Sequential Monte Carlo Instant Radiosity

Peter Hedman, Tero Karras and Jaakko Lehtinen

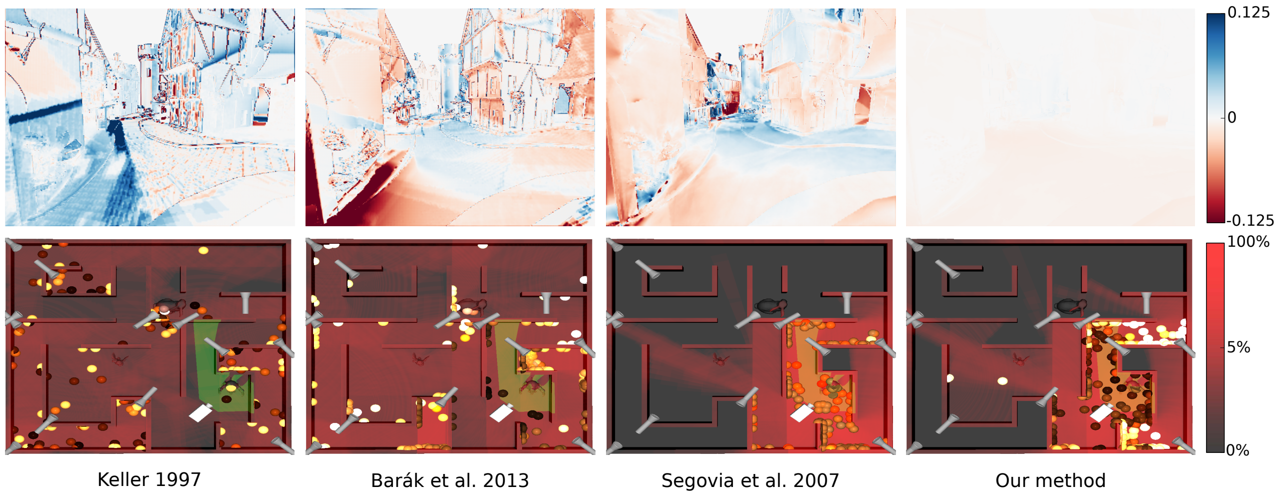


Fig. 3: Top: Temporal stability. The images show the difference between two consecutive frames in the Citadel scene (© Epic Games) where the light is moving. Our method (right) keeps the illumination stable. **Bottom:** An overhead view of a scene, the light sources, the camera and the virtual point lights (VPLs). The view frustum is denoted by the green/yellow region, the VPLs are color coded according to their intensities, and the red tint indicates the fraction of VPLs illuminating a given point. Our method and Segovia et al. [1] only place VPLs where they illuminate surfaces visible to the camera.

Abstract—Instant Radiosity and its derivatives are interactive methods for efficiently estimating global (indirect) illumination. They represent the last indirect bounce of illumination before the camera as the composite radiance field emitted by a set of virtual point light sources (VPLs). In complex scenes, current algorithms suffer from a difficult combination of two issues: it remains a challenge to distribute VPLs in a manner that simultaneously gives a high-quality indirect illumination solution for each frame, and to do so in a temporally coherent manner. We address both issues by building, and maintaining over time, an adaptive and temporally coherent distribution of VPLs in locations where they bring indirect light to the image. We introduce a novel heuristic sampling method that strives to only move as few of the VPLs between frames as possible. The result is, to the best of our knowledge, the first interactive global illumination algorithm that works in complex, highly-occluded scenes, suffers little from temporal flickering, supports moving cameras and light sources, and is output-sensitive in the sense that it places VPLs in locations that matter most to the final result.

Index Terms—Instant Radiosity, Global Illumination.

1 Introduction

Global illumination, the complex interaction of light and the three-dimensional environment, is a key component in perceived image realism. Physically-based rendering algorithms numerically solve equations that describe the radiation transport process. Unfortunately, most robust algorithms are currently too slow for real-time rendering of scenes of realistic extent and complexity. Performance requirements limit the number of samples available to the point that leaves results riddled with visually unacceptable artifacts. To first order, the rich literature on interactive global illumination algorithms aims to circumvent these issues by trading variance for bias; intuitively, a smooth and temporally stable result is perceptually preferable to high variance and flickering.

In addition to the obvious goal of physical accuracy, we identify the following key desirables for an interactive global illumination algorithm:

1) Low variance in space: little noise,
2) Low variance in time: temporal coherence,
3) Support for multi-bounce illumination,
4) Support for moving cameras and light sources,
5) Computing only what matters to the final image and
6) Support for dynamic scenes and complex materials.

Desirable 1, noise-free images, has received an enormous

- *P. Hedman was with NVIDIA Research, the University of Helsinki and University College London during the completion of this work.*
E-mail: p.hedman@cs.ucl.ac.uk
- *T. Karras is with NVIDIA Research.*
E-mail: tkarras@nvidia.com
- *J. Lehtinen is with NVIDIA Research and Aalto University.*
E-mail: jaaakko.lehtinen@aalto.fi

Manuscript received January 30, 2017

amount of attention in the literature. Temporal coherence (Desirable 2) is also a well-studied problem; several image-space temporal filtering techniques share values across frames by reprojection, effectively raising the sampling rate considerably.

In the interactive setting, however, the combination of all the desirables have been left with significantly less attention. The key challenge in large and highly occluded scenes, such as complete building models or typical computer game levels, is that most light paths do not contribute to the image. Focusing computational resources on things that will affect the final image (i.e., avoiding wasting cycles for no visible effect, Desirable 5) is in such cases hard, because one needs to determine what parts of the scene illuminate geometry visible to the camera. Our results show that not accounting for precise visibility severely degrades quality in these settings (Figure 3). Finally, supporting full, multi-bounce global illumination (Desirable 3) renders convenient indirect light path parametrizations based on surfaces visible to the light source, e.g. Reflective Shadow Maps [2], unusable.

In this paper, we describe an algorithm that, to our best knowledge, simultaneously achieves goals 1-5 in diffuse scenes to a greater extent than previous techniques by extending the well-known Instant Radiosity algorithm [3], which approximates the illumination in the image with *virtual point lights* (VPLs), to better adapt to challenging view-light configurations in a temporally coherent manner. Our main contribution is a VPL sampling algorithm that:

- Minimizes temporal flickering by keeping the VPL distribution stable even if illumination changes and the camera is moving (Desirables 2, 4)
- Distributes VPLs according to the amount of light they bring the image without de-emphasizing small regions in the image (Desirable 5),
- Supports multi-bounce indirect illumination (Desirable 3).

We use the standard interleaved sampling technique [4] as a component in our method to remove image-space noise (Desirable 1). Support for fully dynamic scenes and arbitrary materials (Desirable 6) remains future work.

The present paper extends a previous conference version [5] by describing a VPL candidate sampler significantly less prone to flicker when previously hidden regions become visible for the first time, a detailed study of the factors that affect final render quality — in particular, teasing apart the effects of VPL distribution vs. visibility determination method, shadow maps or ray tracing — as well as giving numerous additional details.

2 Previous work

Interactive global illumination is a diverse field of study too large to be fully reviewed here. The focus of this paper is to reduce the temporal variance of Instant Radiosity [3] and place the VPLs where they are most needed for the camera view. Therefore we concentrate on the most closely related work. Refer to the survey by Dachsbacher et al. [6] for a comprehensive overview of VPL-based methods and to the survey by Ritschel et al. [7] for a broader selection of algorithms that simulate global illumination.

2.1 Rendering with VPLs

The brightness I_p of a pixel p in the image $\mathbb{P}$ is determined by the measurement equation as the integral

$$I_p = \int\limits_{\mathcal{P}} f_p(\bar{\mathbf{z}})\mathrm{d}\bar{\mathbf{z}} = \int\limits_{\mathcal{P}_l}\int\limits_{\mathcal{P}_c} f_p(\bar{\mathbf{x}},\bar{\mathbf{y}})\mathrm{d}\bar{\mathbf{x}}\mathrm{d}\bar{\mathbf{y}} \tag{1}$$

over all light transport paths $\bar{\mathbf{z}} \in \mathcal{P}$ [8]. We write each path $\bar{\mathbf{z}}$ as the concatenation $\bar{\mathbf{z}} = (\bar{\mathbf{x}}, \bar{\mathbf{y}})$ of a 2-vertex camera path $\bar{\mathbf{x}} \in \mathcal{P}_c$ and a $(N-2)$-vertex light path $\bar{\mathbf{y}} \in \mathcal{P}_l$. VPL-based methods sample one set of I light paths (or VPLs) $(\bar{\mathbf{y}}_1, \ldots, \bar{\mathbf{y}}_I)$ to estimate the value of all pixels in the image. A set of J camera paths (or *pixel samples*) $(\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_J)$ is then sampled for each pixel p to form the unbiased estimate

$$I_p \approx \frac{1}{I}\frac{1}{J}\sum_{i=1}^{I}\sum_{j=1}^{J}\frac{f_p(\bar{\mathbf{x}}_j,\bar{\mathbf{y}}_i)}{p(\bar{\mathbf{x}}_j)p(\bar{\mathbf{y}}_i)}. \tag{2}$$

Many-lights methods efficiently render images illuminated by a large number of point lights. Lightcuts [9] hierarchically clusters together point lights estimated to be non-essential when computing the brightness of a pixel. These methods are orthogonal to our work as they do not specify how to generate the point lights. While we have evaluated our algorithm in an interactive setting, any many-light rendering method can be used with our method to efficiently render pictures in an offline setting.

Reflective Shadow Maps (RSMs) is a fast method to simulate single-bounce indirect illumination by placing VPLs at every texel in a shadow map [2]. Approximations can be made to reduce the time spent computing visibility [10]. These are orthogonal to our contribution as our focus is on how to generate the VPLs. Performance can also be increased with Interleaved Sampling [4], [11], [12], where adjacent pixels compute the indirect illumination using disjoint subsets of VPLs. We use a similar method to compute the illumination from the VPLs.

A limitation of VPL-based methods is the singularity that occurs when VPLs and pixel samples lie close to each other or due to the narrow emission profile of VPLs on surfaces with glossy materials. As per standard practice, we avoid this by clamping the maximum contribution of a VPL. More specialized methods, e.g. recursively sampling paths when clamping occurs [13] or virtual spherical lights [14], can be used to increase quality.

2.2 VPL sampling algorithms

The original Instant Radiosity algorithm [3] emits photons from the light sources and deposits a VPL at every surface interaction. This is problematic for large scenes, as it does not concentrate the VPLs in regions where they bring indirect light to the image. This issue is addressed by generating VPLs that are distributed according to the total brightness they cause in the image. Segovia et al. [15] achieve this by resampling the VPLs from a larger set of candidates. Georgiev and Slusallek [16] use rejection sampling. Segovia et al. [1] place VPLs using a variant of Metropolis Light Transport [8]. Simon et al. [17] sample VPLs from the product of two complementary photon maps [18], one generated from the light sources and one from the camera. Ignoring the visibility between pixel samples and VPLs, Ritschel et al. [19]

present a fast method to resample VPLs to approximately match the amount of light they bring the image. While these techniques improve the quality of the indirect illumination in individual frames, they do not provide any means to keep it temporally stable. Consequently, they cannot be used in an interactive setting as the number of VPLs needed to keep the temporal noise at acceptable levels becomes prohibitive.

2.3 Temporal coherence

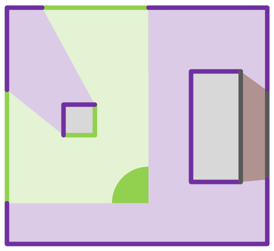
Temporal coherence is an important aspect of interactive rendering [20]. In the context of VPL-based rendering, Laine et al. [21] achieve temporally stable indirect illumination by only moving a few VPLs each frame. However, in addition to only supporting a single indirect bounce, their method is not suited for large scenes as it samples the VPL oblivious of the camera. Knecht et al. [22] improve the stability of the indirect illumination with temporal reprojection filtering. As they do not specify how to generate the VPLs, their method is complementary to ours. Hašan et al. [23] group point lights into clusters and reuse shaded results from the clusters over multiple frames in an animated video. As their method relies on a priori knowledge of the animation it cannot be used in an interactive setting. Wald et al. [24] enforce temporal coherence by fixing the random number sequence used to generate the VPLs. While more VPLs are allocated to sources that bring more indirect light to the image, the distribution suffers if a single source – e.g. the sun – illuminates most of the scene. The method cannot fully ensure temporal stability with moving lights.

Prutkin et al. [25] and Barák et al. [26] devise methods to improve the temporal stability of the VPL sampling method described by Ritschel et al. [19]. As both methods use RSMs to generate the VPLs, they only support single-bounce indirect illumination. They also ignore visibility when resampling the VPLs, which degrades the quality of the VPL distribution in heavily occluded scenes. Most importantly, as both methods enforce temporal coherence *in the RSMs* rather than in world space, they cannot ensure temporally stable illumination if a light source moves. Prutkin et al. [25] cluster the VPLs to form area lights and achieve temporal stability by seeding the clustering algorithm with the area light centers from the preceding frame. Their method is unsuitable for walkthroughs as they only account for the camera during initialization. Barák et al. [26] use Quasi-Monte Carlo and Metropolis-Hastings sampling to resample in a temporally stable fashion. We compare against this method in our experiments.

3 Method

Our two-fold goal is to distribute VPLs in the scene in a way that results in a high-quality diffuse indirect illumination field on the surfaces visible to the camera and evolve the distribution frame-to-frame in a manner that minimizes temporal flickering.

Taking inspiration from Segovia et al. [15], we achieve our first goal by only placing VPLs *on surfaces indirectly visible to the camera*, i.e. surfaces that can be seen from at least one of the surfaces in the image. (Naturally, only these surfaces can cast indirect light onto the image.) The inset shows directly visible surfaces in green, indirectly visible surfaces in purple and other surfaces in gray. To further improve quality, we place more VPLs on surfaces that reflect a lot of light; a type of importance sampling that improves quality in the brighter areas of the image at the expense of the darker areas. In our comparisons, we show that it is essential to consider indirect visibility this way; techniques that strive for view-adaptivity *without* accounting for visibility may use the VPL budget in a way that leaves the image quality severely degraded.



Because the camera (and potentially light sources) are moving, the VPL distribution needs to adapt to the current view-light configuration. This makes temporal coherence a potential issue. Our second goal, stability over frames, is achieved by allowing only a fixed number of VPLs to move between consecutive frames. Concretely, the majority of VPLs from the previous frame survive into the next frame, but ones that are determined the least useful — e.g. not indirectly visible any more, or in a location where VPLs are oversampled w.r.t. the indirect illumination — are discarded. New VPLs are then sampled in a manner that attempts to maintain a distribution proportional to the strength of indirect illumination (see above). The remainder of the section details the specifics of the algorithm steps.

3.1 Target Distribution for VPLs

We begin by expanding the measurement contribution function f_p in Equation 2. Let $\mathbf{x}$ be the 3D position of the "primary hit", i.e. the last vertex of the camera path $\bar{\mathbf{x}}$. Similarly, let $\mathbf{y}$ be the first vertex of the light path $\bar{\mathbf{y}}$. The Monte Carlo estimate for pixel p is

$$I_p \approx \frac{1}{I}\frac{1}{J}\sum_{i=1}^{I}\sum_{j=1}^{J}\frac{W(\bar{\mathbf{x}}_j \leftarrow \mathbf{y}_i)G(\mathbf{x}_j \leftrightarrow \mathbf{y}_i)L(\mathbf{x}_j \leftarrow \bar{\mathbf{y}}_i)}{p(\bar{\mathbf{x}}_j)p(\bar{\mathbf{y}}_i)}, \quad (3)$$

where $W(\bar{\mathbf{x}} \leftarrow \mathbf{y})$ is the importance of a camera path $\bar{\mathbf{x}}$ towards the point $\mathbf{y}$ (including the bidirectional scattering distribution function at $\mathbf{x}$), $G(\mathbf{x} \leftrightarrow \mathbf{y})$ is the geometry term between the two points (including visibility) and $L(\mathbf{x} \leftarrow \bar{\mathbf{y}})$ is the radiance reflected from the last vertex of the light path $\bar{\mathbf{y}}$ towards the point $\mathbf{x}$.

The variance of the resulting image can be reduced by using importance sampling for placing the VPLs, i.e. finding a suitable $p(\bar{\mathbf{y}})$. However, it is not immediately clear what the best importance sampling distribution is. It is common practice to distribute VPLs proportionally to the total power they bring to the image, as done by e.g. Segovia et al. [1] and Simon et al. [17]. This is problematic for temporal coherence: small but brightly illuminated regions in the image are prone to flickering (this is evident in the accompanying videos; cf. in particular the corridor sequences in Soda Hall).

We find it better to distribute the VPLs according to a metric agnostic to the image-space size of the regions they illuminate. A good choice would be to base sampling on the maximum pixel brightness caused by the VPL in the image, i.e. have $p(\bar{\mathbf{y}})$ be proportional to the maximum of the product of the W, G and L terms from Equation (3) over all pixels. As this is hard achieve in general scenes, we make two assumptions. First, we treat all indirectly visible surfaces as diffuse, so that VPLs emit uniformly in all directions, i.e. $L(\mathbf{x} \leftarrow \bar{\mathbf{y}}) = L(\bar{\mathbf{y}})$. Second, we replace the view importance and geometry terms with unity, i.e. we ignore the materials of the primary hits and their geometric configuration w.r.t. indirectly visible surfaces and instead treat all of them with equal importance. Together, these assumptions mean we strive to distribute VPLs according to their radiosity:

$$p(\bar{\mathbf{y}}) \propto L(\bar{\mathbf{y}}). \tag{4}$$

The next sections show how this is achieved in practice. While our results show this leads to high-quality solutions in diffuse scenes, we believe existing work on efficiently estimating illumination bounds can be used for tightening these for more general scenes. We leave this as future work.

VPLs vs. Light Paths: Until now, we have assumed, like many Instant Radiosity algorithms, that a VPL is synonymous with a single light path $\bar{\mathbf{y}}$. Similarly to Segovia et al. [15] and Simon et al. [17], we instead marginalize over all light paths that end up at the same VPL location $\mathbf{y}$. That is, instead of having the VPL emit proportional to a single path from the source, we compute its total radiosity as an average over several paths. From now on, we consider a VPL just to be the point $\mathbf{y}$. Given unbiased estimates for the radiosities of the VPLs, the estimate

$$I_p \approx \frac{1}{I}\frac{1}{J}\sum_{i=1}^{I}\sum_{j=1}^{J}\frac{W(\bar{\mathbf{x}}_j \leftarrow \mathbf{y}_i)G(\mathbf{x}_j \leftrightarrow \mathbf{y}_i)L(\mathbf{y}_i)}{p(\bar{\mathbf{x}}_j)p(\mathbf{y}_i)}, \tag{5}$$

where $p(\mathbf{y}_i)$ is the probability density of the VPL point $\mathbf{y}_i$ with respect to the surface area measure, is still unbiased.[1] To simplify notation, we define the *intensity* $I(\mathbf{y})$ of a VPL $\mathbf{y}$ to be the ratio

$$I(\mathbf{y}) = \frac{L(\mathbf{y})}{p(\mathbf{y})} \tag{6}$$

of its estimated radiosity and the probability density of the point $\mathbf{y}$. Note that if the VPLs are distributed according to their radiosities, i.e. $p(\mathbf{y}) \propto L(\mathbf{y})$, then all VPLs will have the same intensity.

3.2 Sequential Sampling Algorithm

Our goal is to incrementally maintain a temporally coherent VPL distribution with uniform intensities when moving from one frame to the next. Our main idea is to incrementally replace VPLs that have the smallest intensities with new VPLs that have larger intensities. This drives the distribution towards uniform intensity.

The key challenge is a combination of two factors. First, light and camera movement cause the target *probability density function* (PDF) for VPL placement to change over time: moving lights change the radiosity of indirectly visible surfaces and camera movement changes the set of indirectly visible surfaces itself. Second, to maintain temporal stability, we wish to re-use as many VPLs from the previous frame as possible. This brings up a crucial point. As we migrate a VPL distribution from a previous frame, we have no tractable way of analytically computing the PDF $p(\mathbf{y}_i)$ in the new frame: it depends on the sampling decisions made in all previous frames. Instead, we rely on the insight that the probability density on a surface patch can be estimated as the fraction of the VPLs that lie on it normalized by its area. Consequently, we approximate the PDFs of all VPLs using k-nearest neighbor density estimation (Section 3.2.4). This allows us to adapt to any given VPL distribution and drive it towards a uniform intensity distribution.

1. The details are easy but technical and non-essential here.

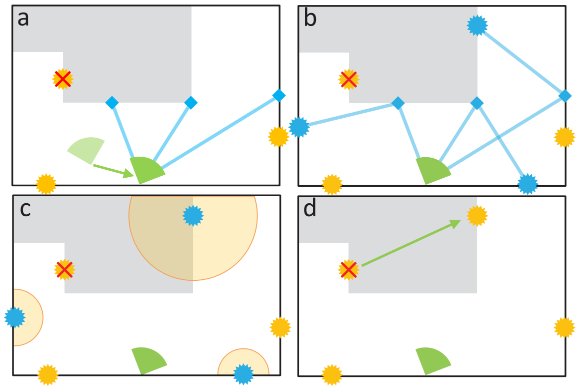


Fig. 4: Overview. VPLs shown with yellow. (**a**) At every frame, we first sample points (in blue) on the surfaces visible to the camera (in green). We also invalidate some VPLs as described in Sections 3.2.2 and 3.2.5 (shown with a red cross). (**b**) We generate candidate VPLs (in blue) by tracing rays from the directly visible points. (**c**) We then estimate the intensities of these candidates (larger circles correspond to stronger intensities). (**d**) Finally, we replace the invalidated VPLs with the candidates that have the strongest intensities.

Algorithm 1 Sequential Monte Carlo Instant Radiosity

```
1:  $\mathcal{Y} \leftarrow$ INITIALIZEVPLS()
2:  for each new frame do
3:      $\mathcal{X} \leftarrow$ RAYCASTFROMCAMERA()
4:      INVALID $\leftarrow \mathcal{Y} \setminus$ INDIRECTLYVISIBLEVPLS($\mathcal{X}, \mathcal{Y}$)
5:      $\hat{\mathcal{Y}} \leftarrow$ RAYCASTFROMPOINTS($\mathcal{X}$)
6:      $\hat{\mathcal{Y}} \leftarrow$ RESAMPLECANDIDATES($\hat{\mathcal{Y}}$)
7:      ESTIMATERADIOSITIES($\hat{\mathcal{Y}}$)
8:      ESTIMATEPROBABILITYDENSITIES($\hat{\mathcal{Y}}$)
9:      $M_{min} \leftarrow$ COUNTUNDERSAMPLEDREGIONS($\mathcal{Y}, \hat{\mathcal{Y}}$)
10:     while |INVALID| $< M_{min}$ do
11:         $\mathbf{y}_{min} \leftarrow$ FINDSMALLESTINTENSITY($\mathcal{Y} \setminus$ INVALID)
12:         INVALID $\leftarrow$ INVALID $\cup \{\mathbf{y}_{min}\}$
13:     $\mathcal{Y} \leftarrow \mathcal{Y} \setminus$ CLAMPSIZE(INVALID, $M_{max}$)
14:     while $|\mathcal{Y}| < I$ do
15:         $\mathbf{y}_{max} \leftarrow$ FINDLARGESTINTENSITY($\hat{\mathcal{Y}}$)
16:         $\hat{\mathcal{Y}} \leftarrow \hat{\mathcal{Y}} \setminus \{\mathbf{y}_{max}\}$
17:         $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{\mathbf{y}_{max}\}$
18:     ESTIMATERADIOSITIES($\mathcal{Y}$)
19:     ESTIMATEPROBABILITYDENSITIES($\mathcal{Y}$)
20:     RENDERFRAME($\mathcal{Y}$)
```

Algorithm: See Figure 4 for an overview and Algorithm 1 for a pseudo-code description of the sampling algorithm. Here, I is the total VPL budget and $M_{\max}$ is the number of VPLs allowed to move between frames. A lower $M_{\max}$ results in faster updates, but gives the sampler

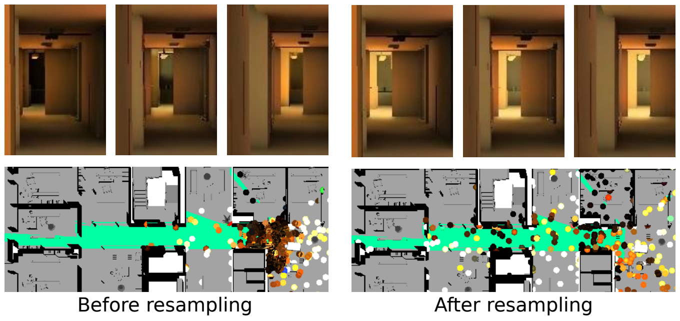


Fig. 5: Generating VPL candidates. Left: Initial VPL candidates obtained by tracing random rays from the directly visible points. **Right:** Approximately uniform distribution obtained by weighted resampling. **Top:** Cutouts from three frames in the Soda Hall sequence showing the illumination in a room just appearing at the far end of a corridor. Missing illumination, visible as a brief temporal flicker, can be observed when weighted resampling is not employed. **Bottom:** Overhead view of the VPL candidate distribution. The view frustum is denoted by the green region.

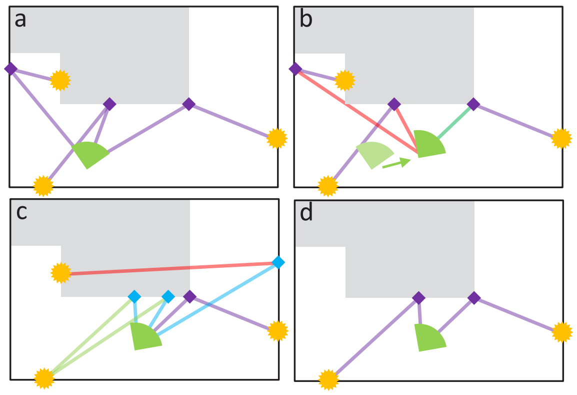


Fig. 6: Verifying indirect visibility. (**a**) Each VPL (in yellow) is associated with a point (in purple) visible to both the camera and the VPL. (**b**) When the camera moves, some VPLs can be validated as indirectly visible if their associated points are still seen by the camera (shown with green). (**c**) We validate more VPLs by casting rays from points directly visible to the camera (in blue) towards randomly chosen VPLs. (**d**) If a ray cast succeeds, we mark the target VPL as valid and associate it with the point that cast the ray.

less room to produce a high-quality distribution. Unless mentioned otherwise, we set $I = 2048$ and $M_{max} = 16$.

When a frame begins, we need to decide which of the existing VPLs, if any, to replace. To ensure that the VPLs are located on the indirectly visible surfaces, we first invalidate all VPLs that cannot be verified as indirectly visible (lines 3-4, Section 3.2.2).

We then generate a set of candidate VPLs by tracing rays from the directly visible surfaces (line 5). We ensure that the candidates are uniformly distributed on the indirectly visible surfaces using weighted resampling (line 6 and Section 3.2.1). We then estimate their intensities by performing density estimation and computing their radiosities (lines 7 and 8, Sections 3.2.3 and 3.2.4). The radiosities and probability densities of VPLs carried over from the previous frame are not re-estimated at this point. We then use these intensities to detect undersampled regions (line 9) — bright indirectly visible surfaces where VPL density is too low — and iteratively fill them in. This happens by removing VPLs from oversampled regions and generating new VPLs in undersampled regions (lines 10-17). Oversampling and undersampling are easily detected by examining the intensities of the VPLs (Section 3.2.5). Finally, the intensities are re-estimated for the remaining VPLs and the image is rendered.

We initialize the algorithm by sampling VPLs on the indirectly visible surfaces identically to how we generate candidate VPLs. To ensure that the VPLs are well distributed, we run 100 iterations of the algorithm before rendering the first frame.

3.2.1 Generating VPL candidates

While our algorithm is primarily concerned about the distribution of the final VPLs $\mathcal{Y}$, we have also observed that the distribution of VPL candidates $\hat{\mathcal{Y}}$ can have a significant impact on how quickly the illumination stabilizes in newly visible areas (see the supplemental videos). In practice, we have found it beneficial to use a uniform distribution with respect to surface area so that every indirectly visible surface receives a sufficient number of samples.

To generate the VPL candidates $\hat{\mathcal{Y}}$, we first trace rays in randomly chosen directions from the directly visible points and estimate the probability densities of the resulting intersection points. We then select $\hat{\mathcal{Y}}$ as an approximately uniformly distributed subset using weighted resampling without replacement (cf. Figure 5).

In all experiments, we use a resampling rate of 16:1 to filter $16 \cdot I$ intersection points down to I VPL candidates. We implement the resampling by drawing I random elements from the set of intersection points in parallel. We avoid duplicates by keeping track of the selected elements using atomics, and retry the operation several times if necessary. To limit the worst-case execution time, we stop after 8 retries.

3.2.2 Resolving indirect visibility

We estimate indirect visibility using a mail-boxing scheme that associates each VPL with a point visible to both the camera and the VPL, cf. Figure 6. When a new VPL is created we first associate it with the point it was generated from. When the camera moves, we can easily validate a VPL as indirectly visible by checking if its associated point is still visible to the camera. A similar scheme has been used for occlusion culling [27].

As this is not an exhaustive visibility test, some of the remaining invalid VPLs may be indirectly visible. We approximately correct for this with an iterative scheme that employs the directly visible points we sample at the start of each frame. At each iteration, we check the indirect visibility for any remaining invalid VPLs by casting rays from these points towards V randomly chosen invalid VPLs. If a ray cast succeeds, we mark the chosen VPL as valid and associate it with the point that chose it. In our experiments we set $V = 16$ and perform two iterations per frame. During the second iteration, we have the benefit of not shooting rays toward VPLs that were already validated during the first iteration. While our approach does not ensure an exact result, the rest of our method is robust to occasional erroneous classifications.

3.2.3 Estimating Radiosities

Any global illumination algorithm can be used to estimate the VPL radiosities. In our experiments, we use 16 samples from a path tracer with next event estimation [28]. To ensure that the estimate remains temporally coherent, we make sure that the path tracer generates similar paths for the same VPL throughout the lifetime of the VPL. We achieve this by initializing the pseudo-random number used by path tracer with a VPL-specific seed.

3.2.4 Estimating probability densities

We approximate the density of VPLs on the surfaces with k-nearest neighbor (kNN) density estimation based on the same assumptions used by photon mapping [18]. That is, instead of performing a costly search along the surfaces in the scene, we assume that the k-nearest neighbors we find with an unconstrained 3D search all lie on the same, planar surface. We then approximate the probability density of a VPL $\mathbf{y}$ w.r.t. the surface area measure as

$$p(\mathbf{y}) \approx \frac{k}{I} \frac{1}{r_k(\mathbf{y})^2 \pi}, \tag{7}$$

where I is the total number of VPLs and $r_k(\mathbf{y})$ is the distance to the kth nearest neighbor of $\mathbf{y}$ among the *current* VPLs in $\mathcal{Y}$.

We use a different k when estimating PDFs for candidate VPLs (lines 8 and 6) and for rendering (line 19). For candidates, the probability densities are used to decide which candidates are selected. We set $k = 1$ to ensure that candidates close to any current VPLs are unlikely to be selected. This prevents VPLs from clumping together and encourages blue-noise properties in the resulting distribution.

When computing intensities for final rendering, we want to maximize quality of the estimated probability density. To achieve this, we choose k in a way that can be shown to result in a consistent estimate [29]:

$$k = \max(1, \left\lfloor \frac{\sqrt{I}}{32} \right\rfloor). \tag{8}$$

3.2.5 Detecting undersampled regions

To detect undersampled regions on indirectly visible surfaces (regions with too few VPLs in relation to their brightness), we observe that the estimated probability density $p(\mathbf{y})$ of any VPL which lies in such a region will be small relative to its radiosity $L(\mathbf{y})$. As a consequence, each such candidate will have an uncharacteristically large intensity $I(\mathbf{y})$ compared to the current set of VPLs $\mathcal{Y}$. Based on this, we compute the number M_{min} of VPLs to invalidate by counting how many candidates have stronger intensities than a large majority of the current VPLs. Specifically, we set

$$M_{min} = \min(M_{max}, M_{strong}), \tag{9}$$

where M_{strong} is the number of candidates in $\hat{\mathcal{Y}}$ with stronger intensities than 95% of the current VPLs in $\mathcal{Y}$. We then simply remove the M_{min} VPLs with the lowest intensities and, while we have room in the total VPL budget, choose the highest-intensity candidate VPLs to replace them (lines 10-17).

3.3 Discussion

Our sample-evolve-resample approach is essentially a form of *particle filter*, a subset of Sequential Monte Carlo (SMC) methods designed for online sampling of evolving distributions using weighted samples ("particles") [30]. Standard particle filters evolve the distribution by mutating samples surviving a resampling step by drawing from an importance distribution derived from the old samples with either closed-form PDFs or Markov Chain Monte Carlo methods. This approach conflicts with our goal of keeping a subset of the samples strictly fixed. More precisely, as our sampling process is a combination of sequential resampling and discard heuristics, it is impossible to derive an exact importance density for the resulting VPLs, as required by standard SMC approaches. Similar to Hämäläinen et al. [31], who in their motion synthesis algorithm combine analytic importance sampling with heuristics that do not admit closed-form densities, we address this by computing the samples' *empirical density* using k-nearest neighbor density estimation instead. This renders our algorithm biased, but it remains consistent: increasing the number of VPLs (while decreasing their clamping) tends to the correct solution.

4 Implementation

All steps of our algorithm are performed on the GPU in either OpenGL or CUDA. We use the OptiX Prime library [32] to cast the rays necessary for sampling new VPL candidates and verifying indirect visibility and in our CUDA path tracer we use it for estimating VPL radiosities. We use CUDA perform the kNN search with the help of a bounding volume hierarchy (BVH) constructed using the fast tree building algorithm by Karras [33].

Similarly to Segovia et al. [12], we compute the illumination from the VPLs using deferred rendering and interleaved sampling [4]. We partition the image into tiles of size 4×4 and use a different subset of the VPLs to compute the indirect illumination for each pixel in a tile. We remove the resulting structured noise with a cross-bilateral filter [34], [35] whose weights are determined by the dot product between the normals and the distance between the world-space locations of the pixel samples.

We use paraboloid shadow maps to determine the visibility between pixel samples and VPLs [36]. To avoid the need for finely tessellated geometry, we generate the paraboloid shadow maps by resampling cube shadow maps. To increase performance, we translate and rotate each cube map so that three of its faces cover the entire hemisphere. Finally, we render the three faces using generalized perspective projection [37] to ensure that the center of projection is at the at the VPL location. Unless mentioned otherwise, we use 128×128 shadow maps in our experiments. The resolution was chosen as a compromise between indirect illumination quality and GPU memory usage (cf. Section 5.4).

Similarly to Laine et al. [21], we lazily update the shadow maps only for VPLs that move between frames. This saves computation with the caveat that dynamic objects cannot be accounted for in the shadow maps.

TABLE 1: Quantitative comparison. We track the luminance of 8 pixels over time, comparing them with a path-traced reference. We show results from five different sequences, all illuminated by area lights. The rows summarize the overall error (**Average** $|E|$) and the temporal instability (**Average** $|E'|$) of the competing VPL sampling methods for all tracked pixels in a sequence. The smallest error is in bold.

				Average $\lvert E\rvert$				**Average $\lvert E'\rvert$**			
Scene	**Length**	**Lights**	**#VPLs**	**MIR**	**IR**	**TCAS**	**Ours**	**MIR**	**IR**	**TCAS**	**Ours**
Soda Hall	30 s	14 (static)	2048	0.0262	0.0551	0.0526	**0.0206**	0.0098	0.0077	0.0057	**0.0044**
Epic Citadel	23 s	1 (static)	2048	0.0492	0.0827	0.0626	**0.0473**	0.0241	0.0364	0.0227	**0.0222**
Maze	22 s	13 (static)	512	0.0287	0.0869	0.0380	**0.0221**	0.0216	0.0132	0.0113	**0.0075**
Crytek Sponza	20 s	1 (dynamic)	2048	0.0288	**0.0215**	0.0352	0.0962	0.0072	0.0044	0.0027	**0.0014**
Epic Citadel	21 s	1 (dynamic)	2048	0.0246	0.0892	0.0650	**0.0240**	0.0129	0.0323	0.0198	**0.0071**

5 Experiments

We now evaluate the quality and performance of our method. To examine how the VPL distribution affects the quality of the final animated result, we compare with representative previous algorithms while keeping the VPL budget, method for determining visibility (standard shadow mapping), sampling pattern (interleaved sampling), and noise filtering (cross bilateral filter) the same between the algorithms — only the positions and intensities of the VPLs change. To clearly compare temporal stability, we do not use temporal reprojection filtering on the indirect illumination. We quantify error by tracking the absolute error and temporal stability of the resulting indirect illumination. We encourage the reader to watch the supplemental videos for a thorough qualitative comparison of these methods.

We also examine the performance and single frame quality of our method. We study performance with a timing breakdown and evaluate quality by comparing the indirect illumination from our method with path traced reference images. In these comparisons, we show how the different methods of determining visibility (shadow maps and ray casting) influences the quality of the indirect illumination. Finally, we also analyze how the error of our method diminishes as more VPLs are used.

We simulate three bounces of indirect illumination in all experiments. To make the results agree better with perceived quality, we perform all measurements after applying the Reinhard tone mapping operator [38]. It is well known that VPL algorithms have severe issues with high-frequency textures. In all experiments, we use an untextured version of the scene for the VPL generation and use average material colors as albedos. The path traced reference images use the same approximation.

5.1 Comparison methods

We compare our method to Instant Radiosity (IR) [3] (baseline method, no view adaptivity and no special consideration for temporal coherence), Metropolis Instant Radiosity (MIR) [1] (high quality VPL distribution but no temporal coherence) and Temporally Coherent Adaptive Sampling (TCAS) [26] (view adaptivity, temporal coherence, but limited to a single indirect bounce and does not account for indirect visibility). Of these, MIR can be seen as a gold standard for single frame image quality as it produces, independently for each frame, a VPL distribution that matches the power brought to the image.

IR [3] generates VPLs by emitting photons from the light sources, so it may not use the full VPL budget if some photons exit the scene before depositing the second or third bounce VPLs. This is visible in Epic Citadel where many photons leave the scene early, exacerbating the structured noise caused by interleaved sampling.

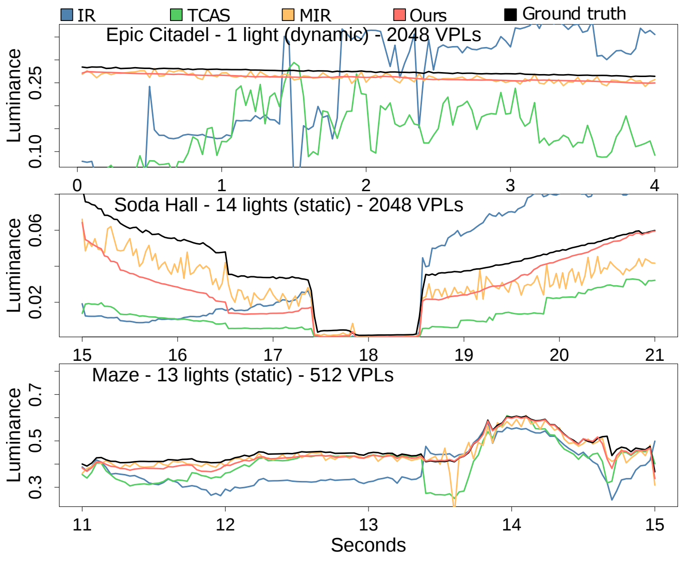


Fig. 7: Example sequences. The plots display how the luminance for a single pixel as estimated by the comparison methods changes over time in three of our sequences.

MIR [1] relies on light paths that are distributed according to the radiance they bring the camera, which the original implementation generates using Metropolis Light Transport [8]. As the visibility and the materials are not complicated in our test scenes, we instead generate these paths by resampling paths from our path tracer. We use a resampling rate of 256 : 1.

TCAS [26] uses Metropolis-Hastings to directly sample VPLs from reflective shadow maps. Barák et al. propose to use five of Metropolis-Hastings iterations per frame, but we found that this produces unacceptable quality in larger scenes. Instead, we use 16 iterations in our tests.

5.2 Distribution quality

The image quality and temporal stability of our method is best demonstrated in the supplemental videos. We also perform a numerical comparison by tracking the luminance of 8 pixels over time in five sequences (see Figure 7 for examples). We compute ground truth values for these pixels using 692100 samples from our path tracer. Table 1 shows how closely the indirect illumination follows the reference with the average absolute error $|E|$ over all pixels in the sequences. To capture perceptually displeasing temporal high

TABLE 2: Breakdown of the time spent computing indirect light in the scenes. All timings are in milliseconds. **Left to right:** the scene (Scene) ; number of triangles (#Tris) ; number of VPLs (#VPLs) ; generating candidate VPLs, detecting undersampled regions and replacing invalidated VPLs (Cand. sampling) ; resampling candidate VPLs (Cand. resampling) ; verifying indirect visibility (Indirect Visibility) ; estimating VPL radiosities (Radiosity) ; kNN density estimation and building the BVH (kNN) ; total time for VPL sampling (Total) ; rendering shadow maps (Shadow maps) ; interleaved shading and including building the G-Buffer (Interl. shading) ; total time for final rendering (Total) ; total time for indirect illumination (Total).

			Sampling VPLs						**Rendering**			**Total**
Scene	**#Tris**	**#VPLs**	Cand. sampling	Cand. resampling	Indirect visibility	Radiosity	kNN	Total	Shadow maps	Interl. shading	Total	
Epic Citadel	373K	2048	3.3	3.0	1.6	11.0	2.6	21.5	8.7	20.0	28.7	50.2
Soda Hall	551K	2048	2.3	2.2	1.2	7.8	1.7	15.2	15.9	19.1	35.0	50.2
Crytek Sponza	262K	2048	2.7	2.5	2.0	9.8	1.6	18.6	6.0	24.4	30.4	49.0
Maze	471K	512	2.2	1.6	1.1	6.8	1.6	13.3	11.7	10.5	22.2	35.5

frequency changes in the indirect illumination (“flickering”), we compute for each pixel the finite difference

$$E' = \frac{\Delta \text{Error}}{\Delta t} \tag{10}$$

of the error with respect to time. In the table we show the average absolute E' over all pixels in the sequences. Note that E' shows how the error fluctuates in the images, not in the scene, and can be non-zero even if the VPLs stay fixed, but the camera moves.

We chose our test scenes to reflect the difficult occlusion characteristics typical for video game levels. This especially true for Soda Hall and Epic Citadel, where only a small portion of the scene is visible at any given time and the set of indirectly visible surfaces (where VPLs can bring light to the image) is small in comparison to the entire scene. This highlights the importance of accounting for precise indirect visibility. First, in Soda Hall even the view-adaptive TCAS algorithm places VPLs in brightly illuminated rooms behind walls where they do not affect the image. Second, bright sunlight illuminates a majority of the Epic Citadel scene. Here, when the camera lies on the street level, the set of illuminated surfaces is much larger than the set of indirectly visible surfaces. Only MIR and our method produce distributions that match the image well.

As the supplemental videos, Figure 7 and Table 1 all show, our method outperforms the comparison methods in terms of temporal stability, more so when the light sources are moving (e.g. Epic Citadel). We see that the improved temporal stability does not affect the image quality, as the error made by our method is comparable to MIR. To see the effect of our importance sampling distribution (Section 3.1), we encourage the reader to compare our method to MIR in the Soda Hall sequence and observe new rooms appearing at the end of long corridors. Note that TCAS naturally has a larger error than the other methods as it does not support multi-bounce indirect illumination.

5.3 Rendering performance

We measure performance by rendering images in 1920×1080 on a PC with an Intel i7 4820K CPU, 16 GB RAM and a NVIDIA GTX Titan X GPU. See Table 2 for a timing breakdown we performed by measuring the speedup from disabling each component of the algorithm and scaling the results to match the total frame time. Most time is spent on interleaved shading, rendering shadow maps and computing VPL radiosities. Less time can be spent computing radiosities by reducing the number of indirect bounces. In scenes with simple visibility, VPL candidate resampling can be disabled for a slight increase in performance. Interleaved sampling can be sped up using larger tiles with the risk of introducing structured noise or oversmoothing the indirect illumination. The performance of most components is linear w.r.t. the number of VPLs, except for the kNN search (a negligible part of the frame time) and rendering shadow maps which depends on the budget M_{max} of VPLs that can move each frame.

Table 2 shows that more than half of the frame time is spent on parts shared by all VPL rendering algorithms (rendering shadow maps and interleaved shading). Therefore, other sampling methods can be at most twice as fast as ours. As the comparison methods do not limit how many VPLs move between frames, rendering shadow maps significantly limits their performance [21], favoring our technique. We feel these bounds are tight enough to not warrant a study using highly tuned implementations. Hence we do not compare timings and we use the same VPL budget for all methods instead of performing equal-time comparisons.

5.4 Illumination quality

We now study the single frame quality of our method. Figure 8 compares path traced ground truth renderings computed using 32 768 paths per pixel to our results computed using a fixed budget of 2048 VPLs. We attempt to maximally separate the two main factors that contribute to final image quality, namely, the VPL distribution itself and the method for determining pixel-to-VPL visibility. The former is at the core of the present method; the latter is important, as shadow maps suffer from well-known artifacts particularly at lower resolutions.

We rendered the VPL results in three different settings: 1) shadow mapping with interleaved sampling (i.e., baseline method, current feasible use case), 2) shadow mapping with exhaustive sampling of all VPLs (i.e., maximal quality from the same VPL distribution with shadow maps), and 3) exhaustive ray traced visibility between all pixels and all VPLs (i.e., maximal quality from the same VPL distribution with perfectly accurate visibility). In all scenes, we manually clamped the geometry term between the VPLs and the primary hits to a value suitable for 2048 VPLs.

To further highlight the differences between VPLs and path tracing, we computed, in addition to standard path tracing, a modified path traced result where we applied clamping similar to VPL rendering. More precisely, we

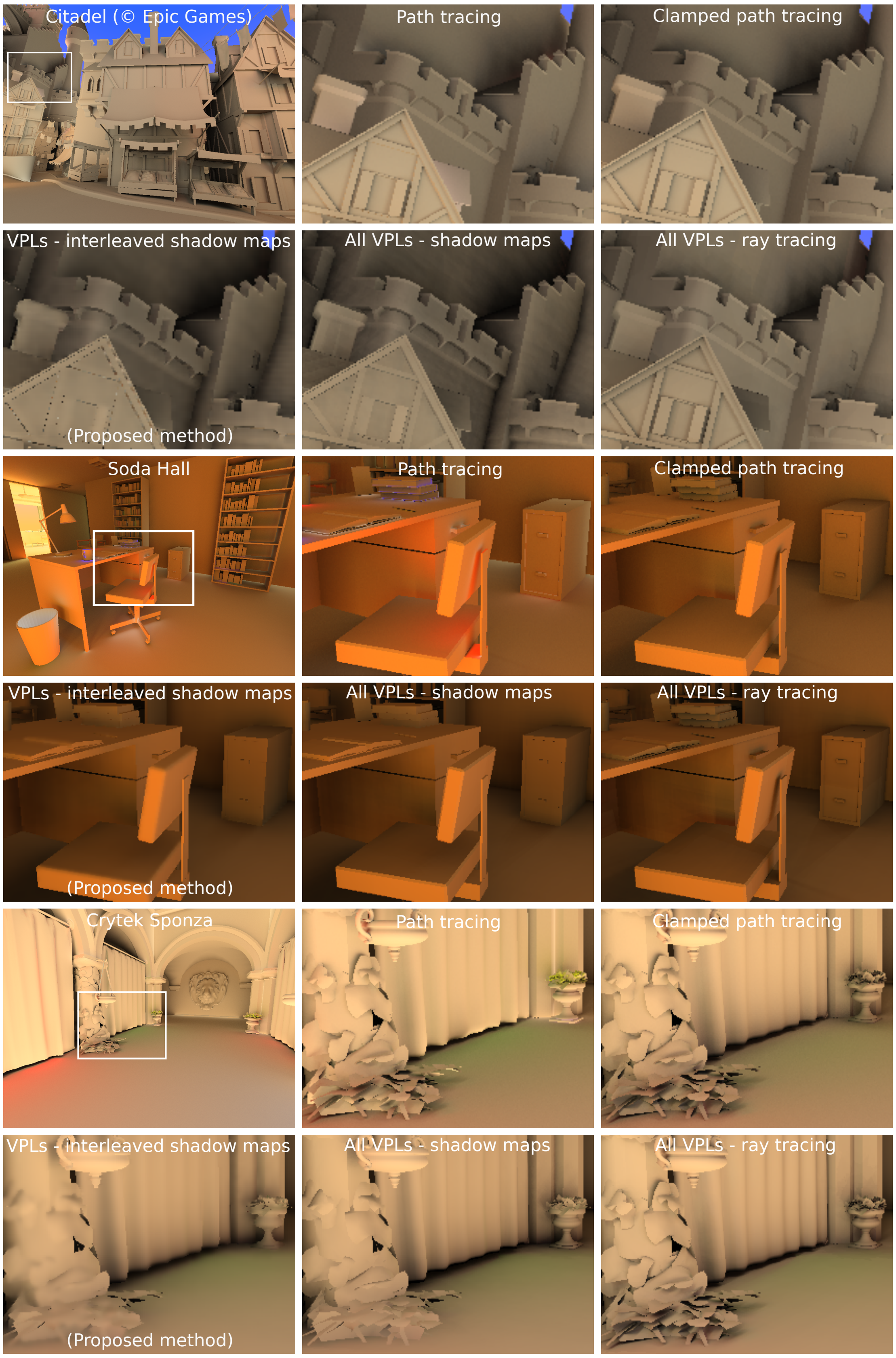


Fig. 8: Comparison between path traced references and VPL renderings using different methods of determining visibility between VPLs and pixels. The scenes were rendered with full global illumination and diffuse texture maps. To emphasize the differences between the methods, we only show the indirect illumination. **Top to bottom:** Citadel (© Epic Games), Soda Hall and Crytek Sponza. Each scene has been rendered using path tracing, path tracing with simulated clamping (**Clamped path tracing**), as well as the proposed method that uses shadow maps and interleaved sampling (**VPLs - interleaved shadow maps**). Additionally, results computed by exhaustively sampling all VPLs by shadow maps (**All VPLs - shadow maps**), and by exhaustively sampling all VPLs with shadow rays (**All VPLs - ray tracing**) are shown. For each scene, all three VPL renderings use the same set of 2048 VPLs. See Section 5.4 for discussion.

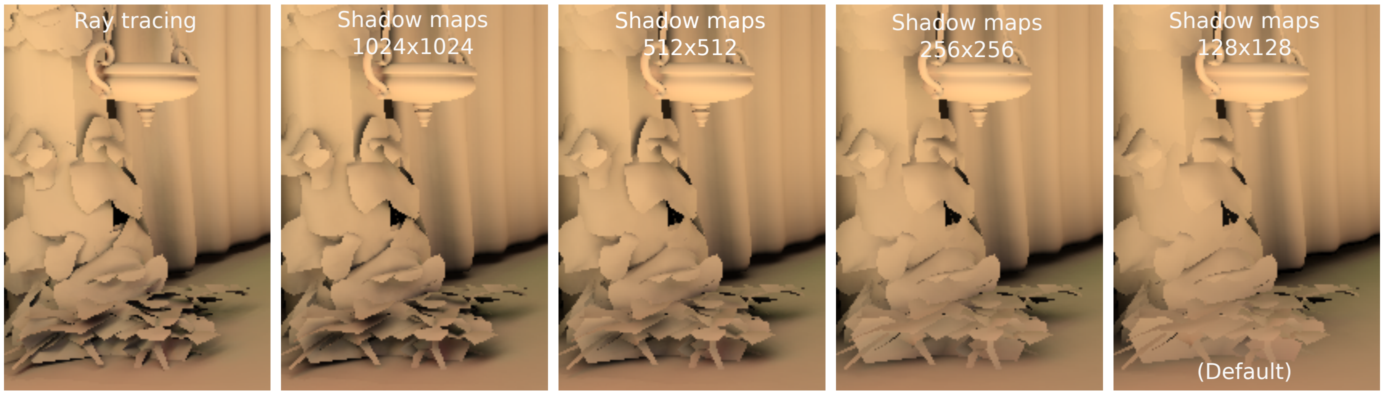


Fig. 9: The effect of shadow map resolution on the indirect illumination. Each image was rendered using the same set of 2048 VPLs generated by our method. **Left to right:** Ray traced reference image (38 MB for BVH), 1024 × 1024 shadow maps (8 GB), 512 × 512 shadow maps (2 GB), 256 × 256 shadow maps (512 MB) and 128 × 128 shadow maps (128 MB). The other experiments use a default resolution of 128 × 128 for the shadow maps.

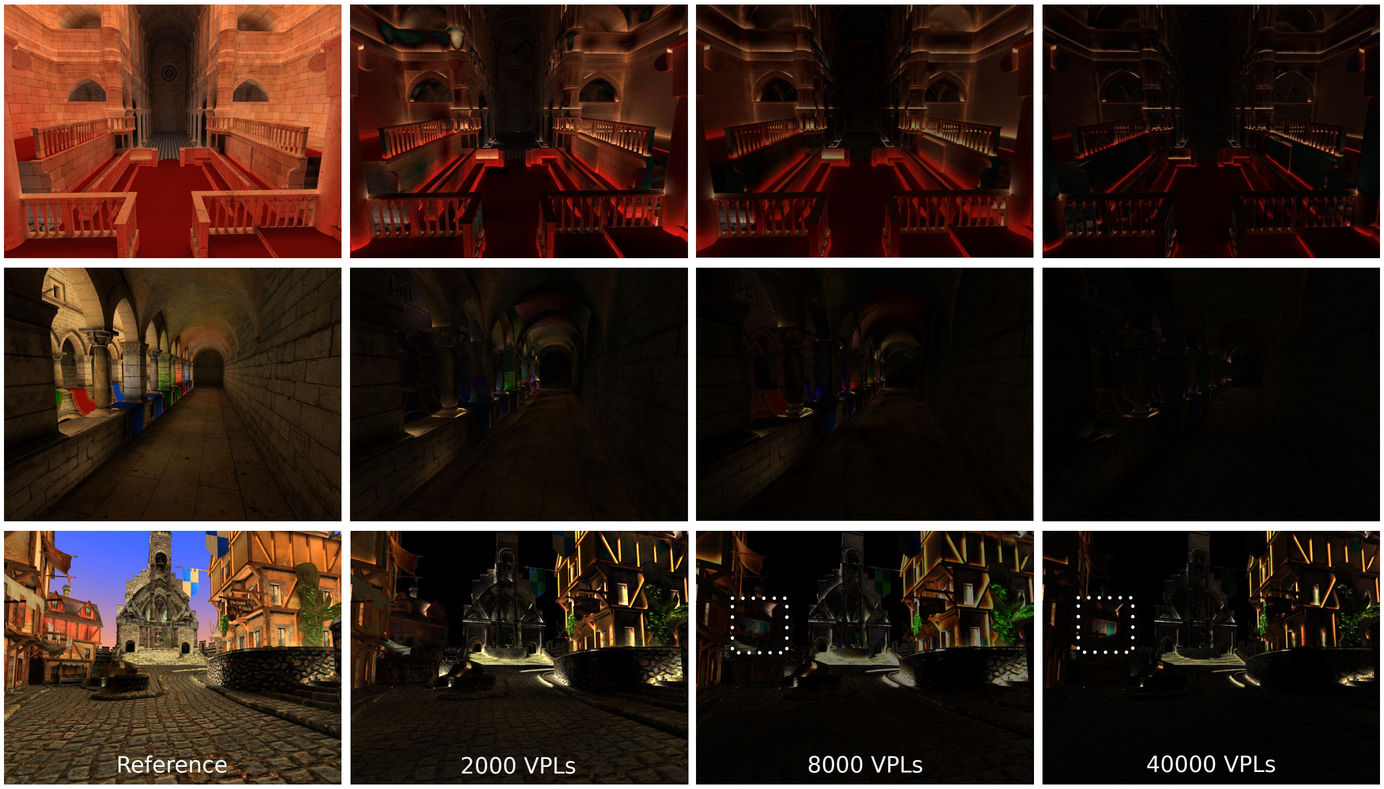


Fig. 10: The error of the indirect illumination produced by our method with different VPL budgets. In this experiment, we render the images with ray traced visibility between the VPLs and the pixels. **Left to right:** Reference image, 2× absolute error (2K VPLs), 2× absolute error (8K VPLs), 2× absolute error (40K VPLs). **Top to bottom:** Sibenik Cathedral, Crytek Sponza, Citadel (© Epic Games). Most of the remaining error with 40K VPLs is attributed to energy loss due to clampling. However, as we discuss in the text, the intensities of some VPLs in Citadel have been over-estimated.

clamp the geometry term of all light transport path segments that originate from the primary hits, except for those that directly connect to a light source. This effectively corresponds to using an infinite number of VPLs, and the result approximates the maximal quality obtainable with our chosen level of clamping. Comparing the two path traced images, we observe, as expected, that clamping removes energy from the close-by interreflections.

As is to be expected, the use of shadow maps washes out the details in the indirect illumination. The shadows next to the window in Epic Citadel are less pronounced and the walls in the background are clearly affected by shadow acne. In Soda Hall, the shadows on the back of the chair disappear together with more subtle effects such as the shadows on the books and the contact shadow of the pen on the table. In Crytek Sponza the shadows on the foliage almost completely disappear, making it difficult to distinguish the individual leaves. Interleaved sampling exacerbates this loss of detail. In Epic Citadel this manifests as aliasing artifacts near the edge of the roof and in Crytek Sponza the foliage becomes even less pronounced.

Interestingly, apart from mild shadow faceting, the images produced by ray casting are nearly indistinguishable from the path traced images with simulated clamping, suggesting that a budget of 2048 VPLs is quite sufficient for capturing the indirect light field in these scenes apart from near-field interactions removed by clamping. This also shows clearly that the inaccuracy of the visibility results produced by shadow maps is a significant negative factor in the total quality of VPL rendering. Although this result is only an existence proof — at present, exhaustive sampling of all VPLs with shadow rays is infeasible — it suggests that more research on making use of fast modern ray tracers in VPL rendering is warranted.

Finally, we study the effect of shadow map resolution on the results: while no known solutions exist for pixel-perfect shadow mapping apart from irregular sampling [39], clearly, increasing the resolution of the shadow maps mitigates the problems. In Figure 9 we change the resolution of the shadow maps and adjust the depth bias accordingly. We find that a shadow map resolution of 1024×1024 results in quality roughly on par with ray traced visibility, although in this case the shadow maps require an unwieldy 8 GB of storage. However, increasing the resolution has only a limited impact on performance as our sampling algorithm only requires a few shadow maps to be rendered each frame. Furthermore, in most experiments rendering shadow maps is limited by geometry processing, not fill rate.

In Figure 10 we show how the quality of the indirect illumination produced by our method increases with the VPL budget. To focus on the error of our sampling method, we only render indirect illumination and evaluate the visibility between VPLs and pixel samples with ray casts. Note that the error is much reduced when the VPL budget is increased to 40000, where most of the remaining error is explained by aliasing and energy loss due to clamping. However, one surface in Epic Citadel still exhibits noticeable error. The probability density of strong VPLs illuminating it has been underestimated as they lie on a thin structure (a clothes line). This is to be expected, as kNN density estimation cannot detect such high-frequency variation in the PDF with a restricted sample budget.

6 Conclusions

The experiments show that our method renders temporally coherent multi-bounce diffuse indirect illumination at real-time rates in large and heavily occluded scenes. We achieve this with an incremental, adaptive sampling algorithm that reuses VPLs from previous frames and places new VPLs to equalize their intensities while accounting for the movement of the camera and the lights. We believe this is the first time these goals have been accomplished in highly occluded scenes with a single-frame illumination quality on par with the state of the art that does not enforce temporal coherence. Naturally, as our algorithm takes up the entire GPU, usage in the tightly constrained performance envelopes of actual products is currently infeasible.

In contrast to many prior VPL rendering algorithms, we use GPU ray tracing in the sampling process. Given the recent advent of efficient BVH builders and their increasing support for dynamic scenes, we believe this trade-off is worthwhile particularly considering the future. Our experiments show that our method produces high quality indirect illumination with only 2048 VPLs, where the most visible artifacts of are caused by shadow maps and clamping. We feel it is an interesting avenue to reduce these artifacts by combining our sequential sampler with techniques for VPL importance sampling [9], [40], enabling the use of ray tracing when determining VPL-pixel visibility. In fact, an early version of this work features an initial study in this direction [41].

Although we have not shown results with dynamic scenes, our method (like prior methods) is able to *illuminate* moving objects even if they do not affect the light flow. This is due to the way we resolve visibility, as shadow maps with moving objects need to be re-rendered every frame. This limitation can be overcome by approximating visibility for the shadow maps [10], although this reduces the quality of the indirect illumination. In contrast, ray tracing demonstrably produces high quality indirect illumination and the acceleration structure only needs to be rebuilt once every frame. We see a strong trend towards replacing shadow maps with ray casts, even in dynamic scenes.

Acknowledgments

We thank Timo Aila, Samuli Laine, Tobias Ritschel, Perttu Hämäläinen, Clément Godard, Corneliu Ilisescu, Moos Hueting, Daniel Worrall and Stefan Garbin for brainstorming and helpful comments. Thanks to Lars Hedman for help with the text and his support. Thanks to Epic Games for Citadel and Erik Sintorn for help with it. Thanks to Marko Dabrovic for Sibenik, Frank Meinl for Crytek Sponza, the U.C. Berkeley WALKTHRU project for Soda Hall and the Stanford Computer Graphics Lab for Bunny and Armadillo. This work was partly funded by the EU project CR-PLAY (no 611089) www.cr-play.eu and EPSRC EP/K023578/1.

References

[1] B. Segovia, J.-C. Iehl, and B. Péroche, “Metropolis instant radiosity,” *Comput. Graph. Forum*, vol. 26, no. 3, pp. 425–434, 2007.

[2] C. Dachsbacher and M. Stamminger, “Reflective Shadow Maps,” in *Proc. I3D ’05*, 2005, pp. 203–231.

[3] A. Keller, “Instant Radiosity,” in *Proc. SIGGRAPH ’97*, 1997, pp. 49–56.

[4] A. Keller and W. Heidrich, “Interleaved sampling,” in *Proc. EGWR ’01*, 2001, pp. 269–276.

[5] P. Hedman, T. Karras, and J. Lehtinen, “Sequential Monte Carlo Instant Radiosity,” in *Proc. I3D ’16*, 2016, pp. 121–128.

[6] C. Dachsbacher, J. Křivànek, M. Hašan, A. Arbree, B. Walter, and J. Novàk, “Scalable realistic rendering with many-light methods,” *Comput. Graph. Forum*, vol. 33, no. 1, pp. 88–104, 2014.

[7] T. Ritschel, C. Dachsbacher, T. Grosch, and J. Kautz, “The state of the art in interactive global illumination,” *Comput. Graph. Forum*, vol. 31, no. 1, pp. 160–188, 2012.

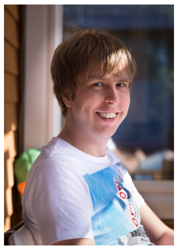
[8] E. Veach and L. J. Guibas, “Metropolis Light Transport,” in *Proc. SIGGRAPH ’97*, 1997, pp. 65–76.

[9] B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg, “Lightcuts: A scalable approach to illumination,” vol. 24, no. 3, pp. 1098–1107, Jul. 2005.

[10] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz, “Imperfect shadow maps for efficient computation of indirect illumination,” vol. 27, no. 5, pp. 129:1–129:8, Dec. 2008.

[11] I. Wald, T. Kollig, C. Benthin, A. Keller, and P. Slusallek, “Interactive Global Illumination Using Fast Ray Tracing,” in *Proc. EGWR ’02*, 2002, pp. 15–24.

[12] B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche, "Non-interleaved Deferred Shading of Interleaved Sample Patterns," in *Proc. Graphics Hardware '06*, 2006, pp. 53–60.
[13] T. Kollig and A. Keller, "Illumination in the presence of weak singularities," in *Monte Carlo and Quasi-Monte Carlo Methods*, 2004, pp. 245–257.
[14] M. Hašan, J. Křivánek, B. Walter, and K. Bala, "Virtual spherical lights for many-light rendering of glossy scenes," vol. 28, no. 5, pp. 143:1–143:6, Dec. 2009.
[15] B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche, "Bidirectional Instant Radiosity," in *Proc. EGSR '06*, 2007, pp. 389–397.
[16] I. Georgiev and P. Slusallek, "Simple and Robust Iterative Importance Sampling of Virtual Point Lights," in *Proc. Eurographics (short papers)*, 2010.
[17] F. Simon, J. Hanika, and C. Dachsbacher, "Rich-VPLs for improving the versatility of many-light methods," *Comput. Graph. Forum*, vol. 34, no. 2, pp. 575–584, 2015.
[18] H. W. Jensen, *Realistic Image Synthesis Using Photon Mapping*. Natick, MA, USA: A. K. Peters, Ltd., 2001.
[19] T. Ritschel, E. Eisemann, I. Ha, J. D. K. Kim, and H.-P. Seidel, "Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes," *Comput. Graph. Forum*, vol. 30, no. 8, pp. 2258–2269, 2011.
[20] D. Scherzer, L. Yang, O. Mattausch, D. Nehab, P. V. Sander, M. Wimmer, and E. Eisemann, "Temporal coherence methods in real-time rendering," *Comput. Graph. Forum*, vol. 31, no. 8, pp. 2378–2408, Dec. 2012.
[21] S. Laine, H. Saransaari, J. Kontkanen, J. Lehtinen, and T. Aila, "Incremental Instant Radiosity for Real-Time Indirect Illumination," in *Proc. EGSR'07*, 2007, pp. 277–286.
[22] M. Knecht, C. Traxler, O. Mattausch, W. Purgathofer, and M. Wimmer, "Differential Instant Radiosity for mixed reality," in *Proc. ISMAR*, 2010, pp. 99–107.
[23] M. Hašan, E. Velázquez-Armendariz, F. Pellacini, and K. Bala, "Tensor Clustering for Rendering Many-Light Animations," in *Proc. EGSR '08*, 2008, pp. 1105–1114.
[24] I. Wald, C. Benthin, and P. Slusallek, "Interactive Global Illumination in Complex and Highly Occluded Environments," in *Proc. EGRW '03*, 2003, pp. 74–81.
[25] R. Prutkin, A. Kaplanyan, and C. Dachsbacher, "Reflective Shadow Map Clustering for Real-Time Global Illumination," in *Proc. Eurographics (short papers)*, 2012.
[26] T. Barák, J. Bittner, and V. Havran, "Temporally coherent adaptive sampling for imperfect shadow maps," *Comput. Graph. Forum*, vol. 32, no. 4, pp. 87–96, 2013.
[27] T. Aila and V. Miettinen, "dPVS: An occlusion culling system for massive dynamic environments," *IEEE Comput. Graph. Appl.*, vol. 24, no. 2, pp. 86–97, 2004.
[28] J. T. Kajiya, "The rendering equation," in *Proc. SIGGRAPH '86*, 1986, pp. 143–150.
[29] D. O. Loftsgaarden and C. P. Quesenberry, "A nonparametric estimate of a multivariate density function," *The Annals of Mathematical Statistics*, vol. 36, no. 3, pp. 1049–1051, 1965.
[30] O. Cappe, S. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proc. IEEE*, vol. 95, no. 5, pp. 899–924, May 2007.
[31] P. Hämäläinen, S. Eriksson, E. Tanskanen, V. Kyrki, and J. Lehtinen, "Online motion synthesis using sequential monte carlo," *ACM Trans. Graph.*, vol. 33, no. 4, 2014.
[32] "OptiX," https://developer.nvidia.com/optix, 2015.
[33] T. Karras, "Maximizing Parallelism in the Construction of BVHs, Octrees, and k-d Trees," in *Proc. High-performance Graphics*. Eurographics Association, 2012, pp. 33–37.
[34] E. Eisemann and F. Durand, "Flash photography enhancement via intrinsic relighting," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 673–678, 2004.
[35] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama, "Digital photography with flash and no-flash image pairs," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 664–672, 2004.
[36] S. Brabec, T. Annen, and H.-P. Seidel, "Shadow Mapping for Hemispherical and Omnidirectional Light Sources," in *Proc. Computer Graphics International*, 2002, pp. 397–408.
[37] R. Kooima, "Generalized Perspective Projection," http://csc.lsu.edu/ kooima/pdfs/gen-perspective.pdf, 2008.
[38] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic tone reproduction for digital images," vol. 21, no. 3, Jul. 2002.
[39] T. Aila and S. Laine, "Alias-free Shadow Maps," in *Proc. EGSR '04*, 2004, pp. 161–166.
[40] S. Popov, R. Ramamoorthi, F. Durand, and G. Drettakis, "Probabilistic connections for bidirectional path tracing," *Comput. Graph. Forum*, vol. 34, no. 4, 2015.
[41] P. Hedman, "Sequential Monte Carlo Instant Radiosity," Master's thesis, University of Helsinki, 2015.



Peter Hedman Peter Hedman is a PhD student in computer vision at University College London, supervised by Gabriel Brostow. Before this, he worked on interactive global illumination as a research intern at NVIDIA. His research interests involve image-based rendering, 3D reconstruction (using stereo and depth cameras) and physically based rendering. His interests also extend to GPU computing and real-time graphics. In 2016 he received the prize for the most distinguished Master's thesis from MAL, the Finnish Academic Association for Mathematics and Natural Sciences. He also received the 2016 Rabin Ezra scholarship for doctoral students in computer graphics, imaging and vision.

Tero Karras works as Principal Research Scientist at NVIDIA Research, which he joined in 2009. His research interests include GPU computing, parallel algorithms, real time ray tracing, and machine learning. He has had a pivotal role in NVIDIA's ray tracing research, especially related to efficient construction of acceleration structures.

Jaakko Lehtinen is an Associate Professor of computer science at Aalto University, and a Principal Research Scientist with NVIDIA Research in Helsinki, Finland. His research interests include physically-based rendering, Monte Carlo methods, and appearance modelling and acquisition.