High Performance Graphics (2016)
Ulf Assarsson and Warren Hunt (Editors)

Deep G-Buffers for Stable Global Illumination Approximation

M. Mara, M. McGuire, D. Nowrouzezahrai, and D. Luebke

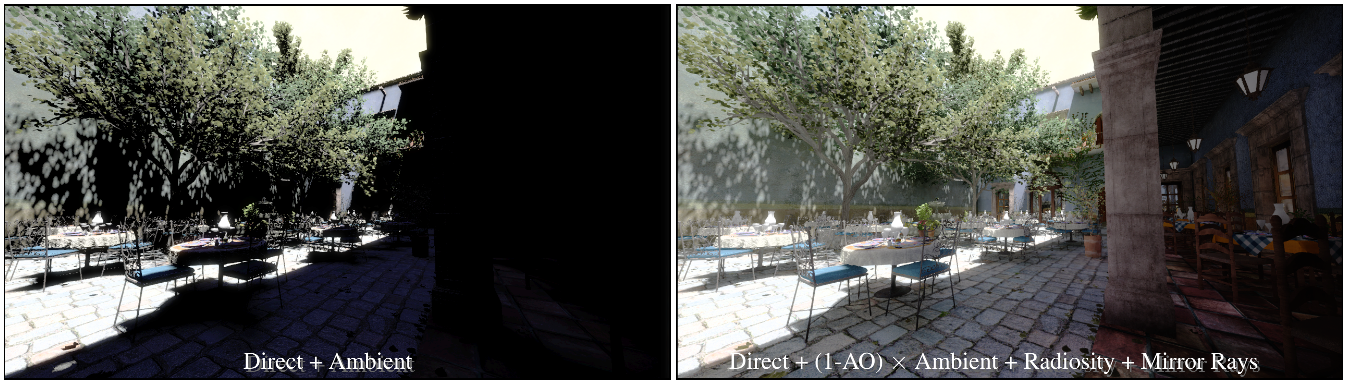


Figure 1: *Left:* Direct and hemispherical ambient illumination in *San Miguel* (6.5M triangles, 968 draw calls). *Right:* Direct lighting, approximate radiosity, mirror reflections, and AO computed from a two-layer Deep G-buffer in 5 ms at 1080p on NVIDIA GeForce 980. The G-buffer was generated in a single 5.8 ms geometry pass. See our evaluation section for faster results on more game-like scenes.

Abstract

We introduce a new hardware-accelerated method for constructing Deep G-buffers that is 2x-8x faster than the previous depth-peeling method and produces more stable results. We then build several high-performance shading algorithms atop our representation, including dynamic diffuse interreflection, ambient occlusion (AO), and screen-space mirror reflection effects.
Our construction method is order-independent, guarantees a minimum separation between layers, operates in a (small) bounded memory footprint, and does not require per-pixel sorting. Moreover, addressing the increasingly expensive cost of pre-rasterization, our approach requires only a single pass over the scene geometry. We include the pseudocode for our Deep G-buffer construction in the paper and the full source code of our technique in our supplemental document.

1. Introduction

Recent advances in graphics hardware have promoted the development and adoption of high-quality, dynamic, high-performance approximate global illumination techniques in interactive visualization and game engines. Here, maintaining consistent shading for dynamic view and lighting conditions, as well as adhering to extreme performance constraints, are more important than full physical accuracy. Among these interactive approximations, screen- and voxel-based techniques are the most adopted approaches due to their ability to balance robustness and performance.

Screen-space methods, the most common of which are variants of screen-space ambient occlusion (AO), map very well to current GPUs and can very efficiently approximate coarse hemispherical shading effects. Despite their popularity, well-known shortcomings include underestimated shading variation and inconsistent shading induced from their view-dependent sampling. On the other hand, world space voxel-based solutions can complement these limitations (often in tandem with screen-space techniques), however they are not as widely adopted due to scalability concerns and an inability to handle higher-frequency effects.

Given the benefits of combining screen- and voxel-space approaches, and motivated by Deep Geometry Buffers (G-buffers), we present a **practical** technique for constructing two-layer Deep G-buffers entirely on the GPU. We leverage our Deep G-buffers to devise **robust** and **fully interactive** shading algorithms. We focus on providing solutions that address the three bolded constraints above, which are essential to interactive graphics and gaming.

We improve the robustness of existing screen-space AO approaches, as well as extending them to indirect illumination effects with negligible additional cost, by carefully combining compression, cache management, sampling, and reconstruction. These applications all build atop our new GPU-friendly data structure: a deep geometry buffer with minimum separation, generated efficiently using a single pass over the scene geometry. While our shading applications are radiometrically approximate, we characterize the nature of the sampling errors we introduce and discuss our ability to scale to fully-converged physically accurate shading, assuming the necessary additional computational budget. While we only consider global illumination applications, we note that our Deep G-buffers can also be used to improve the robustness of other screen-space effects, including distribution effects like depth-of-field and motion blur, or reprojection-based shading techniques.

Our Deep G-buffer generation method is 1.5-2.0 times faster than depth peeling for producing the second layer at full resolution (Table 5), and can be up to five times faster than depth peeling at low resolution (Figure 11). Moreover, it designed to scale with current GPU architecture trends and usage scenarios. We minimize memory traffic by reading geometry from DRAM once and processing both layers on chip simultaneously, because memory speeds have increased slowly between GPU generations compared to ALU throughput due to parallelism. We observe that next-generation game engines have increasing pre-rasterization GPU workloads, such as tessellation and skinning, that make multiple passes over source geometry prohibitively expensive.

We are ultimately motivated by the potential applications of efficient and accurate Deep G-buffer generation, of which robust and dynamic indirect illumination is of immediate interest. We detail the implementation of our construction and shading techniques, both of which are straightforward to understand and implement, and we provide full pseudocode for the construction algorithm, full C++ and GLSL source code, and a standalone interactive indirect illumination demonstration application with complex scene content. Despite its simplicity, the design of our approach and its applications treat the complex interplay of bandwidth management and execution on modern GPU architectures, as well as providing a robust and efficient quasi-Monte Carlo integration schemes amenable to these architectures. We analyze our approach's quality and performance trade-offs, detailing the conditions under which it is guaranteed to execute efficiently.

Following the near- and far-field shading decomposition [AFO05], most existing interactive lighting solutions in modern game engines rely on combining approaches based on coarse decompositions of detail and scale. Our AO, indirect lighting, and ray-traced reflection applications all target the finest scale of dynamic radiometric detail and are fully compatible with existing coarse-scale solutions, such as precomputed static light probe shading. Despite the increasing geometry and material complexity in modern game content, normal maps and texture maps will likely remain an integral component of interactive content generation pipelines, due to the difficulties of scaling geometry and materials to sub-pixel resolutions. Similarly, interactive global illumination will continue to increase in scope and efficiency at coarse scales, but we purport that the illumination techniques presented in this paper are likely to remain useful for finer-scale lighting effects, at least in the foreseeable future of, e.g., interactive gaming.

Contributions Our two-layer Deep G-buffers builds on several ideas, including techniques that apply multiple views or layers to improve screen-space shading effects [SA07, RGS09, VPG13, DS05, Hac05, BS09]. Specifically, our contributions include:

1. an efficient Deep G-buffer construction method with minimum separation, on modern GPUs, in a single pass (Section 2.2),
2. a scalable AO algorithm for Deep G-buffers (Section 3.1),
3. a robust and coherent indirect illumination algorithm for Deep G-buffers (Section 3.2),
4. a camera-space quasi-Monte Carlo sampling sequence based on an empirical analysis of sampling strategies (Section 3.4),
5. a screen-space ray-tracer for Deep G-buffers (Section 3.6),
6. an extensive quantitative performance analysis (Section 4), and
7. a qualitative analysis of our shading approximation errors, used to motivate our Deep G-buffer spacing constraints (Section 4).

We validate the quality and performance of our approach with image and video sequences captured (interactively) on complex scenes, in Section 4 and our supplemental material. We provide full source code for an optimized implementation of our Deep G-buffer indirect illumination and AO solutions, both with temporal filtering. Our results improve upon existing widely-adopted and optimized single-layer shading solutions.

1.1. Related Work

Generating Layers Several approaches can render multiple geometry layers from a single view. In order of decreasing memory, these include: clip-space voxelization [Sch12, CG12], F-, A-, and ZZ-buffers, *k*-buffers and other bounded A-buffer approximations [LV00, MB07, BCL*07, SML11, Sal13], frequency A-buffer approximations [YK07, SA09, JB10], and depth peeling [Eve01, BM08]. Of these approaches, *depth peeling* is particularly interesting for effects that benefit most from a small number (i.e., two or three) of depth layers, since it has the smallest memory footprint.

Prior work has shown that the quality and robustness of screen-space global illumination can be significantly improved using even one additional layer [SA07, RGS09, VPG13]. As such, computing the second-closest camera-facing surface is an important operation and state-of-the-art approaches for computing these surfaces, using a single depth peel, either require two passes over the geometry [BM08] or a single pass with programmable blending [Sal13]. Furthermore, neither of these strategies guarantees a minimum separation between layers, which we show is necessary to compute high-quality and consistent shading. Our approach has much higher performance, does not require programmable blending, operates in a bounded and modest memory footprint, and is order-independent, requiring only one rasterization pass over the geometry.

Indirect Lighting Our shading models are related to directional occlusion [RGS09], a recent interactive AO variant [VPG13], and horizon-based AO [BS09, GN15]. The former uses multiple views, whereas the latter uses a two-layer depth buffer without any minimum separation guarantees. We conceptually extend these approaches to support multiple layers *with* minimum separation, and we show how to efficiently apply these structures to compute an arbitrary number of indirect bounces and specular reflections. Our entire shading approach is incorporated into a scalable gathering framework [MML12] and bears similarities to previous image-space gathering techniques [DS05, SHRH09, NRS14].

2. Generating Deep G-Buffers in a Single Pass

Motivating Multiple Layers Modern rasterization is tailored to visible surface determination and "local" shading operations. When "global" scene information is necessary for shading, rasterizing multiple views or layers can help to fill the gap. Shadow mapping is perhaps the earliest such example, where depth rasterized from the light's view is used to compute shadows from the camera's view. Reflective shadow maps [DS05] and orthographic depth peeling [Hac05] extend this idea to more complex effects, and other works have improved the robustness of screen-space techniques using many views [SA07, RGS09, VPG13].

Motivating Single-Pass Generation In order to better understand the penalty of rendering multiple views or multiple layers, we surveyed several industrial experts in high-performance game engine development and solicited rendering profiles from their respective engines: in all cases, the experts consistently report that one sixth to one third of a frame's render time is spent on operations that occur *prior* to rasterization in the graphics pipeline, including scene graph traversal, frustum and occlusion culling, tessellation, displacement mapping, procedural geometry generation, skeletal animation, and various transformations [Bra13, Buk13, McG13]. This implies that, even in the limiting case of populating a one-pixel G-buffer with zero rasterization or pixel processing overhead, processing all of the scene geometry twice in order to generate just two different views incurs a significant and often prohibitive cost. Furthermore, the pre-rasterization cost of the graphics pipeline has been increasing as culling and geometry processing become more sophisticated, and given the increase of animated/dynamic content.

Motivating Minimum Depth Separation We observe in practice that the second-closest surface to the camera is often not the second-most relevant surface for capturing information useful to shading: decals, non-convex geometry, and finer geometric details often introduce local structure that occludes the most useful secondary surface. For example, traditional depth peeling in *Sponza* reveals the second fold of the column's molding, and not the more radiometrically relevant red tapestry behind the column (Figure 2). To resolve this local structure problem, we enforce a *minimum separation* distance between layers. When generating our Deep G-buffers, we select only those fragments that are immediately accessible after a certain distance Δz past the visible surfaces.

Note that a k-buffer cannot resolve this problem in bounded memory, even with single-pass programmable blending variants [Sal13]. One would need more than a $k = 2$ buffer to guarantee minimum separation, since the goal is to output two *specific* layers from a $k = \infty$ buffer, not the first two layers. That is, until all surfaces have been rasterized, each pixel has no way of knowing the minimum acceptable depth for the second layer, so all surface fragments must be stored. Given this limitation, we proceed to describe a set of algorithms to robustly identify these important secondary surfaces within a small, bounded memory footprint.

2.1. A Strawperson Two-Pass Generation Algorithm

Listing 1 outlines a multi-pass depth peeling algorithm to generate a Deep G-buffer that respects our minimum depth constraint at frame t. Here, each frame buffer render target is a texture array, a feature supported by current GPUs, and we denote the depth buffer layers as $Z_t[0]$ and $Z_t[1]$. The geometry shader applies the current transformation T_t to each triangle, which comprises all model-view-projection and skinning transformations.

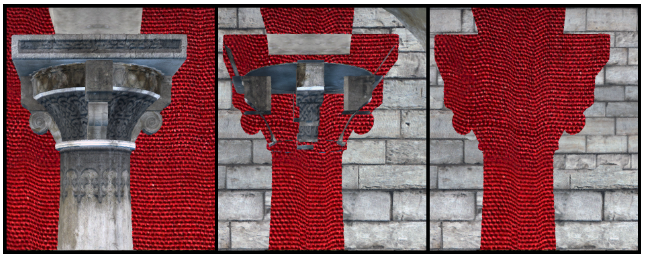

Primary   Traditional Peeling   Minimum Separation

Figure 2: Depth peeling (center) provides little additional information in areas with local structure, compared to nearest-surface rendering (left). Our minimum separation helps to capture the most relevant next significant surface.

For $\Delta z = 0$, this algorithm corresponds to traditional depth peeling [BM08] and, for $\Delta z > 0$, it guarantees a minimum separation. The pixel shader applies an arbitrary shading function S. For G-buffer generation, S would simply output material properties. It is possible (and often preferable on modern GPUs) to implement this algorithm using two separate frame buffers, without texture arrays and a geometry shader. We chose this structure to make the analogy and notation clear in the following section.

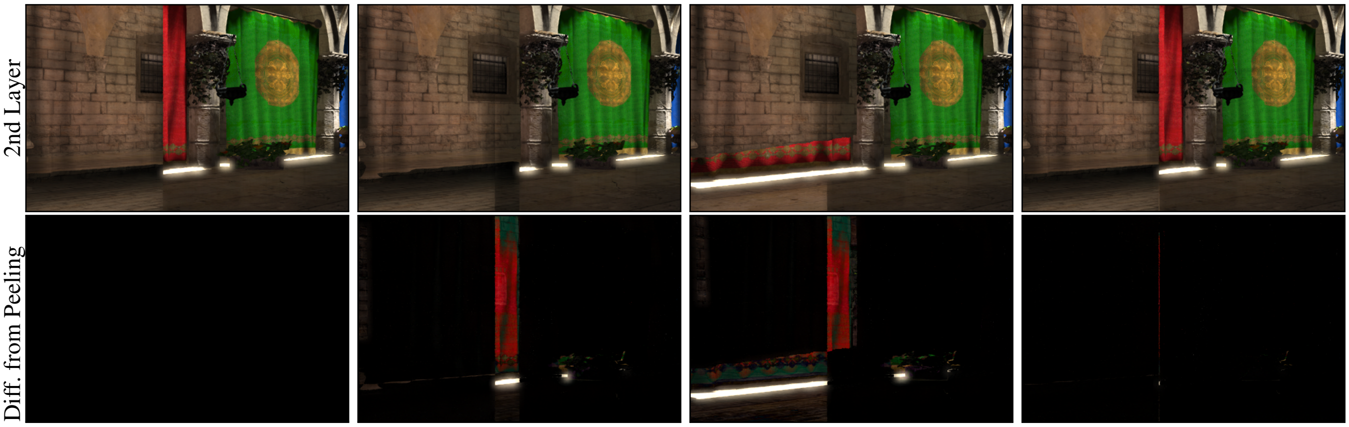
```
// 1st Pass
submit geometry with:
  geometryShader(tri):
     emit $T_t$(tri) to layer 0
  pixelShader($x, y, z$):
     return $S(x, y, z)$

// 2nd Pass
submit geometry with:
  geometryShader(tri):
    emit $T_t$(tri) to layer 1
  pixelShader($x, y, z$):
    if ($z > Z_t[0][x, y] + \Delta z$): return $S(x, y, z)$
    else: discard the fragment
```

Listing 1: A strawman *two-pass* Deep G-buffer generator with minimum separation Δz, using depth peeling. Our method (Section 2.2) improves significantly on this approach.

2.2. Efficient Single-Pass Deep G-buffer Generation

Listing 2 generates a two layer Deep G-buffer with minimum separation *in a single pass* over the geometry by rendering to both layers simultaneously. To identify fragments in the second layer, we require an oracle to predict the depth buffer's first layer *before* that buffer has been rendered to. We will detail four algorithm variants, each corresponding to a different oracle approximation, along with an optimized version of the REPROJECT variant.

DELAY Variant By adding a frame of latency, so that the next



(a) DELAY (b) PREVIOUS (c) PREDICT (d) REPROJECT

Figure 3: Top: second-layer surfaces captured by variants of Listing 2 with a moving camera in *Sponza*. Bottom: differences from ground truth produced by Listing 1. DELAY is perfect but has latency, while REPROJECT is nearly as good and adds no latency.

```
1   submit geometry with:
2     geometryShader(tri)
3         emit $T_t$(tri) to layer 0
4         emit $T_t$(tri) to layer 1
5         if (VARIANT == Delay) || (VARIANT == Predict):
6           emit $T_{t+1}$(tri) to layer 2
7
8     pixelShader($x,y,z$):
9       switch (layer):
10      case 0: // 1st layer; usual G-buffer pass
11        return $S(x,y,z)$
12
13      case 1: // 2nd G-buffer layer: choose the comparison texel
14        if (VARIANT == Delay) || (VARIANT == Predict):
15          $L = 2$ // Comparison layer
16          $C = (x,y,z)$ // Comparison texel
17        else if (VARIANT == Previous):
18          $L = 0; C = (x,y,z)$
19        else if (VARIANT == Reproject):
20          $L = 0; C = (x_{t-1}, y_{t-1}, z_{t-1})$
21
22        if ($z_C > Z_{t-1}[L][x_C, y_C] + \Delta z$): return $S(x,y,z)$
23        else: discard the fragment
24
25      case 2: // Depth only write to predict $Z_{t+1}[0]$; no shading
26        return // We only reach this case for Delay and Predict
```

Listing 2: Our efficient *single-pass* Deep G-buffer generator with minimum separation Δz.

frame's transformations T_{t+1} are known at render time, we can perfectly predict the next frame's first depth layer. Frame t reads (line 22) from the oracle computed in the previous frame, and generates the oracle for frame $t+1$ (lines 4, and 25-26) to satisfy the induction. This variant gives perfect output but requires one frame of latency; in certain cases (e.g., triple buffering) such a latency may already be present but, typically, we would like to avoid it.

PREVIOUS Variant By simply using the previous frame's first depth layer as an approximate oracle, approximation error increases only as object and camera motion increase. This can be acceptable in some cases for several reasons: first, errors will only appear in the second layer, not on visible surfaces; second, the errors are only in the minimum separation as the second layer still captures only surfaces at the correct positions at time t; lastly, there will only be errors in final moving objects, and we know that the perception of motion overrides the perception of precise intensities and even shape [SA11].

PREDICT Variant We can predict T_{t+1} using velocities from any underlying physics/animation simulation, or extrapolation from vertices at $t-1$ and t. When velocity prediction is accurate, this variant yields perfect results (equivalent to DELAY), but **without** latency. When it is inaccurate, the same disadvantages and arguments that hold for the PREVIOUS variant apply here.

REPROJECT Variant Here, we apply *reverse reprojection* [NSL*07] to perform a minimum separation test against the first depth layer from frame $t-1$: we use vertex positions from $t-1$ to compute the screen coordinates and depth C for the visibility test. Note that old depth values are not warped forward: instead, *visibility* is computed in the "past". This is susceptible to errors around moving objects, but less so than PREDICT since it can use perfect hindsight velocities from $t-1$. Note that many techniques require such velocities for use e.g. in screen-space motion blur and antialiasing.

Figure 3 (top) compares the second layer surfaces obtained from each variant, with fast camera motion in *Sponza*; Figure 3 (bottom) compares to ground truth minimum separation. PREVIOUS and PREDICT can produce large errors, while REPROJECT limits errors to tight regions around silhouettes and adds no latency. As such, we identify it as our principal solution.

Optimized REPROJECT Variant Simple geometry shaders are less expensive on the newest generation of GPUs. Furthermore, as of the NVIDIA Maxwell architectures, we can bypass generating multiple triangles in the geometry shader by using the new pass-through geometry shader and viewport multicast features of the hardware (exposed in OpenGL via the `NV_geometry_shader_passthrough` and `NV_viewport_array2` extensions). We compare the straightforward implementation of the REPROJECT variant to the optimized version leveraging these new features in the results section.

3. Applications to Global Illumination

Several applications can benefit from our layered Deep G-buffers, including stereo image reprojection, depth of field, transparency, motion blur, and global illumination. We focus on the latter.

We first extend screen-space AO to Deep G-buffers (Section 3.1), modulating local light probe shading by AO. Despite the popularity of screen-space AO, indirect illumination extensions have yet to find widespread adoption. We suspect this is primarily due the additional artifacts present in such single-layer screen-space solutions and we address this issue by generalizing our robust AO solution to single-bounce indirect illumination (Section 3.2). Multi-bounce indirect illumination (Section 3.3) is much more challenging as it requires a higher numerical integration sampling for low-error results. We extend our indirect solution to multiple bounces, adding temporal smoothing and reverse reprojection to amortize the additional computation, which reduces the cost to that of our single bounce solution per frame. Computing indirect illumination with Deep G-buffers is similar to reflective shadow mapping [DS05]: the main differences are that, by operating exclusively in camera space, we can amortize cost by using work already performed in a deferred-shading pipeline, allowing us to simulate more complex effects that involve objects visible to the viewer but not to the light. In our final application, we apply Deep G-buffers to mirror reflection tracing (Section 3.6).

As future work, we plan to investigate glossy reflections by either modifying the reflection rays to use pre-filtered incident lighting (computed on each layer of the Deep G-buffer) or by modifying the BSDF in our indirect illumination algorithm, depending on the footprint of the glossy integration lobe.

3.1. Ambient Occlusion

We extend Scalable Ambient Obscurance [MML12] (SAO) to leverage our layered Deep G-buffer, devising a sampling scheme that further improves its quality (Section 3.4). The original SAO algorithm compensates for undersampling behind primary surfaces (which dominates its error) with a coarser, biased estimator. Our improvements produce a more plausible shading falloff, avoid view-dependent halos on moving objects, and reduce noise.

Ambient Visibility $(1-AO)$ at a (view-space) point X is:

$$AV(X) = \max\left(0, 1-\sqrt{\frac{\pi}{N}\sum_{i=1}^{N}\max\left(0, A_i^0, A_i^1\right)}\right) \tag{1}$$

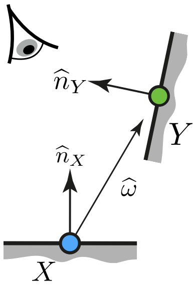
where we sample over occluding surfaces, $A_i^j = O(X, R(Z[j], i))$, N is the sample count, $R(Z,i)$ reconstructs the position of the i^{th} sample surface using the depth buffer Z, and O is the occlusion at X due to a sample at Y:

$$O(X,Y) = \left(1-\frac{\vec{v}\cdot\vec{v}}{r^2}\right)\cdot\max\left(\frac{\vec{v}\cdot\hat{n}_X-\beta}{\sqrt{\vec{v}\cdot\vec{v}+\varepsilon}}, 0\right), \tag{2}$$

where $\vec{v} = Y - X$, r is the sample pattern radius (see Section 3.4), and $\hat{n}_X$ is the normal at X. Equation 1 corresponds roughly to SAO's AV with a union of occluders in both layers, but without any of the ad-hoc falloff terms.

Our improved sampling (Section 3.4) benefits from explicit normals, and we pack camera-space Z and normal values for the two layers into a single texture each (see Table 1; note, radiosity inputs are unused for AO). For all our applications, we employ a modified bilateral reconstruction that includes normal and plane weights to prevent blurring across surface discontinuities [SGNS07].

3.2. Single-bounce Diffuse Indirect Illumination



Soler et al. [SHRH09] proposed a screen-space radiosity approximation that we extend in a radiometrically well-founded fashion. After doing so, we extend the approach to use Deep G-buffers, including performance and aesthetically motivated modifications.

The incident irradiance $E(X)$ at X due to outgoing diffuse radiance $B(Y)$ from the closest point Y in direction $\hat{\omega}$ is [CG85]

$$E(X) = \int_{\Omega}\frac{B(Y)}{\pi}\max(\hat{n}_X\cdot\hat{\omega}, 0)\, d\hat{\omega}\,. \tag{3}$$

We estimate this integral numerically as

$$E(X) \approx \frac{2\pi}{M}\sum_{\text{samples}} B(Y)\max(\hat{\omega}\cdot\hat{n}_X, 0), \tag{4}$$

where $\hat{\omega} = \vec{v}/||\vec{v}||$. The highest-quality version of our approximation samples N points Y from both G-buffer layers, but only uses the M for which both

$$(\hat{\omega}\cdot\hat{n}_X) > 0 \text{ and } (\hat{\omega}\cdot\hat{n}_Y) < 0. \tag{5}$$

As with AO, we assume mutual visibility between X and Y. We can significantly reduce the bandwidth requirements for our sampling process by omitting the second test in Equation 5 since, in this case, we need not access n_Y for each sample. Eliminating this test introduces bias in our indirect illumination approximation but, by allowing us to increase the effective sampling rate, we achieve a reduction in variance. Thus, the user can choose to do so depending on whether a less biased estimator is preferable to one that reduces noise. Incident irradiance at X is reflected as outgoing radiance as

$$B(X) = E(X)\cdot\rho_X\cdot\text{boost}(\rho_X), \tag{6}$$

where ρ_X is the diffuse reflectivity at X. We amplify it by

$$\text{boost}(\rho) = \frac{\max_\lambda \rho[\lambda] - \min_\lambda \rho[\lambda]}{\max_\lambda \rho[\lambda]}, \tag{7}$$

where λ is the wavelength or color channel. If we so choose, we can use $\text{boost}(\rho) = 1$ to conserve energy; if not, this boosting function can be used to emphasize scattering from saturated surfaces to enhance the perception of color bleeding. This is a common post-processing operation in interactive graphics, as it helps with visualizing intermediate results as well as providing a high-level aesthetic control often desirable in entertainment applications (i.e., see [Hal10]).

Texture Format	*Contents*	
RGBA8	Layer 0 Normal n (Oct16)	Layer 1 Normal n (Oct16)
R11G11B10F	Layer 0 Previous Bounce Radiosity B	
R11G11B10F	Layer 1 Previous Bounce Radiosity B	
RG32F	Layer 0 Camera-space z	
	Layer 1 Camera-space z	

Table 1: Input to our indirect illumination algorithm, packed into 160 bits/pixel to minimize bandwidth and fetch instructions.

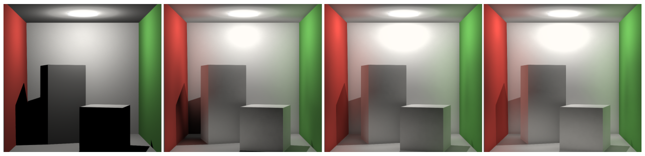
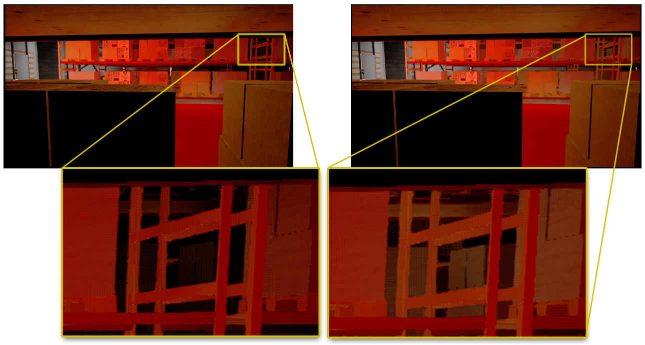
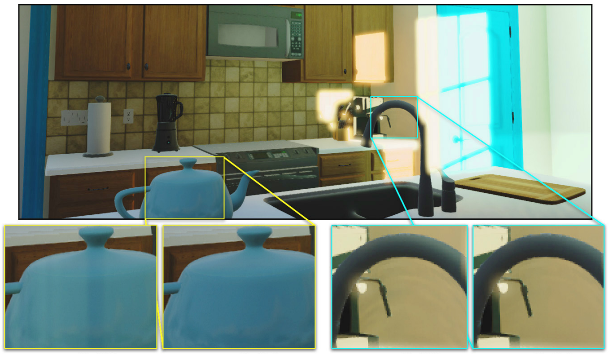


Figure 4: Illumination after 1, 2, and 100 frames = bounces.



a) Propagating first layer b) Deep propagation

Figure 5: Gathering indirect light in *Warehouse* from two G-buffer layers but only propagating within the first underestimates multi-bounce lighting (a) in areas of high depth complexity compared to two-layer propagation (b).



a) Incremental reprojection error b) Variance from disocclusions

Figure 6: Temporal filtering artifacts under vertical camera movement in *Kitchen* at $\alpha = 0.98$. Choosing $\alpha = 0.85$ eliminates these.

The diffuse radiance $B(Y)$ in the initial input is simply the Lambertian shading from (boosted) direct illumination. We iteratively re-apply Equations 4 and 6 (i.e., over multiple frames) to synthesize multiple indirect bounces (Section 3.3).

In addition to the indirect illumination, our shading pass computes a confidence value M/N at each pixel, corresponding to the fraction of samples that contribute to the final result. At pixels where confidence is close to 1, many nearby points were identified in the Deep G-buffer to produce a robust indirect shading approximation. At pixels where the confidence is ≈ 0, most samples from the Deep G-buffer were not representative of surfaces that could reflect light towards the pixel (i.e., since they were backfacing), so the result is unreliable. During final shading, we linearly interpolate between a coarse-scale or precomputed lighting solution and our dynamic Deep G-buffer indirect illumination, according to the confidence. All our results use static radiance and irradiance probes for the coarse-scale fallback, a common industry solution [GS12]; however, light maps, sparse voxel lighting, irradiance volumes, or per-vertex lighting are all viable alternatives.

Our indirect illumination algorithm uses Deep G-buffers as input in addition to data packed according to Table 1. Careful bandwidth management (both for DRAM and cache) is essential to high-performance computation on modern GPUs, and so data packing both optimizes the cache and amortizes the cost of issuing and executing texture fetches. We pack frequently sampled data into low precision and memory-adjacent locations, including camera-space depth (which, combined with projection information and texel location fully describes 3D scene position) for both layers into a single buffer, and we use the OCT16 encoding [CDE*14] to pack both layers' normals into a single RGBA8 buffer.

We additionally implement the cache coherence optimization of McGuire et al. [MML12], where a depth MIP-map computed using rotated-grid downsampling has been shown to improve performance when sampling over a large radius in screen space.

3.3. Multi-bounce Indirect Illumination

Computing multiple bounces of indirect light requires N integration samples per bounce iteration and, in order to decouple render cost from the number of bounces, we incorporate information *across* frames in two ways: first, we only increment illumination by one bounce per frame using *progressive computation*; second, we *filter temporally* using an extension of our bilateral reconstruction across time to pixels from the previous frame. In each case, we reverse-reproject sample locations to account for motion, which differs from our reverse-reprojection depth oracle (Section 2.2) but shares the same benefits and drawbacks.

Progressive Computation We accumulate additional light bounces using the previous frame's final indirect irradiance buffer E_{t-1} in Equation 6, simulating n bounces in n frames (Figure 4). Reprojection avoids ghosting in the presence of dynamic objects, but light will still linger for many frames on a surface. To reduce this artifact, we damp the forward propagation of E_{t-1} by a factor $0 < \delta \leq 1$, which (intentionally) underestimates illumination. We

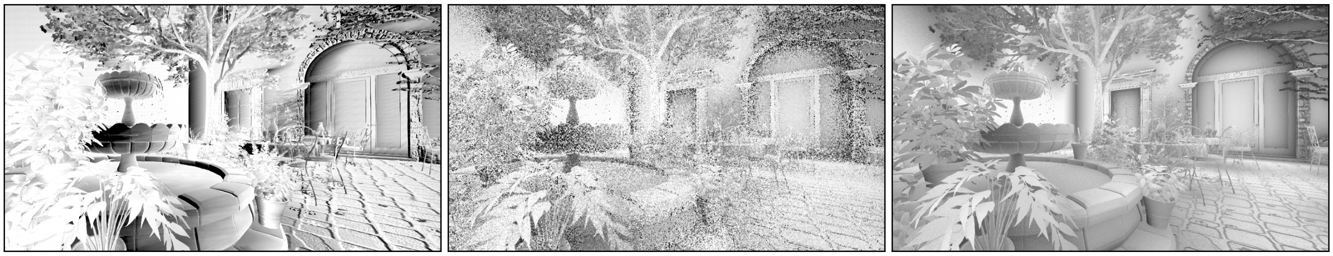


Figure 7: AO from 99 spiral taps *(left)* without rotation and a suboptimal τ, *(center)* unbiased in 2D via pattern rotation, and *(right)* with τ chosen using our screen-space QMC optimization. Results are shown without reconstruction to better illustrate the noise reduction.

```
// tau[N-1] = optimal number of spiral turns for N samples
const int tau[ ] = {1, 1, 2, 3, 2, 5, 2, 3, 2, 3, 3, 5, 5, 3, 4,
7, 5, 5, 7, 9, 8, 5, 5, 7, 7, 7, 8, 5, 8, 11, 12, 7, 10, 13, 8,
11, 8, 7, 14, 11, 11, 13, 12, 13, 19, 17, 13, 11, 18, 19, 11, 11,
14, 17, 21, 15, 16, 17, 18, 13, 17, 11, 17, 19, 18, 25, 18, 19,
19, 29, 21, 19, 27, 31, 29, 21, 18, 17, 29, 31, 31, 23, 18, 25,
26, 25, 23, 19, 34, 19, 27, 21, 25, 39, 29, 17, 21, 27};
```

Listing 3: Discrepancy-minimizing number of turns τ.

compensate for this bias with a small amount of environment lighting from static light probes, all according to our confidence value.

We also propagate indirect illumination *across* layers, which is essential for multiple bounces in scenes with high depth complexity (see Figure 5). The marginal cost of propagating to the second layer is negligible since it shares gathered samples from the first layer.

Temporal Filtering To reduce any remaining undersampling noise we apply an exponentially-weighted moving average $E_t = E(1-\alpha) + \mathrm{reproject}(E_{t-1})\alpha$ but use $E_t = E$ for pixels where the reprojected point is not within 1cm of either layer, which is indicative of an incorrect velocity estimate. We recommend (and use) $\alpha = 0.85$, except where noted. For $\alpha \geq 0.95$ we observe dynamic lighting latency and two types of artifacts may appear in each frame (Figure 6): despite detecting failed reprojections, ghosting can still result from incrementally accumulated reprojection errors (each within the 1cm threshold), and rejecting too many samples due to reprojection disocclusion increases the variance per pixel.

3.4. Quasi-Monte Carlo Sampling

For our AO and indirect illumination, we distribute N samples around each shade point in a spiral pattern with τ turns and radius r_p, similarly to McGuire et al. [MML12], however we optimize the pattern's parameters to minimize (2D) discrepancy for quasi-Monte Carlo (QMC) integration. We amortize computation over layers by sampling the same points in each. The i^{th} sample at (x,y) is accessed from texel $(x,y) + h_i\,\hat{u}_i$, where $h_i = r_p \kappa_i$, $\hat{u}_i = (\cos\theta_i, \sin\theta_i)$, $\theta_i = 2\pi\kappa_i\tau + \phi$, and $\kappa_i = (i+0.5)/N$. We rotate all samples by an azimuthal angle ϕ chosen according to a hash on (x,y), and the sample MIP level m_i is $m_i = \lfloor \log_2(h_i/q) \rfloor$. The constant q is the screen-space radius at which we first increment MIP levels, chosen based on the texture cache size.

We precompute the optimal values of τ (to the nearest integer; see Listing 3) that minimize discrepancy for each N and choose the appropriate value at run-time, whereas McGuire et al. [MML12] manually computed $\tau = 7$ for a fixed $N = 9$, and so their shading quality is suboptimal when $\tau = 7$ is used for $N \neq 9$. Figure 7 illustrates the impact of our optimized QMC sample placement: all three images have equal render time and use 99 AO samples. The left-most image has high discrepancy ($\tau = 7$) and exhibits banding since all samples use $\phi = 0$, the center image rotates along ϕ, but the impact of discrepancy still manifests itself as visual clusters of noise; choosing the optimal τ yields a higher quality result (right).

3.5. Recommended Radiosity Parameters

There are five content-independent parameters for our indirect illumination method. These should be chosen based on the desired performance and image quality tradeoff. We recommend three parameter sets in Table 2, which are also supplied as presets in our demo application. Increasing the number of samples N (from which we determine the number of spiral turns τ, according to Listing 3) reduces variance. Increasing the number of spatial reconstruction filter taps reduces noise in the final image, but also blurs high-frequency illumination. Including the n_Y test (Equation 5) improves contrast and reduces bias. Increasing the minimum MIP level when computing indirect illumination can increase variance in the low-frequency terms, leading to large-scale flickering, but has a significant impact on performance since it affects cache coherence. The Deep G-buffer input fills a guard band around the frame to help stabilize results under camera motion. The output can fill a more narrow guard band since it only contributes to the previous bounce's result. Thus, a user can increase performance at the expense of robustness for multi-bounce indirect light by reducing the fraction of the guard band for which we compute indirect illumination.

We tuned the High Performance parameter set to minimize evaluation time for the lowest image quality we found acceptable. It barely suppresses flicker and noise artifacts and provides heavily biased results, but it is still stable and fast. This is what one might desire for a game with strict performance constraints. We tuned the High Quality parameter set until further parameter changes led to negligible increase in quality. The Balanced parameter set is at the knee in our perceived quality vs. performance curve, and we recommend it for games with smoother camera movement.

3.6. Reflection Ray Tracing

We adapt screen-space mirror reflection [SKS11] to Deep G-buffers, and Section 4 illustrates results with this effect in addi-

Preset	*Indirect Samples (N)*	*Reconstruction Filter Taps*	*Use $\hat{n}_Y$ Test (Equation 5)*	*Minimum MIP Level*	*Fraction of Guard Band Shaded*
HIGH PERFORMANCE	13	9	No	3	10%
BALANCED	14	11	Yes	2	50%
HIGH QUALITY	30	13	Yes	0	80%

Table 2: Three parameter sets for our radiosity algorithm.

tion to our indirect illumination. We march reflection rays in camera space, projecting each point into both G-buffer layers: we treat rays that lie within $[z, z+\Delta z]$ of either of the G-buffers' (x,y,z) positions for a pixel as a hit and, here, outgoing radiance is simply the incoming radiance along the reflection direction. After a maximum distance, or once the ray exits the guard band, we revert to mirror reflection environment map lookups. Our supplement includes a full implementation with this feature.

4. Evaluation

We evaluate our single-pass layered Deep G-buffer **construction** with minimum separation, and its application to global illumination (GI) in several scenes (see Table 3). All results were measured at 1080p (i.e., 1920×1080) on a NVIDIA GeForce GTX 980 GPU.

4.1. Performance

Table 5 shows that both of our single-pass construction REPROJECT variants outperform depth peeling on all scenes. On complex scenes such as *San Miguel*, our optimized REPROJECT variant provides even more of a performance improvement the standard REPROJECT variant. Table 4 illustrates that the incremental cost of including an additional layer for GI computation is small. Our algorithms amortize the cost of pixel iteration, sample tap computation, and framebuffer overhead – only bandwidth costs increase measurably when adding more samples.

Scene	Indirect Illumination [ms]			AO
	Max Perf.	*Balanced*	*Max Quality*	[ms]
Kitchen	2.1 + 0.5	3.2 + 0.4	5.4 + 1.0	1.4 + 0.1
Sponza	2.0 + 0.7	3.4 + 0.5	6.3 + 0.9	1.4 + 0.0
Old City	2.1 + 0.4	3.5 + 0.4	6.1 + 0.5	1.8 + 0.1
Dockside	1.8 + 0.5	3.2 + 0.3	6.1 + 0.3	1.7 + 0.1
Op925	2.2 + 0.5	3.6 + 0.3	6.3 + 0.3	1.7 + 0.0
San Miguel	2.2 + 0.5	3.5 + 0.5	6.0 + 0.7	1.7 + 0.0

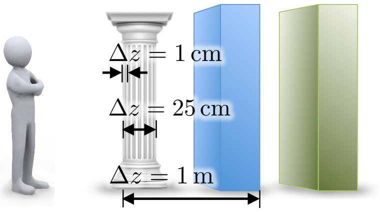
Table 4: Execution times for two-layer Deep G-buffer GI (including spatial and temporal reconstruction filtering), formatted as 1st layer time + 2nd layer time. Amortizing the overhead reduces the incremental cost for the 2nd layer. For scenes with mirror reflectors, ray tracing cost *Kitchen*: 1.3 + 0.3; *Dockside*: 1.7 + 0.1; and *San Miguel*: 1.0 + 0.2.

		Layer 2 [ms]				
	Layer 1	*Depth*	REPROJECT		PREVIOUS	PREDICT/
Scene	[ms]	*Peel*	Optimized	Standard		DELAY
San Miguel	4.1	4.0	**1.7**	2.7	2.7	25.8
Kitchen	2.2	2.2	**0.3**	0.4	0.3	1.5
Sponza	1.5	1.5	**0.9**	**0.9**	0.9	3.4
Dockside	2.1	2.0	**1.1**	1.2	1.1	7.4
Op925	2.7	2.8	**0.8**	1.2	1.2	8.3
Old City	1.2	1.1	**0.7**	0.9	0.9	6.1
Office	0.2	0.2	**0.1**	**0.1**	0.1	0.2
Warehouse	2.4	2.4	**0.9**	1.0	1.0	3.4

Table 5: Deep G-buffer construction times at 1080p, with the fastest method for a full-resolution second layer in **bold** for each scene/row. Our optimized REPROJECT variant produces the second layer in about half the time of single-layer rendering, on all scenes.

4.2. Parameter Selection

G-buffer construction depends on a scene-dependent minimum separation constant, Δz. If Δz is too small, then the second layer will capture superfluous local detail. If Δz is too large, then the second layer will capture surfaces that are too distant, potentially missing important features: e.g., in the inlined figure, $\Delta z = 1$ m fails to capture the blue wall behind the column and will instead "see through" to the green wall.



Scene	*Source*	*Tris.*	*Chars.*	*Meshes*
Office	g3d.sf.net	10k	0	17
Kitchen	turbosquid.com	370k	0	77
Warehouse	turbosquid.com	640k	34	89
Sponza	Crytek	850k	0	56
Old City	turbosquid.com	1.2M	0	100
Dockside	*Call of Duty: Black Ops 2*	2.3M	8	20
Op925	*Battlefield 3*	2.8M	32	66
San Miguel	Evolucién Visual	5.9M	0	1196

Table 3: Triangle, animated character, and mesh counts for scenes.

We have, however, consistently observed robust and stable image quality for a wide range of Δz settings, even on scenes with high depth complexity. We use $\Delta z = 50$ cm for every result in this paper. Supplemental Figure 2 illustrates the impact of the temporal weight α on undersampling noise: our temporal filter is an exponentially-weighted moving average, so the useful range of α is on the high end of the $[0,1)$ interval. We generally recommend $\alpha = 0.85$.

4.3. Image Quality

As few as two layers can significantly improve the appearance of scenes that have high depth variance and depth complexity. Figures 15 and 16 illustrate our robustness to occlusion and viewpoint changes in scenes with indirect illumination, and supplemental Figures 3 and 4 highlight the importance, and validate the necessity, of enforcing a minimum separation distance in addition to the additional layer: we capture important lighting features by enforc-

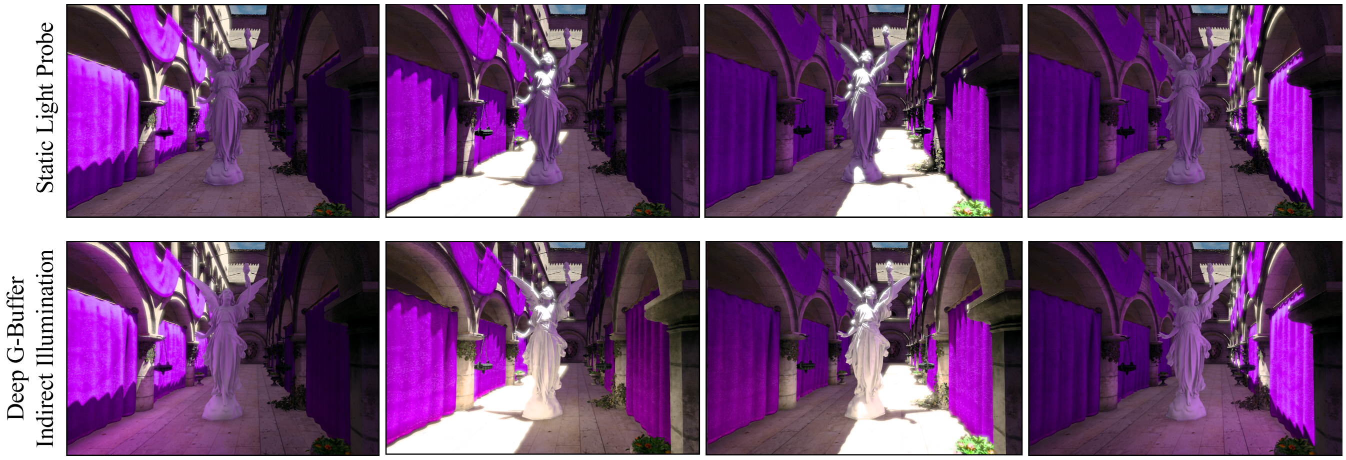
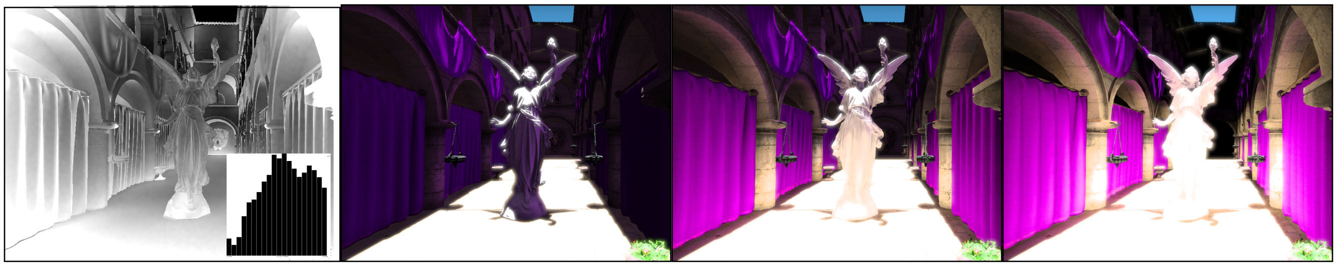
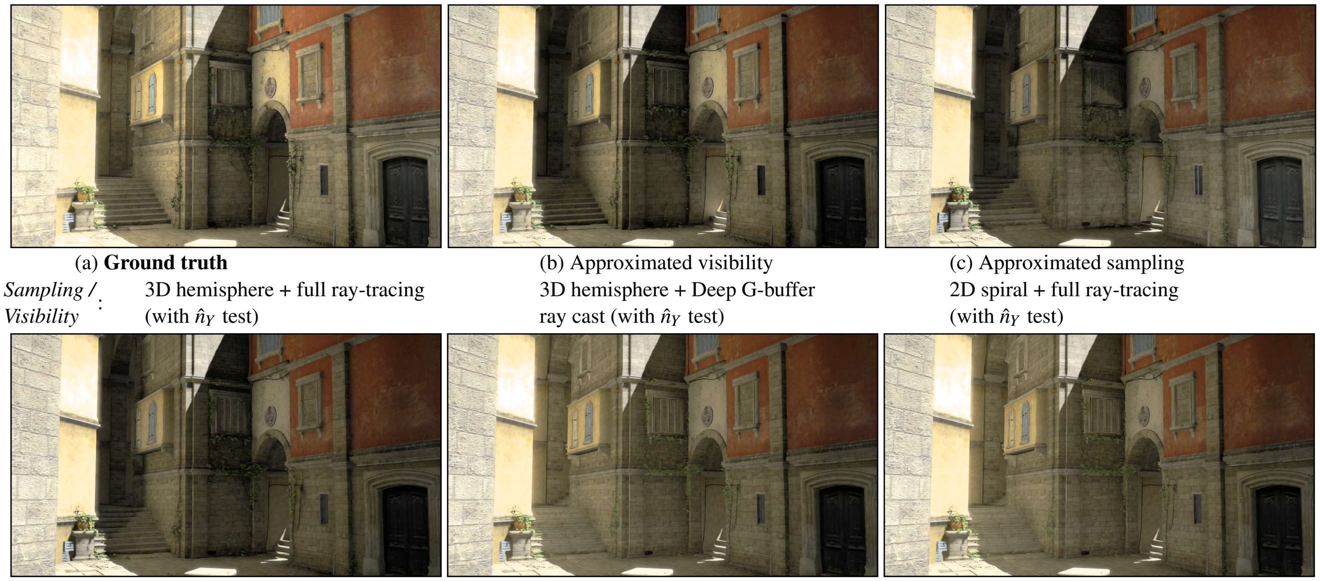


Figure 8: *Sponza* lit with dynamic lighting from a static light probe lighting solution (top), and indirect illumination computed with our Deep G-buffers (bottom), where global illumination captures color bleeding and reproduces plausible large-scale soft shadowing.



a) Indirect Light Confidence
b) Light Probe Fallback
c) + Deep G-buffer GI
d) Composite w/ Uniform Confidence

Figure 9: a) A context-sensitive confidence weight (with histogram shown inset) blends between b) static light probe GI and Deep G-buffer indirect lighting, producing c) a more robust result than either alone or d) simple averaging.



(a) **Ground truth**
Sampling / Visibility: 3D hemisphere + full ray-tracing (with $\hat{n}_Y$ test)

(b) Approximated visibility
3D hemisphere + Deep G-buffer ray cast (with $\hat{n}_Y$ test)

(c) Approximated sampling
2D spiral + full ray-tracing (with $\hat{n}_Y$ test)

(d) Section 3.2
Sampling / Visibility: 2D spiral + Deep G-buffer ray cast (with $\hat{n}_Y$ test)

(e) No visibility test, variant 1
2D spiral + no visibility (with $\hat{n}_Y$ test)

(f) No visibility test, variant 2
2D spiral + no visibility (**without** $\hat{n}_Y$ test)

Figure 10: Experimental results on the qualitative impact of each of our simplifying assumptions for indirect lighting.

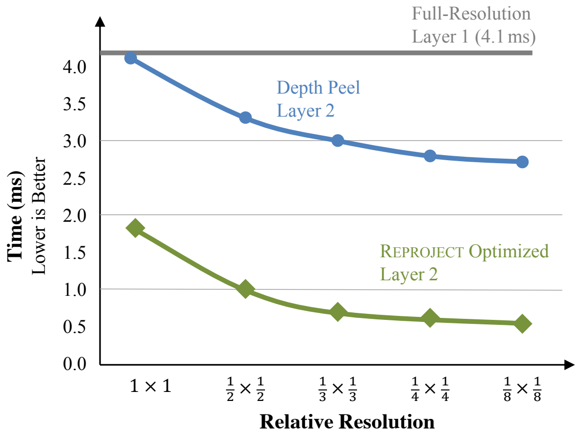


Figure 11: Performance of two methods for generating the Deep G-buffer's 2nd layer vs. resolution in *Sponza*. Our REPROJECT algorithm is twice as fast as traditional DEPTHPEEL. As the resolution for the second layer decreases, the incremental cost of generating a second layer with REPROJECT asymptotically drops to only 25% of that of the first layer, a 4x savings over naive depth peeling.

ing minimum separation and, more importantly, accuracy improvements (i.e., compared to 8-layer depth-peeled references) due to minimum separation are larger than that of including many more depth peeled layers.

In general, our Deep G-buffer approach increases the quality of existing screen-space shading approaches (e.g., see supplemental Figure 3 for AO and Figure 13 for specular reflections) with only moderate performance and storage costs, and we quickly approach the quality of many-layer depth peeled references: Figures 11 and 14 highlight the scalability of our approach with the resolution of our second layer, as well as our scalability compared to traditional depth peeling. Our single-pass construction consistently outperforms depth peeling by more than a factor of 2, in addition to generating higher quality results due to our robust minimum separation criterion.

Figures 1, 8, 15, and 16 also confirm that a layered Deep G-buffer can provide sufficient information to indirectly illuminate large regions that receive no direct light, provided that direct light appears somewhere in the framebuffer (e.g., the 2$^{\text{nd}}$ layer or guard band). These results inherently depend on the viewpoint, but in a manner that has two desirable properties: indirect illumination and AO fade-out smoothly as surfaces approach glancing angles, avoiding temporal "popping" artifacts; moreover, our results remain self-consistent for surfaces that are in (or nearly in) view.

Our indirect lighting approximation has four sources of error:

1. it can overestimate E by assuming Y is visible from X,
2. it underestimates E by not taking surfaces outside the Deep G-buffer into account,
3. our spiral sampling pattern introduces bias, and
4. ignoring the sample backface (i.e., $\hat{n}_Y$) test overestimates E.

Figure 10 explores the qualitative impact of each of these error sources on the final rendering. Here, we would like to identify the point at which a user could perceive differences between ground truth after our approximation, as opposed to a quantitative numerical error analysis. Figure 10a uses full world-space ray tracing against triangles, unbiased QMC hemispherical sampling, and the full backface test to produce a ground truth indirect + direct illumination result. The remaining results in Figure 10 use every valid combination of the aforementioned error-introducing approximations, culminating our fastest approximation in Figure 10f. The principal visual artifacts that we observed were *under*-estimation of indirect light when not using world-space ray tracing (i.e., Figure 10a vs. b) and *over*-estimation of indirect light when omiting the $\hat{n}_Y$ backface test (i.e., Figure 10e vs. f). Some other minor artifacts include varying (and often complimentary) degrees of global contrast loss, but we note that the ground truth offline result (Figure 10a) and our fastest real-time approximation (Figure 10f) compare reasonably well to each other.

In these examples, we find that the mutual visibility approximation contributes less to the perceptible error than the spiral sampling bias, likely due to the fundamental underlying screen-space assumption: nearby surfaces that face each other often have small amounts of mutual occlusion. A distant or backfacing surface contributes little indirect illumination, so its visibility does not significantly impact the final result.

We use reverse reprojection in multi-bounce indirect illumination for both progressive computation and temporal filtering. In each case, reverse reprojection creates disoccluded regions ("holes") at newly revealed locations. Figure 12 illustrates the effect of disocclusion on progressive computation (and the impact on filtering is comparable). Since the 2$^{\text{nd}}$ layer can fill in many disocclusions, and indirect illumination has a wide gather kernel, the perceptible impact on the final image is small.

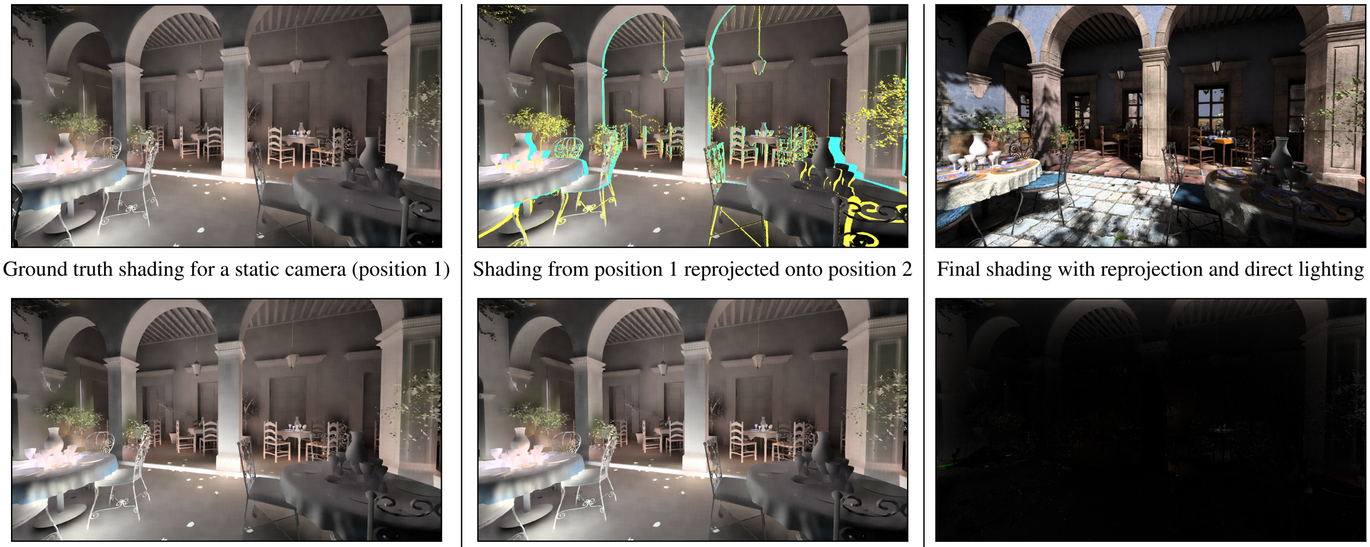
5. Conclusions

We presented an efficient single-pass method for constructing layered Deep G-buffers that respect a novel minimum separation criterion, and we demonstrated the robustness and utility of this structured in the context of several shading applications. We detailed four GPU-accelerated variants of our construction algorithm, solving the problem of determining the first layer's depth before it is ever rendered. Our indirect illumination sampling is based on radiometric first principles, and we showed how Deep G-buffers can be applied to sampling-based shading integral estimates.

While multiple layers increase robustness in these important use cases, we showed that *both* the minimum separation criterion and our single-pass implementation are essential to generating high-performance, high-quality results. Finally, we described a sampling and spatio-temporal reconstruction strategy optimized for both image quality and performance.

Discussion Our results illustrate, sometimes surprisingly, that one can reach a rendering quality normally associated to offline global illumination, but instead using our high-performance Deep G-buffers. Our techniques fail gracefully and in ways that self-identify undersampled regions, allowing fallbacks to coarser-scale lighting solutions such as precomputed light probes (which we demonstrate) or dynamic sparse voxel octrees.

All of our single-pass Deep G-buffer construction methods can



Ground truth shading for a static camera (position 1)

Shading from position 1 reprojected onto position 2

Final shading with reprojection and direct lighting

Ground truth shading for a static camera (position 2)

Shading with reprojection from position 1 to 2

Error due to reprojection at position 2 (scaled 3×)

Figure 12: Impact of reprojection on indirect lighting: cyan marks disocclusions in layer 1, and yellow marks disocclusions in both layers.

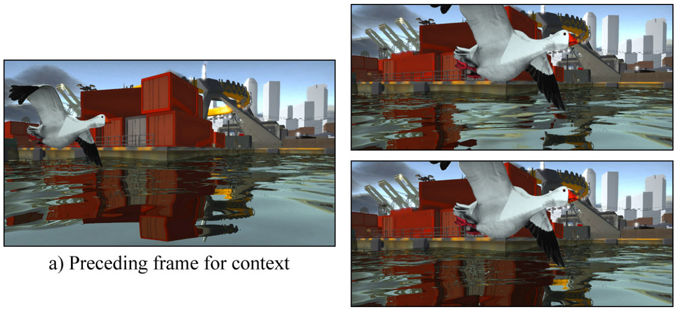
generalize from 2 to k G-buffer layers, but our PREDICTION variant requires rendering $2k-1$ layers per frame ($k-1$ for depth-only). The REPROJECTION (and less desirable PREVIOUS) variants require only k render layers per frame.

Acknowledgements

Aaron Lefohn at NVIDIA challenged us to solve this problem in a single pass and the NVIDIA Architecture group provided much essential information. All of the 3D models that we may redistribute are packaged in the G3D Innovation Engine http://g3d.williams.edu. Our source code and an extended version of this paper are online at http://graphics.cs.williams.edu/papers/DeepGBuffer16/.

References

[AFO05] ARIKAN O., FORSYTH D. A., O'BRIEN J. F.: Fast and detailed approximate global illumination by irradiance decomposition. *ACM ToG 24* (2005), 1108–1114. 2



b) Reflection with one layer

a) Preceding frame for context

c) Reflection with two layers

Figure 13: A second layer makes the red containers' reflection in *Dockside* more stable to passing foreground objects.

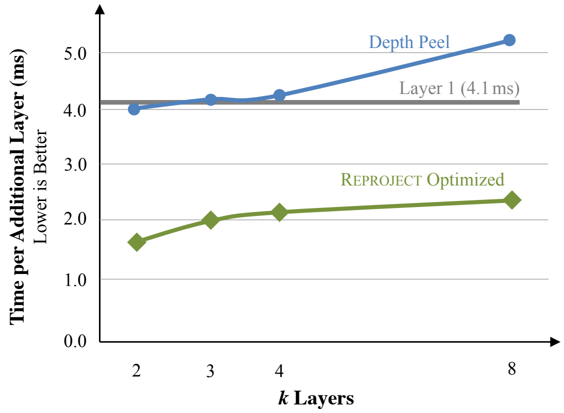


Figure 14: Deep G-buffer generation in *San Miguel*. We subtract the constant overhead of the first layer and show the additional time to complete all k layers, divided by $k-1$ to reveal amortized cost. Main result: optimized REPROJECT is 2x as fast as depth peeling.

[BCL*07] BAVOIL L., CALLAHAN S. P., LEFOHN A., COMBA JO A. L. D., SILVA C. T.: Multi-fragment effects on the GPU using the k-buffer. In *I3D* (2007), ACM, pp. 97–104. 2

[BM08] BAVOIL L., MYERS K.: *Order independent transparency with dual depth peeling*. Tech. rep., NVIDIA, 2008. 2, 3

[Bra13] BRAINERD W.: Profiling results on Playstation4 at Activision Maine, October 2013. Personal comm. 3

[BS09] BAVOIL L., SAINZ M.: Multi-layer dual-resolution screen-space ambient occlusion. In *ShaderX^7*, Engel W., (Ed.). 2009. 2

[Buk13] BUKOWSKI M.: Profiling results on NVIDIA GeForce 670 at Vicarious Visions, October 2013. Personal comm. 3

[CDE*14] CIGOLLE Z. H., DONOW S., EVANGELAKOS D., MARA M., MCGUIRE M., MEYER Q.: A survey of efficient representations for independent unit vectors. *JCGT 3*, 2 (April 2014), 1–30. 6

[CG85] COHEN M. F., GREENBERG D. P.: The hemi-cube: a radiosity solution for complex environments. *SIGGRAPH* (July 1985), 31–40. 5

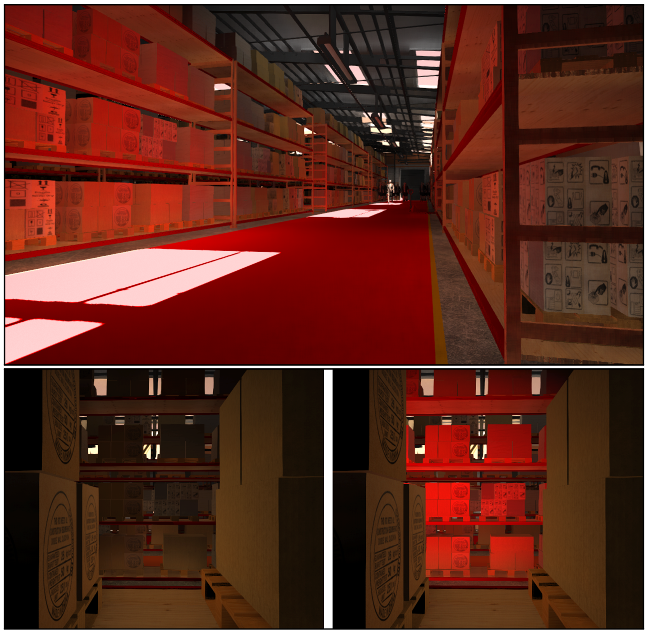


Figure 15: Single-layer shading misses the indirect bounce off the red floor (bottom left), yielding inconsistent shading in *Warehouse*; using two layers (bottom right) corrects this.

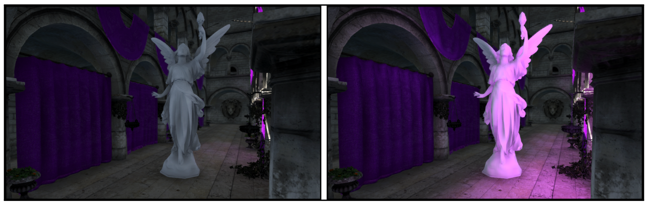


Figure 16: A single layer fails to capture indirect light from the purple banners behind the column in *Sponza* (left), whereas our 2-layer Deep G-buffer captures enough light for plausible GI (right).

[CG12] Crassin C., Green S.: *Octree-based sparse voxelization using the GPU hardware rasterizer*. CRC Press, 2012. 2

[DS05] Dachsbacher C., Stamminger M.: Reflective shadow maps. In *I3D* (2005), ACM, pp. 203–231. 2, 3, 5

[Eve01] Everitt C.: *Interactive order-Independent transparency*. Tech. rep., NVIDIA, 2001. 2

[GN15] Giraud A., Nowrouzezahrai D.: Practical shading of height fields and meshes using spherical harmonic exponentiation. In *EGSR Experimental Ideas & Imp.* (2015), Eurographics. 2

[GS12] Gilabert M., Stefanov N.: Deferred radiance transfer volumes. GDC Talk. 6

[Hac05] Hachisuka T.: *High-Quality Global Illumination Rendering Using Rasterization*. GPU Gems 2, Addison-Wesley, 2005, ch. 38. 2, 3

[Hal10] Halén H.: Style and gameplay in the Mirror's Edge, July 2010. Stylized Rendering in Games SIGGRAPH Course. 5

[JB10] Jansen J., Bavoil L.: Fourier opacity mapping. In *I3D* (2010), ACM, pp. 165–172. 2

[LV00] Lokovic T., Veach E.: Deep shadow maps. In *SIGGRAPH* (2000), ACM Press, pp. 385–392. 2

[MB07] Myers K., Bavoil L.: Stencil routed A-buffer. In *SIGGRAPH Sketches* (2007), ACM. 2

[McG13] McGuire M.: Profiling results on NVIDIA GeForce 660 at Unknown Worlds, October 2013. Personal comm. 3

[MML12] McGuire M., Mara M., Luebke D.: Scalable ambient obscurance. In *HPG* (June 2012). 2, 5, 6, 7

[NRS14] Nalbach O., Ritschel T., Seidel H.-P.: Deep screen space. *I3D* (2014), 79–86. 2

[NSL*07] Nehab D., Sander P. V., Lawrence J., Tatarchuk N., Isidoro J. R.: Accelerating real-time shading with reverse reprojection caching. In *Graphics Hardware* (2007), Eurographics, pp. 25–35. 4

[RGS09] Ritschel T., Grosch T., Seidel H.-P.: Approximating dynamic global illumination in image space. *I3D* (2009), 75–82. 2, 3

[SA07] Shanmugam P., Arikan O.: Hardware accelerated ambient occlusion techniques on GPUs. In *I3D* (2007), ACM, pp. 73–80. 2, 3

[SA09] Sintorn E., Assarsson U.: Hair self shadowing and transparency depth ordering using occupancy maps. In *I3D* (2009), ACM, pp. 67–74. 2

[SA11] Suchow J. W., Alvarez G. A.: Motion silences awareness of visual change. *Curr. Bio. 21*, 2 (2011), 140 – 143. 4

[Sal13] Salvi M.: Pixel synchronization. In *SIGGRAPH Advances in Real-time Rendering Course*. 2013. 2, 3

[Sch12] Schwarz M.: Practical binary surface and solid voxelization with Direct3D 11. In *GPU Pro 3*. A K Peters, 2012, pp. 337–352. 2

[SGNS07] Sloan P.-P., Govindaraju N. K., Nowrouzezahrai D., Snyder J.: Image-based proxy accumulation for real-time soft global illumination. In *Pacific Graphics* (2007), IEEE, pp. 97–105. 5

[SHRH09] Soler C., Hoel O., Rochet F., Holzschuch N.: *A Fast Deferred Shading Pipeline for Real Time Approximate Indirect Illumination*. Tech. rep., INRIA, 2009. 2, 5

[SKS11] Sousa T., Kasyan N., Schulz N.: Secrets of CryEngine 3 graphics technology. In *SIGGRAPH Courses* (2011), ACM. 7

[SML11] Salvi M., Montgomery J., Lefohn A.: Adaptive transparency. In *HPG* (2011), ACM, pp. 119–126. 2

[VPG13] Vardis K., Papaioannou G., Gaitatzes A.: Multi-view ambient occlusion with importance sampling. In *I3D* (2013), ACM, pp. 111–118. 2, 3

[YK07] Yuksel C., Keyser J.: *Deep Opacity Maps*. Tech. rep., Dept. of Comp. Sci., Texas A&M University, 2007. 2