# Interactive Indirect Illumination Using Adaptive Multiresolution Splatting

Greg Nichols and Chris Wyman, *Member*, *IEEE*

**Abstract**—Global illumination provides a visual richness not achievable with the direct illumination models used by most interactive applications. To generate global effects, numerous approximations attempt to reduce global illumination costs to levels feasible in interactive contexts. One such approximation, reflective shadow maps, samples a shadow map to identify secondary light sources whose contributions are splatted into eye space. This splatting introduces significant overdraw that is usually reduced by artificially shrinking each splat's radius of influence. This paper introduces a new multiresolution approach for interactively splatting indirect illumination. Instead of reducing GPU fill rate by reducing splat size, we reduce fill rate by rendering splats into a multiresolution buffer. This takes advantage of the low-frequency nature of diffuse and glossy indirect lighting, allowing rendering of indirect contributions at low resolution where lighting changes slowly and at high-resolution near discontinuities. Because this multiresolution rendering occurs on a per-splat basis, we can significantly reduce fill rate without arbitrarily clipping splat contributions below a given threshold—those regions simply are rendered at a coarse resolution.

**Index Terms**—Global illumination, interactive rendering, reflective shadow maps, multiresolution splatting.

## 1 Introduction

Indirect illumination represents light reaching a surface after previously interacting with other surfaces. While this lighting adds tremendously to visual richness and scene realism, the costs to track multibounce light reflections often prove prohibitive. Because of this, interactive applications frequently forgo complex global illumination entirely or use approximation techniques such as ambient occlusion [1], light maps, or precomputed radiance transport [2]. Unfortunately, these techniques impose limits of their own, often ignoring color bleeding or restricting the motion of geometry, lights, or viewer.

However, physically accurate global illumination may be unnecessary in most contexts. Tabellion and Lamorlette [3] found that even in visually demanding applications, such as feature films, single bounce indirect illumination provides plausible lighting. Accepting this limitation avoids tracing complex light paths that add little to a scene and dramatically simplifies the rendering equation.

One single bounce approach uses an augmented shadow map, called a reflective shadow map (RSM), to either gather during a final deferred render pass [4] or scatter indirect illumination via splatting [5]. Both techniques build on the idea of instant radiosity [6], where pixels in the shadow map represent virtual point lights (VPLs) used as secondary light sources for computing indirect lighting. This approach effectively reformulates the rendering equation from a complex integral over surfaces to a sum over all texels in the shadow map. The key to achieving performance then lies in reducing the costs of this summation.

This paper proposes a novel multiresolution splatting technique that reduces costs for RSM-based indirect illumination. Previous techniques either gathered light from a subset of the virtual point lights or splatted light into limited regions. Our approach instead recognizes that each virtual light potentially affects the whole scene, but due to the low-frequency nature of indirect illumination many pixels receive radiance quite similar to their neighbors and can be processed as a group. This idea is similar to hierarchical radiosity approaches [7], but instead works in image space. We divide the image into regions with similar indirect illumination, which we call subsplats. When indirect light from VPLs is splatted into our multiresolution buffer, each subsplat is rendered as a single fragment into an appropriate layer of the buffer. Effectively, we use a piecewise-constant illumination approximation inside subsplats and reduce fill rate by rendering subsplats at varying resolutions (rather than always splatting into an image-resolution buffer). The buffer layers are upsampled and additively blended, and a new interpolation technique removes discretization artifacts from the final indirect illumination.

Our most general formulation defines different subsplats (i.e., regions of near-constant indirect illumination) for each virtual point light, allowing the rendering of arbitrary BRDFs. Recomputing subsplats for each VPL quickly becomes the bottleneck, so we propose an alternative approach using the same set of subsplats for an entire frame. This significantly improves performance, though only diffuse materials are properly handled.

Like other RSM approaches our technique does not consider visibility, and therefore does not converge to correct single bounce indirect illumination. In spite of this limitation, our method yields plausible results at reasonable framerates. In many applications, the lack of strict physical accuracy is an acceptable compromise.

• *The authors are with the University of Iowa, 14 MacLean Hall, Iowa City, IA 52242-1419. E-mail: {gbnichol, cwyman}@cs.uiowa.edu.*

*Manuscript received 25 Feb. 2009; revised 19 May 2009; accepted 11 Aug. 2009; published online 18 Aug. 2009.*
*Recommended for acceptance by M. McGuire and E. Haines.*
*For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCGSI-2009-02-0044.*
*Digital Object Identifier no. 10.1109/TVCG.2009.97.*


Fig. 1. (a) Direct light only; (b) indirect light generated with our method; (c) the combined image. This scene is generated at 29 fps with fully dynamic lighting, geometry, and camera.

## 2 Previous Work

Widespread illumination research has enabled numerous techniques for interactive complex lighting. Generally, the simplest approaches add global effects by precomputing lighting, for instance, using radiosity [8], and baking in the results to surface textures. Clearly this precludes dynamic lighting and significant scene modifications, but quality depends on precomputation time and runtime evaluation is cheap.

Ambient occlusion [1] approximates indirect illumination by darkening direct lighting based upon occlusion from neighboring geometry. Although this produces only a coarse approximation, it cheaply reproduces effects such as darkened corners that often arise from indirect lighting. Precomputed ambient occlusion provides a cheaper alternative to complex radiosity solutions, but maintains the assumption of static geometry. Bunnell [9] describes an iterative ambient occlusion approximation for simple dynamic models. Other techniques [10], [11] precompute occlusion "fields" for rigid objects, allowing these objects to move while occluding nearby geometry.

Another class of techniques precomputes light transport and combines it with dynamic illumination at runtime using a simple dot product. Using a spherical harmonic basis to store the precomputed transport, Sloan et al. [2] allow rendering of low-frequency lighting on static geometry. This works best with environmental lighting, though further work [12] also allows local lights. Transport fields [13], [14] extend this approach to allow scenes with a few rigid, dynamic objects. Related techniques [15], [16] allow simple deformable models, but without indirect illumination.

Instant radiosity [6] introduces the concept of VPLs, which are emitted from scene illuminants using a quasirandom walk. Multiple hardware rendering passes use each VPL in turn as a point light, accumulating lighting and using shadow maps to account for indirect visibility. This technique directly demonstrates the cost of visibility queries in global illumination—each object is rasterized once for each virtual light. Laine et al. [17] reduce visibility costs for instant radiosity by reusing shadow maps between some VPLs. Ritschel et al. [18] precompute coherent shadow maps, allowing for faster visibility queries on rigid objects; later work [19] simultaneously samples visibility from 1,024 lights, producing an approximation that is imperfect, but sufficient for most indirect illumination.

Dachsbacher et al. [20] and Dong et al. [21] explore techniques that avoid explicitly computing visibility.

### 2.1 Reflective Shadow Maps

Rather than computing or approximating visibility, reflective shadow maps [4] entirely ignore visibility for indirect rays, assuming that viewers will not notice incorrect visibility for secondary illumination. Reflective shadow maps build on the idea of instant radiosity, using shadow mapping hardware to generate virtual lights directly instead of using quasirandom path tracing. The map itself consists of a standard shadow map augmented by additional buffers to store surface normals, positions, and reflected flux (see Fig. 3)—essentially a light-space G-buffer [22].

The original method [4] uses a gathering approach to sample nearby locations in the reflective shadow map for each pixel in the final image (see Fig. 2); eye-space interpolation reduces illumination artifacts due to low sampling rates in both eye and light space. Later work reformulates reflective shadow mapping using a shooting algorithm [5], splatting each VPL's contribution onto a nearby region in eye space. This technique extends to render glossy materials and simple caustics by elongating the splat size based upon the material's BRDF, and importance sampling the shadow map allows selection of a good set of VPLs based upon flux distribution.

One of the problems with splatting illumination from VPLs is excessive overdraw. In theory, each VPL can affect final illumination everywhere in the scene; using 1,000 point lights requires computing contributions for 1,000 splats at each eye-space pixel. Dachsbacher and Stamminger [5] reduce overdraw by restricting splat sizes. Beyond a certain distance from each VPL, indirect contributions are ignored. This gives an ideal parameter for tuning performance, but darkens illumination significantly as splat sizes shrink.

### 2.2 Min-Max Mipmaps

Our technique makes use of the min-max mipmap, which is similar to a subdivided quad-tree [23]. Each layer in such a mipmap stores both the maximum and the minimum (instead of the average) of all corresponding texels from finer resolutions in the map. Sampling any texel of a min-max mipmap gives the largest and smallest values in that texel's image-space region. Recent uses of min-max mipmaps include the rendering of soft shadows [24], geometry intersection [25], and dynamic height field rendering [26].

### 2.3 Splatting and Multiresolution Approaches

Interactive techniques often rely on splatting, as gathering frequently proves less amenable to GPU acceleration. Gautron et al. [27] approximate global illumination using

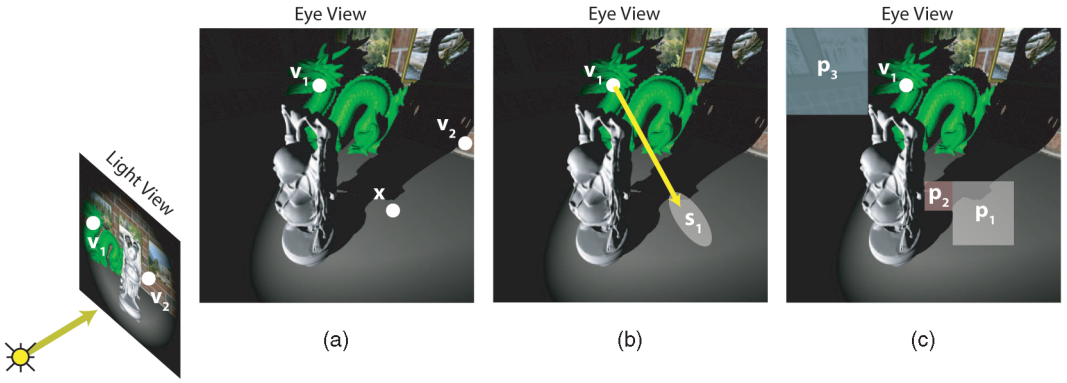


Fig. 2. (a) Reflective shadow maps sum contributions from nearby virtual point lights $v_i$ for each pixel $x$ in image space. (b) Reformulating this process as a splatting algorithm (center) processes each VPL just once, but splats $s_i$ frequently have limited area of effect in order to reduce overdraw. (c) While every VPL $v_i$ potentially illuminates any pixel in eye space, many regions (such as patches $p_1$, $p_2$, and $p_3$) have relatively constant contributions from these lights. We propose computing illumination contributions for each patch $p_i$ just once, by rendering a pixel-sized *subsplat* into an appropriate layer of our multiresolution illumination buffer. For each $v_i$ we draw a subsplat for $p_3$ into a coarse buffer, a subsplat for $p_1$ into a finer level, and a subsplat for $p_2$ into an even higher resolution buffer.

a splat-based renderer with a radiance cache. Shanmugam and Arikan [28] use billboards as splats to compute ambient occlusion on surfaces within the splat's influence. Sloan et al. [29] use splats to accumulate indirect illumination from spherical proxy geometry. Caustic mapping [30] frequently uses splats to represent photon energy, varying splat size to account for divergent photons and reduce sampling noise [31].

Even offline illumination rendering has investigated splatting as an alternative to gathering techniques, allowing more accurate illumination on high-frequency geometry and the removal of low-frequency noise [32].

However, most interactive splat-based illumination algorithms simply assume splats must be rendered at full resolution, or clamp splats to a "reasonable" size to maintain performance. Point-based rendering [33] and volume rendering [34] use multiresolution splatting to achieve interactive speeds. Lehtinen et al. [35] use an adaptive hierarchical point sampling method to induce a basis function for PRT, which is rendered using GPU-based splatting. Offline rendering techniques frequently use multiresolution and hierarchical techniques to reduce computational costs (e.g., hierarchical radiosity [7]).

We draw inspiration from recent caustic work [36] that renders illumination from splats into a multiresolution image. This allows capturing illumination from very large splats into coarse buffers and fine illumination details in high-resolution buffers while maintaining small splat sizes, dramatically reducing the overhead introduced by overdraw.

Finally, many of the ideas described in this paper were introduced in an earlier work [37]. This paper extends our earlier work by exploring alternative thresholding techniques for controlling rendering quality and detecting image-space lighting discontinuities. We discuss applicability to not only diffuse surfaces but also more general nondiffuse BRDFs. We also provide a more in-depth exploration of the effect various parameters have on visual quality and performance and further discussion of possible artifacts and how to avoid them.

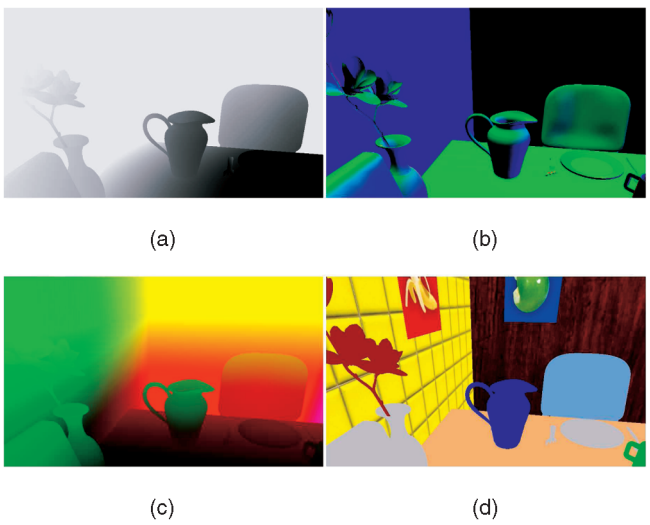


Fig. 3. Components of a reflective shadow map: (a) a linear distance from the eye, (b) a surface normal, (c) a world-space fragment position, and (d) a reflected flux.

## 3 Algorithm

Like Dachsbacher and Stamminger's algorithm [5] and other splatting approaches, we splat illumination from each virtual point light onto the scene. Naive splatting approaches render one splat for each VPL into a single resolution buffer. Choosing the optimal resolution for this buffer is nontrivial. Indirect lighting from a point light generally changes quite smoothly as the $1/r^2$ falloff gradually goes to zero; even a coarse buffer often suffices to capture this slowly changing illumination. Because splats are rendered in eye space, however, high-frequency normal variations and depth discontinuities can introduce rapid changes into the illumination that cannot be adequately captured with a coarse buffer. While this problem can be eliminated by simply rendering the splats at a higher resolution, doing so imposes much higher fill rate requirements and thus negatively impacts performance.

Instead of splatting into a single-resolution buffer, we propose a multiresolution approach. We begin with a full-screen quad for each VPL. This quad is divided into a set of primitives called *subsplats*: each subsplat covers just a single

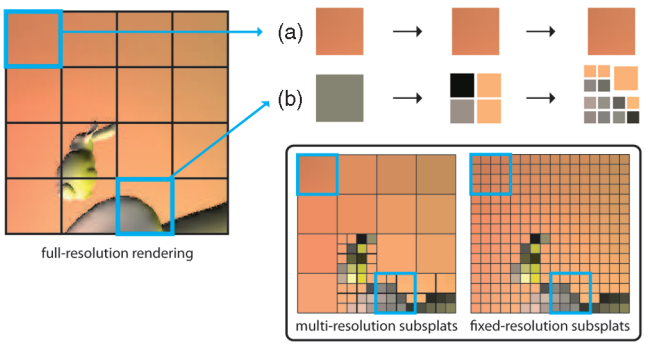


Fig. 4. A full-screen quad divided into single-texel subsplats, each of which are then adaptively refined twice. Here, (a) the first subsplat remains at its current resolution after each refinement pass. (b) The second subsplat is refined to a higher resolution. Three of the resulting subsplats are further refined, while the fourth remains at its current resolution. Summation of the multiresolution subsplats (lower right) and for comparison, a fixed-resolution summation where *all* subsplats are refined to the highest resolution. Multiresolution refinement allows comparable quality with many fewer subsplats.

texel at some resolution, though its image-space area may contain up to several thousand pixels. The initial set of subsplats is produced at very low resolution: $16^2$ or even $8^2$. We then adaptively refine the subsplats, subdividing those that are too coarse. After repeated refinements, the final set of subsplats is rendered, with some subsplats output to a $16^2$ buffer, some refined and output to a $32^2$ or $64^2$ buffer, and some refined all the way to the final output image. The "splat" is the summation of this set of multiresolution subsplats. Fig. 4 illustrates this concept, depicting the results as two different subsplats are refined twice.

This idea has roots in various previous techniques. It could be seen as a variant of hierarchical radiosity [7], where patches are chosen based upon image-space rather than object-space constraints. Radiance caching [27] typically focuses illumination samples near edges, and Tole et al. [38] rely on image-space criteria to better select cache samples for an interactive render. Mitchell [39] describes a method of reconstructing an image from nonuniform samples.

Ultimately, our algorithm simply allows splatting-based techniques to reduce fill rate costs by avoiding redundant computations, grouping them, and rendering to the coarsest buffer allowable for each group. In our algorithm each subsplat covers a single texel, though that texel might lie in a low-resolution buffer and thus affect hundreds of pixels in the final image.

The rest of this section describes, in greater detail, the steps of our algorithm. A quick breakdown follows:

1. Compute reflective shadow map and direct light;
2. Select VPLs used to splat indirect illumination;
3. Generate mipmap to detect discontinuities;
4. Create and iteratively refine the list of subsplats;
5. Render to multiresolution illumination buffer;
6. Upscale and combine buffer for total indirect light;
7. Add direct and indirect light for final result.

Steps 1 and 2 are described in Section 3.1, steps 3-5 are described in Sections 3.2-3.4, and steps 6 and 7 are described in Section 3.5.

### 3.1 Reflective Shadow Map and VPLs

We begin by generating a reflective shadow map [5] by rendering from the light, storing world-space position, distance from the light, surface normal, and reflected flux for each texel (e.g., Fig. 3). Next, we render from the eye, computing only direct light with shadow mapping. During this step, we also generate a G-buffer [22] containing data needed for deferred shading (i.e., world-space position, normal, and distance from the eye).

We then sample the reflective shadow map to create VPLs. In our implementation, we always use the same fixed grid of RSM sampling locations to create VPLs, without regard to the content of the RSM. A flux-based importance sampling or quasirandom sampling may ultimately give better quality.

### 3.2 Min-Max Mipmap Creation

To correctly refine subsplats, we need to identify image-space discontinuities. To do this we use a *min-max mipmap*, where each element stores maximum and minimum values rather than the average values in a standard mipmap. To best detect depth and normal discontinuities, we investigated several different types of min-max mipmaps, described in the following sections.

#### 3.2.1 Min-Max Mipmaps for Depth Discontinuities

We begin by describing the construction of a conventional min-max mipmap using linear depth values. We start with the full-resolution linear depth buffer (computed in Section 3.1) and run the mipmap generation process. We halve the resolution at each step, computing for each output element the maximum and minimum values of the four input elements. When completed, sampling a texel within the min-max mipmap gives us the minimum and maximum depth values in that texel's image-space area. This allows efficient detection of depth discontinuities: if the difference between these values is greater than a threshold, then we consider a depth discontinuity to exist within that texel's region.

This method conservatively detects discontinuities, causing excess subsplat subdivision when a surface is viewed at a steep angle. In this case, depth differences along smooth surfaces viewed obliquely will be incorrectly treated as discontinuities. To address this, we investigated the use of a depth *derivative* to detect discontinuities. We compute a screen-space depth derivative, calculating

$$\sqrt{(dz/dx)^2 + (dz/dy)^2}$$

for each texel at the highest resolution. When computing each lower resolution mipmap texel, we select only the largest of the four input texels, generating a max-mipmap rather than a min-max mipmap. When refining subsplats, an area contains a discontinuity if the depth derivative for that region is greater than a threshold.

#### 3.2.2 Min-Max Mipmaps for Normal Discontinuities

Detecting surface normal discontinuities using a min-max mipmap is less straightforward. As the goal is to detect significant differences in surface normal orientation, we note that surface normals with significantly different

orientations vary significantly in at least one component. Therefore, we generate three sets of min-max mipmaps, one for each component of the unit surface normal. If the difference between the max and min values of *any* of the normal components exceeds a threshold, we consider it a normal discontinuity. In practice, this method works well for detecting sharp surface features.

Because we generate three separate min-max mipmaps, this approach consumes excess memory. As an alternative, we experimented with detecting surface normal discontinuities based on variations in surface curvature. Given the normals $\vec{N}_1$ and $\vec{N}_2$ of two neighboring texels, the curvature $\kappa$ of the surface between them can be estimated by [40]

$$\kappa = 2 * \sin\left[\frac{\arccos(\vec{N}_1 \cdot \vec{N}_2)}{2}\right]. \quad (1)$$

Similar to the depth derivative, we compute surface curvature for each pair of neighboring texels at the highest-resolution mipmap level, and choose the largest input at each additional step. When detecting discontinuities, we sample the min-max mipmap and compare the largest curvature value to a threshold to determine whether a significant change exists in the surface within that texel.

### 3.3 Refining the List of Subsplats

We begin with a very coarse set of subsplats at the coarsest resolution in our *illumination buffer*, the multiresolution image used to accumulate our indirect illumination. This coarse set is rarely sufficient to represent indirect illumination for the entire image at a reasonable level of quality. To generate an adequate set of subsplats, many of them must be refined to higher resolutions. In our implementation, we use a $16^2$ buffer for the coarsest resolution; we therefore start with 256 subsplats.

During each refinement step, each subsplat can either be rendered as a point at a coarse resolution or refined into four new subsplats corresponding to the next layer in the illumination buffer. Since these subdivided subsplats each represent a fragment in a higher resolution layer, each refinement quadruples the required fill rate. Thus, the key to performance is determining when to refine a subsplat and when a coarser one suffices.

In general indirect illumination changes slowly, based upon a distance-squared falloff from the light and cosine falloffs dependent on surface patch orientation. In simple scenes, coarse sampling and bilinear interpolation give plausible lighting. However, complex models create depth discontinuities along silhouettes and normal discontinuities along creases that introduce rapid changes to the indirect illumination seen by a viewer.

We detect these discontinuities by sampling the min-max mipmaps. Sampling these mipmaps at the subsplat's resolution allows us to detect significant depth or normal variations within the subsplat. If the difference between the max and min depth values exceeds a threshold, or the difference in any of the normal components exceed a similar threshold, then we consider a discontinuity to exist within that subsplat.

If a discontinuity exists, we subdivide the subsplat into four higher resolution subsplats (see Fig. 5). During each pass, we iteratively refine the subsplats, subdividing some and not others:

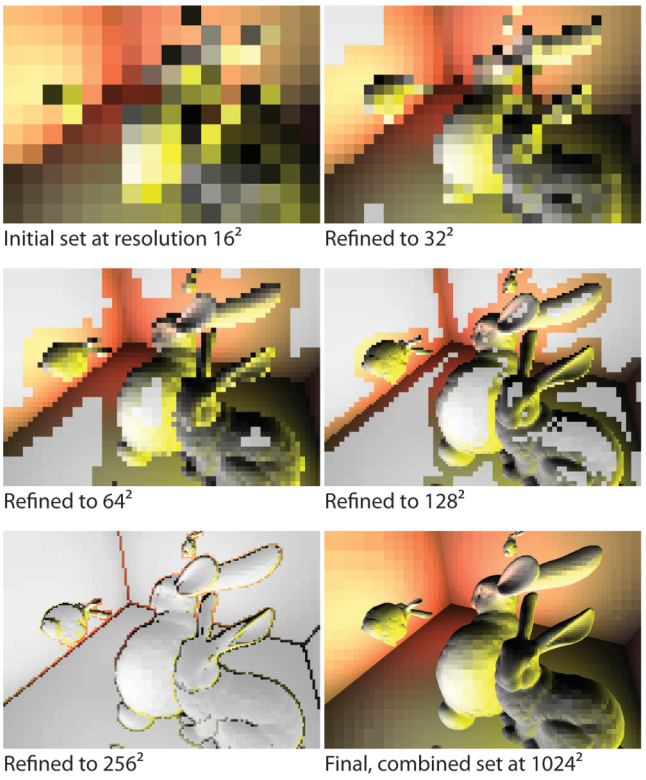


Fig. 5. Subsplat refinement occurs in areas with depth or normal discontinuities. In this case, we demonstrate how an initial set of subsplats at $16^2$ might be refined four times. For clarity, the set of unrefined subsplats at each step is displayed in gray scale. The combined result at $1{,}024^2$, after six refinement passes (lower right).

```
SubsplatList inputList;
SubsplatList outputList;

for all (subsplats s ∈ inputList) do
   if ( ContainsDepthDiscontinuity( s ) or
        ContainsNormalDiscontinuity( s ) ) then
      outputList.Add( s.TopLeftChild() );
      outputList.Add( s.TopRightChild() );
      outputList.Add( s.BottomRightChild() );
      outputList.Add( s.BottomLeftChild() );
   else
      outputList.Add( s );
```

Each refinement pass doubles the resolution of the indirect illumination. If the resulting set of subsplats has reached the desired resolution, they can be rendered. Otherwise, the set of subsplats is used as the input to a subsequent pass.

#### 3.3.1 Refinement for Arbitrary BRDFs

Complex lighting models often give rise to regions of focused light, which must be rendered at high resolution to be properly reproduced. Refinement based solely on depth and surface normal discontinuities is insufficient in these cases: if a highlight occurs on a surface without discontinuities, the refinement of that surface will likely be too coarse to adequately reproduce them.

To address this, we explored a variation of our method that uses discontinuities in *illumination*, rather than depth or surface normal, to guide the refinement process. When

evaluating whether to subdivide a subsplat, we compute the indirect illumination in each of the subsplat's quadrants and compare the results. We refine the subsplat if any of the samples differ significantly:

```
SubsplatList inputList;
SubsplatList outputList;

for all (subsplats s ∈ inputList) do
  l1 = IlluminationAtPoint( s.TopLeft() )
  l2 = IlluminationAtPoint( s.TopRight() )
  l3 = IlluminationAtPoint( s.BottomRight() )
  l4 = IlluminationAtPoint( s.BottomLeft() )

  if ( SamplesDiffer( l1, l2, l3, l4 ) ) ) then
    outputList.Add( s.TopLeftChildPoint() );
    outputList.Add( s.TopRightChildPoint() );
    outputList.Add( s.BottomRightChildPoint() );
    outputList.Add( s.BottomLeftChildPoint() );
  else
    outputList.Add( s );
```

Many ways exist to detect a "significant difference" in illumination samples. In our implementation, we achieve reasonable results by computing the maximum and minimum values of each component of the inputs, and comparing their differences to a threshold $\mathbf{T}$:

```
boolean SamplesDiffer( l1, l2, l3, l4 )

// component-wise max/min of the input samples
maxValues = max( max( max( l1, l2 ), l3, l4 );
minValues = min( min( min( l1, l2 ), l3, l4 );

if ( (maxValues.r - minValues.r) > T or
     (maxValues.g - minValues.g) > T or
     (maxValues.b - minValues.b) > T ) then
  return true;
else
  return false;
```

Because this style of refinement depends on point samples to detect illumination discontinuities, very sharp highlights may go undetected, and areas that contain them may not be sufficiently refined. Thus, this method may not yield sufficient quality using BRDFs that often exhibit high-frequency illumination detail (i.e., mirror-like BRDFs). For many BRDFs, though, refining subsplats in this manner effectively captures illumination behavior, allowing the plausible reproduction of lighting features and enabling our method to be used with complex materials (such as the Phong material shown in Fig. 6). The performance implications of this variation of our method are explored in Section 4.2.

### 3.4 Rendering Indirect Illumination

After iterative refinement, we have a large list of subsplats. Our implementation requires a three-tuple to store each subsplat, with one value specifying the subsplat's output resolution, and two values specifying its screen-space location.

During rendering, a vertex shader positions each subsplat in the correct layer of the illumination buffer, a fragment shader computes indirect illumination for each subsplat, and additive blending accumulates all contributions. For each subsplat, our indirect illumination is


Fig. 6. Indirect lighting from a Buddha with a Phong BRDF; here, subsplats are refined separately for each VPL. To emphasize the contribution, a blue tint was added to indirect light reflected by the Buddha. (a) Direct lighting, (b) indirect lighting, and (c) combined result. Indirect illumination rendered at $256^2$ at 23 fps.

$$I(x_s, x_l) = \rho_l(\vec{L}, \vec{V_{ls}})\rho_s(\vec{V_{sl}}, \vec{E})\Phi_l \frac{\langle \vec{V_{ls}}, \vec{N_l}\rangle \langle \vec{V_{sl}}, \vec{N_s}\rangle}{\pi |\vec{V_{ls}}|^2}, \quad (2)$$

where $x_s$ and $x_l$ are the shaded point and the virtual light point, $\vec{N_s}$ and $\vec{N_l}$ are the surface normals at $x_s$ and $x_l$, $\Phi_l$ is the flux to the VPL at $x_l$, $\rho_s$ and $\rho_l$ are the BRDFs at $x_s$ and $x_l$, $\vec{L}$ is the vector from $x_s$ to the light, $\vec{E}$ is the vector from $x_s$ to the eye, and $\vec{V_{ls}}$ and $\vec{V_{sl}}$ are, respectively, the vectors from $x_l$ to $x_s$ and from $x_s$ to $x_l$. These values are retrieved from the appropriate location either in the RSM or the G-buffer. Here, $\langle \vec{A}, \vec{B}\rangle = \max(\vec{A} \cdot \vec{B}, 0)$.

After rendering all subsplats, the illumination buffer contains all indirect illumination split into disjoint components at various resolutions. Naively summing the layer contributions gives a blocky representation of total indirect light (see Fig. 7) that requires interpolation.

### 3.5 Upsampling and Combination

Fig. 7 demonstrates the artifacts from naive methods of combining multiresolution illumination: blocky illumination when using nearest neighbor upsampling, and strange multiresolution haloing and ringing when using bilinear interpolation. Clearly, neither is acceptable.

The problem lies in the complex structure of the illumination buffer. Each texel contains either all of the illumination for that eye-space location, or none at all. Linear interpolation between a texel containing all of its relevant light and one containing no illumination makes little sense, as it spreads energy from texels containing energy to those deemed too coarse or too fine. Given that roles may be reversed at other levels in the illumination buffer, multiresolution linear interpolation leads to ringing and haloing. This arises from the varying regions of support for the interpolation filter at multiple scales.

We address this with a unique upsampling scheme. Upscaling progresses from coarsest to finest layers in the illumination buffer. At each resolution, texels containing illumination information are linearly interpolated with neighboring texels of the same resolution, whether they were originally rendered at that resolution or upsampled from a lower resolution during a previous pass:

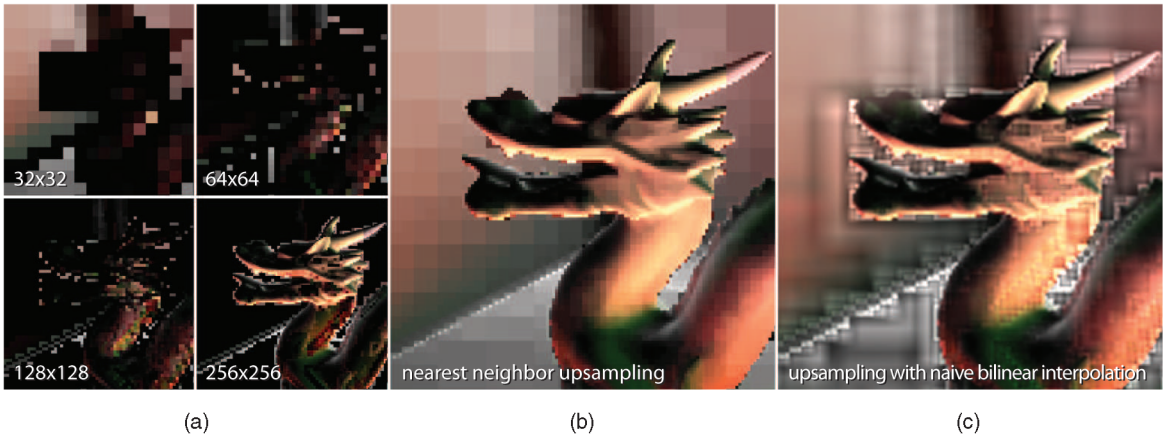


Fig. 7. Each illumination buffer level contains pieces of the indirect illumination at a different resolution. Artifacts from combining the illumination buffers using (a) nearest neighbor upsampling and (b) naive bilinear interpolation.

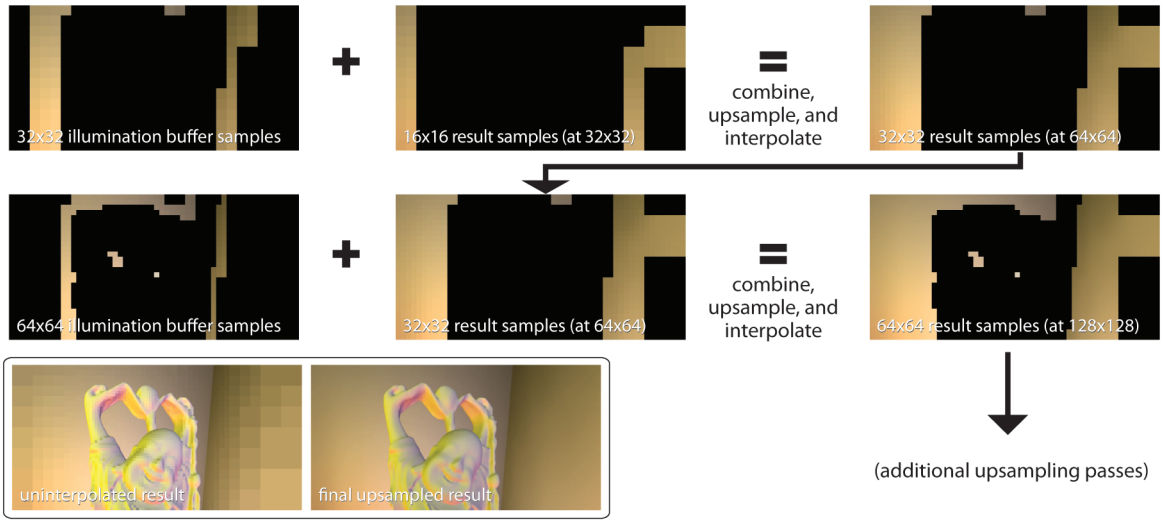


Fig. 8. Two upsampling passes, from $32^2$ to $64^2$ to $128^2$. At each level, missing samples are combined with interpolated data from previous levels. The result is interpolated to higher resolution, and used as the input for the next upsampling pass. The noninterpolated summation (lower left), and the final interpolated and upsampled result (lower middle).

```
for all layers l ∈ illumination buffer, coarsest to finest do
  for all (texels t ∈ l) do

    // 3x3 input texels both from the illumination
    // buffer and from previous passes
    inputTexels[9];
    renderedTexels[9];
    interpolationWeights[9];

    outputTexel = EmptyTexel();
    totalWeight = 0;

    for (i = 0 to 8) do
      if HasData(inputTexels[i]) or
         HasData(renderedTexels[i]) then
         totalWeight += interpolationWeights[i];
         outputTexel += interpolationWeights[i] *
              (renderedTexels[i] + inputTexels[i]);

    outputTexel/ = totalWeight;
```

After each layer has been upsampled, the result then becomes the input for the next resolution. To avoid the haloing and ringing shown in Fig. 7, each upsampling pass only outputs interpolated texels in locations where the illumination buffer already contained data for that resolution: energy is never pulled from empty areas nor spread into them, and all texels that contained no energy at the start of each pass remain empty. Fig. 8 demonstrates this process graphically through two upsampling steps.

After processing all the layers, we have a single combined and upsampled image that varies smoothly, without ringing artifacts. Furthermore, this method does not spread energy across major discontinuities. Our refinement process ensures that areas with these discontinuities are refined into high-resolution subsplats. Because each texel is interpolated only with texels of equal resolution and those upsampled from coarser resolutions, energy stays on the correct side of a discontinuity.

## 4 Implementation

We implemented our method using OpenGL and GLSL on a machine with a dual-core 3 GHz Pentium 4 and a GeForce

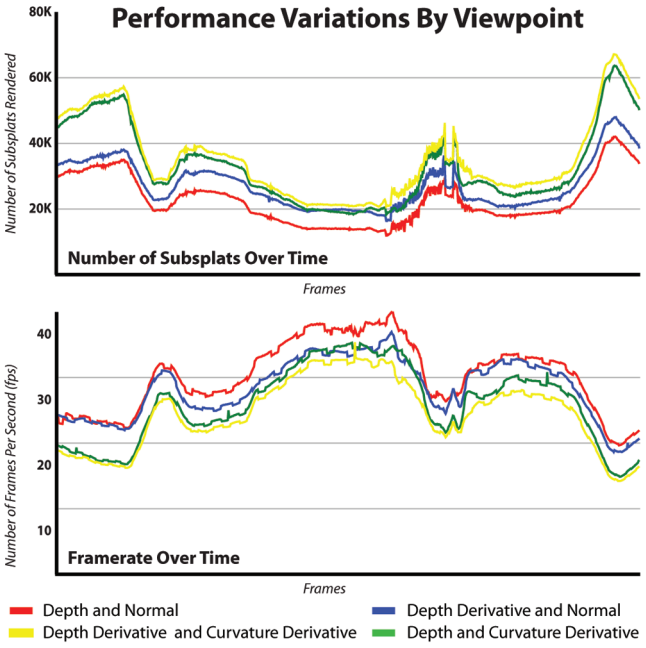


Fig. 9. Framerates and subsplat counts during a flythrough of the dragon scene. Four different combinations of depth and normal discontinuity detection are illustrated, rendered at $1{,}024^2$ with similar visual quality.

GTX 280. Our implementation uses OpenGL's geometry shader and transforms feedback extensions. We use a geometry shader to subdivide subsplats, selectively either dividing each input into four new outputs, or passing it through unchanged; the resulting subsplats are output to a vertex buffer object. We repeat the refinement process until the highest resolution subsplats have reached an appropriate resolution.

All images in this paper were generated using a final output resolution of $2{,}048^2$, which was downsampled to a $1{,}024^2$ window for an antialiased rendering.

### 4.1 Discontinuities and Min-Max Mipmaps

We found that neither depth derivatives nor surface curvature offer a substantive advantage in detecting discontinuities. Min-max mipmaps constructed with depth derivatives sometimes produce a smaller set of subsplats than those built with direct linear depth values. However, they are slightly more costly to construct due to the additional samples required to calculate the derivative. In practice, using a depth derivative min-max mipmap is slightly slower than using one constructed with linear depth values for a similar level of quality.

Min-max mipmaps built using surface curvature values do not fare as well. In particular, detection of gradually curving surfaces is problematic using curvature mipmaps: even if a great deal of change occurs within the complete image-space area of a low-resolution texel, curvature between neighboring high-resolution texels may be minimal. To detect these gradual changes, a very low threshold is required, leading to unnecessary subdivisions and a corresponding performance hit.

Fig. 9 demonstrates the relative performance of these discontinuity detection methods as the scene viewpoint changes. Although each method was configured to produce

TABLE 1
Refinement Costs for the Buddha with a Phong BRDF, with Subsplats Refined Separately for each VPL

| Indirect Resolution | $16^2$ VPLs | $32^2$ VPLs | $64^2$ VPLs | $128^2$ VPLs |
|---|---|---|---|---|
| $16^2$ | 65.5K<br>0.0ms<br>95 fps | 262.1K<br>0.0ms<br>60 fps | 1.05M<br>0.0ms<br>25 fps | 4.19M<br>0.0ms<br>7.4 fps |
| $32^2$ | 120.5K<br>3.9ms<br>65 fps | 484.9K<br>15.9ms<br>32 fps | 1.94M<br>49.4ms<br>13 fps | 7.20M<br>201.6ms<br>2.83 fps |
| $64^2$ | 220.4K<br>11.7ms<br>52 fps | 893.9K<br>28.0ms<br>21 fps | 3.57M<br>86.3ms<br>6.4 fps | N/A |
| $128^2$ | 421.8K<br>17.7ms<br>39 fps | 1.73M<br>37.1ms<br>14 fps | 6.66M<br>149.0ms<br>4.0 fps | N/A |
| $256^2$ | 852.0K<br>35.3ms<br>27 fps | 3.22M<br>60.1ms<br>8.9 fps | N/A | N/A |

*Each cell contains the number of refined subsplats, refinement time, and the resulting framerate. "N/A" entries required excessive memory and execution time.*

results of similar quality, an absolute qualitative comparison is difficult. In particular, the different normal discontinuity detection methods each accentuate certain features. For example, when reproducing a gently curved surface with the surface curvature method, a very low threshold must be used to match the quality of the surface normal method. However, this also results in sharp, detailed reproductions of small surface features that would be smoothed over by the surface normal method at anything but a very low threshold. Additionally, different viewpoints can yield results of varying quality depending on the detection method used.

### 4.2 VPLs and Subsplat Refinement

Our initial implementation refined a separate list of subsplats for each VPL as described in Section 3.3.1, computing illumination during refinement to guide the splitting process. With this approach, generating and refining a list of subsplats for each VPL is the most expensive part of our method, often by a large margin. Refining a single subsplat into four new subsplats using the method outlined in Section 3.3.1 takes roughly 30 nanoseconds, and rendering it takes a comparable amount of time. At these speeds, hundreds of thousands of subsplats can be refined and rendered while easily maintaining interactive rates. However, with each VPL generating tens of thousands of subsplats, the refinement and rendering costs quickly become unmanageable. As Table 1 illustrates, subsplat counts can easily climb into the millions, even at low VPL counts and relatively low resolutions. Additionally, because this method requires the calculation of illumination during refinement, more expensive BRDFs increase these costs yet further.

For diffuse scenes we found that splats were being split almost identically for each VPL. Noting this, we implemented an alternate approach that performs splat refinement just once per frame, avoiding illumination calculations and refining solely based on normal and depth discontinuities.

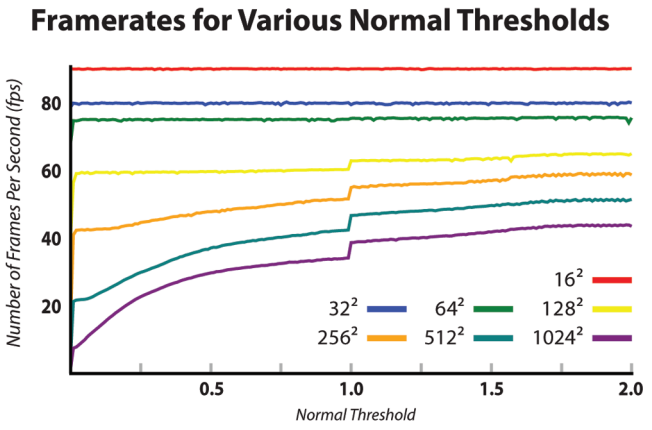


Fig. 10. Framerates of our method in the dragon scene with an increasing normal refinement threshold, at various resolutions of indirect illumination.

All VPLs then reuse the same subsplats for rendering. While there is no guarantee that normal and depth discontinuities are completely predictive of areas that will need refinement, we found that in diffuse scenes, this approach yields a significant increase in performance while maintaining similar image quality. Unless otherwise stated, all images and results in this paper are generated using this approach.

### 4.3 Adding Surface Detail

The number of subsplats generated has a significant impact on the performance of our technique. Since this number is heavily influenced by thresholds within the refinement process, these thresholds can serve as a parameter for tuning performance. This is particularly true of the threshold used to detect normal discontinuities, which ranges from $[0..2]$; a higher threshold allows faster framerates, at the expense of fine surface details. Fig. 10 illustrates this effect for one viewpoint in the dragon scene. While having no effect at low resolutions, at high resolutions the normal threshold substantially affects the overall framerate. This parameter's effect on image quality can be observed in Fig. 11: with high normal thresholds, insignificant surface features of the Buddha do not trigger refinement. These areas are then rendered at low resolution, effectively blurring them out.

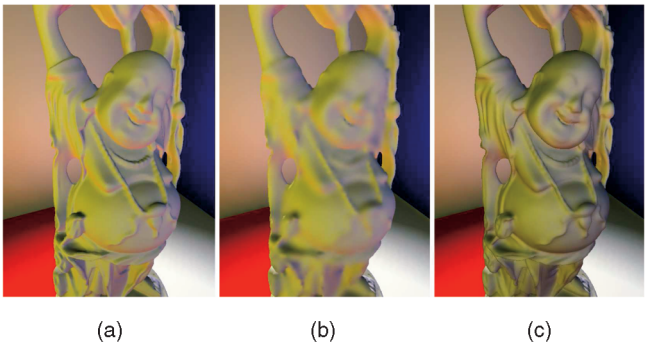


Fig. 11. The Buddha model rendered at $1{,}024^2$. (a) The image, rendered with a normal splitting threshold of 0.25, runs at 8 fps; (b) and (c) the images use a threshold of 1.4 and run at 32 fps. (c) The image approximates surface detail with the technique described in Section 4.3, using $\alpha = 0.5$.

TABLE 2
Framerates at Various Resolutions of Indirect Illumination

| Indirect Resolution | Simple Teapot (6.3K triangles) | Spinning Buddha (250K triangles) | Dragon & Bunny (405K triangles) | Flying Bunnies (417K triangles) |
|---|---|---|---|---|
| $128^2$ | 68 fps | 60 fps | 59 fps | 52 fps |
| $256^2$ | 50 fps | 42 fps | 42 fps | 41 fps |
| $512^2$ | 33 fps | 30 fps | 28 fps | 31 fps |
| $1024^2$ | 22 fps | 24 fps | 21 fps | 25 fps |

One solution is to approximate the missing detail, modulating the indirect illumination using the surface normal's alignment toward the camera. For each image-space pixel of illumination $c$, with view vector $\vec{V}$ and surface normal $\vec{N}$ at the same location, and a parameter $\alpha$ (ranging from $[0..1]$) controlling the effect's intensity:

$$c_{out} = c * (\alpha * \langle \vec{V}, \vec{N} \rangle + (1 - \alpha)) \quad (3)$$

Ideally, instead of modulating the indirect illumination with the viewing vector, we would introduce cosine falloff by modulating with the vector to each VPL. However, once the indirect illumination has been splatted into the illumination buffer, VPL locations are no longer easily accessible. While our method is somewhat ad hoc, it gives visually plausible results and incurs almost no runtime cost. This allows the use of higher normal thresholds, which in turn leads to faster framerates. This approximation may cause some areas to appear overly dark, which may be objectionable in some circumstances. The third panel of Fig. 11 illustrates the effect of replacing surface detail using this method.

## 5 Results and Discussion

Like other splatting approaches for indirect illumination, the performance bottleneck in our technique is the cost of rendering the indirect subsplats. The resolution of each subsplat does not matter: in our approach, subsplats each cover a single texel, so the cost of rendering them is independent of resolution. This also allows us to render subsplats as points, rather than triangle meshes, quads, or point sprites. We achieved a 5 percent speed increase simply by switching from quads to points.

As demonstrated by the data in Table 2, our performance has little dependency on geometric scene complexity: when rendering indirect illumination at $1{,}024^2$, performance is similar whether the scene contains a single teapot or many complex models. Geometry is rasterized twice, one rendering from the light and one from the eye. Only at fairly low resolutions does the geometric complexity of a scene have a significant effect on performance. However, our method is sensitive to the *visual complexity* of a scene. When looking at a flat wall, few subsplat refinements are required; complex geometry requires refinement around edges, creases, and crevices. High-frequency random geometry may require a uniformly dense refinement, where naive splatting would outperform our multiresolution approach.

As the viewpoint moves within the scene, changing the visual complexity of the current view, total subsplat count

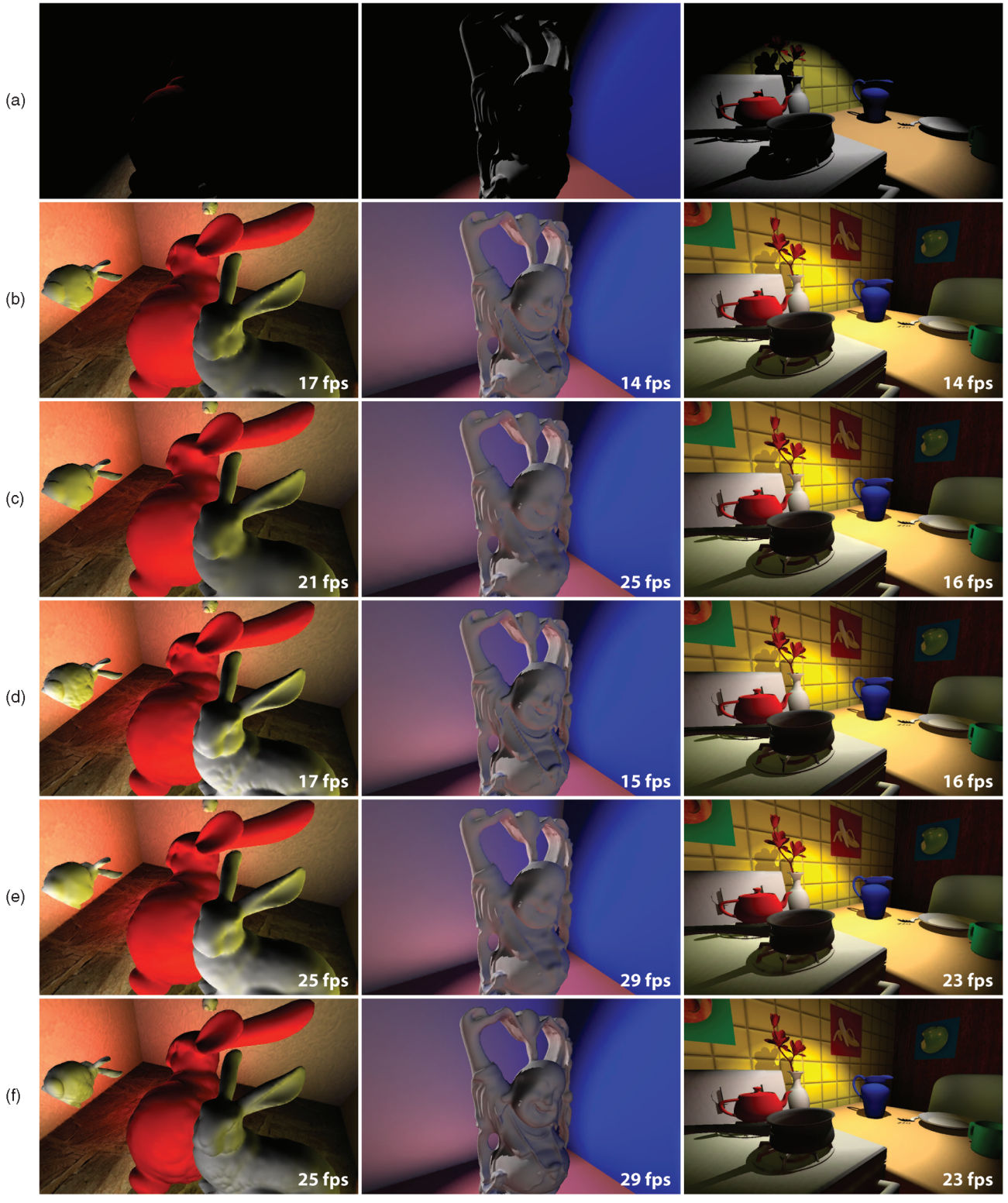


Fig. 12. Example scenes using our technique, with indirect illumination rendered at $1{,}024^2$. These scenes were rendered with (a) direct illumination only; indirect illumination generated with a curvature min-max mipmap, using threshold (b) 0.1 and (c) 0.3; indirect illumination generated with a surface normal min-max mipmap as described in Section 3.2, using normal threshold (d) 0.2 and (e) 1.5; (f) using normal threshold 1.5, and replacing surface detail using the method described in Section 4.3 (with $\alpha = 0.5$).

can vary up to 300 percent. Since splatting remains the bottleneck in our method, overall performance is strongly tied to the number of subsplats rendered, an effect that can be easily observed in Fig. 9.

As with other splatting techniques, we ignore visibility considerations when accumulating indirect illumination, which can result in an overly bright image. VPLs on a scene's ceiling illuminate not only the surface of a table, but

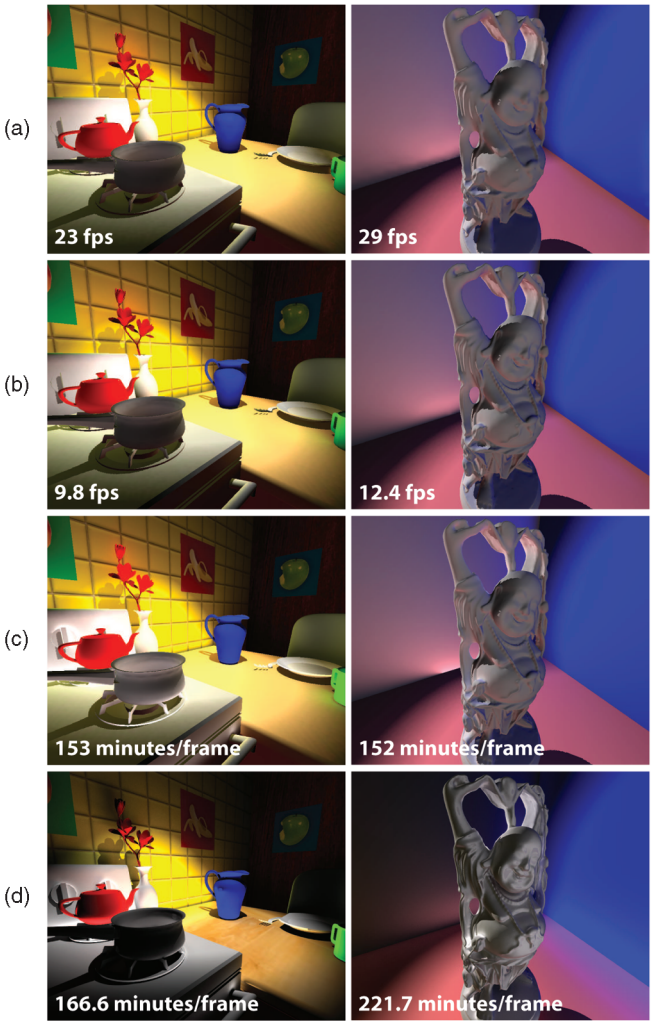


Fig. 13. Compare (a) our work to (b) reflective shadow mapping, with near indistinguishable results. One problem our work shares with RSMs is the need to clamp the minimum $r^2$ falloff to avoid point singularities. When using every texel of the RSM as a VPL, no clamping is required and generates (c) a brighter image (lower middle). (d) A path traced comparison shows that the main differences are lack of indirect visibility, clamping the $r^2$ falloff to avoid singularities, and an approximation that all RSM texels subtend the same solid angle. The solid angle approximation causes our work and RSM implementation to appear dimmer toward the inside of the spotlight and brighter toward the outside.

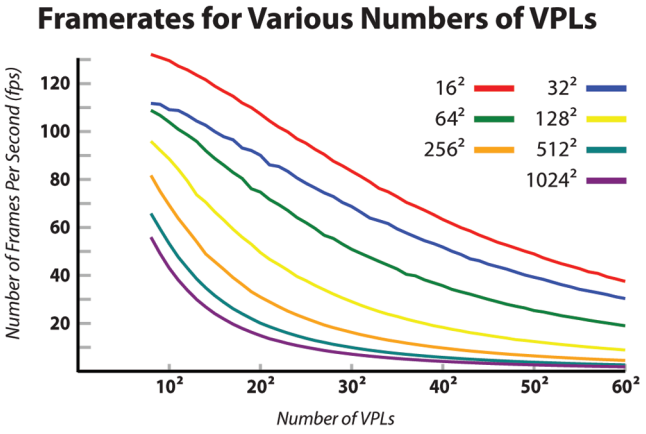


Fig. 14. Framerates of our method in the dragon scene using increasing numbers of VPLs, at various resolutions of indirect illumination.

also the floor below the table. One method of addressing this might be to approximate visibility independently, e.g., using ambient occlusion, and modulate the results. In the future, we also hope to explore multiresolution techniques that account for visibility.

Fig. 1 demonstrates the dragon and bunny scene rendered with direct illumination only, and with indirect illumination subdivided to a maximum refinement of $512^2$. Fig. 12 depicts several different scenes under varying rendering parameters, rendered with a maximum refinement level of $1{,}204^2$. In the second and third rows, discontinuities were detected using min-max mipmaps built using the surface curvature method described in Section 4.1, using a low threshold for the images in the second row and a higher threshold for those in the third row. The remaining three rows were rendered with min-max mipmaps built from surface normals, using a low surface normal threshold for the fourth row, and a high threshold for the fifth and sixth rows. The sixth row illustrates the effects of the reintroduction of detail using the ad hoc method described in Section 4.3. The visual appearance of complex objects, especially those close to the camera, depends heavily on refinement parameters. On the other hand, scenes with high geometric complexity, such as the kitchen scene in the rightmost column, often suffer little degradation in quality even when rendered with higher normal thresholds.

In Fig. 13, we compare the results of our technique to naive reflective shadow maps and a ground truth image produced by a Monte Carlo path tracer.

Fig. 14 explores performance on the dragon scene, showing variations due to the number of VPLs and refinement passes. With no refinement, indirect illumination is rendered at $16^2$ resolution. Each refinement doubles the resolution (up to $1{,}204^2$ after six passes).

### 5.1 VPL Sampling Artifacts

We regularly sample the reflective shadow maps to choose our virtual lights. While this sampling scheme is simple, it can result in flickering under animation as VPL locations jump on and off surfaces in successive frames. In simple scenes, we found that $16^2$ to $32^2$ VPLs gave smooth results under animation, with minimal flickering. More complicated scenes (in which VPLs jump between surfaces more often) require larger number of VPLs to achieve smooth results. This represents a tradeoff; increasing the number of VPLs reduces performance (as demonstrated by Fig. 14). A more sophisticated VPL sampling scheme may reduce the number required and minimize flickering.

### 5.2 Upsampling Artifacts

Our upsampling method assumes that adjacent texels in the illumination buffer can be safely interpolated together without visual artifacts. This relies heavily on sufficient subsplat refinement, which in turn depends on the successful detection of discontinuities. Suboptimal behavior of either of these processes results in a set of subsplats that does not adequately represent the scene. Additionally, when discontinuities are present in low-resolution subsplats, energy is blurred across the discontinuity, causing

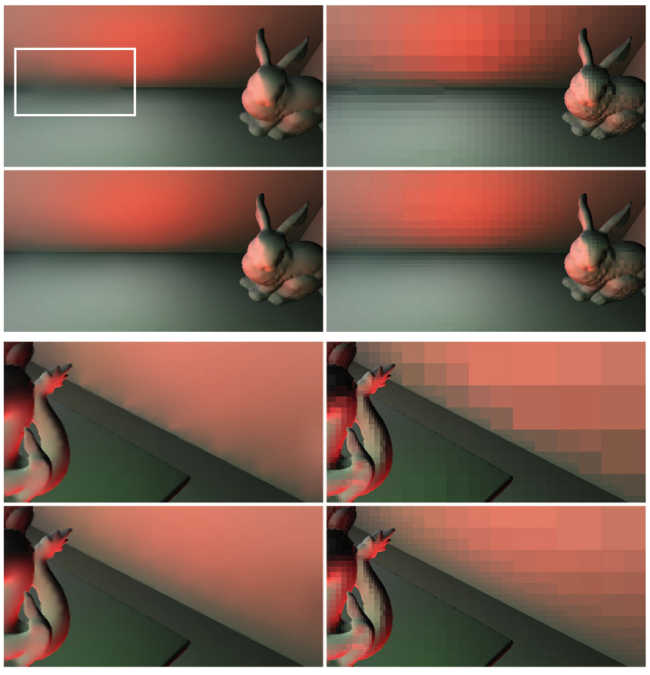


Fig. 15. Examples of upsampling artifacts, including (a) interpolated and (b) uninterpolated versions. Top group: artifacts arising from insufficient refinement along coarse mipmap boundaries (top row) and correct refinement (bottom row). Bottom group: artifacts from insufficient refinement along diagonals (top row) and correct refinement (bottom row).

distracting artifacts that flicker and pop under animation as subsplats are re-refined each frame.

Fig. 15 presents two examples of artifacts arising from inadequate refinement. In the top example, a horizontal edge lies on a mipmap boundary, and is not correctly detected as a discontinuity. The min-max mipmap can be constructed to detect discontinuities such as these by sampling outside the usual neighborhood, allowing refinement when a discontinuity is present not just in the cell being refined, but in any adjacent cell. However, this modification causes an excessive amount of refinement, incurring as much as a 40 percent speed penalty.

Artifacts also occur when a mipmap cell is situated diagonally to a discontinuity, as shown in the lower half of Fig. 15. To address these, when evaluating whether to refine a given subsplat, we detect discontinuities not just in the current subsplat, but also in all subsplats situated diagonally. This does result in overrefinement (and a roughly 2-3 percent speed penalty), but it is necessary to minimize these particularly distracting artifacts.

## 6 Future Work

The performance of our method depends on the set of refined subsplats, and visual quality depends on the correct interpolation and upsampling of that set. We plan to further address both of these aspects of our method, improving both the refinement and upsampling processes, leading to an increase in performance and visual quality. In addition, several other avenues of future work exist, each of which would improve the applicability and performance of our method.

First, we are working toward choosing VPLs intelligently rather than rendering each subsplat with the same fixed set. While we liken our approach to hierarchical radiosity, we currently use adaptive subdivision only for the receiving surfaces of the scene. Applying similar ideas to the selection and refinement of VPLs will allow us to simulate the effects of a large number of VPLs, while in actuality using a relatively small number. We also plan to intelligently select the subset of VPLs used during rendering, further improving performance.

Furthermore, we plan to continue exploring the use of our method with nondiffuse surfaces. The primary limitation of our current approach is the excess generation of subsplats, which quickly overwhelms memory and bandwidth resources. This can be improved with a more intelligent refinement method. One approach might first generate a list of subsplats independently of any VPL, as we currently do with diffuse surfaces, and then iteratively refine subsplats only as needed for each VPL. The result would be a dramatic reduction in the total number of subsplats, increasing performance.

Finally, we believe that our multiresolution approach is applicable to any number of additional rendering problems, such as caustics or shadows, that do not require all areas of an image to be high resolution.

## 7 Conclusion

This paper introduced a novel multiresolution splatting technique, and described its application to the rendering of indirect illumination with a reflective shadow map. Our method reduces the cost of rendering indirect illumination by rendering each piece of it at the lowest possible resolution. This allows indirect illumination to be rendered at high resolutions at interactive rates, without artificially restricting each VPL's ability to contribute illumination to the entire scene.

We discussed a variant of our method that works efficiently with diffuse surfaces, and a more general form that allows its use with arbitrary BRDFs. We discussed artifacts that can arise and potential solutions, presented a simple and efficient method of adding plausible detail to indirect illumination, and evaluated the effects of various parameters and configurations on performance. Finally, we discussed future directions and applications of our method. We believe that our multiresolution splatting methods will enable the rendering of many different global illumination methods.

## References

[1] S. Zhukov, A. Iones, and G. Kronin, "An Ambient Light Illumination Model," *Proc. Eurographics Rendering Workshop*, pp. 44-45, 1998.

[2] P.-P. Sloan, J. Kautz, and J. Snyder, "Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments," *ACM Trans. Graphics*, vol. 21, pp. 527-536, 2002.

[3] E. Tabellion and A. Lamorlette, "An Approximate Global Illumination System for Computer Generated Films," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 469-476, 2004.

[4] C. Dachsbacher and M. Stamminger, "Reflective Shadow Maps," *Proc. Symp. Interactive 3D Graphics and Games*, pp. 203-231, 2005.

[5] C. Dachsbacher and M. Stamminger, "Splatting Indirect Illumination," *Proc. Symp. Interactive 3D Graphics and Games*, pp. 93-100, 2006.

[6] A. Keller, "Instant Radiosity," *Proc. ACM SIGGRAPH,* 1997.
[7] P. Hanrahan, D. Salzman, and L. Aupperle, "A Rapid Hierarchical Radiosity Algorithm," *Proc. ACM SIGGRAPH,* 1991.
[8] M.F. Cohen and J.R. Wallace, *Radiosity and Realistic Image Synthesis.* Academic Press Professional, citeseer.ist.psu.edu/cohen93radiosity.html, 1993.
[9] M. Bunnell, "Dynamic Ambient Occlusion and Indirect Lighting," *GPU Gems 2,* Addison-Wesley, pp. 223-233, 2005.
[10] J. Kontkanen and S. Laine, "Ambient Occlusion Fields," *Proc. Symp. Interactive 3D Graphics and Games,* 2005.
[11] M. Malmer, F. Malmer, U. Assarsson, and N. Holzschuch, "Fast Precomputed Ambient Occlusion for Proximity Shadows," *J. Graphics Tools,* vol. 12, no. 2, pp. 59-71, 2007.
[12] A.W. Kristensen, T. Akenine-Möller, and H.W. Jensen, "Precomputed Local Radiance Transfer for Real-Time Lighting Design," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 1208-1215, 2005.
[13] K. Zhou, Y. Hu, S. Lin, B. Guo, and H.-Y. Shum, "Precomputed Shadow Fields for Dynamic Scenes," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 1196-1201, 2005.
[14] K. Iwasaki, Y. Dobashi, F. Yoshimoto, and T. Nishita, "Precomputed Radiance Transfer for Dynamic Scenes Taking into Account Light Interreflection," *Proc. Eurographics Symp. Rendering,* pp. 35-44, 2007.
[15] J. Kautz, J. Lehtinen, and T. Aila, "Hemispherical Rasterization for Self-Shadowing of Dynamic Objects," *Proc. Eurographics Symp. Rendering,* pp. 179-184, 2004.
[16] Z. Ren, R. Wang, J. Snyder, K. Zhou, X. Liu, B. Sun, P.-P. Sloan, H. Bao, Q. Peng, and B. Guo, "Real-Time Soft Shadows in Dynamic Scenes Using Spherical Harmonic Exponentiation," *ACM Trans. Graphics,* vol. 25, no. 3, pp. 977-986, 2006.
[17] S. Laine, H. Saransaari, J. Kontkanen, J. Lehtinen, and T. Aila, "Incremental Instant Radiosity for Real-Time Indirect Illumination," *Proc. Eurographics Symp. Rendering,* 2007.
[18] T. Ritschel, T. Grosch, J. Kautz, and S. Mueller, "Interactive Illumination with Coherent Shadow Maps," *Proc. Eurographics Symp. Rendering,* 2007.
[19] T. Ritschel, T. Grosch, M.H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz, "Imperfect Shadow Maps for Efficient Computation of Indirect Illumination," *Proc. ACM SIGGRAPH Asia,* 2008.
[20] C. Dachsbacher, M. Stamminger, G. Drettakis, and F. Durand, "Implicit Visibility and Antiradiance for Interactive Global Illumination," *ACM Trans. Graphics,* vol. 26, no. 3, p. 61, 2007.
[21] Z. Dong, J. Kautz, C. Theobalt, and H.-P. Seidel, "Interactive Global Illumination Using Implicit Visibility," *Proc. Pacific Graphics,* pp. 77-86, 2007.
[22] T. Saito and T. Takahashi, "Comprehensible Rendering of 3-d Shapes," *Proc. ACM SIGGRAPH,* pp. 197-206, 1990.
[23] H. Samet, *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.
[24] G. Guennebaud, L. Barthe, and M. Paulin, "Real-Time Soft Shadow Mapping by Backprojection," *Proc. Eurographics Symp. Rendering (EGSR),* pp. 227-234, 2006.
[25] N.A. Carr, J. Hoberock, K. Crane, and J.C. Hart, "Fast gpu Ray Tracing of Dynamic Meshes Using Geometry Images," *Proc. Graphics Interface Conf.,* pp. 203-209, 2006.
[26] A. Tevs, I. Ihrke, and H.-P. Seidel, "Maximum Mipmaps for Fast, Accurate, and Scalable Dynamic Height Field Rendering," *Proc. Symp. Interactive 3D Graphics and Games,* 2008.
[27] P. Gautron, J. Křivánek, K. Bouatouch, and S.N. Pattanaik, "Radiance Cache Splatting: A GPU-Friendly Global Illumination Algorithm," *Proc. Eurographics Symp. Rendering,* pp. 55-64, 2005.
[28] P. Shanmugam and O. Arikan, "Hardware Accelerated Ambient Occlusion Techniques on Gpus," *Proc. Symp. Interactive 3D Graphics and Games,* pp. 73-80, 2007.
[29] P.-P. Sloan, N. Govindaraju, D. Nowrouzezahrai, and J. Snyder, "Image-Based Proxy Accumulation for Real-Time Soft Global Illumination," *Proc. Pacific Graphics Conf.,* pp. 97-105, 2007.
[30] M. Shah, J. Konttinen, and S. Pattanaik, "Caustics Mapping: An Image-Space Technique for Real-Time Caustics," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 2, pp. 272-280, Mar./Apr. 2007.
[31] C. Wyman and C. Dachsbacher, "Reducing Noise in Image-Space Caustics with Variable-Sized Splatting," *J. Graphics Tools,* vol. 13, no. 1, pp. 1-17, 2008.
[32] R. Herzog, V. Havran, S. Kinuwaki, K. Myszkowski, and H.-P. Seidel, "Global Illumination Using Photon Ray Splatting," *Computer Graphics Forum,* vol. 26, no. 3, pp. 503-513, 2007.
[33] S. Rusinkiewicz and M. Levoy, "Qsplat: A Multiresolution Point Rendering System of Large Meshes," *Proc. ACM SIGGRAPH,* pp. 343-352, 2000.
[34] D. Laur and P. Hanrahan, "Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering," *Proc. ACM SIGGRAPH,* pp. 285-288, 1991.
[35] J. Lehtinen, M. Zwicker, E. Turquin, J. Kontkanen, F. Durand, F. Sillion, and T. Aila, "A Meshless Hierarchical Representation for Light Transport," *ACM Trans. Graphics,* vol. 27, no. 3, 2008.
[36] C. Wyman, "Hierarchical Caustic Maps," *Proc. Symp. Interactive 3D Graphics and Games,* 2008.
[37] G. Nichols and C. Wyman, "Multiresolution Splatting for Indirect Illumination," *Proc. Symp. Interactive 3D Graphics and Games (I3D '09),* 2009.
[38] P. Tole, F. Pellacini, B. Walter, and D. Greenberg, "Interactive Global Illumination in Dynamic Scenes," *Proc. ACM SIGGRAPH,* pp. 537-546, 2002.
[39] D.P. Mitchell, "Generating Antialiased Images at Low Sampling Densities," *Proc. ACM SIGGRAPH '87,* pp. 65-72, 1987.
[40] R.B. Fisher, *From Surfaces to Objects: Computer Vision and Three Dimensional Scene Analysis.* John Wiley & Sons, Inc., 1989.

**Greg Nichols** received the BA degree in computer science from the Central College in 2002, and the master's degree in computer science from the University of Iowa in 2008. He is currently a graduate student at the University of Iowa, and expects to complete his PhD in computer science in 2010. His research deals primarily with interactive global illumination.

**Chris Wyman** received the BS degree in mathematics and computer science from the University of Minnesota in 1999, and the PhD degree in computer science from the University of Utah in 2004. He is an assistant professor in the Department of Computer Science at the University of Iowa. His research interests focus on interactive global illumination, including both diffuse color bleeding and specular focusing of light, but also extend to other interactive and realistic rendering problems, visualization, and perceptual issues in rendering. He is a member of the IEEE.