# Micro-Rendering for Scalable, Parallel Final Gathering

T. Ritschel* T. Engelhardt† T. Grosch* H.-P. Seidel* J. Kautz‡ C. Dachsbacher†

MPI Informatik* Universität Stuttgart† University College London‡

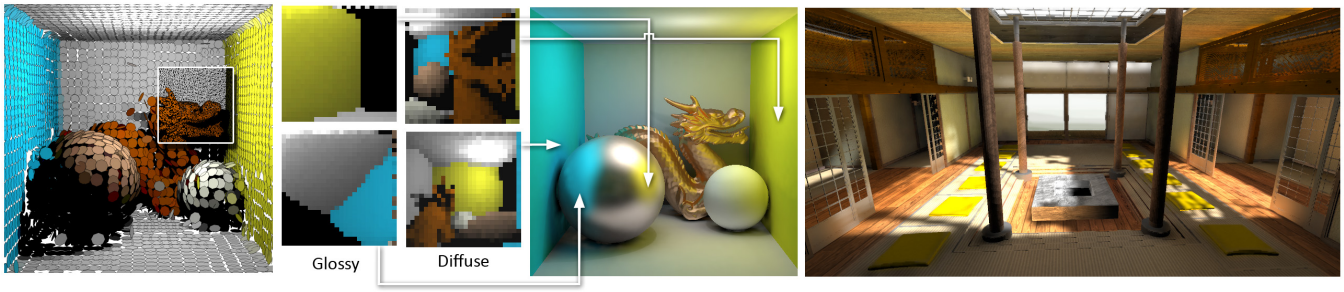


***Figure 1:*** *Our method computes global illumination by rasterizing many thousands of tiny micro-buffers (middle left) in parallel, using a sub-linear point rendering technique with an importance-warped projection. Two interior levels of the hierarchy with 1M points are shown on the left. The middle image renders at 1.1 Hz (512 × 512 res.). The right scene (700K triangles converted to 1M points) renders at 0.7 Hz.*

## Abstract

Recent approaches to global illumination for dynamic scenes achieve interactive frame rates by using coarse approximations to geometry, lighting, or both, which limits scene complexity and rendering quality. High-quality global illumination renderings of complex scenes are still limited to methods based on ray tracing. While conceptually simple, these techniques are computationally expensive. We present an efficient and scalable method to compute global illumination solutions at interactive rates for complex and dynamic scenes. Our method is based on parallel final gathering running entirely on the GPU. At each final gathering location we perform *micro-rendering*: we traverse and rasterize a hierarchical point-based scene representation into an importance-warped *micro-buffer*, which allows for BRDF importance sampling. The final reflected radiance is computed at each gathering location using the micro-buffers and is then stored in image-space. We can trade quality for speed by reducing the sampling rate of the gathering locations in conjunction with bilateral upsampling. We demonstrate the applicability of our method to interactive global illumination, the simulation of multiple indirect bounces, and to final gathering from photon maps.

**CR Categories:** I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Shading

**Keywords:** global illumination, real-time rendering, GPU, final gathering

## 1 Introduction

High-quality global illumination at interactive speed is a difficult challenge, especially in complex and fully dynamic scenes. On the one hand, conventional methods for off-line rendering produce photo-realistic results, but do not allow interactivity. On the other hand, available interactive or real-time techniques often harness the computational power available in modern GPUs, but suffer from various limitations, such as static geometry [Sloan et al. 2002], precomputed movement [Dachsbacher et al. 2007], low-frequency lighting [Sloan et al. 2007], or low-glossy indirect illumination [Ritschel et al. 2008].

In principle, a global illumination solution – at least for one indirect bounce – can be computed easily: render the directly illuminated scene from every visible point (either using rasterization or ray tracing) and convolve it with the BRDF. While this method, commonly called final gathering [Ward et al. 1988], is conceptually simple, it is quite expensive, and most GPU-based global illumination techniques try to avoid it, often leading to rather intricate methods. We return to the final gathering idea and propose a GPU-based rasterization approach that turns it into a highly efficient interactive technique. There are two main challenges that we face. First, the costly standard triangle-based rasterization needs to be avoided as it is inefficient when executed many times (e.g., for every surface point), and second, importance sampling is vital to reduce unnecessary computation but is difficult to incorporate on the GPU.

We present *micro-rendering* to address these issues: At the core, we perform final gathering of the incident illumination at a large number of visible surface points. Our final gathering technique operates in parallel on the GPU, and rasterizes the scene into a large number of *micro-buffers* by traversing a hierarchical point-based representation of it. The micro-buffers employ a non-linear projective mapping to allow for importance sampling of the BRDFs. Convolution of the incident illumination with the BRDF is a simple sum of each micro-buffers' content due to importance sampling. For high-quality results, we perform final gathering at every image location. Faster renderings can be achieved by performing final gathering at a subset of all image locations followed by bilateral upsampling [Sloan et al. 2007]. Our main contributions include:

- A novel, scalable GPU-based micro-rendering technique to efficiently gather incident radiance in large and dynamic scenes.
- A method to perform BRDF-based importance warping for rasterization of a point-based hierarchical scene representation.
- Techniques for the efficient computation of multiple-bounce indirect illumination and photon mapping walkthroughs.

We demonstrate our method for complex, fully dynamic scenes with indirect, diffuse and glossy illumination, for final gathering from photon maps, and for radiosity-style global illumination.

### 1.1 Related Work

We briefly review related work, with a focus on final gathering and global illumination (GI).

**Final Gathering** Final gathering refers to the process of calculating the amount of indirect illumination at a surface point. For instance, it is an integral part of photon mapping [Jensen 1996]. It is usually the most time consuming part in GI and, not surprisingly, many algorithms aim at reducing the final gathering cost.

Irradiance caching [Ward et al. 1988] performs final gathering only at a sparse set of locations in the image, and interpolates irradiance values at other locations. Irradiance gradients [Ward and Heckbert 1992] additionally compute the gradient of the irradiance to enable better interpolation. Křivánek et al. [2005] extended irradiance caching to glossy surfaces by storing incident radiance instead of irradiance. Our micro-rendering technique speeds up the final gathering process itself and is independent of the cache placement strategy.

The lightcuts method [Walter et al. 2005] pursues a similar goal, but assumes that all (indirect) lighting is represented as a hierarchical collection of point lights. For every visible surface point, a cut through the hierarchy is computed, and the contributions of all clustered point lights are summed up; visibility is resolved by raycasting toward the center of each cluster. In contrast, our method uses a hierarchy of point samples, which we rasterize in parallel into micro-buffers at less cost while enabling more accurate visibility. Recently, lightcuts were also extended to combine several cuts through a hierarchy of lights, visibility, and BRDFs [Cheslack-Postava et al. 2008]. While this extension allows for interactive performance, it does assume static geometry. Matrix row-column sampling [Hašan et al. 2007] computes a single set of point lights instead of varying it per image location. However, importance sampling is not possible and rendering quality is limited due to the low number of gathering samples used. Recently, Christensen [2008] proposed a CPU-based method to speed up final gathering for diffuse and moderately glossy scenes using a point-based representation of direct illumination stored in an octree. At each irradiance cache location, distant points are rasterized into a cube map and nearby points are raycast, but since no importance warping is used, glossiness is directly limited by the buffer's resolution. Our method follows a similar goal: speeding up global illumination through a hierarchical representation of illumination. In contrast to Christensen [2008], our method exploits GPU compute power and employs importance sampling for arbitrary BRDFs – both for rasterization and on-demand raycasting. In concurrent work, Wang et al. [2009] demonstrate interactive global illumination using GPU-based final gathering with ray-tracing. It enables complex lighting effects but relies on sparse gathering locations for efficiency.

**Real-time GI** Real-time global illumination for static scenes is possible with a number of different techniques. One of the early methods is precomputed radiance transfer (PRT) [Sloan et al. 2002]. Most of the PRT variants require static geometry, although some recent extensions also allow the movement of rigid objects [Iwasaki et al. 2007]. Low-resolution dynamic scenes are possible, assuming low-frequency illumination [Ren et al. 2006; Sloan et al. 2007]. Recently, Lehtinen et al. [2008] presented an interactive PRT-based illumination method for static scenes using a hierarchical, point-based representation. Light transport was simulated in a similar manner to radiosity [Cohen and Wallace 1993]. We also employ a hierarchical point-based representation but only for emitters; receiver points are selected in image-space and gathering is only performed for those.

**Interactive GI in Dynamic Scenes** Interactive frame rates for GI in moderately complex scenes can be achieved using anti-radiance [Dachsbacher et al. 2007], but computing a link hierarchy for fully dynamic scenes requires additional work [Meyer et al. 2009]. Bunnel [2005] approximated ambient occlusion and indirect illumination in small scenes using a hierarchy of linked disc elements. Instant radiosity [Keller 1997] is the basis for many interactive GI approaches but is too slow for complex scenes in its basic form. However, it can be efficiently implemented on GPUs using reflective shadow maps [Dachsbacher and Stamminger 2005; Dachsbacher and Stamminger 2006] when indirect visibility is ignored. Imperfect shadow maps [Ritschel et al. 2008] achieve interactive frame rates for moderately complex and fully dynamic scenes using approximate visibility, but ultimately fail to handle large scenes due to a non-hierarchical point representation. Further, indirect shadows are generally smoothed out considerably. In contrast, our approach enables high-quality, indirect illumination for both diffuse and glossy scenes of high geometric complexity.

**Discussion** Most of the above techniques share the goal of evaluating the rendering equation more densely, when one of the factors inside the integral is high, and less densely everywhere else. This goal is also the inspiration of our technique but in contrast to many of the above techniques we evaluate everything on the fly. Traditional importance sampling in ray tracing [Dutré et al. 2006] is probably the best-known method that tries to focus computation on where it is most needed. It has usually been coupled with ray tracing, since visibility needs to be checked in arbitrary directions. We make this idea amenable to GPU-based rendering by rasterizing our hierarchy of points into warped micro-buffers, effectively performing importance sampling.

### 1.2 Overview

Our method allows the rendering of global illumination in fully dynamic scenes. It is scalable such that the user can trade rendering quality for speed, with a smooth transition ranging from fast previews to solutions that are close to ground truth. At the heart of our method is an efficient micro-rendering technique which performs a BRDF importance sampled final gathering of the incident radiance. The term "micro" refers to the low overhead of launching the rendering, as well as to the low resolution of the frame buffer. We demonstrate that our method runs an order of magnitude faster than previous approaches and reaches preview quality at interactive speeds of up to 10 frames per second. The key points of our algorithm are:

- We generate a hierarchical point-based representation of the scene's surfaces for adaptive level-of-detail rendering.
- Our novel micro-rendering technique facilitates a highly parallel rendering of arbitrary (in our case hemispherical) mappings on the GPU, in order to gather the incident radiance at many surface points at the same time.
- We integrate importance sampling into the micro-rendering allowing us to efficiently compute final gathering for diffuse and glossy surfaces with arbitrary BRDFs.
- For preview quality we compute final gathering at a subset of the image pixels and use bilateral upsampling [Sloan et al. 2007]. High-quality renderings at approximately 0.5 to 1 frames per second perform final gathering at every pixel ($512 \times 512$ res.).

Micro-rendering is beneficial for many different global illumination methods. It directly renders one-bounce indirect illumination when point samples are directly lit, and multiple bounces when used with instant radiosity techniques. It can also be used to compute and display radiosity solutions, and for interactive walkthroughs of photon mapping results with final gathering.

## 2 Scalable, Parallel Final Gathering

In this section we describe all steps of the micro-rendering method in detail, starting with the basic technique, which is then extended with importance sampling and bilateral upsampling.

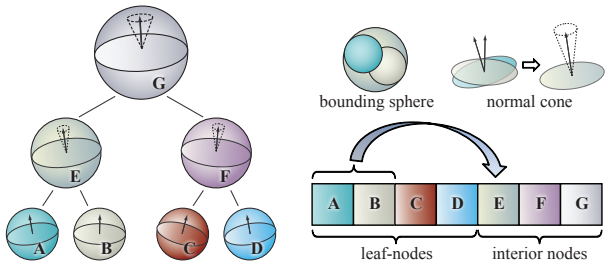


***Figure 2:*** *We represent the scene's surfaces using a point hierarchy, similar to QSplat, which is stored as a complete binary tree. This allows for easy traversal and fast updates at run-time.*

## 2.1 Hierarchical Point-Based Representation

Our micro-rendering method is based on a hierarchical point-based representation of the scene, since point-based representations allow for efficient level-of-detail rendering on GPUs [Dachsbacher et al. 2003; Ritschel et al. 2008], often provide a simple point selection criterion [Rusinkiewicz and Levoy 2000], and have shown to be well-suited for approximative global illumination [Christensen 2008]. In comparison to triangle-based rendering, point-based rendering has a lower setup and rasterization cost for low image resolutions. Similar to QSplat, we use a hierarchy of bounding spheres, where leaf nodes represent a single surface element (an oriented disc with radius $r$) and interior nodes represent a collection of surface elements. For rendering, this hierarchy is traversed starting from the root node: each node's bounding sphere is projected into micro screen space to compare its size to a given threshold. This determines if the respective disc is rendered and the traversal is terminated, or if its child-nodes are to be tested recursively. We replace the 'size-in-screen-space'-test of the original QSplat method with a test based on the solid-angle subtended by a node. This criterion will allow us to define *warped projective mappings*, and thus to integrate importance sampling easily. Note that warped mappings would be difficult to combine with triangle rasterization.

**Point Hierarchy Generation** The point-based representation of the scene is generated in an offline preprocessing step. First, we create random points on the triangles of the scene, proportional to the triangle areas, using a best candidate sampling. For every point sample, we store the triangle index and the barycentric coordinates of the point relative to its triangle. The barycentric coordinates allow us to recompute positions and normals for deforming geometry [Ritschel et al. 2008]. The point density, which is initially constant, determines the radius of the point samples. Under deformations we scale the points' radii to compensate for the varying point densities.

These point samples form the leaf nodes of our hierarchical point representation (Fig. 2). We build the hierarchy by computing a binary-space partitioning of the point samples, which we store as a complete binary tree (hence the number of points $n$ is a power of two). This enables us to compute skip-pointers on-the-fly during traversal, instead of storing additional offsets. The construction first sorts the leaf nodes and works as follows: we take the list of leaf nodes (the initially created point samples) as input and determine along which coordinate axis the point list has the largest extent. We then sort all points along this axis, split the list into two parts with an equal number of points, and recursively process both sublists in the same manner. In total, the cost for sorting all points is $O(n \log^2 n)$ for $n$ points. As we use a complete tree, the order of the points in the list implicitly defines the hierarchy. Consider the example shown in Fig. 2: nodes A to D are the leaf nodes after sorting. As the tree is complete, nodes A and B are children of the node E, and so on. For all interior nodes, we compute the minimum bounding sphere enclosing all child nodes, as well as the cone of normals (stored as direction plus cone angle).

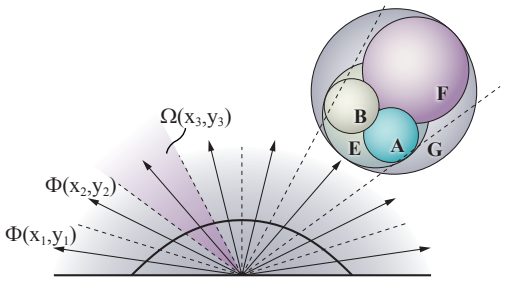


***Figure 3:*** *Every pixel $(x_i, y_i)$ of a micro-buffer corresponds to a direction $\Phi(x_i, y_i)$ and subtends a solid angle $\Omega(x_i, y_i)$. The point hierarchy is traversed and rasterized such that nodes project to no more than one pixel in the micro-buffer. In this example, the nodes A, B, and F, of the point hierarchy in Fig. 2, are selected.*

**Deforming and Moving Geometry** For deforming geometry, we leave the hierarchy itself unchanged and only update the per-node data. At run-time at the beginning of every frame, we recompute the leaf nodes' positions and normals, and update the interior nodes, i.e., we recompute the minimum bounding sphere and cone of normals, containing the two child-nodes' bounding spheres and normals, respectively. This process works bottom-up by successively merging two nodes at a time, yielding a total of $O(n)$ operations for $n$ leaves. This keeps the run-time cost for maintaining the point hierarchy low, and we can reasonably handle deforming geometry. For moving objects we create separate point hierarchies. The normals and normal cones are used for lighting computation and back-face culling during the point hierarchy traversal.

## 2.2 Final Gathering Using Micro-Rendering

Final gathering is used for high-quality renderings to compute the indirect illumination at every visible surface point $\mathbf{p}$. It involves gathering incident radiance $L_{\text{in}}(\omega_i)$ from direction $\omega_i$ of the upper hemisphere at $\mathbf{p}$. Usually BRDF importance sampling is used to gather more from directions that contribute more to the reflected radiance towards the observer. Due to the typically large number of gather directions involved, this is an expensive operation and commonly used in the context of offline rendering only. Our method enables parallel final gathering and achieves interactive frame rates through micro-rendering, which has been developed with the high parallelism of contemporary and future GPUs in mind. In the following we first detail the basic micro-rendering procedure for a single gather point. In Section 3 we then describe the implementation details, and how we ensure that the computational power of such hardware is utilised to a very high degree.

Micro-rendering generates images using the mapping $\Phi(x, y) = \omega$ relating a pixel $(x, y)$ of the micro-buffer to a gather direction $\omega$. We denote the solid angle subtended by the pixel under this mapping as $\Omega(x, y)$ (see Fig. 3). Such a mapping can be any standard hemispherical projection; however, we use our own mapping as described in the next subsection. The micro-buffers store an index of the nearest visible node as well as its distance at every pixel (i.e., we maintain an index and depth buffer); A micro-buffer is typically small, ranging from $8 \times 8$ to $24 \times 24$ pixels in our examples.

The basic image formation process starts with computing the cut in the point hierarchy, which also determines which point sample is visible for every micro-pixel. We then gather the incident radiance for every micro-pixel, and convolve it with the BRDF, yielding the radiance reflected towards the observer. Importance sampling can be integrated easily at little additional cost by changing the mapping function $\Phi(x, y)$ appropriately (Section 2.3).

**Point Hierarchy Cut** We compute the cut using a depth-first search in the point hierarchy starting from the root node. For each node, we evaluate the selection criterion: we first compute

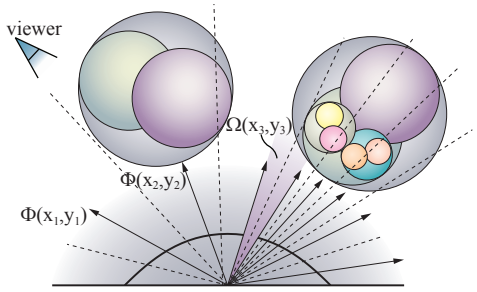


***Figure 4:*** *The warping function $\Phi(x_i, y_i) = \omega_i$ relates pixels in the micro-buffer to directions distributed according to the BRDF. Consequently the solid angle, $\Omega(x_i, y_i)$, corresponding to each pixel varies as well.*

the direction $\omega_i$ to the node's center, the solid angle $\Omega_i$ that it subtends, and the pixel $(x_i, y_i) = \Phi^{-1}(\omega_i)$ that the direction maps to. If $\Omega_i > \Omega(\Phi^{-1}(\omega_i))$, then the node is larger than 1 pixel under the projective mapping, and we will proceed with the child-nodes. Otherwise, we perform a depth test and if the node's distance is smaller than the depth value at $(x_i, y_i)$, we store its index and update the depth buffer.

**On-demand Ray Casting** When encountering leaf nodes, a further refinement is not possible. Such nodes potentially project onto several pixels, with possibly distorted shapes. We opt to resolve the exact visibility of such nodes using ray casting after computing the cut. To this end, we store their indices in a *post-traversal list*, which we maintain for every micro-rendering, for later processing. That is, after traversal and rasterizing all 1-pixel-sized nodes, we cast rays, one for every pixel in the micro-buffer, to find the closest intersection with the nodes in the post-traversal list and update the micro-buffer accordingly. From this, we obtain an accurate micro-rendering with water-tight surfaces.

**Convolution** After ray casting, the micro-buffer stores the index to the nearest visible node in the point hierarchy for every pixel. To obtain the reflected radiance, we compute the radiance $L_{\text{in}}(\omega_i)$ from every node, weight it by the respective solid angle, multiply by the BRDF, and sum up all contributions. Note that there are different strategies to obtain $L_{\text{in}}(\omega_i)$. We can determine it dynamically by extracting the position and normal of each node and computing the lighting including shadowing using shadow maps (for direct lighting or instant radiosity lighting). When using only diffuse surfaces, we can store the reflected radiance inside the point hierarchy [Christensen 2008], or we can obtain a radiance estimate for every node from a photon mapping solution (Fig. 8). We demonstrate results for all these strategies, and also describe a radiosity-like light propagation scheme using the point hierarchy in Section 4.

## 2.3 BRDF Importance Sampling

We have yet to specify the mapping $\Phi(x, y)$ that relates a pixel to a gather direction $\omega$. One could use a standard hemispherical parameterization, such as a Nusselt projection: $\Phi(x, y) = (x, y, \sqrt{1 - x^2 - y^2})$. However, if the surface point $\mathbf{p}$ is highly glossy, much of the information stored in the micro-buffer is practically irrelevant, as incident radiance from only a small solid angle (and thus few pixels) will be reflected towards the viewer. Therefore, we apply importance sampling of the BRDF. In the context of micro-rendering, this means that we require more pixels in the micro-buffer to correspond to important sample directions.

This can be achieved by defining the mapping $\Phi(x, y)$ appropriately. For a given point $\mathbf{p}$ in the scene, we know the current viewing direction $\omega_o$. We take the 2D light-dependent slice $f_{\text{r}}^{\omega_o}(\omega)$ of the BRDF $f_{\text{r}}(\omega_{\text{o}}, \omega)$ at that point, i.e., we fix the view direction $\omega_o$, and parameterize it by the x- and y-coordinate of $\omega$: $f_{\text{r}}^{\omega_o}(\omega_{\text{x}}, \omega_{\text{y}})$. After normalizing the slice with $1/\rho$, where $\rho = \int f_{\text{r}}^{\omega_o}(\omega)\text{d}\omega$, we

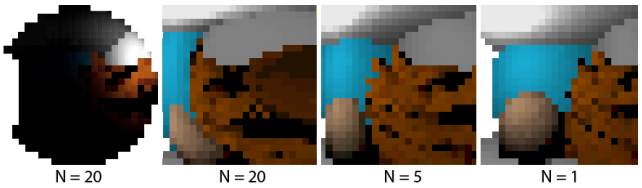


***Figure 5:*** *Micro-buffer for a glossy BRDF with a standard hemispherical mapping (BRDF -weighted) and with importance-warped mapping (Phong, $N = 20$). The importance-warped micro-buffer uses the available space more efficiently. On the right, we show the importance-warped micro-buffer for Phong $N = 5$ and $N = 1$.*

regard it as a 2D probability distribution function (PDF). We compute its inverse cumulative marginal and conditional distributions, $M^{-1}$ and $C^{-1}$, which are used to map from uniformly distributed $x$ and $y$ (the pixels) to $\omega_{\text{x}}(x) = M^{-1}(x)$ and $\omega_{\text{y}}(y) = C^{-1}(y|\omega_{\text{x}}(x))$. These are then used to define the importance-warped mapping $\Phi(x, y) = \left(\omega_{\text{x}}(x), \omega_{\text{y}}(y), \sqrt{1 - \omega_{\text{x}}(x)^2 - \omega_{\text{y}}(y)^2}\right)$. This particular mapping performs importance sampling according to $\omega_{\text{z}} f_{\text{r}}(\omega_{\text{o}}, \omega)$. The cosine term $\omega_{\text{z}}$ is implicitly included due to the chosen parameterization of the BRDF slice. Fig. 4 illustrates a BRDF-based mapping function $\Phi(x, y)$ and its associated $\Omega(x, y)$. Fig. 5 shows a micro-buffer for a glossy gather sample with and without importance-warping.

While for some BRDFs this mapping can be derived analytically (e.g., specular Phong component), we opt for generality; i.e., we always tabulate the PDF and compute the inverse distributions numerically. Note that we require the inverse mapping $\Phi^{-1}(\omega) = (x, y)$ in order to project a node onto our micro-buffer, while the forward mapping is required for ray casting. In practice, we compute both the forward and inverse mapping. Further, we need to know what the subtended solid angle of a pixel is, i.e., we need to define $\Omega(x, y)$ as well. We use a first-order approximation by taking the magnitude of the gradient of $\Phi(x, y)$. Note that we jitter the local coordinate system slightly at every gather sample to avoid banding artifacts.

As the micro-buffer is now importance-warped, a simple *sum* of all pixels scaled by $\rho$ yields the convolved indirect illumination; no more multiplication with the BRDF $f_{\text{r}}(\omega_{\mathbf{o}}, \omega)\omega_{\text{z}}$ or the pixel's solid angle is required.

**Discussion** Rendering with highly glossy materials is demanding in terms of warping, raycasting and point sampling. Although warping itself does not restrict glossiness if analytic formulas for mapping BRDFs are available, tabulation might miss features, effectively limiting glossiness. Highly glossy BRDFs also require more point samples in the hierarchy, because surface edges and textures can become visible in reflections. Also, the projection of more nodes is likely to be larger than 1 pixel in diameter, thus requiring raycasting.

Similar to our method, Jensen [1995] uses an importance-warped table to roughly record directions that contribute most to the reflected radiance in order to improve importance sampling for path tracing. This table is computed from nearby photons in a photon map only, thus possibly containing holes. In contrast, we use a BRDF-warped micro-buffer to accurately record all incident lighting, directly yielding reflected radiance after summation.

## 2.4 Bilateral Upsampling

For interactive previews we compute the indirect illumination at a lower resolution image only, and upsample the shading results using bilateral upsampling [Sloan et al. 2007] to the full resolution; direct lighting is always computed at full resolution. During upsampling, we avoid interpolating across discontinuities in the geometry, i.e., across silhouettes or over large differences in normal. For this, we compute interpolation weights as proposed in Sloan et al. [2007]. Indirect illumination is typically low-frequency and thus can be in-

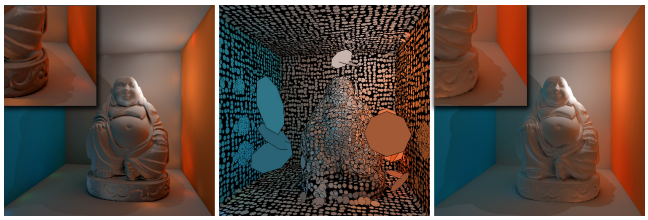

***Figure 6:*** *Multi-bounce indirect illumination for instant radiosity. Instead of directly using instant radiosity to illuminate a scene (left), we shade the points in our hierarchical scene representation with instant radiosity (visualized in the middle) and perform a final gathering step (right, 0.7 Hz, no upsampling). The additional final gathering step removes many of the artifacts of instant radiosity.*

terpolated and, in addition, direct illumination often masks possible artifacts.

Nevertheless, we can detect pixels where the interpolation is deficient. This is the case when the interpolation weights are nearly zero. For such pixels we can compute the indirect illumination in an additional render pass rather than interpolating (similar to [Dachsbacher and Stamminger 2005]). Unfortunately, a high overhead due to additional memory transfers (in our CUDA implementation), not the micro-rendering cost, slows down this approach significantly.

Preview quality using indirect illumination computed at $1/16$-th image resolution (i.e., ¼ in each dimension) typically renders at 4 to 10 Hz, whereas the full simulation runs at 0.5 to 1.0 Hz. There are more sophisticated techniques [Ward and Heckbert 1992] to determine where to compute indirect illumination, and how to interpolate irradiance. Note that these techniques could be combined with our method, but the integration into a GPU-based framework is challenging.

# 3 Implementation

Micro-rendering has been designed to exploit the parallelism of GPUs. We implemented our method using NVIDIA's CUDA and thus we will use the respective terminology in this section.

## 3.1 Data Structures

Our micro-rendering implementation is based on two important data structures: the scene's geometry stored as a point hierarchy, and the micro-buffers and post-traversal lists for efficiently computing parallel final gathering.

We create the point hierarchy in a preprocessing step (Section 2.1), and store it as an array in global memory that is used by all rendering threads. For each node, we allocate 128 bits storing a node's position and radius ($4 \times 16$ bits), normal and surface albedo (both quantized to $3 \times 5$ bits, packed into one 32 bit value), and the cone angle (quantized to 8 bits). In contrast to the original QSplat data structure, we store absolute positions and radii to allow direct access to interior nodes, and to avoid stack maintenance which is expensive on GPUs. This packing leaves 24 bits free, which we experimentally used to store the reflected radiance for every point sample. As expected, the clamping to 8 bits per channel corrupts the results, and we thus opt for storing the reflected radiance as half-floats in a separate array instead. The update of the point hierarchy data is done using CUDA at the beginning of every frame.

The micro-buffers are allocated in local memory, storing 32 bits for every pixel, out of which 8 bits are used for the depth component, and 24 bits for the node index. Thus our current implementation is limited to a point hierarchy with a maximum of $2^{24}$ nodes (i.e., $2^{23}$ point samples), which is significantly more than what we used in our examples. In addition to the micro-buffer, we allocate the post-traversal list of the same size providing space for the indices of the nodes whose visibility is resolved using ray casting.

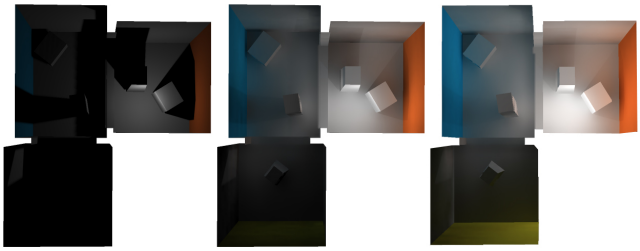

***Figure 7:*** *Direct illumination (left), one-bounce indirect illumination (middle) and two-bounce indirect illumination (right). Multiple bounces are computed at 5.0 Hz by gathering incident illumination at the hierarchy level 12 (4 k nodes) followed by a final gathering step at $128 \times 128$ surface locations. The final image is obtained from bilateral upsampling with anti-aliasing.*

## 3.2 Parallel Micro-Rendering using CUDA

In order to perform as many micro-renderings in parallel as possible, we launch one CUDA thread for each micro-rendering, i.e., every gather sample. Each thread first computes the BRDF warping functions, and then continues to compute the point hierarchy cut for the respective gather sample. Note that we compute the BRDF warping for every single micro-rendering, which allows for arbitrary, spatially-varying BRDFs. The output of this first step is a partially finished micro-buffer and the post-traversal list. Ray casting is typically only necessary for 10% to 30% of the micro-renderings (i.e., the post-traversal list is empty for the other 70%-90%). The threads with a non-empty list cast one ray for every micro-pixel, according to the tabulated warping function, and intersect it with the nodes stored in their respective post-traversal list to find the closest intersection. We also experimented with splitting the micro-rendering into two kernels: computing the warping and the point hierarchy cut in one thread, and next casting rays through all pixels of a micro-buffer in parallel by using many threads. However, the increased parallelism did not amortize due to the overhead of memory transfers required for this approach. Instead of ray casting we have also experimented with rasterization of large (multi-pixel) point samples, which turns out to be costly due to the warped projective mapping. We found the required accuracy, i.e., no holes in the micro-buffer, to be cheapest to achieve with ray casting.

A parallel execution works best if the instruction sequences of CUDA threads are as similar as possible. This is the case if two threads compute a similar cut and post-traversal list. In order to support this, we enumerate all micro-renderings according to a 3D Morton-order space-filling curve to provide high spatial coherence.

After filling the micro-buffers, each thread computes the reflected radiance by summing up its content. We then hand the results to OpenGL for bilateral upsampling and final display.

# 4 Applications

Our micro-rendering method can be used in various ways to achieve high-quality interactive renderings, and full-resolution renderings comparable to off-line ray tracing methods such as PBRT [Pharr and Humphreys 2004]. In this section we outline four applications.

**One-Bounce Indirect Illumination** When directly lighting the point hierarchy, e.g. using shadow mapping techniques, and then using micro-rendering for final gathering, we achieve renderings with one bounce of indirect light. As shown in Fig. 1 we can capture all $L\{S|D\}^2E$ light paths.

**Multiple Bounces with Instant Radiosity** The basic technique can be extended to handle additional diffuse bounces by using instant radiosity to generate a set of virtual point lights [Keller 1997], which is then used to illuminate the points in our hierarchy. This enables us to render $LD^*\{S|D\}^2E$ light paths (Fig. 6). Note that

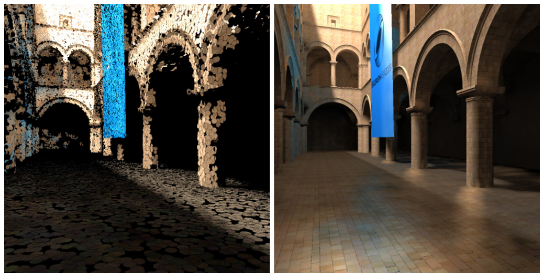

***Figure 8:*** *Final gathering for photon mapping. We first perform density estimation on the photon map (computed offline) for every point in our hierarchy (left). On the GPU, we then perform final gathering (right, note the glossy floor), which allows for interactive walkthroughs of photon mapped scenes (at 2 Hz) using bilateral upsampling with anti-aliasing from 128×128 to 1024×1024.*

the typical instant radiosity artifacts, such as bright splotches and shadow aliasing, diminish thanks to final gathering.

**Multiple Bounces with Radiosity** The hierarchical point representation of the scene's surfaces can also be used to compute light transport similar to hierarchical radiosity. We use micro-rendering to perform gathering in a Jacobi-iteration scheme: in every iteration we gather the indirect lighting at the point samples from other surfaces in the hierarchy to compute the reflected radiance. It is usually sufficient to do this for an interior node level and then to update the illumination of the entire hierarchy using push-pull [Cohen and Wallace 1993]. We initialize this computation with the direct illumination at all points. The final rendering pass consists of performing micro-rendering at every pixel, effectively displaying the radiosity solution using final gathering; see Fig. 7 (right).

**Photon Mapping** We can use our method to interactively display solutions stored in diffuse photon maps. In a preprocess, we obtain a radiance estimate for each leaf node of the point hierarchy through density estimation from the photon map, and compute the radiance values for the interior nodes using pull steps. Micro-rendering is then used for final gathering; see Fig. 8.

## 5 Results

In the following, we present our results rendered at interactive frame rates on a quad-core 2.4Ghz CPU with an NVIDIA GeForce 280 GTX. As mentioned before, all scene components (geometry, lighting, BRDFs) are allowed to change on-the-fly. Unless otherwise mentioned, result images are rendered at 512 × 512 pixels using 24 × 24 micro-buffers with one-bounce indirect illumination and direct lighting of the points in the hierarchy using shadow maps.

Fig. 14 shows four different scenes rendered with our method with gather samples at every pixel and at every 4th pixel (in each dimension) using bilateral upsampling, and we compare to a reference solution computed with Monte-Carlo path tracing [Pharr and Humphreys 2004]. Broadly speaking, the differences between the

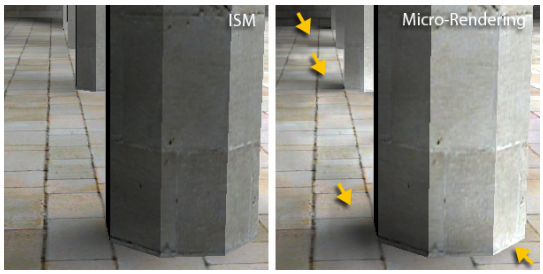


***Figure 9:*** *Comparison between ISMs [Ritschel et al. 2008] and our proposed method. At the same rendering speed (5 Hz) our method achieves higher quality than ISMs.*

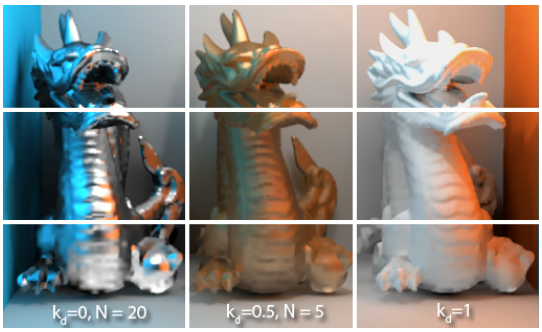


***Figure 10:*** *Final gathering at every pixel (0.83 Hz, top), 1/4 of all pixels (2.2 Hz, middle), and 1/16 (4.2 Hz, bottom). Preview quality can be achieved with a very low number of gather samples for diffuse or low-glossy materials.*

reference solution (taking minutes to hours to compute) and ours are minor. When using bilateral upsampling, the differences are slightly more perceptible; however, we achieve interactive frame rates of up to 10 Hz.

Fig. 9 compares our method using bilateral filtering to ISMs [Ritschel et al. 2008]. We adjusted the number of gather locations to achieve roughly the same speed as ISMs. Note how our method achieves superior quality in indirect shadows compared to ISMs. Furthermore, ISMs can only be used for low-glossy scenes, otherwise VPLs will become visible; this precludes scenes as shown in Fig. 1.

The number of required gather locations is scene dependent, and scenes with higher geometric complexity or glossy materials naturally require more samples. In Fig. 10 we compare the rendering quality using different numbers of sample locations. For diffuse surfaces 1/16 resolution is visually sufficient, whereas for the glossy dragon we need more locations (1/4 resolution).

Fig. 11 demonstrates what happens if we omit the post-traversal ray casting step. Holes occur in the micro-buffers and the rendering quality decreases, especially around edges. Post-traversal ray casting is therefore an integral step in our method.

In Fig. 12, we compare the influence of the size of the micro-buffer on rendering quality. 24 × 24 pixels (used for all other figures) is the maximum we can fit into local memory on our GPU but even lower resolutions still achieve acceptable results at a higher frame rate. However, note that at any resolution there is a possibility that we miss small holes (e.g., see the plant in Fig. 14), as is the case for all rasterization-based methods.

The performance of our method is sub-linear in the number of input points, which is to be expected from a hierarchical representation; see the red curve in Fig. 13, which indicates running time vs. scene complexity (dynamic horse scene represented with more and more points). The blue curve indicates that tree update is sub-linear until a hardware limitation is reached (at around $2^{20}$ point samples). As shown by the green curve in Fig. 13, we can process

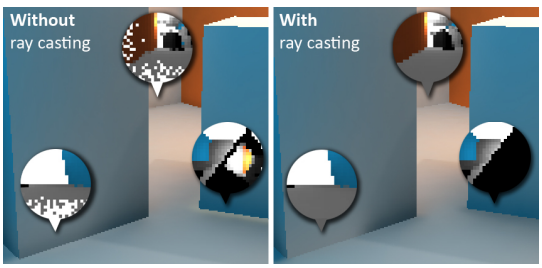


***Figure 11:*** *Micro-rendering with (2.5 Hz) and without (2.9 Hz) the post-traversal ray casting (256×256). Ray casting is an integral step that is needed for high-quality results, especially around edges.*

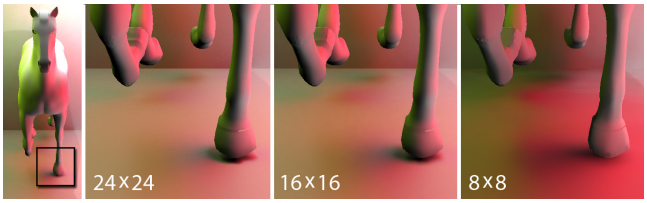


***Figure 12:*** *Influence of micro-buffer size on rendering quality (256 × 256). We use 8 × 8 (3.2 Hz), 16 × 16 (1.5 Hz), and 24 × 24 (0.7 Hz). Smaller sizes are faster but quality decreases.*

more gather samples per second when the total number of gather samples increases. This is to be expected due to the increased coherency between gather samples.

The computation time of our GPU-based final gathering technique is (roughly) split as follows for a typical scene, such as the teaser. 1.5% is spent on updating the hierarchy, 2% on building the view-dependent per-pixel mappings $\Phi(x, y)$, 18% on evaluating them, 60% on rasterizing the point hierarchy, 8% on ray casting, and 11% on bilateral upsampling, tonemapping and direct lighting.

### 5.1 Discussion and Limitations

In typical scenes, our method is able to compute about 150M final gathering samples with importance sampling (a 512 × 512 image with 24 × 24 micro-buffers at every pixel renders at about 1Hz, including tree update, shading, etc.). CPU-based ray tracing can send out about 10M rays per second per core, if the rays are coherent [Shevtsov et al. 2007], i.e., about an order of magnitude less than our method. Currently reported numbers on GPU-based ray tracing indicate that about 20M rays can be traced in dynamic scenes (including a complete rebuild of the acceleration structure) [Zhou et al. 2008].

While our method deals well with complex scene and arbitrary BRDFs, it has certain limitations. When glossy surfaces are present in a scene, more gather samples are required. In this case our simple regular gather sample distribution is not ideal and more elaborate distributions should be used [Křivánek et al. 2005]. As mentioned earlier, we have opted to prevent banding artifacts by jittering the local coordinate system at each pixel. As a result some noise is visible in our images. While our method renders one-bounce indirect caustics – simply by gathering from highly glossy surfaces – more than two specular bounces are not supported. Other effects, such as transparent objects and refractions are currently not simulated.

We implemented our method in CUDA to explore the potential of parallelizing the various micro-rendering tasks (see Section 3). In the end, performing all tasks in a single kernel was fastest. This suggests that a pure OpenGL implementation might be just as fast.

## 6 Conclusions and Future Work

We have presented a novel technique for scalable and parallel final gathering that enables the efficient computation of indirect illumination. It has been designed to harness the power of modern GPUs, and can trade rendering quality for computation time in a simple and intuitive manner. It handles large, fully dynamic scenes with diffuse and glossy surfaces. We demonstrate various applications of our method including single and multiple bounce indirect illumination, and the computation and display of radiosity solutions. Further, it can be used to compute final gathering for interactively rendering photon mapping results, yielding a speedup of an order of magnitude over a standard CPU implementation.

There are several possible avenues for future research. Our method is geared towards diffuse and glossy surfaces; highly specular surfaces require a sufficient number of gather locations as well as point samples to capture the illumination. We would like to investigate alternatives such as radiance caching on the GPU to handle

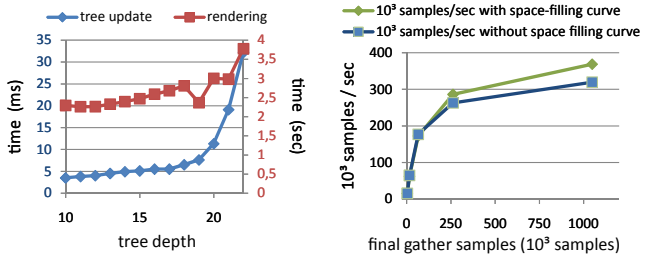


***Figure 13:*** *Left: the red curve indicates computation time vs. scene complexity (measured as tree depth, equals $2^N$ point samples). It indicates sub-linear complexity. The blue curve outlines tree-updating time. Right: the number of processed gather samples per second vs. the total number of gather samples in the image (green with, blue without space-filling curve to support spatial coherence).*

specular surfaces. Our hierarchical point representation allows for large and complex geometry; however, scenes with a high depth complexity, such as buildings with many rooms, would benefit from portal decompositions and efficient culling techniques.

## References


BUNNELL, M. 2005. Dynamic ambient occlusion and indirect lighting. In *GPU Gems 2*, M. Pharr, Ed. Add. Wesley, 223–233.

CHESLACK-POSTAVA, E., WANG, R., AKERLUND, O., AND PELLACINI, F. 2008. Fast, realistic lighting and material design using nonlinear cut approximation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 27*, 5, 128:1–128:10.

CHRISTENSEN, P. 2008. Point-based approximate color bleeding. Tech. Rep. 08-01, Pixar Animation Studios.

COHEN, M., AND WALLACE, J. 1993. *Radiosity and Realistic Image Synthesis*. Academic Press Professional.

DACHSBACHER, C., AND STAMMINGER, M. 2005. Reflective shadow maps. In *Proc. I3D*, 203–213.

DACHSBACHER, C., AND STAMMINGER, M. 2006. Splatting indirect illumination. In *Proc. I3D*, 93–100.

DACHSBACHER, C., VOGELGSANG, C., AND STAMMINGER, M. 2003. Sequential point trees. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3, 657–662.

DACHSBACHER, C., STAMMINGER, M., DRETTAKIS, G., AND DURAND, F. 2007. Implicit visibility and antiradiance for interactive global illumination. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3.

DUTRÉ, P., BALA, K., AND BEKAERT, P. 2006. *Advanced Global Illumination*. AK Peters.

HAŠAN, M., PELLACINI, F., AND BALA, K. 2007. Matrix row-column sampling for the many-light problem. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3, 26.

IWASAKI, K., DOBASHI, Y., YOSHIMOTO, F., AND NISHITA, T. 2007. Precomputed radiance transfer for dynamic scenes taking into account light interreflection. In *Proc. EGSR*, 35–44.

JENSEN, H. W. 1995. Importance driven path tracing using the photon map. In *Proc. ESGR*, 326–335.

JENSEN, H. W. 1996. Global illumination using photon maps. In *Proc. EGSR*, 21–30.

KELLER, A. 1997. Instant radiosity. In *SIGGRAPH '97*, 49–56.

KŘIVÁNEK, J., GAUTRON, P., PATTANAIK, S., AND BOUATOUCH, K. 2005. Radiance caching for efficient global illumination computation. *IEEE TVCG 11*, 5, 550–561.

LEHTINEN, J., ZWICKER, M., TURQUIN, E., KONTKANEN, J., DURAND, F., SILLION, F., AND AILA, T. 2008. A meshless hierarchical representation for light transport. *ACM Trans. Graph. (Proc. SIGGRAPH) 27*, 3, 37:1–37:9.

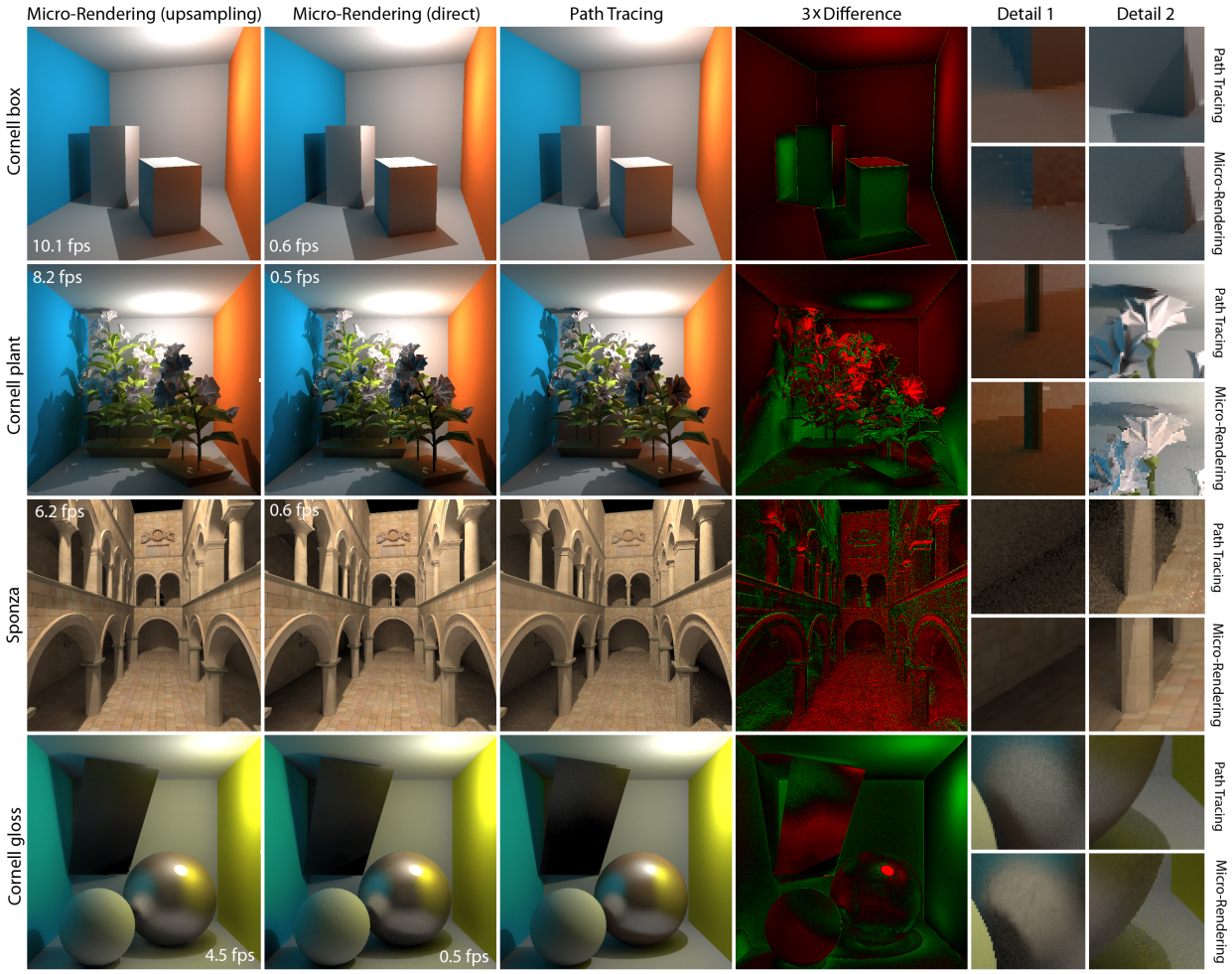


***Figure 14:*** *Comparison between 1. Fast preview images (1/16 resolution and upsampling), 2. Non-filtered micro-rendering for all* $512 \times 512$ *pixels and 3. PBRT path tracing. The simple Cornell box achieves high-quality global illumination results even with bilateral upsampling. The geometrically complex plant scene shows some slight differences (see insets), which we attribute to the discrete micro-buffers. For the Sponza scene, our method produces results that are indistinguishable from the reference rendering. In fact, bilateral upsampling removes noise and produces the visually most pleasing result. We also achieve very similar results for the glossy scene. However, as expected, bilateral upsampling changes the glossy reflection on the sphere slightly. The error images are scaled by a factor of three.*


MEYER, Q., EISENACHER, C., STAMMINGER, M., AND DACHSBACHER, C. 2009. Data-parallel hierarchical link creation for radiosity. In *Proc. EGPGV*, 65–70.

PHARR, M., AND HUMPHREYS, G. 2004. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann.

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

RITSCHEL, T., GROSCH, T., KIM, M. H., SEIDEL, H.-P., DACHSBACHER, C., AND KAUTZ, J. 2008. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 27*, 5, 129:1–129:8.

RUSINKIEWICZ, S., AND LEVOY, M. 2000. QSplat: A multiresolution point rendering system for large meshes. In *Proc. SIGGRAPH*, 343–352.

SHEVTSOV, M., SOUPIKOV, A., AND KAPUSTIN, A. 2007. Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes. *Computer Graphics Forum (Proc. Eurographics) 26*, 3, 395–404.

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

SLOAN, P.-P., GOVINDARAJU, N., NOWROUZEZAHRAI, D., AND SNYDER, J. 2007. Image-based proxy accumulation for real-time soft global illumination. In *Proc. Pacific Graphics*, 97–105.

WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. 2005. Lightcuts: A scalable approach to illumination. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3, 1098–1107.

WANG, R., WANG, R., ZHOUN, K., PAN, M., AND BAO, H. 2009. An efficient GPU-based approach for interactive global illumination. *ACM Trans. Graph. (SIGGRAPH) 28*, 3, 91:1–91:8.

WARD, G., AND HECKBERT, P. 1992. Irradiance gradients. In *Proc. EGSR*, 85–98.

WARD, G., RUBINSTEIN, F., AND CLEAR, R. 1988. A ray tracing solution for diffuse interreflection. In *Computer Graphics (Proc. SIGGRAPH)*, vol. 22, 85–92.

ZHOU, K., HOU, Q., WANG, R., AND GUO, B. 2008. Real-time kd-tree construction on graphics hardware. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 27*, 5, 126:1–126:11.