibility integral from Equation 2 analytically with high performance. To achieve real-time frame rates in dynamic scenes it is generally understood that approximations must be made. The space of possible approximations can be loosely bisected: those that may yield objectionable artifacts (e.g. light leaks, aliasing), and those that yield plausible but inaccurate results. Approximations of the first type typically allow the balance between image quality and performance to be tuned. Unfortunately, the computation required to achieve a given level of image quality depends on the spatial relationship between the eye, light, and geometry. In dynamic scenes this relationship is often unknown a priori, leading to unpredictable results. For this reason, we use approximations of the second type. These often yield more modest performance gains, but require no tuning and offer better error bounds. Our algorithm uses two: single point of projection and independent evaluation of occluders. The visual impact of each is summarized here and explored further by Johnson [2008].

5.1 Single Point of Projection

The visibility term of Equation 2 is often estimated by determining occlusion from one point (rather than many) on the surface of an area light. In actuality, the view of an occluder varies from point to point across the light surface tracing out a penumbra on a more distant object. The magnitude of this parallax effect can be estimated heuristically, as is done here. The value is proportional to the width of the penumbra cast by one object onto another, which is itself the ratio of the distances from the light to the occluder and the light to the object in shadow [Parker et al. 1998]. This estimate is widely used in place of computing visibility from multiple points on the light, as the error is not significant or objectionable even when the light is large relative to occluders.

However, there is a second effect due to parallax in which hidden silhouette edges become visible as seen from different points on the light [Assarsson et al. 2003]. The contribution of these edges can impact the shape of the penumbrae and the apparent size of the umbrae cast by the object, but cannot be estimated effectively from only a single point on the light. The resulting error is seen primarily



(a) Ray traced shadows. (b) Irregular shadow mapping. (c) [Annen et al. 2008].

Figure 7: *The image quality produced by ray tracing* (a)*, our algorithm* (b)*, and Annen et al.* (c) *compared. The inset highlights a region that is challenging for many algorithms to render accurately. A key feature of irregular shadow mapping is the high quality of the umbrae* and *penumbrae and the seamless transition between the two. The resulting images compare favorably with those produced by a ray tracer, simultaneously capturing high-frequency shadow details and smooth, low-frequency penumbrae. This result can be examined against that of backprojection, bitmask, and percentage closer soft shadows through Figure 7d - f of Annen et al. 2008. The image in* (c) *is generously provided by Thomas Annen and is used with permission.*

as umbrae which are reduced in width, with correspondingly larger penumbrae, and no high-frequency or other objectionable artifacts are introduced. The incidence of this error can be reduced by incorporating visibility information from multiple points on the light surface. For example, Assarsson et al. represent a single large area light as a collection of smaller lights [Assarsson et al. 2003]. This strategy is likewise compatible with our algorithm.

5.2 Independent Evaluation of Occluders

Many soft shadow algorithms evaluate the visibility term from Equation 2 per-occluder. However, these partial visibility terms are not independent, leading to a loss of information on the degree of overlap between occluders. Thus, the total accumulated occlusion is overestimated by an unknown amount C as shown in Equation 5. Here, $\tilde{V}(object_i, \mathbf{x})$ is 1 when a point on the light $\mathbf{x}$ is occluded by object i and 0 otherwise, and $\tilde{V}(object_i, \mathbf{x}) = 1 - V(object_i, \mathbf{x})$. It is expensive to accurately compute the value of C. Doing so requires clipping an incoming occluder to an arbitrarily-shaped silhouette representing the composition of prior occluders. Bitwise coverage masks can simplify this clipping, but at the cost of introducing aliasing into the penumbrae [Schwarz and Stamminger 2007]. In practice, algorithms typically approximate C. This value is bounded by 0 (i.e. no occluders overlap) and the sum of the area of all but the largest occluder (occluders maximally overlap). Some algorithms compute the average overlap between occluders [Assarsson et al. 2003] while others use the maximum [Haines 2001; Wyman and Hansen 2003; Chan and Durand 2003].

$$\int_{\mathbf{x}\in A} V(\mathbf{x})d\mathbf{x} = 1 - \left[\left(\sum_{i=1}^{N} \int_{\mathbf{x}\in A} \tilde{V}(object_i, \mathbf{x})d\mathbf{x}\right) - C\right] \qquad (5)$$

Our algorithm assumes $C = 0$ (i.e. no occluders overlap). This approximation is fast to compute and accurate in many common cases. It is approximate only when two or more partially-occluding surfaces overlap within the area sample (i.e. the actual value of C is greater than 0). Even here, no visually objectionable artifacts are introduced. Estimated penumbrae are continuous and visually pleasing, but are biased towards darker values. In the limit, a penumbra of zero width can result, but this extreme case requires a large number of silhouette edges to be (nearly) colinear after projection into the light-view image plane, and is infrequent.

6 Further Related Work

The approximations discussed above are part of a larger space of solutions for solving the visibility integral of Equation 2. This space can be structured according to methods for approximating the domain of integration, and methods for computing the integrand V. Interactive soft shadow algorithms employ a combination of methods, and differences in operation and performance, and image quality (Table 3) derive from the specific set used. Our soft shadow algorithm is distinguished primarily in the method used to determine V. The algorithm does not determine occlusion from the light in eye space using shadow geometry[1], or a discretized representation of the scene geometry (i.e. shadow map). Rather, receiver points are stored in a light-view spatial acceleration structure, and occlusion is computed at these locations in light space directly from the scene geometry.

6.1 Restricted Light Geometry

Observe that many sources of light in the real world (e.g. sun, incandescent bulbs, fluorescent fixtures) are simple shapes with symmetry, and the visual impact of light shape on penumbrae is subtle. As a result, area light sources are frequently defined as planar rectangles or discs. For rectangular lights, a simple linear parameterization can be used to obtain a uniform distribution of samples across the light surface [Laine and Aila 2005; Laine 2006; Agrawala et al. 2000]. Soft shadow algorithms that employ bilinear [Eisemann and Décoret 2006] or percentage closer filtering [Fernando 2005] to estimate penumbrae from umbral silhouettes, implicitly assume a square or rectangular area light due to their use of square or rectangular filter kernels. Disc-shaped lights mimic omnidirectional spheres of constant intensity [Brabec and Seidel 2002; Haines 2001], and produce a sinusoidal intensity falloff which can be approximated with a Bernstein cubic interpolation function [Parker et al. 1998; Wyman and Hansen 2003; Chan and Durand 2003]. The simplicity of the circle also enables fast analytic integration of occlusion resulting in high-quality penumbrae (Equation 3).

[1] Light-space penumbra quads are used to determine a coarse solution for V. However, the final visibility computation employs no shadow primitives.

	Plausible But Inaccurate			Noticeably Implausible		
Algorithm	**Inner Penumbra**	**Outer Penumbra**	**Accurate Overlap**	**No Aliasing**	**No Light Leaks**	**All Frequency**
[Agrawala et al. 2000]	✓	✓	✓		✓	✓
[Annen et al. 2008]	✓	✓		✓	✓	★
[Assarsson et al. 2003]	✓	✓		✓	✓	✓
[Bavoil et al. 2006]	✓	✓	✓			✓
[Brabec and Seidel 2002]	✓	✓			✓	✓
[Chan and Durand 2003]		✓		★★		✓
[Eisemann and Décoret 2006]	✓	✓		✓	✓	★
[Fernando 2005]	✓	✓	✓		✓	✓
[Guennebaud et al. 2007]	✓	✓			✓	✓
[Haines 2001]		✓		✓	✓	✓
[Johnson et al. 2009]	✓	✓		✓	✓	✓
[Ren et al. 2006]	✓	✓	✓	✓	✓	
[Schwarz and Stamminger 2007]	✓	✓	✓		✓	✓
[Soler and Sillion 1998]	✓	✓	✓			✓
[Wyman and Hansen 2003]		✓			✓	✓
[Zhou et al. 2005]	✓	✓	✓	✓	✓	

Table 3: *The visual qualities of several soft shadow algorithms is compared. Ours is marked in gray. Accuracy is loosely proportional to the number of check marks. The absence of a check in a* plausible but inaccurate *column indicates the algorithm yields plausible-looking but potentially incorrect penumbrae, while the absence of a check in an* noticeably implausible *column denotes the potential for objectionable artifacts. Penumbrae are divided into* inner *and* outer *regions by the hard silhouette of an object. The lack of either yields penumbrae which are misaligned and too narrow.* Accurate Overlap *denotes accurate computation of the area of overlap between occluders. Inaccuracies here commonly result in a bias towards darker penumbrae.* Alias-free *algorithms produce no sawtooth or banding patterns, or flickering in penumbrae during object-light motion.* Light leaks *appear as intensity discontinuities in otherwise unbroken regions of shadow.* All-frequency *algorithms do not impose artificial bounds on the frequency content of illumination via undersampling in frequency space, or via prefiltering of undersampled depth (i.e. shadow) maps. Many algorithms reduce the incidence of these artifacts using various means, but do not resolve the underlying issue(s). The accuracy of our algorithm is similar to that of Assarsson et al., which employs eye-space shadow geometry.* ★ *Frequency content is lost by prefiltering the shadow map.* ★★ *Texture mapping can re-introduce aliasing.*

6.2 Proxy Scene Geometry

Just as arbitrarily-shaped lights complicate the visibility integral, finely tessellated scene geometry can be similarly computationally challenging. An alternative is to compute visibility from simplified "proxy" geometry rather than the original scene geometry. Doing so can reduce the number of silhouette edges and reduce the complexity of the visibility computation per edge. For example, Ren et al. replace the occluding geometry with a hierarchy of spheres [Ren et al. 2006]. The size of each sphere approximates the scale of the local geometry replaced by that sphere. This strategy simplifies the spherical harmonic rotations and exponentiations required to compute the degree of occlusion at a point in the scene visible from the eye. However, finer shadow details are lost in regions with greater geometric complexity than that expressed by the proxy geometry.

A more common approach is to represent the scene geometry (explicitly or implicitly) as thin planar occluders parallel to the light source. Soler and Sillion explicitly decompose occluding geometry into planar elements and convolve these elements with the light to produce soft shadow textures [Soler and Sillion 1998]. Similarly, Eisemann and Décoret create planar proxy geometry by uniformly subdividing scene geometry by distance from the light, and projecting the contents of each partition into a plane parallel to the light [Eisemann and Décoret 2006]. These methods can achieve relatively high performance, but are susceptible to light leaks between planar elements.

Alternatively, several methods use a conventional shadow map as a discrete representation of occluder geometry [Schwarz and Stamminger 2007; Guennebaud et al. 2007; Bavoil et al. 2006]. Shadow map texels are backprojected onto the light, and the projected area is compared with the area of the light to produce a visibility estimate. This approach can achieve interactive [Schwarz and Stamminger 2007] and even real-time performance [Guennebaud et al. 2007] in simple scenes, but high-frequency geometric features (and thus finer shadow details) are lost due to the discretization. Moreover, a straightforward implementation of this technique can introduce holes in occluders resulting in light leaks.

6.3 Shadow Geometry

While proxy geometry is used as a stand-in for scene geometry, *shadow geometry* delimits regions of the scene inside the umbra or penumbra of an occluder. The penumbra cast by a silhouette edge forms a wedge. The faces of this wedge can be represented with shadow polygons. Assarsson et al. and Forest et al. compute penumbral occlusion at each receiver point corresponding to a pixel covered by these shadow primitives [Assarsson et al. 2003; Forest et al. 2008]. The result is combined with the occlusion computed from a second set of shadow primitives which delimit the bounds of umbrae. Haines utilizes shadow geometry composed of sheets and cones forming the faces and corners of penumbra wedges, shaded with a gradient mimicking the transition from light to shadow [Haines 2001]. This geometry is rendered to a texture which is then projected back onto the scene. Chan and Durand generate a set of polygons representing the extents of penumbrae as seen from the light [Chan and Durand 2003]. This geometry is rasterized into a light-view "smoothie buffer". Penumbral occlusion is computed by sampling this buffer, while umbral occlusion is computed by sampling a classical shadow map. In all three cases, penumbral regions are defined geometrically and can be rendered directly, yielding plausible looking shadows without aliasing or band limiting. However, the shape and structure of the shadow geometry often derives from assumptions about the light geometry, resulting in approximations particularly at the joints between neighboring shadow polygons [Chan and Durand 2003].

6.4 Precomputation and Band-Limiting

Part or all of the visibility computation can be moved offline and the results stored in a form readily accessible at runtime. However, precomputing visibility or illumination requires making assumptions about the spatial relationship between elements of the scene. These assumptions lead to restrictions on the motion of the camera or light [Agrawala et al. 2000] and / or on rigid body motion or deformation of scene geometry [Zhou et al. 2005].

Ren et al. precompute low-frequency visibility information using spherical harmonic exponentiation [Ren et al. 2006]. As spherical harmonic coefficients represent illumination in frequency space, band-limiting bounds the storage requirements of the precomputed data. This method yields interactive frame rates and handles the difficult case of self-shadowing in deformable models. It also works well for large area lights and environment maps, but shadows from small local light sources are unconvincing.

6.5 Resampling / Filtering Approximate Visibility

Alternatively, the visibility computation can be minimized via resampling and / or filtering an approximate solution to achieve often plausible looking (but potentially inaccurate) penumbrae. For example, the hard shadow silhouettes found via classical shadow mapping, can be blurred using bilinear filtering. The performance of these methods benefits from hardware-accelerated filtering and from the relative inexpense of the approximate visibility function.

Soler and Sillion use an explicit convolution kernel to produce penumbrae from an image of the occluders as seen from the light. The resulting soft shadow textures are then re-projected back onto scene geometry [Soler and Sillion 1998]. The constant-width filter used is accurate only for planar scene geometry parallel to the light. Fernando uses a variable-width kernel with percentage closer filtering to blur hard shadow boundaries in a classical shadow map on a per-sample basis [Fernando 2005]. The size of the filter is proportional to the penumbra width and is estimated as the ratio of the distances from the light to the occluder and the light to the shadowed geometry. The algorithm is simple to implement but is bandwidth intensive since the number of shadow map texels retrieved from memory grows as the square of the kernel width. Further, the resulting penumbrae can exhibit aliasing artifacts since the shadow map is a discretized representation of the occluding geometry. Mipmapping and summed area tables [Lauritzen 2007] can be used to prefilter shadow maps to avoid this aliasing, but doing so results in the loss of high frequency shadow information [Annen et al. 2008; Eisemann and Décoret 2006].

6.6 Closely Related Work

Concurrent with our work, Sintorn et al. have developed a similar algorithm [Sintorn et al. 2008]. As in our approach, the receiver points are stored in a light-view spatial acceleration structure. Their structure is a 2D variation of the 3D perspective-correct grid used here. Further, the method for determining umbral occlusion, and the use of shadow geometry to estimate the set of receiver points affected by a silhouette, are also similar. This similarity is not surprising. The grid-based data structure and use of light-space shadow geometry in particular are logical extensions of ideas we introduced in hard irregular shadow mapping [Johnson et al. 2004].

The two methods differ mainly in the computation of penumbral occlusion and in the optimizations used. For example, Sintorn et al. estimate the integral of the visibility function in Equation 2, by evaluating V at several (i.e. 128 - 1024) points on the light surface. In our algorithm, we evaluate this integral analytically, avoiding a potential source of aliasing. Moreover, our algorithm reduces overdraw associated with the penumbra quads via the optimizations in Subsection 2.5. These optimizations contribute to consistently higher performance in scenes of comparable geometric complexity: 15 - 33 fps at image sizes of between 800×600 and 1600×1200 (our algorithm) versus 3 - 7 fps at 512×512 [Sintorn et al. 2008].

7 Conclusion

Though extensively studied, there remains a gap in the solution space for real-time soft shadow rendering. Specifically, no existing algorithm has been demonstrated to achieve both high image quality *and* good performance in dynamic scenes from modern games. Existing approaches vary in the visibility function V and in the method used to estimate the integral of V over the light surface, but image quality and performance are primarily influenced by the former. Most soft shadow algorithms determine V from shadow geometry or a shadow map. Methods which use *eye-space* shadow geometry can produce penumbrae comparable to a beam tracer, but the performance is constrained by overdraw. Alternatively, algorithms based on shadow mapping can achieve high performance, but are subject to aliasing, incorrect self shadowing, and light leaks. The incidence of these artifacts can be reduced through prefiltering, resampling, and oversampling, but such solutions do not address the root causes: misalignment of the eye and light-view sample patterns and loss of geometric detail due to the discretization.

Our algorithm is distinct from existing methods in two ways. First, receiver point coordinates are stored in an explicit light-space spatial acceleration structure. Second, the occlusion calculation uses both shadow geometry and a shadow map. These properties allow umbral and penumbral occlusion to be computed efficiently and with few if any discernible artifacts. Performance loss from overdraw is reduced through the use of light-space rather than eye-space shadow geometry ("penumbra quads"), while artifacts endemic to classical shadow mapping are avoided by aligning the eye and light-view sample patterns, and by analytically area-sampling occluding geometry. Our method incurs overhead from the per-frame construction of a spatial acceleration structure, and construction and rasterization of shadow primitives. In spite of this overhead, soft irregular shadow mapping is comparable in performance to all but one other recent approach, with substantially higher image quality than any other known method in the real-time performance regime. More broadly, the technique provides an example of using real-time graphics hardware to build (per frame) and traverse irregular data structures, enabling new kinds of real-time graphics algorithms based on carefully-directed sampling.

Acknowledgements

The authors would like to thank the following individuals and organizations who contributed to this work. Jeff Boody developed an early Larrabee implementation of this algorithm that guided the work presented here. The image seen in Figure 7c was generously provided by Thomas Annen. We are grateful to Techland for granting us permission to use scenes from Call of Juarez in our analysis. The portion of this work undertaken at the University of Texas at Austin was funded in part by the Intel Corporation and NSF CAREER award #CCF-0546236.

References

AGRAWALA, M., RAMAMOORTHI, R., HEIRICH, A., AND MOLL, L. 2000. Efficient image-based methods for rendering soft shadows. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive*

Techniques, ACM Press / Addison-Wesley Publishing Co., New York, NY, 375–384.

AILA, T., AND LAINE, S. 2004. Alias-free shadow maps. In *Proceedings of the Eurographics Symposium on Rendering 2004*, Eurographics, 161–166.

ANNEN, T., DONG, Z., MERTENS, T., BEKAERT, P., SEIDEL, H.-P., AND KAUTZ, J. 2008. Real-time, all-frequency shadows in dynamic scenes. In *SIGGRAPH '08: ACM SIGGRAPH 2008 Papers*, ACM Press, New York, NY.

ASSARSSON, U., DOUGHERTY, M., MOUNIER, M., AND AKENINE-MÖLLER, T. 2003. An optimized soft shadow volume algorithm with real-time performance. In *HWWS '03: Proceedings of the ACM SIGGRAPH / EUROGRAPHICS Conference on Graphics Hardware*, Eurographics, Aire-la-Ville, Switzerland, Switzerland, 33–40.

BAVOIL, L., CALLAHAN, S. P., AND SILVA, C. T. 2006. Robust soft shadow mapping with depth peeling. SCI Institute Technical Report UUSCI-2006-028, University of Utah.

BLYTHE, D. 2006. The Direct3D 10 system. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, ACM Press, New York, NY, 724–734.

BRABEC, S., AND SEIDEL, H.-P. 2002. Single sample soft shadows using depth maps. In *Graphics Interface (GI 2002 Proceedings)*, 219–228.

CHAN, E., AND DURAND, F. 2003. Rendering fake soft shadows with smoothies. In *EGRW '03: Proceedings of the 14th Eurographics Workshop on Rendering*, Eurographics, Aire-la-Ville, Switzerland, Switzerland, 208–218.

EISEMANN, E., AND DÉCORET, X. 2006. Plausible image based soft shadows using occlusion textures. In *Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing, 19 (SIBGRAPI)*, IEEE Computer Society, 155–162. Oliveira Neto, Manuel Menezes deCarceroni and Rodrigo Lima (Eds.).

FERNANDO, R. 2005. Percentage-closer soft shadows. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, ACM Press, New York, NY, 35.

FOREST, V., BARTHE, L., AND PAULIN, M. 2008. Accurate shadows by depth complexity sampling. In *Proceedings of Eurographics 2008*, Eurographics, 663–674. Computer Graphics Forum Volume 27, Number 2.

GUENNEBAUD, G., BARTHE, L., AND PAULIN, M. 2006. Real-time soft shadow mapping by backprojection. In *Eurographics Symposium on Rendering*, Eurographics, 227–234.

GUENNEBAUD, G., BARTHE, L., AND PAULIN, M. 2007. High-quality adaptive soft shadow mapping. *Computer Graphics Forum 26*, 3, 525–533.

HAINES, E. 2001. Soft planar shadows using plateaus. *Journal of Graphics Tools 6*, 1, 19–27.

INTEL CORPORATION, 2008. First details on a future Intel design codenamed "Larrabee", August. http://www.intel.com/pressroom/archive/releases/20080804fact.htm.

JOHNSON, G. S., MARK, W. R., AND BURNS, C. A. 2004. The irregular z-buffer and its application to shadow mapping. Technical Report TR-04-09, Department of Computer Sciences, The University of Texas at Austin, April.

JOHNSON, G. S., LEE, J., BURNS, C. A., AND MARK, W. R. 2005. The irregular z-buffer: Hardware acceleration for irregular data structures. *ACM Transactions on Graphics 24*, 4, 1462–1482.

JOHNSON, G. S. 2008. *A Hybrid Real-Time Visible Surface Solution for Rays With a Common Origin and Arbitrary Directions*. PhD thesis, Department of Computer Sciences, University of Texas at Austin.

LAINE, S., AND AILA, T. 2005. Hierarchical penumbra casting. *Computer Graphics Forum 24*, 3, 313–322.

LAINE, S., AILA, T., ASSARSSON, U., LEHTINEN, J., AND AKENINE-MÖLLER, T. 2005. Soft shadow volumes for ray tracing. *ACM Transactions on Graphics 24*, 3, 1156–1165.

LAINE, S. 2006. *An Incremental Shaft Subdivision Algorithm for Computing Shadows and Visibility*. Master's thesis, Helsinki University of Technology.

LAURITZEN, A. 2007. *Summed-Area Variance Shadow Maps*. Addison-Wesley, July, ch. 8, 157–181.

LEFOHN, A. E., SENGUPTA, S., AND OWENS, J. D. 2007. Resolution-matched shadow maps. *ACM Transactions on Graphics 26*, 4, 20–42.

PARKER, S., SHIRLEY, P., AND SMITS, B. 1998. Single sample soft shadows. Tech. Rep. UCS-98-019, Computer Science Department, University of Utah, October.

REN, Z., WANG, R., SNYDER, J., ZHOU, K., LIU, X., SUN, B., SLOAN, P.-P., BAO, H., PENG, Q., AND GUO, B. 2006. Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. *ACM Transactions on Graphics 25*, 3, 977–986.

SCHWARZ, M., AND STAMMINGER, M. 2007. Bitmask soft shadows. *Computer Graphics Forum 26*, 3 (September), 515–524.

SEILER, L., CARMEAN, D., SPRANGLE, E., FORSYTH, T., ABRASH, M., DUBEY, P., HANRAHAN, P., JUNKINS, S., LAKE, A., AND SUGERMAN, J. 2008. Larrabee: A many-core x86 architecture for visual computing. In *SIGGRAPH '08: ACM SIGGRAPH 2008 Papers*, ACM Press, New York, NY.

SINTORN, E., EISEMANN, E., AND ASSARSSON, U. 2008. Sample based visibility for soft shadows using alias-free shadow maps. In *Proceedings of the Eurographics Symposium on Rendering 2008*, Eurographics, 1285–1292. Computer Graphics Forum Volume 27, Number 4.

SOLER, C., AND SILLION, F. X. 1998. Fast calculation of soft shadow textures using convolution. In *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press, New York, NY, 321–332.

WYMAN, C., AND HANSEN, C. 2003. Penumbra maps: Approximate soft shadows in real-time. In *EGRW '03: Proceedings of the 14th Eurographics Workshop on Rendering*, Eurographics, 202–207.

ZHOU, K., HU, Y., LIN, S., GUO, B., AND SHUM, H.-Y. 2005. Precomputed shadow fields for dynamic scenes. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM Press, New York, NY, 1196–1201.