Eurographics Symposium on Rendering 2010
Jason Lawrence and Marc Stamminger
(Guest Editors)

Volume 29 (2010), Number 4

Interactive, Multiresolution Image-Space Rendering for Dynamic Area Lighting

Greg Nichols, Rajeev Penmatsa, and Chris Wyman

University of Iowa, USA

Abstract

Area lights add tremendous realism, but rendering them interactively proves challenging. Integrating visibility is costly, even with current shadowing techniques, and existing methods frequently ignore illumination variations at unoccluded points due to changing radiance over the light's surface. We extend recent image-space work that reduces costs by gathering illumination in a multiresolution fashion, rendering varying frequencies at corresponding resolutions. To compute visibility, we eschew shadow maps and instead rely on a coarse screen-space voxelization, which effectively provides a cheap layered depth image for binary visibility queries via ray marching. Our technique requires no precomputation and runs at interactive rates, allowing scenes with large area lights, including dynamic content such as video screens.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—

1. Introduction

Interactive rendering has long relied on illumination from infinitesimal points, where efficient lighting and visibility are well understood. Illumination from more complex area sources requires integrating both visibility and radiance over the light's surface. Given framerate constraints in interactive applications, per-pixel computation of such integrals is infeasible. Current solutions include precomputing light transfer [SKS02], discretely sampling the light source [Kel97], and caching samples in object-space data structures [KG09] that require updates for dynamic geometry.

Researchers have developed numerous algorithms for soft shadows that integrate light over an area light [HLHS03], but most break with larger lights and few allow radiance fluctuations over the emitter. A few sophisticated algorithms account for radiance variations in penumbral regions using 4D lookup tables [AAM03] or summed area tables [GBP06], but they generally treat unoccluded radiance as coming from a single-colored emitter, using the average source radiance.

This paper presents a novel image-space technique for interactive lighting from area lights that draws inspiration from recent global illumination research. In particular, recent techniques demonstrate that rendering of slowly varying

Figure 1: *Our multiresolution image-space approach enables illumination from dynamic area lights. (Top) Compare with and without area lighting. (Bottom) No precomputation is needed, allowing fully dynamic geometry. Image-space computations decouple algorithmic and geometric complexity, maintaining interactivity with over 500,000 polygons.*

lighting can be accelerated via multiresolution image-space computations [NSW09], and can rely on crude visibility approximations [RGK*08].

We present **three main contributions**:

1. A multiresolution approach that gathers illumination from dynamic area lights without visibility. This runs in real time for diffuse and non-diffuse BRDFs.
2. A multiresolution algorithm using screen-space voxelization to quickly compute per-pixel area light visibility.
3. An incremental refinement that considers both illumination and visibility variations when choosing an appropriate resolution for rendering illumination.

Since we avoid precomputation, our work handles dynamic lights, viewpoint, and geometry (see Figure 1). Image-space computations allow our algorithm to scale with visual complexity rather than geometric complexity. Additionally, iterative refinement provides a clear quality/performance tradeoff: simply stop sooner for improved performance. However, our discrete geometric representation can introduce aliasing.

2. Previous Work

Researchers have long developed techniques to interactively compute light visibility. Until recently, these focused on visibility from point lights and approximate methods for uniform area lights. Due to space constraints, we focus on the most relevant of these techniques, and discuss recent work for area lights and global illumination with characteristics similar to our work.

2.1. Single Shadow Map Visibility Approximations

Standard shadow maps [Wil78] quickly compute visibility from point lights. The image-space computations scale well with increased scene complexity and enable efficient post processing [RSC87, Fer05] for approximating more complex effects, such as soft shadows. Most soft shadow map algorithms [HLHS03] rely on crude, but plausible, visibility approximations and ignore radiance variations over the light.

Recent work aims to improve accuracy. Backprojection soft shadows [GBP06] replace each shadow map texel by a micropolygon, allowing computation of analytical per-patch visibility. Disjoint micropolygons introduce light leakage and overshadowing, but more sophisticated meshing reduces the problem [SS07]. Backprojection creates a per-fragment list of potential occluders; to avoid searching all micropolygons, hierarchical shadow maps [GBP06, SS07] accelerate identification. While these hierarchies break for large lights, subdividing the lights [YFGL09] improves quality.

While backprojection soft shadow maps accurately compute visibility from area lights, they largely ignore variations due to multicolored lights. A few techniques use 4D lookup tables [AAM03] or summed area tables [GBP06] to represent varying radiance. Bitmask soft shadows [SS07] use a bitmask approach that inspires our work. However, none of these algorithms consider how radiance variations affect unoccluded fragments.

One common problem for all single map algorithms stems from use of a single silhouette edge to approximate occlusions. While sufficient for small lights, it presents challenges for large lights, especially as occluders approach the light.

2.2. Multiple Shadow Map Visibility Approximations

Using many shadow maps avoids this problem, at the cost of additional render passes. Heckbert and Herf [HH97] suggest rendering shadow maps from many samples. While this converges for increasing sampling density, rendering hundreds of shadow maps proves a serious bottleneck.

Annen et al. [ADM*08] split lights into multiple samples and use fast per-sample soft shadowing [AMB*07], though they handle an order of magnitude fewer samples than our work due to memory limits and shadow map construction costs. Coherent shadow maps [RGKM07] precompute and compress shadow map per-object visibility, enabling moving geometry and complex lighting if object convex hulls do not overlap [RGKS08]. However, this requires substantial precomputation and memory overhead.

An ingenious way to accelerate multi-shadow map illumination uses crude, imperfect maps [RGK*08] and relies on averaging over thousands of shadow lookups to remove low-resolution shadow map artifacts. To efficiently create imperfect shadow maps, a preprocess densely point-samples the scene. While relatively inexpensive, this preprocess limits dynamic scenes to deformations that need no resampling. Recent work [REG*09] shows point primitives also work with more complex illumination and material properties, albeit at significantly increased cost.

2.3. Multi-Layer Visibility Approximations

Shadow maps help accelerate many complex illumination problems, but extending the depth buffer to contain 3D data, e.g., using layered depth images [SGwHS98], provides an interesting alternative. Agrawala et al. [ARHM00] precompute visibility for interactive rendering, and Im et al. [IHK05] achieve near-interactive soft shadowing with layered depth.

Multi-layer techniques often use depth peeling [Eve01] to achieve interactivity, and recent work simultaneously peels many layers [LHLW09]. One visibility approximation using layer depth is screen-space ambient occlusion [Mit07, BS09]. While it interactively captures visibility from large area lights, it poorly handles highly directional lighting.

We use another layered visibility approximation: image-space scene voxelization [DCB*04, ED06]. Voxelization discretizes the volume enclosed by the view frustum, with

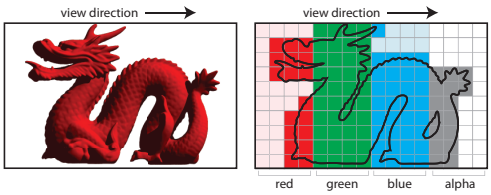


Figure 2: *An example of voxelization, viewed from the side. Each quanta of depth, represented by a single bit in the color buffer, identifies whether geometry is present in the voxel.*

each framebuffer pixel representing multiple single-bit voxels (see Figure 2). Voxelization occurs in a single render pass, requiring under a millisecond even for relatively complex scenes. This representation proves useful for numerous applications [ED08], such as refraction, translucency, and collision detection. We observe that screen-space voxelization efficiently creates a perspective grid [HM08], suitable as a simple ray acceleration structure (similar to Woo [WA90]).

2.4. Interactive Global Illumination

While our work computes only *direct* light from area sources, we drew inspiration from interactive global illumination research that explores similar techniques. Numerous precomputed radiance transport algorithms [SKS02, NRH03] use area or environmental lights, but also impose strict limits on scene dynamism. Recent work [SR09] obtains interactive scene relighting by relaxing limits on viewpoint and light movement; however, substantial precomputation still precludes dynamic geometry.

Instant radiosity [Kel97] approximates indirect light as a sum of direct illumination from point lights. While initially unsuitable for real time rendering, Dachsbacher and Stamminger [DS05] developed an interactive shadow map-based approximation, which also works with imperfect shadow maps [RGK*08].

Naive reflective shadow mapping [DS05] has fill-rate limitations, due to the cost of gathering illumination at all visible fragments. Light propagation volumes [KD10] reduce costs by projecting lights to an spherical harmonic basis and using only these SH coefficients during rendering, though this limits the illumination frequencies achievable. Nichols et al. [NSW09] introduce a multiresolution gather that, in effect, acts as an image-space illumination cache. This reduces costs in regions with slowly changing illumination. Direct area illumination exhibits similar behavior, and we adapt their image-space hierarchy. Object-space illumination caches, common in radiance caching [KG09] may also work; however, we prefer to avoid object-space data structures requiring updates for dynamic geometry.

3. Multiresolution Light from Area Sources

Direct illumination from an area light requires integrating over the light for each pixel, combining light contributions, material reflectance, and any occlusions. Using an area formulation of the rendering equation [Kaj86], we write this:

$$L(\mathbf{x},\vec{\omega}) = \int_{\mathbf{y}\in S} f_r(\vec{\omega},\mathbf{x}\to\mathbf{y})I(\mathbf{y})V(\mathbf{x}\to\mathbf{y})G(\mathbf{x}\to\mathbf{y})dA, \quad (1)$$

where $\mathbf{x}$ is the point to shade, $\vec{\omega}$ the viewing direction, S the light surface, I the light intensity, $\mathbf{x}\to\mathbf{y}$ the vector from $\mathbf{x}$ to the light sample $\mathbf{y}$, V the binary visibility between $\mathbf{x}$ and $\mathbf{y}$, and $G(\mathbf{x}\to\mathbf{y})$ the geometry term:

$$G(\mathbf{x}\to\mathbf{y}) = \frac{\cos(\mathbf{x}\to\mathbf{y},\vec{N}_\mathbf{x})\cos(\mathbf{y}\to\mathbf{x},\vec{N}_\mathbf{y})}{||\mathbf{x}\to\mathbf{y}||^2}.$$

$\vec{N}_\mathbf{x}$ and $\vec{N}_\mathbf{y}$ are the surface normals at $\mathbf{x}$ and $\mathbf{y}$. In an interactive context, sampling provides the only feasible solution, so we rewrite Eq. 1 as a sum over discrete light samples:

$$L(\mathbf{x},\vec{\omega}) \approx \sum_{i=0}^{N} f_r(\vec{\omega},\mathbf{x}\to\mathbf{y}_i)I(\mathbf{y}_i)V(\mathbf{x}\to\mathbf{y}_i)G(\mathbf{x}\to\mathbf{y}_i)A(\mathbf{y}_i). \quad (2)$$

Here, light samples $\mathbf{y}_i$ can be thought of as virtual point lights, allowing algorithms akin to recent global illumination research (e.g., [Kel97, DS05]).

3.1. Overview

We have three main contributions. Section 3.2 ignores the visibility in Eq. 2 and adapts recent multiresolution image-space rendering [NSW09] to interactively gather from virtual point lights (VPLs) on the area light. While this retargeting is relatively straightforward, area illumination enables opportunities for improved performance and, unlike Nichols et al. [NSW09], allows rendering of non-diffuse materials. Section 3.3 considers *only* the visibility from Eq. 2 and applies screen-space voxelization [ED08] to coarsely represent the scene. We introduce an incremental, multiresolution ray marching approach that interactively approximates light visibility. Section 3.4 combines these into a multiresolution approach to interactively render area illumination, allowing both visibility and radiance variations over the light.

3.2. Multiresolution Gathering Without Visibility

Illumination from area sources generally changes smoothly, giving low frequency lighting where coarser than per-pixel illumination sampling should suffice. One could apply object-space caching [KG09], but this requires data structures that incur expensive updates in dynamic environments. We instead build on recent multiresolution image-space work [NW09], akin to an image-space cache, with a cheap per-frame build cost that allows dynamic geometry.

First we simplify area illumination in Eq. 2 by ignoring visibility (setting $V = 1$) and assuming a diffuse material with albedo ρ, giving the following equation:

$$L(\mathbf{x},\vec{\omega}) \approx \frac{\rho}{\pi}\sum_{i=0}^{N} \frac{I(\mathbf{y}_i)A(\mathbf{y}_i)\cos(\mathbf{x}\to\mathbf{y},\vec{N}_\mathbf{x})\cos(\mathbf{y}\to\mathbf{x},\vec{N}_\mathbf{y})}{||\mathbf{x}\to\mathbf{y}||^2}.$$

This per-sample contribution resembles the per-VPL contribution used by Nichols and Wyman [NW09], suggesting a similar multiresolution approach also works for direct area lighting. Their approach works as follows:

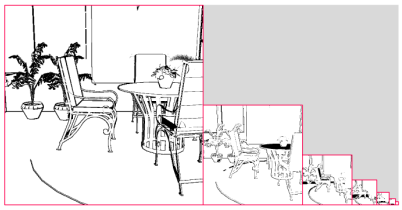
1. Render an eye-space G-buffer [ST90];
2. Compute multi-scale image-space depth and normal variations by creating a mipmap structure over the G-buffer;
3. In parallel, set stencil bits in a multiresolution buffer at the texels where illumination will be gathered (where depth & normal variations exceed a threshold);
4. In parallel, gather illumination only at specified texels;
5. Combine and upsample to full-screen resolution.

We observe that illumination discontinuities for area sources occur at locations similar to indirect light: at image-space depth and normal boundaries. We use the depth and normal metrics from prior work, computing per-pixel depth derivatives (in Step 2) as $\sqrt{(dz/dx)^2+(dz/dy)^2}$, and creating a max-mipmap. We use a similar computation for normals, performed on a per-component basis; normal discontinuities occur when at least one component of the normal changes significantly.

Step 3 stencils all resolutions in parallel, via a full-screen pass over a flattened multiresolution buffer:

```
for all (fragments f ∈ quad) do
  if ( ∀i, f ∉ MipmapLevel( i ) ) then
    continue; // Fragment not actually in multires buffer
  i ← CurrentMipmapLevel( f );
  if ( HasDepthOrNormalDiscontinuity( f, i ) ) then
    continue; // Patch not valid (needs subdivision)
  if ( NoDiscontinuity( f, i+1 ) ) then
    continue; // Coarser patch did not need subdivision
  SetStencil( f );
```

The resulting stencil has bits set for fragments to be sampled:



We gather illumination from VPLs on the area light, using the stencil to cull unneeded computations. Figure 3 shows the resulting area lighting, with coarse samples in slowly varying regions and denser samples near discontinuities.

For area lighting, a priori knowledge of the source location enables multiresolution improvements infeasible for global illumination. First, gathering light on emitters is unnecessary; we stencil them out, resulting in dramatic performance improvements in scenes with large light sources.

Second, while indirect light often arrives incoherently, direct light often has a preferred direction. For point sources, a cheap $\vec{N}\cdot\vec{L}<0$ identifies unlit fragments. For area lights, we

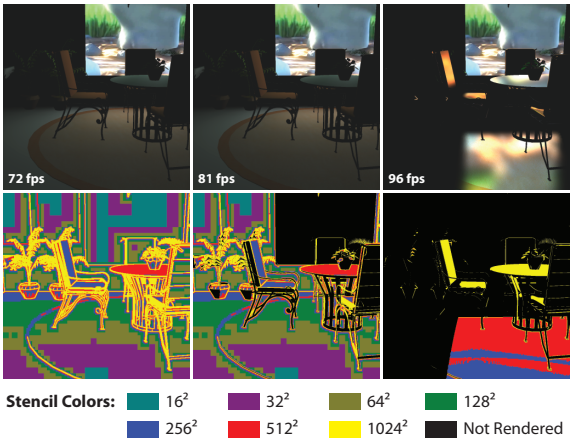


Figure 3: *The indoor garden with diffuse (left, center) and Phong (right) materials. We show the final rendering above a pseudocolor image depicting the resolution we rendered each region. Our naive approach (left) gathers illumination everywhere. Discarding fragments on or facing away from the light (center) improves performance. Phong surfaces (right) are refined extensively or trivially discarded.*

detect geometry facing away from the light by testing $\vec{N}\cdot\vec{L}_j$ for vectors $\vec{L}_j$ towards the corners (or bounding box) of the light. If $\vec{N}\cdot\vec{L}_j<0$ for all j, we trivially discard the fragment. Additionally, for one-sided lights, such as television screens, we also discard fragments behind the source.

3.2.1. Non-Diffuse Materials

Non-diffuse materials require keeping the BRDF inside the sum in Eq. 2. This complicates multiresolution rendering, as shiny materials introduce high frequencies not considered by image-space depth and normal metrics. Specular surfaces need additional refinement to capture these frequencies.

Seemingly, further refinement degrades performance by adding additional samples. But while shiny materials reflect high frequencies near the reflection direction, they often reflect little light in others. This suggests traditional importance sampling techniques directly translate into material-specific refinement metrics for multiresolution rendering.

Consider a Phong BRDF without a diffuse component, reflecting light according to the term $(\vec{R}\cdot\vec{L})^n$. For even low shininess n, this reduces off-specular illumination significantly. We discard fragments where the Phong lobe misses the light. We identify these fragments by locating where the reflection vector $\vec{R}$ misses the light, and checking that the Phong lobe does not overlap the source (by guaranteeing $(\vec{R}\cdot\vec{L}_j)^n$ falls below a threshold for all j).

Frequencies in a Phong reflection depend on lobe size. Large lobes give blurry reflections representable via coarser sampling. When we create our depth and normal mipmaps, we also compute the cross-sectional Phong lobe size at the

light. Image-space sample spacing must be denser than the lobe width, which we add as a Phong-specific refinement metric. We separate diffuse and Phong components into separate gather passes (see Figure 3), which should work for many other BRDFs (e.g., Ashikhmin-Shirley [AS00]).

As we use 256 VPL samples, highly specular materials can show reflections of individual VPLs. We avoid this using a simplistic adaptive strategy: when a specular lobe includes too few VPLs (we used a threshold of 9) we directly sample the light texture 25 times inside the lobe using a mipmap level dependent on the lobe size when it hits the light. We sample on a regular 5×5 grid centered on the reflection direction, with sample spacing dependent on lobe width.

3.3. Voxel-Space Visibility Computations

Interactive rendering algorithms frequently rely on depth maps to approximate the visibility in Eq. 2. Creation of a z-buffer as a rasterization byproduct encourages this ubiquity, and algorithmic improvements naturally start from prior shadow mapping work. However, creation of multi-layer buffers [LHLW09] or thousands of shadow maps [RGK*08] is costly, and accessing the results strains bandwidth and texture cache without providing true 3D data.

Fortunately, screen space voxelization [DCB*04, ED06] provides a compelling alternative. Voxelization is efficient, taking under a millisecond even in complex scenes, and produces a 3D structure where each bit represents the presence or absence of geometry in a single voxel (see Figure 2). Yet, there is a clear tradeoff. 32-bit z-buffers uniquely store 2^{32} depth values, whereas 32-bit voxelizations uniquely represent just 32. But the voxelization can simultaneously store 32 depths while the z-buffer stores just one. We found a 128-bit voxelization worked well for our scenes, though additional render targets could extend this to 2048 bits.

3.3.1. Naive Voxel-Space Visibility

Inspired by the visibility quality attained using imperfect shadow maps [RGK*08], we suggest that image-space voxelization, a similarly crude approximation, can achieve similar quality. Instead of creating thousands of shadow maps and querying each, we propose marching rays through a coarse scene voxelization.

Our naive query spawns a ray traversal between every light sample at each pixel (see Figure 4). Ideally, we would perform a 3D DDA for each traversal, checking all intermediate voxels for intersections. This performed poorly on our GPU, due to varying numbers of loop iterations, so we instead sample the ray uniformly between each fragment and VPL. While a sparse sampling may miss occlusions from thin geometry, we partially compensate for missed occlusions by checking multiple bits in texels fetched from the voxel buffer. This essentially thickens the ray.

Figure 4: *Compare (left) brute force visibility and (right) our incremental refinement. Brute force computations ray march from each pixel to every VPL, whereas incremental computations reuse results from coarser visibility passes.*

3.3.2. Incremental Voxel-Space Visibility

While we need per-pixel visibility to each VPL, this visibility, like illumination, changes slowly throughout our image. We leverage this observation, introducing two algorithms that incrementally refine visibility from sparser samples.

Consider shadows from a point light. By definition, these are coherent except at shadow boundaries. A coarse shadow sampling is correct at low resolution, except near these boundaries. Clearly, this also holds for any single VPL. Instead of computing visibility to each VPL at every pixel, we first sample on a coarser pixel grid (e.g., 64^2) and incrementally refine only near visibility discontinuities (see Figure 5).

When refining, we reuse a VPL's visibility when coarse samples in a 3×3 region agree, only spawning a new visibility ray when inconsistencies arise. When using hundreds of VPLs, usually fewer than 5% vary in any region, significantly reducing the rays needed. To this end, our visibility passes output a binary bitstream (as in bitmask soft shadows [SS07]) with one bit per VPL, allowing identification of regions needing additional queries.

While our incremental approach can converge to a correct solution, this is rarely necessary. For instance, imperfect shadow maps have inaccurate per-VPL visibility that gets averaged out, and ambient occlusion algorithms often sample coarsely and then apply a blur. Given our multiresolution approach from Section 3.2 essentially blurs illumination, we seek to avoid computing per-pixel visibility prior to blurring.

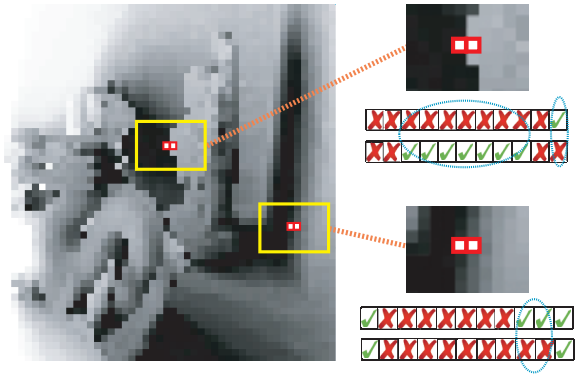


Figure 5: *Each pixel stores a bitstream of visible and occluded VPLs. When refining, we only resample lights whose visibility changes between neighboring pixels. Even at significant discontinuities (top), only a subset of VPL visibilities change. Often visibility changes more slowly (bottom); here only a few VPLs need resampling during refinement.*

Instead of refining visibility everywhere, we only generate denser samples near discontinuities. It turns out that most visibility discontinuities occur at image-space depth and normal boundaries, so we can use the same refinement criteria from Section 3.2. However, these metrics miss discontinuities caused by contact shadows. To detect these discontinuities, we add a bit-counting metric. When enough VPLs change visibility in an image-space neighborhood, refinement is needed. Surprisingly, this threshold can be quite high: in our scenes, we captured light discontinuities by refining only when at least half the VPLs (i.e., 128) were partially visible in a neighborhood. We compare naive voxel-space visibility with a combination of both our incremental improvements in Figure 4.

Given the gradual visibility variations in most scenes, we found interleaved sampling [KH01] on 2×2 pixel blocks worked at all our sampling resolutions. This allows a 4-channel 16-bit render target to store visibility to all 256 VPLs used in our implementation. Interleaved sampling is essentially orthogonal to the incremental refinements proposed above, so our results include all three.

3.3.3. Visibility with Sharp BRDFs

While visibility from a fixed set of 256 VPLs gives good results for diffuse surfaces, for sharper BRDFs individual shadows are clearly visible for the few VPLs reflected in the specular highlight. We propose an alternative for such materials based on the simple adaptive strategy proposed in Section 3.2.1. We send visibility rays to the 25 samples chosen in the specular lobe.

This improves shadow quality, but with only 25 visibility samples some shadow banding remains. We could add additional visibility queries in the lobe, but this increases cost, potentially highlights voxel aliasing, and behaves poorly using our incremental visibility. Instead we use a variance shadow query, inspired by variance shadow maps [DL06].

Instead of storing a single bit per visibility query, we store the distance (and distance squared) to the nearest occluder along the ray. We then estimate the per-query distance variance using a screen space blur, applying the equations described by Donnelly [DL06] to compute the per-VPL light intensity. This eliminates the shadow banding from individual shadows (see Figure 6), as it essentially performs per-query percentage closer filtering.

Unfortunately, this greatly increases storage for visibility queries—from 1 bit to 2 floats per VPL. We manage by sending only 24 visibility queries (ignoring one corner on the 5×5 grid), for a total of 48 floats. Using 2×2 interleaving, this requires only 12 floats per pixel, which fits in three 4-channel render targets.

3.4. Incremental Stencil Rendering

While our algorithms from Sections 3.2 and 3.3 give either area illumination or visibility, we need both simultaneously. Additionally, the algorithms refine image-space regions differently, so separate evaluation of lighting and visibility followed by a per-fragment combination is infeasible. Shadows introduce high frequencies where illumination varies slowly, and for specular materials the reverse can hold.

To address this, we propose an incremental approach to multiresolution rendering. We still rely on a stencil mask to identify which mulitresolution fragments to render. But instead of creating the stencil masks simultaneously before rendering, we create the masks and render one resolution at a time, from coarsest to finest. This allows each stencil mask to depend on illumination variations at coarser levels, in addition to prior metrics (e.g., depth and normal). This requires only minor changes from sequential stenciling:

```
i ← CurrentMipmapLevel( f );
for all (fragments f ∈ quad( mip-level( i ) ) ) do
  if ( HasIlluminationDiscontinuity( f, i+1 ) ) then
    SetStencil( f ); // Coarse patch had light discontinuity;
    continue;      //  re-render at higher resolution
  if ( HasDepthOrNormalDiscontinuity( f, i ) ) then
    continue; // Patch not valid (needs subdivision)
  if ( NoDiscontinuity( f, i+1 ) ) then
    continue; // Coarser patch did not need subdivision
  SetStencil( f );
```

Rapid color changes between coarse samples indicates that higher sampling may capture higher frequencies. When iterating, we compare the light variations between computed samples to a user threshold to determine if additional sampling is needed (see Figure 7).

Our incremental rendering iterates the following steps once per rendering resolution, beginning at the coarsest level and progressing through the finest:

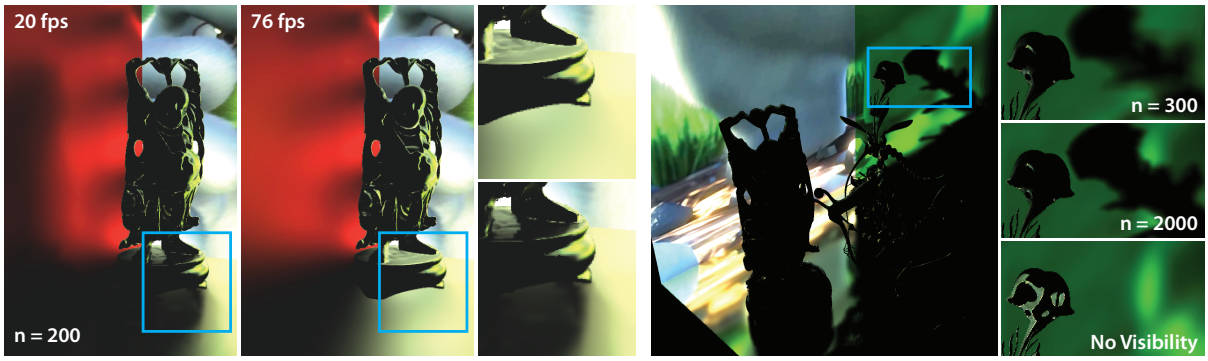


Figure 6: *Two examples of our method with a Phong BRDF, with and without visibility, and varying levels of shininess.*

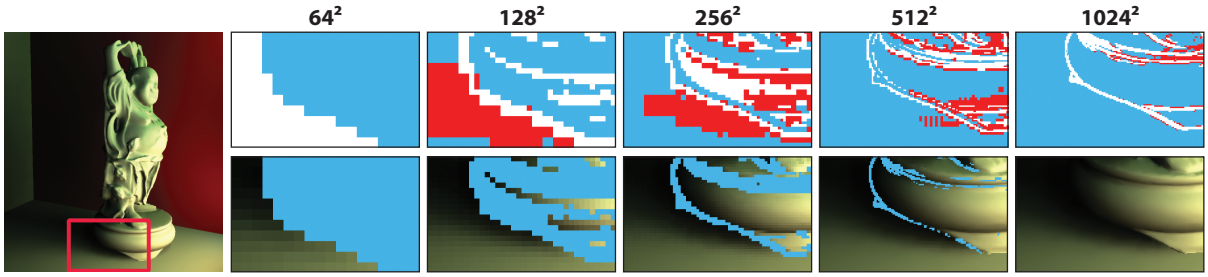


Figure 7: *An example of incremental stencil refinement. (Top) The refinement stencil for each resolution. Areas in blue are not rendered; areas in white are refined due to depth or normal discontinuities, and areas in red are refined due to illumination discontinuities. (Bottom) The resulting image after each step, with unrendered areas in blue.*

1. Create a stencil based on image-space depth, normal, and BRDF frequency metrics *and* any illumination discontinuities observed in coarser levels,
2. At set stencil bits, incrementally compute visibility and illumination to locally approximate Eq. 2.
3. When the stencil is *not* set, upsample and interpolate any coarser results.

Figure 7 depicts incremental stencil rendering. At the lowest resolution, we create a stencil based solely on depth and normal discontinuities. At higher resolutions, observed illumination discontinuities (shown in red) supplement regions known to require refinement. At each level, we upsample and interpolate unrendered areas using prior work [NW09], essentially an edge-aware bilinear interpolation that avoids using uncomputed fragments (those shown in blue).

This enables adaptive refinement at shadow boundaries. Furthermore, the illumination metric can capture high frequency BRDFs. Discontinuities introduced by specularities naturally trigger refinement, though we find that material specific metrics (e.g., Section 3.2.1) typically provide better quality, performance, and avoid aliasing.

4. Results and Discussion

Our timings come from a dual-core 3GHz Pentium 4 and a GeForce GTX 280, using OpenGL. Unless otherwise stated, all images and reported results were for an output resolution of 1024^2. Key GPU functionality required includes integer textures (for voxelization) and early stencil culling (to cull unnecessary fragments).

Figure 1 demonstrates the realism area sources add to a scene. The top compares a living room with and without our area lighting. Figure 6 shows surfaces with a Phong BRDF, comparing our results with and without visibility. The selective stenciling described in Section 3.2 allows us to discard many fragments unprocessed, improving performance.

Figure 8 compares our work, with and without visibility, to a path traced reference and a rendering using 1024 coarse (64^2) shadow maps, similar to rendering with imperfect shadow maps. We created the coarse shadow maps via traditional rasterization, leading to higher quality but poorer performance than ISMs. In general, our work compares favorably to path tracing, and captures higher shadow frequencies than those possible with coarse shadow maps.

4.1. Performance

Ultimately, multiresolution rendering performance depends on refinement quality. Illuminating each fragment requires gathering light from hundreds of VPLs and many visibility queries; thus, speed depends on the number of fragments rendered. We require only a few passes over the geometry, so geometric complexity plays a relatively minor performance role. Like prior multiresolution image-space algorithms, per-

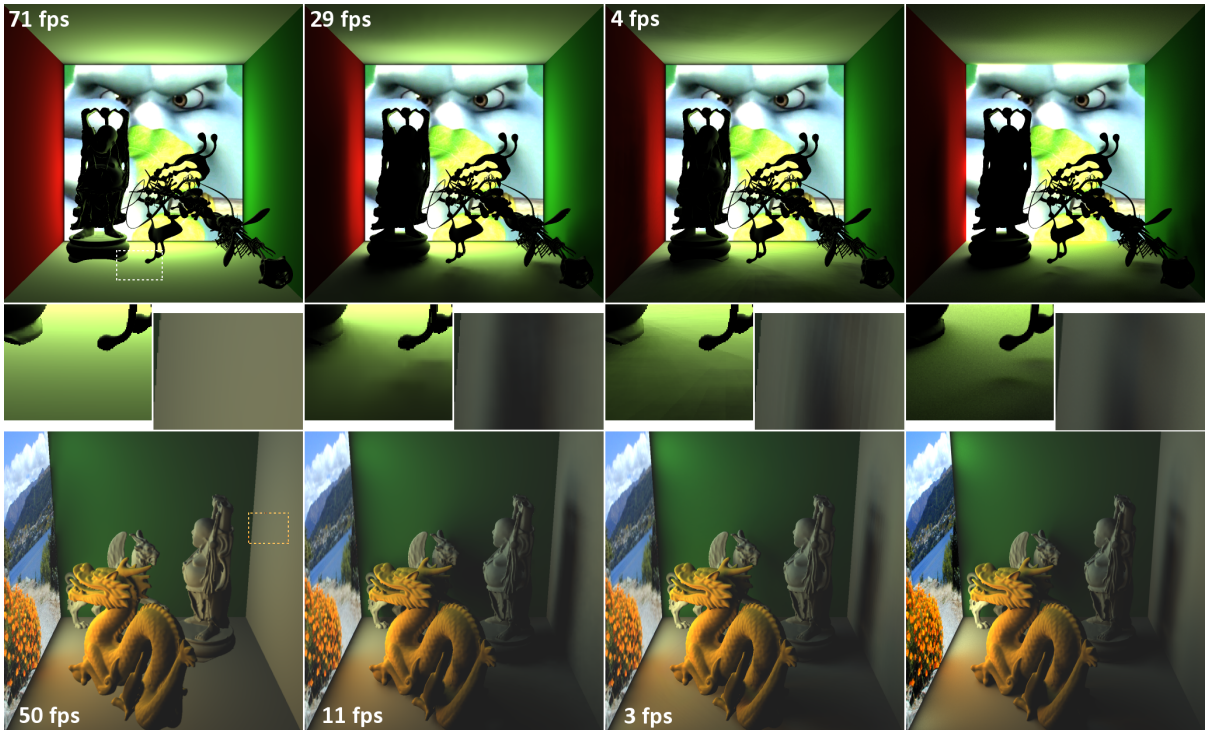


Figure 8: *Two complex scenes (left) without visibility, using our multiresolution approach; (center left) our incremental visibility; (center right) coarse,* 64^2 *shadow maps, with quality similar to imperfect shadow maps; and (right) path tracing.*

formance varies with *visual* complexity. Scenes with high frequency details require more rendered fragments with a corresponding performance hit, regardless of polygon count.

Figure 9 explores this effect for a moving camera in the scene atop Figure 8. While polygon count stays constant, the motion provides varying visual complexity and causes the fragment count to fluctuate each frame. The top graph shows the percentage of fragments rendered at each resolution. At 64^2, we render roughly half of the fragments. Finer resolutions cheaply interpolate and reuse many of these; only 10-15% of the final image needs per-pixel computation.

While we process a small *percentage* at high resolution, 10% of 1024^2 is significantly larger than 50% of 64^2. The bottom graph of Figure 9 reveals the cost for each refinement level. The quantity of fragments at higher resolutions still contribute much of our overall cost. To improve speed, some applications might render at only 256^2 or 512^2 and apply edge-preserving bilateral filters to upsample for final display.

As refinement thresholds directly affect fragment count, they greatly impact performance, as shown in Figure 10. Both strongly affect performance, but the lowest relative threshold has the strongest impact; with a color threshold of 0.005 the normal threshold has a minor impact, whereas the normal threshold dominates for higher color thresholds.

4.2. Voxel Buffer Resolution

We achieve compelling visibility using surprisingly coarse voxelizations. Figure 11 shows a closeup of the 755k triangle, genus 131 YeahRight model with six different voxel resolutions. All our results use 128 bits of depth, via a 32-bit per channel buffer. Visibility from a $128^2 \times 128$ buffer is almost indistinguishable to the $1024^2 \times 128$ visibility. While coarser resolutions give noticeably different results, this may be acceptable for large geometry with low-frequency visibility. Additionally, small voxel buffers increase cache coherence during ray traversal and significantly improve performance.

4.3. Limitations

Coarse voxelization, especially in the z dimension, leads to self occlusions. This occurs when initial visibility ray steps remain inside voxels representing the originating surface (see Figure 12). This is a variant of the "shadow acne" problem that plagues shadow mapping. We address it similarly, adding a bias to push the ray origin away from the surface.

While reducing self-occlusion errors, a bias causes missed contact shadows for thin geometry. In general, our method yields best results for geometry occupying multiple voxels; such geometry exhibits less aliasing during ray marching. Increasing steps per visibility query reduces aliasing as does increasing voxel resolution, though both impact perfor-

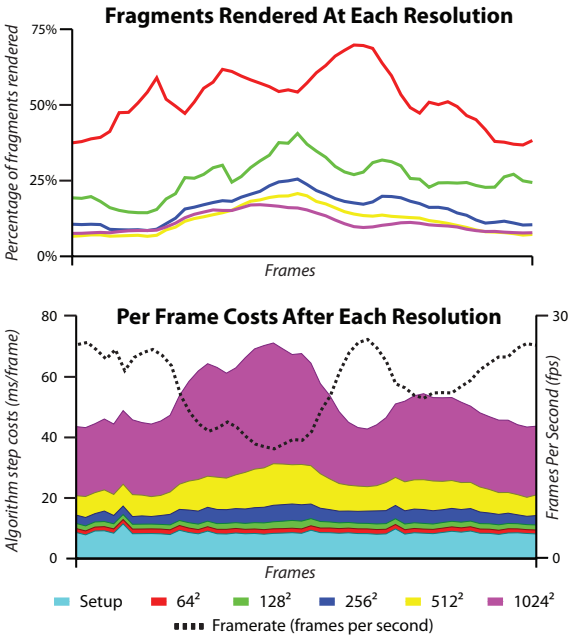


Figure 9: *Percentage of fragments rendered from each resolution (top) and the per-frame costs for each refinement (bottom) as the camera moves through the "Yeah Right" scene. The lower graph also displays the framerate at* 1024^2.

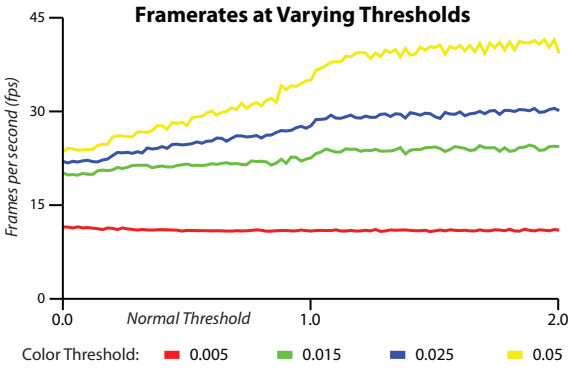


Figure 10: *Speeds with varied normal and color thresholds.*

mance. Using additional render targets would capture higher fidelity voxels (for up to 2048-bit z-resolution). Though, surprisingly, we found a single 128-bit buffer sufficient for all but the thinnest geometry (e.g., the furniture in Figure 3).

As visual complexity dramatically impacts performance, naively adding normal or bump mapping may refine a lot of pixels, degrading performance. We see numerous ways to mitigate this, which we plan to address in future work.

5. Conclusions

We introduced a multiresolution image-space rendering algorithm able to compute direct illumination from dynamic area lights. We described refinement methods to accelerate

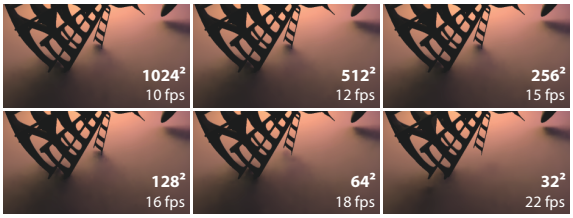


Figure 11: *The basket of the YeahRight model (also in Figure 8) rendered with varying voxel buffer dimensions.*

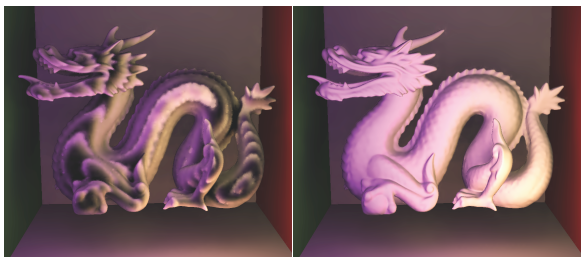


Figure 12: *A dragon with self-occlusion artifacts due to limited voxel resolution. A bias helps alleviate the problem.*

the rendering of diffuse and non-diffuse surfaces, proposed a method of coarsely approximating visibility using screen space voxelization, and combined them using incremental refinement. We achieve interactive speeds for a variety of scenes, require no precomputation, and impose no restrictions on the light, camera, or geometry.

We see many interesting future directions. As visibility dominates our computation, additional refinement techniques may prove more efficient. Additionally, illumination refinement is orthogonal from our proposed visibility queries; incremental lookups into, say, imperfect shadow maps may work better. We uniformly sample the light, which works for many environments, but more complex materials may require adaptive light sampling (e.g. Lightcuts [WFA*05]) to maintain quality. Finally, we believe multiresolution rendering can accelerate other problems, such as ambient occlusion and illumination in participating media.

Acknowledgments

This work was supported by DARPA grant HR0011-09-1-0027. The authors would like to acknowledge generous hardware donations by NVIDIA. 3D models were provided by Google's 3D Warehouse, Keenan Crane, the Stanford repository, and the Utah 3D Animation Repository. Video stills are from "Big Buck Bunny" (copyright © Blender Foundation).

References

[AAM03] ASSARSSON U., AKENINE-MÖLLER T.: A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Trans. Graph. 22*, 3 (2003), 511–520.

[ADM*08] ANNEN T., DONG Z., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Real-time, all-frequency shadows in dynamic scenes. *ACM Trans. Graph. 27*, 3 (2008).

[AMB*07] ANNEN T., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Convolution shadow maps. In *Proc. Eurographics Symposium on Rendering* (2007), pp. 51–60.

[ARHM00] AGRAWALA M., RAMAMOORTHI R., HEIRICH A., MOLL L.: Efficient image-based methods for rendering soft shadows. In *Proceedings of SIGGRAPH* (2000), pp. 375–384.

[AS00] ASHIKHMIN M., SHIRLEY P.: An anisotropic phong brdf model. *Journal of Graphics Tools 5* (2000), 25–32.

[BS09] BAVOIL L., SAINZ M.: Multi-layer dual-resolution screen-space ambient occlusion. In *SIGGRAPH Talks* (2009).

[DCB*04] DONG Z., CHEN W., BAO H., ZHANG H., PENG Q.: Real-time voxelization for complex polygonal models. In *Proceedings of Pacific Graphics* (2004), pp. 43–50.

[DL06] DONNELLY W., LAURITZEN A.: Variance shadow maps. In *Proc. Symp. Interactive 3D Graphics and Games* (2006), pp. 161–165.

[DS05] DACHSBACHER C., STAMMINGER M.: Reflective shadow maps. In *Proc. Symp. Interactive 3D Graphics and Games* (2005), pp. 203–231.

[ED06] EISEMANN E., DÉCORET X.: Fast scene voxelization and applications. In *Proc. Symp. Interactive 3D Graphics and Games* (2006), pp. 71–78.

[ED08] EISEMANN E., DÉCORET X.: Single-pass gpu solid voxelization for real-time applications. In *Proc. Graphics Interface* (2008), pp. 73–80.

[Eve01] EVERITT C.: *Interactive Order-Independent Transparency*. Tech. rep., nVidia, http://developer.nvidia.com/object/interactive_order_transparency.html, 2001.

[Fer05] FERNANDO R.: Percentage-closer soft shadows. In *SIGGRAPH Sketches* (2005).

[GBP06] GUENNEBAUD G., BARTHE L., PAULIN M.: Real-time soft shadow mapping by backprojection. In *Proceedings of the Eurographics Symposium on Rendering* (2006), pp. 227–234.

[HH97] HECKBERT P., HERF M.: *Simulating Soft Shadows with Graphics Hardware*. Tech. Rep. CMU-CS-97-104, Carnegie Mellon University, January 1997.

[HLHS03] HASENFRATZ J.-M., LAPIERRE M., HOLZSCHUCH N., SILLION F.: A survey of real-time soft shadow algorithms. *Computer Graphics Forum 22*, 4 (2003), 753–774.

[HM08] HUNT W., MARK W. R.: Adaptive acceleration structures in perspective space. In *Proceedings of the Symposium on Interactive Ray Tracing* (2008), pp. 11–17.

[IHK05] IM Y.-H., HAN C.-Y., KIM L.-S.: A method to generate soft shadows using a layered depth image and warping. *IEEE Trans. Vis. Comput. Graph. 11*, 3 (2005), 265–272.

[Kaj86] KAJIYA J.: The rendering equation. In *Proceedings of SIGGRAPH* (1986), pp. 143–150.

[KD10] KAPLANYAN A., DACHSBACHER C.: Cascaded light propagation volumes for real-time indirect illumination. In *Proc. Symp. Interactive 3D Graphics and Games* (2010), pp. 99–107.

[Kel97] KELLER A.: Instant radiosity. In *Proceedings of SIGGRAPH* (1997), pp. 49–56.

[KG09] KRIVANEK J., GAUTRON P.: *Practical Global Illumination with Irradiance Caching*. Morgan and Claypool, 2009.

[KH01] KELLER A., HEIDRICH W.: Interleaved sampling. In *Proc. Eurographics Workshop on Rendering* (2001), pp. 269–276.

[LHLW09] LIU F., HUANG M.-C., LIU X.-H., WU E.-H.: Efficient depth peeling via bucket sort. In *Proceedings of High Performance Graphics* (2009), pp. 51–57.

[Mit07] MITTRING M.: Finding next gen: Cryengine 2. In *SIGGRAPH Course Notes* (2007).

[NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph. 22*, 3 (2003), 376–381.

[NSW09] NICHOLS G., SHOPF J., WYMAN C.: Hierarchical image-space radiosity for interactive global illumination. *Computer Graphics Forum 28*, 4 (2009), 1141–1149.

[NW09] NICHOLS G., WYMAN C.: Multiresolution splatting for indirect illumination. In *Proc. Symp. Interactive 3D Graphics and Games* (2009), pp. 83–90.

[REG*09] 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 (2009).

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H.-P., DACHSBACHER C., KAUTZ J.: Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. *ACM Trans. Graph. 27*, 5 (2008).

[RGKM07] RITSCHEL T., GROSCH T., KAUTZ J., MUELLER S.: Interactive illumination with coherent shadow maps. In *Proc. Eurographics Symposium on Rendering* (2007).

[RGKS08] RITSCHEL T., GROSCH T., KAUTZ J., SEIDEL H.-P.: Interactive global illumination based on coherent surface shadow maps. In *Proc. Graphics Interface* (2008), pp. 185–192.

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

[SGwHS98] SHADE J., GORTLER S., WEI HE L., SZELISKI R.: Layered depth images. In *Proc. SIGGRAPH* (1998), pp. 231–242.

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph. 21*, 3 (2002), 527–536.

[SR09] SUN B., RAMAMOORTHI R.: Affine double- and triple-product wavelet integrals for rendering. *ACM Trans. Graph. 28*, 2 (2009), 1–17.

[SS07] SCHWARZ M., STAMMINGER M.: Bitmask soft shadows. *Computer Graphics Forum 26*, 3 (2007), 515–524.

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-d shapes. In *Proceedings of SIGGRAPH* (1990), pp. 197–206.

[WA90] WOO A., AMANATIDES J.: Voxel occlusion testing: a shadow determination accelerator for ray tracing. In *Proc. Graphics interface* (1990), pp. 213–220.

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

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. In *Proceedings of SIGGRAPH* (1978), pp. 270–274.

[YFGL09] YANG B., FENG J., GUENNEBAUD G., LIU X.: Packet-based hierarchal soft shadow mapping. *Computer Graphics Forum 28*, 4 (2009), 1121–1130.