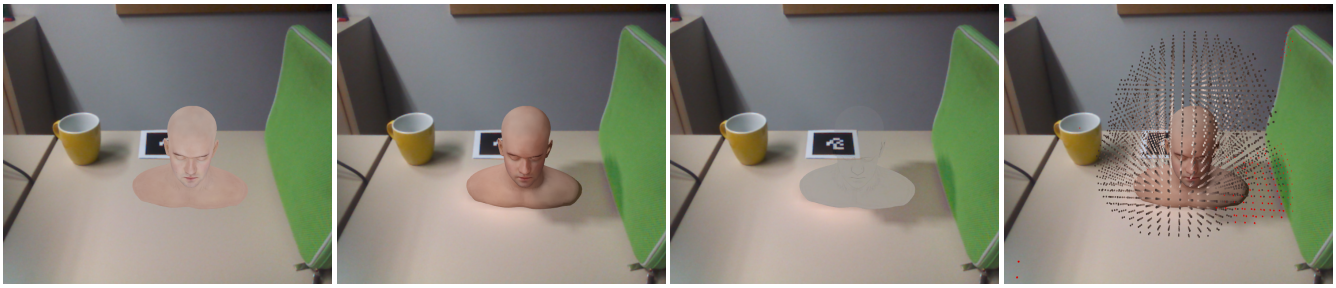
Delta Light Propagation Volumes for Mixed Reality

Tobias Alexander Franke*

Fraunhofer IGD & TU Darmstadt
Germany



(a) No illumination (b) Full DLPV rendering (c) DLPV effects only (d) Volume visualization

Figure 1: Infinite Head model inserted into a real scene with one reconstructed light source: (a) a virtual object is inserted without illumination, (b) visible first bounce around the base as well as low resolution shadow, (d) indirect effects without virtual object for better visualization, (d) visualization of the DLPV (red dots indicate negative values).

ABSTRACT

Indirect illumination is an important visual cue which has traditionally been neglected in mixed reality applications. We present Delta Light Propagation Volumes, a novel volumetric relighting method for real-time mixed reality applications which allows to simulate the effect of first bounce indirect illumination of synthetic objects onto a real geometry and vice versa. Inspired by Radiance Transfer Fields, we modify Light Propagation Volumes in such a way as to propagate the change in illumination caused by the introduction of a synthetic object into a real scene. This method combines real and virtual light in one representation, provides improved temporal coherence for indirect light compared to previous solutions and implicitly includes smooth shadows.

Keywords: Mixed Reality, Real-time Global Illumination

Index Terms: H.5.1 [HCI]: Multimedia Information Systems—Artificial, augmented, and virtual realities I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Display algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1 INTRODUCTION

Apart from geometric registration, fusing synthetic objects with a real context requires believable interaction of real and virtual light. Where shadows provide the visual cues to locate a virtual object in a real scene and transferred light from real light sources let it appear in harmony with its surrounding, the mutual indirect interaction of illumination is a necessary detail to convince an observer that the rendered result is not merely augmented but part of the scene. Such a *mixed reality* (MR) system has applications in movie production, advertisement of unfinished products or cultural heritage visualization.

*email:tobias.franke@igd.fraunhofer.de

While there are a range of global illumination solutions available for offline renderers that produce plausible results (for instance through photon mapping [9]), most real-time MR systems lack such sophistication and often greatly simplify shading. One important aspect that is lost in the process is the mutual influence of indirect illumination between real and virtual objects. Attempts have been made to resolve this issue with precomputed global illumination solutions [10] or Instant Radiosity [17].

In this paper, we present a novel global illumination solution based on Light Propagation Volumes [13] and Differential Rendering [5], which models the change of near-field illumination introduced by the presence of a virtual object. This differential can later be used to relight the real scene. Our solution aims to be completely dynamic and interactive at the same time.

Our *contributions* are summarized as follows:

- A system for interactive near-field indirect illumination to model lighting influence of virtual objects
- A novel volume-based relighting solution for MR at low computational cost without any requirements for precomputation
- Implicit shadows through differential light injection

2 RELATED WORK

2.1 Relighting

Our method focuses on photometric registration of first bounce indirect illumination introduced by the presence of a virtual object in a real scene. Several methods have been proposed to simulate light interaction between real and virtual objects that either track light sources or extract incident illumination from hemispherical captured images.

Fournier et al. [6] use Radiosity to compute a global illumination solution for a virtual object in a reconstructed scene. For virtual objects, the accumulated Radiosity result is used, while for real objects only the additional delta is added. Our method closely relates to this approach.

Debevec [5] captures real light from an HDR lightprobe instead of using synthetic light sources. The resulting images are mapped onto objects to simulate the reflection of real light on virtual objects. Differential rendering is introduced to superimpose lighting interaction between virtual and real surfaces on a background image, which has been subsequently turned into an interactive method in [8] ignoring however any kind of inter-reflection between objects.

Many different methods exist to reconstruct light sources from lightprobes or fish-eye lens cameras. Nowrouzezahrai et al. [20] factorize a low-order spherical harmonic representation of a lightprobe into a directional and a global component. The directional component can be used as regular point light while the global term is applied using matrix radiance transfer. Shadows are cast using PCF shadow mapping. Viriyothai and Debevec [26] sample an environment map using variance minimization techniques on a median cut algorithm. Finally, single light sources can also be tracked with markers.

A recent publication by Karsch et al. [14] describes a method to insert virtual objects into legacy photographs. With the help of user annotations to define occluding geometry and light sources or light shafts, scene geometry and lighting is estimated and several reconstruction processes are avoided. In contrast, our method focuses on real-time insertion into dynamic scenes instead of static images. Light sources and geometry are automatically reconstructed from camera images capturing scene depth, incident environment light or tracked light sources.

2.2 Precomputed methods

Grosch et al. [10] compute scaling factors for subdivided distant illumination regions which are used to linearly combine multiple basis Irradiance Volumes. The final volume is used to query illumination for a surface point of a synthetic object and can transfer indirect illumination from the virtual scene. Our method is also volumetric, but has no precomputation or rigidness requirement.

In a preprocess Kakuta et al. [12] discretize the hemisphere around a virtual object into a polyhedron. A set of basis shadow maps are computed from the position of each vertex of the polyhedron, which are linearly combined with coefficients computed from the luminance per unit area of a hemispherical image of each face. In a similar fashion Shadow Fields [27] are used to insert virtual shadows in [7]. A shell around a virtual object contains precalculated occlusion data. When rendering, intersecting real geometry transfer is augmented with the shadow transfer through a spherical harmonic triple product. The extended and very similar Radiance Transfer Fields [22] have been proposed to be used to transfer indirect light to surrounding real geometry. However, this requires the virtual object to be rigid.

2.3 Real-time global illumination

Many current real-time global illumination methods are based on Instant Radiosity [16], which approximates first bounce indirect illumination with the help of small light sources called virtual point lights (VPL) instead of gathering all contribution along a light path. An elegant method to create such VPLs has been presented in [2]: a Reflective Shadow Map (RSM) is an extension to regular shadow maps, adding normals and albedo. Assuming that pixels in a RSM represent diffuse reflectors one can use it to reconstruct VPLs with position, direction and flux.

VPL-based methods typically add up to high evaluation costs due to significant overdraw or suffer from temporal incoherence when using a low number of indirect lights, which results in flickering of indirect light. VPLs also create local singularities near the surface they are placed on (visible as bright spots) which are usually countered by clamping the contribution of a VPL at the cost of some energy loss. To combat overdraw, Dachsbacher and Stamminger [3] propose splatting VPLs with a clamped influence radius. Nichols and Wyman [19] furthermore exploit the low-frequency nature of indirect diffuse and glossy light. They identify discontinuities in the final image to render splats at lower resolution where no sudden changes occur. Tiled Shading [21] subdivides the image space into regular tiles. In a separate step, for each tile a list of influencing light sources is compiled, reducing the final shading cost.

Several clustering methods exist to increase the amount of VPLs without hurting performance. Prutkin et al. [23] importance sample an RSM and combine several VPLs via k-means clustering into area lights. Light Propagation Volumes (LPV) [13] are inspired by discrete ordinate methods. An RSM is computed and VPLs generated from it are injected into a small volume as low order spherical harmonic coefficients. The injected light is then propagated through the volume in several steps, after which it can be queried for indirect contribution by transforming positions into LPV space. LPVs provide smooth indirect illumination, clustering many lights into a compact representation. We make use of LPVs for our own representation in Section 4.

Differential Instant Radiosity [17] makes use of RSMs to transfer indirect light onto surrounding real geometry. Through differential rendering, first bounce indirect light is extracted and can be added to the background image. To handle a low amount of VPLs without flickering, the authors exploit frame-to-frame coherence and blend illumination from each previous frame by calculating a confidence value for each pixel which depends on the differential of the normal, illumination and position. In contrast, Lensing and Broll [18] use multi-resolution splatting [19] to use a high number of VPLs. Our approach is independent from the number of VPLs and therefore simplifies this process, does not suffer from intrinsic flickering and is evaluated at lower runtime costs.

Dachsbacher et al. [4] avoid calculating explicit visibility with the help of *Antiradiance*. A reformulation of the rendering equation simply avoids any occlusion in the transport operator. To compensate for the extraneous radiance, the same quantity of incident light at each surface is injected and propagated negatively on its backside, canceling out incorrectly transmitted light. Our approach implicitly contains Antiradiance.

Screen-space Directional Occlusion, a method introduced in [24] that is similar to screenspace ambient occlusion (SSAO), samples the neighboring color buffer of a texel to gather near-field indirect illumination. While this method is very flexible, it is prone to the same limitations as SSAO.

3 Reconstruction

3.1 Real light sources

Reconstructing light sources from the real surrounding can be done in many ways. In our implementation, we support two modes. A simple reconstruction of a point light source with an attached marker extracts position only, while we manually set intensity and other parameters to match the real source.

In a second mode, we use a variance minimized median cut algorithm to sample and extract several light sources from a hemispherical camera [26] for each frame. Spherical coordinates (θ, ϕ) for light sources are determined which are then placed on a sphere around the virtual object. In Figure 2, this process has been used to reconstruct three light sources in a room.

3.2 Real scene

The real scene can be partially reconstructed from a depth sensor such as a Microsoft Kinect camera. However, the depth sensor image usually contains artifacts such as noise (e.g., visible variance on planar surfaces) or holes (e.g., from invalid or unavailable depth data). These artifacts affect shading of the real scene later on when incident virtual light is reflected incorrectly and small shadows or bumps appear on otherwise perfectly flat areas. By prefiltering the depth image before extracting world positions and normals we can

Figure 2: A cup rendered with three light sources that are reconstructed from a hemispherical camera image.

reduce variance on surfaces and potentially close smaller holes in the depth image. We follow the approach of [18] and use Guided Image Filtering [11] for this operation.

One should keep in mind that this reconstruction is not complete: the backside of each object, invisible to the depth sensor, is missing. Real light sources illuminating the scene from this direction will not be properly processed, because the geometry to bounce off from is missing. If this is a requirement, other means of reconstruction have to be found.

We estimate simple diffuse surface parameters with the reconstructed light sources, ignoring shadowed areas as well as highlights in the image. For each pixel intensity I_p, we can recover the diffuse parameter k_d with:

$$k_d = \frac{I_p}{\sum_i L_i \left\langle \vec{n}_p, \vec{\omega}' \right\rangle_+} \tag{1}$$

L_i represents the radiance from light source i; $\vec{n}_p$ is the normal at pixel p; $\left\langle \vec{n}_p, \vec{\omega}' \right\rangle_+$ is the geometric term, where $\langle , \rangle_+$ denotes a dot product with negative values clamped to zero and $\vec{\omega}'$ is the incident light direction.

If the real scene is close to the camera, the depth reconstruction provided by the sensor might fail. In this case, we use a manually reconstructed model of the scene.

4 Delta Light Propagation Volumes

4.1 Formal definition

In geometric optics a simplified approximation to the electromagnetic field for light traveling through space and scattering is the radiance field:

$$L : \mathbb{R}^3 \times S^2 \to \mathbb{R}^3 \tag{2}$$

For a point in space $\mathbb{R}^3$ and a direction on a sphere S^2 a mapping exists to radiant flux (usually linear color space RGB $\mathbb{R}^3$). This 5-dimensional rendering equation $L(x, \vec{\omega})$ can be evaluated for instance for a surface point x in direction $\vec{\omega}$.

$$\begin{aligned} L(x, \vec{\omega}) &= L_e(x, \vec{\omega}) + \\ &\int_{S^2} L(x, -\vec{\omega}') f(x, \vec{\omega}, \vec{\omega}') \left\langle \vec{n}_x, \vec{\omega}' \right\rangle_+ d\vec{\omega}' \end{aligned} \tag{3}$$

It is composed from these parts: L_e is the emissive radiance; f is a BSDF; $\left\langle \vec{n}_x, \vec{\omega}' \right\rangle_+$ is the geometric term for normal $\vec{n}_x$ at x and incident light direction $\vec{\omega}'$. We can expand the equation into a Neumann series with the integral expressed as a linear transport operator $\mathbf{T}$. If $\mathbf{T}$ is contractive, then L has a solution:

$$\begin{aligned} L &= L_e + \mathbf{T}L && (4) \\ (\mathbf{I} - \mathbf{T})L &= L_e && (5) \\ L &= (\mathbf{I} - \mathbf{T})^{-1} L_e = \sum_{i=0}^{\infty} \mathbf{T}^i L_e && (6) \end{aligned}$$

Each term $\mathbf{T}^i$ intuitively represents scattering of one bounce of light, where $L_0 = L_e$ is the emissive light only, $L_1 = \mathbf{T}L_e$ is direct light, $L_2 = \mathbf{T}^2 L_e$ the first indirect bounce of light, and so on.

Consider an existing radiance field L^μ. When we insert another non-emissive object O into the scene covered by L^μ, the properties of the radiance field change: light is either blocked or scattered by the newly inserted geometry. To account for this *change* in the radiance field, consider another field L^ρ which has the same light configuration as L^μ and additionally contains O. The change in a radiance field L^μ by introducing O is therefore a *Delta Radiance Field* $L^\Delta = L^\rho - L^\mu$.

$$\begin{aligned} L^\Delta &= L^\rho - L^\mu && (7) \\ &= \sum_{i=0}^{\infty} \mathbf{T}_\rho^i L_e - \sum_{i=0}^{\infty} \mathbf{T}_\mu^i L_e && (8) \\ &= \sum_{i=0}^{\infty} \left[\mathbf{T}_\rho^i L_e - \mathbf{T}_\mu^i L_e \right] && (9) \\ &= \sum_{i=0}^{\infty} \mathbf{T}_\Delta^i L_e && (10) \end{aligned}$$

We want to model direct and first bounce indirect change in a radiance field. Two operators, $\mathbf{T}_\Delta$ and $\mathbf{T}_\Delta^2$ can be applied to incoming light to calculate a new radiance field, which can then be added onto L^μ for relighting purposes to approximate L^ρ.

$$L^\rho \approx L_e + \left(\mathbf{T}_\mu L_e + \mathbf{T}_\Delta L_e \right) + \left(\mathbf{T}_\mu^2 L_e + \mathbf{T}_\Delta^2 L_e \right) \tag{11}$$

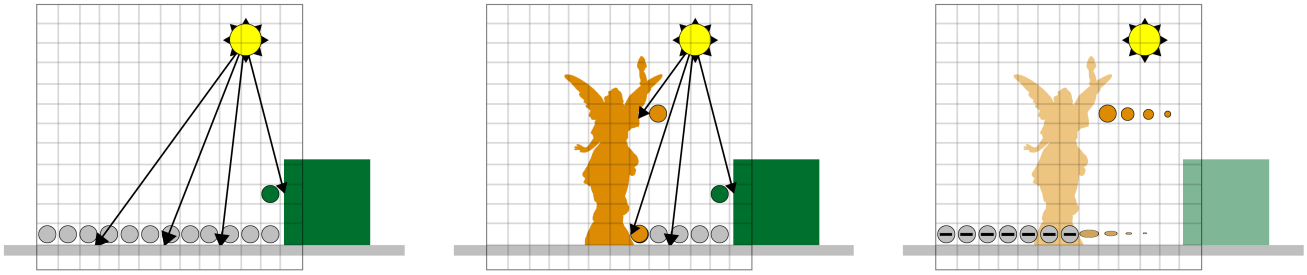
In Section 4.3, we will define both operators $\mathbf{T}_\Delta$ and $\mathbf{T}_\Delta^2$ for interactive rendering.

4.2 Model

Inspired by Radiance Transfer Fields [22] and Differential Rendering, we model a Delta Radiance field L^Δ by extending Light Propagation Volumes. This new representation will be used for mixed reality light transfer. A LPV $V_n(j)$ is a volume of n^3 voxels j which store approximate propagated diffuse indirect light. Similarly to Irradiance Volumes, a LPV covers part of a 3D scene and can be queried for each position in space to retrieve the directional indirect light contribution at this position.

The evaluation of an LPV is done entirely on the GPU: A RSM is computed for a light source, and VPLs generated from it are *injected* into their respective voxels inside the volume (if their position is covered by the volume). The directional distribution intensity of a VPL is saved by converting it into low-frequency, second order spherical harmonic (SH) coefficients. An SH value of band b has b^2 coefficients, therefore one can save the entire volume in three volumetric RGBA textures, one for each color channel.

After injecting all VPLs, the flux of each voxel is scattered to each neighbor voxel on each principal axis. The same process can be mapped to a GPU friendly gathering operation instead, where each voxel collects the contribution over a solid angle from each neighbor. By repeating this step multiple times for each voxel j in the LPV, light is *propagated* through the volume. The accumulated result of all iterations represents the light distribution in the space covered by the LPV, which can be queried with a tri-linear lookup.



(a) Injection in V^{μ} (b) Injection in V^{ρ} (c) The propagated difference V^{Δ}

Figure 3: Creating a DLPV from two RSMs: (a) a regular LPV V^{μ} without the virtual object after injection with R^{μ} and similarly (b) LPV V^{ρ} injected with R^{ρ}. (c) The propagated difference $R^{\rho} - R^{\mu}$ yields the DLPV. The residue indirect light from the virtual object remains as positive contribution, while the shadowed space behind it now has negative values.

Instead of the proposed cascade of LPVs centered around the camera in [13] however, we use one volume which is centered around the virtual object with dimensions usually twice the size of the largest bounding box edge, since we are only interested in modeling change in illumination in the vicinity of the virtual object.

4.3 Construction

We start by reconstructing one or more real direct light sources with position and orientation (e.g., tracked or extracted from a hemispherical image), as well as the real geometry of the scene with the help of a depth sensor or a pre-reconstructed model. We use the reconstructed light sources to furthermore evaluate diffuse surface parameters of the real scene. For each light source we render the virtual object and the reconstructed real geometry of the scene into an RSM R^{ρ}. The same process is repeated for a second RSM R^{μ} which contains only the reconstructed real geometry.

To illustrate our method (see Figure 3), we can now think of two independent LPVs V^{ρ} and V^{μ} created from both RSMs. Applying Equation 7, the difference between both volumes, the *Delta Light Propagation Volume* (DLPV) $V^{\Delta} = V^{\rho} - V^{\mu}$, contains only the change in illumination caused by the introduction of the virtual object O into the scene. We observe that this differential not only includes indirect change of illumination but also Antiradiance (negative voxel values) which implicitly creates shadowed areas when superimposed onto another radiance field.

A naive approach is to create both volumes on the GPU, query both for their radiance contribution for each point in space and add the difference to a background image or radiance field. Instead, we choose to create one volume which will *propagate change in illumination* directly. To this end, we alter the injection phase of a regular LPV and calculate the differential at the level of VPLs directly, avoiding the the memory and propagation cost for one of the volumes.

By injecting not only indirect, but also direct light, we model both operators $\mathbf{T}_{\Delta}$ and $\mathbf{T}^2_{\Delta}$ with one representation. The final value queried from the volume V^{Δ} is added onto the background image at its intersection with the real geometry, shown in Figure 4.

4.3.1 Indirect change $\mathbf{T}^2_{\Delta}$

The radiant intensity $I_p(\vec{\omega})$ of a VPL created from pixel p of an RSM with normal $\vec{n}_p$, reflected flux Φ_p and incident light direction $\vec{\omega}$ that is initially injected into a voxel of an LPV is

$$I_p(\vec{\omega}) = \frac{\Phi_p}{\pi} \langle \vec{n}_p, \vec{\omega} \rangle_+ \tag{12}$$

For a DLPV, we instead inject

Figure 4: Using a DLPV to display shadows cast by blocking direct light as well as indirect bounces. The above image exaggerates the indirect value for better illustration.

$$I^{\Delta}_p(\vec{\omega}) = I^{\rho}_p(\vec{\omega}) - I^{\mu}_p(\vec{\omega}) \tag{13}$$

The outcome of this operation is that VPLs visible in both RSMs are eliminated. New VPLs created from bouncing off the surface of the virtual object O are added, while VPLs now blocked by it appear as negative contribution. To avoid self-illumination by VPLs injected into voxels which also contain the surface they originate from, we shift the position of the VPL inside the volume by half the size of a voxel along its normal before determining the voxel into which the value will be injected, as proposed in [13]. The same procedure is safe-guarding against discretization errors, where VPLs appear behind the surface they are supposed to bounce off from.

4.3.2 Direct change $\mathbf{T}_{\Delta}$

After injection and propagation of indirect change, the volume contains indirect bounces of light with subtle shadows from blocked indirect sources. However, shadows are usually cast by blocking direct incident light, which is why we inject direct light into the volume as well.

For each reconstructed direct light source L_r we use the same RSMs R^{ρ} and R^{μ} to determine surfaces that are directly lit. Analogously to VPLs, for each pixel in the RSM R^{ρ} we reconstruct the surface position but instead use the negative incident direction $\vec{\omega}'$ and flux Φ_r of a reconstructed real light source L_r (from which the RSM was created) to compute the SH value. This value is injected just like a VPL created from that pixel, but with a safe-guard distance of half a cell in direction $\vec{\omega}'$ of L_r.

Figure 5: A textured bust (left) is inserted into the scene, receiving red indirect light from the ground and casting a slight shadow.

4.4 Reducing shadowing artifacts

Voxels $V^{\Delta}(j)$ which have a value $I_j^{\Delta}(\vec{\omega}) < 0$ in direction $\vec{\omega}$ after injection can cause severe shadow bleeding artifacts when propagated. Because the initial injection for indirect bounces is associated to the reconstructed material where the light hits the surface, propagating these values can affect the surrounding real geometry and subtract light incorrectly due to the low spatial approximation of DLPVs. For instance, in Figure 1 one can see thin purple line along the edge of the green briefcase, which is the result of negative green contribution onto the white desk.

Shadow bleeding artifacts cannot be avoided completely and are visible mostly along edges of real reconstructed geometry or bleeding out from below virtual objects. Since this error is related to the light bleeding artifact in [13], we use directional derivatives of the intensity distribution to dampen these values. The same dampening factor also helps to contain self-shadowing.

4.5 Merging DLPVs with the real scene

When compositing the real scene with the rendered virtual result, we use a G-Buffer G that contains the reconstructed albedo $G_a(p)$ of the real scene as well as the albedo of the virtual object, a mask $G_m(p)$ tagging real geometry as 1 and everything else as 0, and normals $G_n(p)$ for each pixel p. Additionally, a buffer $B(p)$ contains the real background image. For a real, reconstructed light source L_i^r, the virtual object is rendered the regular way by adding direct and indirect contribution of V^{ρ} to include bounces from virtual and real geometry on the objects surface (see Figure 5). For all other pixels p of the background image, we determine their position in space, query the DLPV and add its value onto it. The procedure is outlined in Algorithm 1.

for $p \in G$ **do**
  if $G_m(p) = 0$ **then**
    $\left[\left(\sum_i L_i^r \langle G_n(p), \vec{\omega}' \rangle_+\right) + V_n^{\rho}(p)\right] \cdot G_a(p)$;
  else
    $V_n^{\Delta}(p) \cdot G_a(p) + B(p)$;
  end
end

Algorithm 1: Merging a DLPV with the combined scene of virtual and real geometry for real light sources L_i^r.

5 Implementation and Results

We have implemented the system described in Section 4 using Direct3D 11. A Microsoft Kinect camera was used to capture the background image, and a UEye UI-2230-C camera in conjunction with a fish-eye lens to capture surrounding real light. A marker based tracking approach through ARToolKit [15] was used to geometrically register a virtual object. For our tests, when the camera was very close to the scene the Kinect depth buffer could not be used to reconstruct it. In these instances we therefore registered a manually reconstructed model of the real scene.

RSM size	V^{ρ}	V^{Δ} Delta	V^{Δ} Direct
256^2	0.091	0.184	0.18
512^2	0.462	0.862	0.895
1024^2	2.108	4.667	4.494
2048^2	10.74	21.35	20.215

Table 2: Timings taken for a full VPL injection (i.e. every pixel is used as a VPL) when varying the size of both RSM R^{μ} and R^{ρ}. The test scene contains the Stanford Lucy model and a planar ground. All values are in milliseconds.

V^{Δ} size \ Propagation steps	16	32	64	128	256
16^3	0.712	1.815	-	-	-
32^3	2.342	4.552	8.857	-	-
64^3	11.148	22.021	43.23	85.089	-
128^3	79.96	158.36	304.57	625.89	1250.5

Table 3: Timings taken for varying volume sizes of V^{Δ} with varying numbers of propagation steps and the cost in milliseconds.

We first render the reconstructed geometry with the virtual object into an RSM R^{ρ} of size 512^2 pixels and inject all VPLs generated from it into V_{32}^{Δ}. We repeat the process only for the real scene geometry and create an RSM R^{μ}, which we inject as negative intensity using subtractive hardware blending. We use 32 propagation steps to distribute the light inside the DLPV before accessing it in a deferred rendering step to assemble an image. A postprocessing pipeline can optionally add SSAO or bloom effects to bright spots and will tonemap HDR values and gamma correct the output. An example can be seen in Figure 1.

We have gathered timings for the entire pipeline on a test system with an Intel i7 X980 and a NVIDIA GTX 470 in Table 1. After injection, the required propagation time does not differ from a regular LPV. For each reconstructed light source two RSMs need to be calculated. While LPV and DLPV propagation stay approximately the same for each model, on average the delta injection consumes twice the time of a normal injection.

Increasing the size of both RSMs can lead to better sampling of VPLs. The time consumed for injecting every pixel of a RSM into a regular volume as well as a DLPV is listed in Table 2. Up to 1024^2 sampled VPLs amount to a reasonable investment for real-time purposes. Since the injection step essentially clusters VPL contribution into a fixed amount of voxels, the following propagation remains unaffected.

Better spatial sampling of the radiance field can be achieved by increasing the resolution of the DLPV. The impact on performance, listed in Table 3, furthermore depends on the number of propagation steps used to distribute light inside the volume: a higher resolution DLPV implicitly requires more propagation steps for the same distribution coverage. At a size of 64^3 and 32 steps the cost of the propagation dominates the entire rendering pipeline.

We now compare our method to the multi-resolution splatting method by Lensing and Broll [18]. Naive sampling of VPLs in Instant Radiosity can force high VPL counts to avoid artifacts, which

Rendering step	Infinite Head	Cornell Box	Stanford Lucy	Stanford Armadillo
R^{μ}	0.123	0.0629	0.114	0.0675
R^{ρ}	0.152	0.0626	1.174	0.816
V^{ρ} Inject	0.367	0.3674	0.4728	0.368
V^{Δ} Delta Inject	0.734	0.7345	0.7808	0.734
V^{Δ} Direct Inject	0.73	0.73	0.837	0.73
V^{ρ} Propagation	3.619	3.664	4.055	3.652
V^{Δ} Propagation	3.833	3.777	4.437	3.913
Phantom scene	0.3	0.26	0.31	0.32
Virtual object	0.1	0.0198	1.14	0.894
Deferred	1.539	1.526	1.972	1.551
$\sum$	11.497	11.2042	15.2926	13.0455
Generating V^{Δ}	5.572	5.367	7.3428	6.2605

Table 1: Detailed timings per frame in milliseconds taken for each step of the pipeline, with 512^2 VPL injections at 32 propagations. The highlighted rows display the effective time to generate the DLPV V^{Δ}. In the last row the sum of these operations add up to roughly 6 ms on average.

	$\mathbf{d}_{normal}$		
VPLs	10°	5°	1°
1024	5.81	6.32	6.45
4096	9.8	11.9	11.9
16384	11.23	13.5	13.69

Table 4: Multi-resolution splatting method timings taken with a fixed d_{depth} of 1 cm for varying VPL counts and d_{normal} degrees in milliseconds.

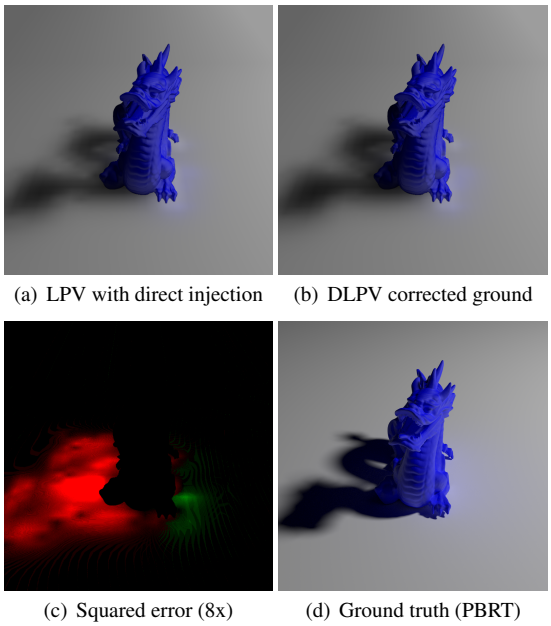
in turn have a high impact on rendering speed. The multi-resolution splatting method, based on a frequency analysis of the current view space, effectively importance samples regions of higher interest and therefore reduces the amount of VPLs needed to render an artifcat-free image.

Timings for the multi-resolution splatting rendering pipeline, displayed in Table 4, were measured on a NVIDIA GTX 285, a comparable match to the NVIDIA GTX 470 used to measure the DLPV time.

In Figure 6 we set up a small test scene with a beach ball. The beach ball features differently colored slices and can lead to temporal inconsistencies when animated or when the viewer camera is moving. In this case the VPL distribution has to be recalculated and can lead to slight flickering. Lensing and Broll report that at a VPL count of 4000 or higher is needed for the beach ball scene to suppress flickering in the animation. More complex scenes require higher VPL counts. In contrast, by clustering 512^2 VPLs into a volume in $\sim$ 2.3 out of 9.6 ms to render the image, a DLPV with 32 propagations handles a VPL count two orders of magnitude higher at the same rendering speed. DLPVs are therefore not constrained by the number of injected VPLs. One should note however that by using this clustering method, many VPLs can get blurred together and may loose details still visible in the multi-resolution splatting method. Another difference between both methods is that the propagation distance in DLPVs is limited to the number of steps and the size of the volume, whereas regular VPL accumulation in Instant Radiosity is not limited by distance factors other than the physical falloff.

6 Discussion

When rendering V^{ρ} to add indirect bounces from real geometries to the virtual object, one could argue to simply evaluate the difference of V^{ρ} and V^{μ} in a shader instead of calculating a DLPV. DLPVs however contain less self-illumination errors than a differential of



(a) LPV with direct injection (b) DLPV corrected ground

(c) Squared error (8x) (d) Ground truth (PBRT)

Figure 7: An error estimation for DLPV rendering. (a) 64^3 voxel LPV rendering with additional direct light injected and propagated for five steps, (b) an LPV of the ground corrected with a DLPV, (c) the squared error enhanced by factor eight between image *a* and *b* (negative errors in red, positive in green, brighter pixels equal higher error), (d) ground truth path traced with PBRT.

two LPVs, since VPLs which do not contribute to the change of illumination are eliminated during injection. Decoupling both volumes has the additional benefit that they can be rendered at different resolutions without introducing artifacts, and antiradiance and radiance propagation in the DLPV can be controlled independently.

DLPVs can be used in conjunction with other shading methods used to calculate the surface transfer on the virtual object O. For instance Precomputed Radiance Transfer [25] can be used to relight rigid objects on the fly. In this case the delta injection saves the bandwidth and propagation cost of an extra LPV V^{ρ}.

In Figure 7(a) a virtual test scene has been rendered with an LPV which also contains direct light propagated for five further steps. In Figure 7(b) the ground plane has been rendered into an LPV

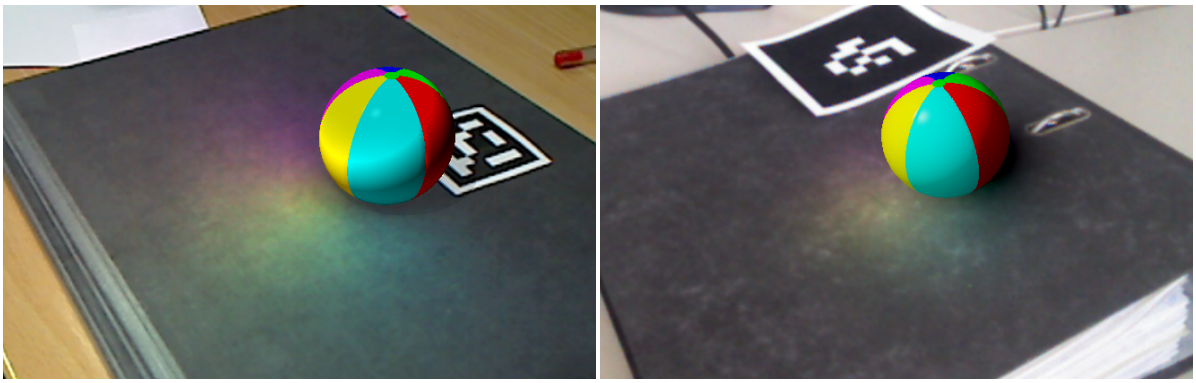


Figure 6: Visual comparison between multi-resolution splatting (left) and DLPV rendering (right): the multi-resolution splatting image was rendered with 4096 VPLs, a d_{normal} of 10° and d_{depth} of 1 cm in 9.8 ms. The DLPV image was produced with 32 propagations and 512^2 in 9.6 ms.

and has been superimposed with a DLPV created for the dragon. The enhanced squared difference 7(c) between both shows the error introduced by using DLPVs: apart from errors which are caused by aliasing (i.e. propagation of values into wrong voxels because of low DLPV resolution), self-illumination issues from neighbor voxels bleeding into shadowed areas or indirectly lit areas cause slightly more energy to be present in the LPV rendering. These neighboring values are eliminated during the injection phase of the DLPV and are therefore not corrected.

7 Limitations

The construction of DLPVs comes with several limitations. As noted in [13], light bleeding and self-illumination are the most apparent issues. In Section 4.4 we have already discussed shadow bleeding, which is the same effect caused by Antiradiance in a coarse volume.

Because of low spatial resolution shadowed areas display heavy aliasing. Increasing the volume size leverages the problem, but at prohibitively increased rendering cost. Cubical filtering can be applied to blur out jagged shadow edges. Another solution is to allow further light propagation of n iterations after injecting direct light into the DLPV which, due to the low frequency nature of the SH values, will smooth out shadow borders. This might in turn however oversaturate the center of a shadow.

An alternative to insert and render shadows is to use regular shadow mapping with R^ρ instead of Antiradiance. However, because the approach with Antiradiance is a volumetric correction term instead of a projection it is important for adjusting scattering effects in the real radiance field. While limited and coarse at the moment, better discretization of the delta volume will eventually allow higher resolution shadows [1].

The size of the volume may not be sufficient to cover the area of light influence caused by the introduction of the object. This will usually be noticeable when shadows or bright indirect spots get cut-off. Resizing the volume can only leverage the problem to a certain extent, as the sampling rate of the volume with respect to its covered space is affected. Higher resolution volumes with larger coverage have higher propagation cost. An alternative method to light propagation could decrease this cost.

DLPVs only support diffuse indirect light transfer. Better basis representation and construction of the volume may combat both problems [1].

8 Conclusion and Future Work

We presented Delta Light Propagation Volumes, a new representation for light transfer from virtual to real objects. DLPVs can overcome issues of temporal coherence and speed limitations when using many VPLs to simulate indirect light influence of a virtual object on a real scene. DLPVs also provide low frequency shadows and can handle virtual and real light in one unified representation.

We want to further explore GPU based voxelization methods to overcome issues resulting from the limited resolution in DLPVs. Sparse voxel octrees [1] may also allow us to display indirect specular reflections. Better discretization of a DLPV will facilitate rendering of higher sampled shadows, combat self-illumination issues and may furthermore enable integration of fine-grained occlusion information in the form of geometry volumes while building on the same theoretical framework.

Acknowledgements

The author would like to thank Philipp Lensing for his help with evaluation data and images, Sebastian Wagner for creating a cup model and Arjan Kuijper for his helpful comments. The head model is courtesy of Inifinite-Realites, the dragon model is courtesy of the Stanford 3D Scanning Repository and the beach ball model is courtesy of TurboSquid.

References

[1] C. Crassin, F. Neyret, M. Sainz, S. Green, and E. Eisemann. Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum (Proceedings of Pacific Graphics 2011)*, 30(7), sep 2011.

[2] C. Dachsbacher and M. Stamminger. Reflective shadow maps. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, I3D '05, pages 203–231, New York, NY, USA, 2005. ACM.

[3] C. Dachsbacher and M. Stamminger. Splatting indirect illumination. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, I3D '06, pages 93–100, New York, NY, USA, 2006. ACM.

[4] C. Dachsbacher, M. Stamminger, G. Drettakis, and F. Durand. Implicit visibility and antiradiance for interactive global illumination. *ACM Trans. Graph.*, 26(3), July 2007.

[5] P. Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 189–198, New York, NY, USA, 1998. ACM.

[6] A. Fournier, A. S. Gunawan, and C. Romanzin. Common illumination between real and computer generated scenes. Technical report, Vancouver, BC, Canada, Canada, 1992.

[7] T. Franke, S. Kahn, M. Olbrich, and Y. Jung. Enhancing realism of mixed reality applications through real-time depth-imaging devices in x3d. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 71–79, New York, NY, USA, 2011. ACM.

[8] S. Gibson and A. Murta. Interactive rendering with real-world illumination. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 365–376, London, UK, UK, 2000. Springer-Verlag.

[9] T. Grosch. Differential photon mapping: Consistent augmentation of photographs with correction of all light paths. In M. Alexa and J. Marks, editors, *Eurographics 2005, EG Short Pres.*, pages 53–56, Trinity College, Dublin, Ireland, 2005. Eurographics Association.

[10] T. Grosch, T. Eble, and S. Mueller. Consistent interactive augmentation of live camera images with correct near-field illumination. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, VRST '07, pages 125–132, New York, NY, USA, 2007. ACM.

[11] K. He, J. Sun, and X. Tang. Guided image filtering. In *Proceedings of the 11th European conference on Computer vision: Part I*, ECCV'10, pages 1–14, Berlin, Heidelberg, 2010. Springer-Verlag.

[12] T. Kakuta, T. Oishi, and K. Ikeuchi. Virtual kawaradera: Fast shadow texture for augmented reality. In *CREST05*, pages 79–85, 2005.

[13] A. Kaplanyan and C. Dachsbacher. Cascaded light propagation volumes for real-time indirect illumination. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, I3D '10, pages 99–107, New York, NY, USA, 2010. ACM.

[14] K. Karsch, V. Hedau, D. Forsyth, and D. Hoiem. Rendering synthetic objects into legacy photographs. *ACM Trans. Graph.*, 30(6), Dec. 2011.

[15] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, IWAR '99, pages 85–94, Washington, DC, USA, 1999. IEEE Computer Society.

[16] A. Keller. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[17] M. Knecht, C. Traxler, O. Mattausch, W. Purgathofer, and M. Wimmer. Differential Instant Radiosity for Mixed Reality. In *Proceedings of the 2010 IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2010)*, pages 99–107, Oct. 2010.

[18] P. Lensing and W. Broll. Instant indirect illumination for dynamic mixed reality scenes. *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 0:109–118, 2012.

[19] G. Nichols and C. Wyman. Multiresolution splatting for indirect illumination. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 83–90, New York, NY, USA, 2009. ACM.

[20] D. Nowrouzezahrai, S. Geiger, K. Mitchell, R. Sumner, W. Jarosz, and M. Gross. Light factorization for mixed-frequency shadows in augmented reality. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 173–179, Washington, DC, USA, 2011. IEEE Computer Society.

[21] O. Olsson and U. Assarsson. Tiled shading. *Journal of Graphics, GPU, and Game Tools*, 15(4):235–251, 2011.

[22] M. Pan, R. Wang, X. Liu, Q. Peng, and H. Bao. Precomputed radiance transfer field for rendering interreflections in dynamic scenes. *Comput. Graph. Forum*, 26(3):485–493, 2007.

[23] R. Prutkin, A. Kaplanyan, and C. Dachsbacher. Reflective shadow map clustering for real-time global illumination. In *Eurographics (Short Papers)*, pages 9–12, 2012.

[24] T. Ritschel, T. Grosch, and H.-P. Seidel. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 75–82, New York, NY, USA, 2009. ACM.

[25] P.-P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.*, 21(3):527–536, July 2002.

[26] K. Viriyothai and P. Debevec. Variance minimization light probe sampling. In *SIGGRAPH '09: Posters*, SIGGRAPH '09, pages 92:1–92:1, New York, NY, USA, 2009. ACM.

[27] K. Zhou, Y. Hu, S. Lin, B. Guo, and H.-Y. Shum. Precomputed shadow fields for dynamic scenes. *ACM Trans. Graph.*, 24(3):1196–1201, July 2005.