Differential Instant Radiosity for Mixed Reality

Martin Knecht* Christoph Traxler† Oliver Mattausch‡ Werner Purgathofer§ Michael Wimmer¶

Institute of Computer Graphics and Algorithms
Vienna University of Technology

Abstract

In this paper we present a novel plausible realistic rendering method for mixed reality systems, which is useful for many real life application scenarios, like architecture, product visualization or edutainment. To allow virtual objects to seamlessly blend into the real environment, the real lighting conditions and the mutual illumination effects between real and virtual objects must be considered, while maintaining interactive frame rates (20-30fps). The most important such effects are indirect illumination and shadows cast between real and virtual objects.

Our approach combines Instant Radiosity and Differential Rendering. In contrast to some previous solutions, we only need to render the scene once in order to find the mutual effects of virtual and real scenes. The dynamic real illumination is derived from the image stream of a fish-eye lens camera. We describe a new method to assign virtual point lights to multiple primary light sources, which can be real or virtual. We use imperfect shadow maps for calculating illumination from virtual point lights and have significantly improved their accuracy by taking the surface normal of a shadow caster into account. Temporal coherence is exploited to reduce flickering artifacts. Our results show that the presented method highly improves the illusion in mixed reality applications and significantly diminishes the artificial look of virtual objects superimposed onto real scenes.

Keywords: Mixed Reality, Real-time Global Illumination, Differential Rendering, Instant Radiosity

Index Terms: I.1.3 [COMPUTER GRAPHICS]: Three-Dimensional Graphics and Realism—Radiosity; H.5.1 [INFORMATION INTERFACES AND PRESENTATION]: Multimedia Information Systems—Artificial, augmented, and virtual realities

1 Introduction

Mixed reality is an attractive and exciting way to present virtual content in a real context for various application domains, like architectural visualizations, virtual prototyping, marketing and sales of not yet existing products and edutainment systems. These kinds of application scenarios demand a believable realistic appearance of virtual objects, providing a perfect illusion for human visual perception. Unfortunately this requirement is not met in common mixed reality systems, where the composed images look disturbingly artificial. One major reason for this is that real illumination conditions and the mutual shading effects between virtual and real objects are completely ignored.

In this paper we present a new global illumination (GI) rendering system that is designed to calculate the mutual influence between real and virtual objects. The aim of the GI solution is to be perceptually plausible without the ambition to be physically accurate. Jacobs and Loscos [14] give a very comprehensive overview and classification of different illumination methods for mixed reality. In their classification our approach would be placed in the "Common Illumination and Relighting" category. Besides calculating the influence between real and virtual objects, we are also able to relight the scene by virtual light sources.

Figure 1: This figure shows a mixed reality scenario where a real spotlight illuminates the Stanford dragon, causing green color bleeding on the real teapot. Rendered at 22 fps.

Our method is based on Debevec's [5] extension of Differential Rendering, which was originally introduced by Fournier et al. [8]. Furthermore it is based on the Instant Radiosity approach [16] combined with imperfect shadow maps [27].

There are some previous methods [9, 10] which are able to take indirect illumination into account. However they need a computationally expensive preprocessing step or are not feasible for real-time applications. An important aspect of our work is an extension of Instant Radiosity to handle real-world objects. Instant Radiosity has the advantage that it does not need any pre-computation and therefore can be easily used for dynamic scenes where object positions change or the illumination is varied through user interaction.

To capture the surrounding environment, we use a fish-eye camera and for now our system uses a pre-modeled representation of the real scene. Figure 1 gives an impression on how our method handles indirect illumination in mixed reality scenarios.

The main contributions presented in this paper are:

- A modified Instant Radiosity approach that is usable for Differential Rendering, while shading is performed only once.
- New higher quality imperfect shadow maps by aligning the splats to the surface normal.
- A novel method to assign virtual point lights to multiple primary light sources.

*e-mail: knecht@cg.tuwien.ac.at
†e-mail: traxler@cg.tuwien.ac.at
‡e-mail: matt@cg.tuwien.ac.at
§e-mail: wp@cg.tuwien.ac.at
¶e-mail: wimmer@cg.tuwien.ac.at

- Reduced temporal flickering artifacts by exploiting temporal coherence.

2 Related Work

Our approach is based on several areas of computer graphics: Image based lighting, real-time global illumination and the compositing of real and rendered image data.

Image Based Lighting Most approaches that deal with illumination in mixed reality applications use an environment map to simulate the incident illumination. There are basically two types of methods to acquire the environment map: outside-in and inside-out methods. Outside-in methods use a camera to take photos or a video stream of a chrome sphere. This chrome sphere reflects the surrounding scene and can be used as an environment map. The inside-out methods use a camera to capture the illumination at the center of the scene.

Debevec [5] as well as Agusanto et al. [1] captured a high dynamic range (HDR) environment map from a chrome sphere. Heymann et al. [13] use a real-time outside-in approach. They place a chrome sphere in the scene and use a marker to find it in the captured video stream. However, all methods need a preprocessing step before the environment map is usable for rendering, either by taking photographs [5], [1] or by extracting the environment map out of the video stream [13].

Ritschel and Grosch [26] used a HDR video camera to capture the surrounding illumination using the standard inversion technique [11]. Sato et al. [29] introduced a stereo vision inside-out approach to calculate the environmental radiance distribution. Furthermore they used the stereo image pair to get a reconstruction of the surrounding scene. While the reconstruction was performed with user interaction, Korn et al. [17] introduced a method which was able to do that at interactive frame rates.

Once the environment is captured, a fast method is needed to extract light sources from the environment map. Dachuri et al. [4] propose a method to efficiently find positions of light sources in an environment map by segmenting it and finding the bright spots. Havran et al. [11], Debevec [6] and Clarberg et al. [2] take the luminance of the environment map as a probability function and distribute samples on it. Havran et al. [11] use their own inverse transform method, while Debevec uses a median cut method to find positions for light sources. To our knowledge [4], [6] and [11] use the CPU to generate the samples. We use hierarchical warping from Clarberg et al. [2], since the algorithm works with mipmap levels of the luminance probability map and thus allows us to perform importance sampling directly on the GPU.

A completely different solution was developed by Madsen and Nielson [19]. They detect shadows in an image and use the date of capture and position of the camera to determine the ratio between direct light from the sun and the sky. This way they are able to simulate the actual lighting conditions without requiring a light probe.

Real-time Global Illumination Algorithms Real-time global illumination (RTGI) is a very active area of research. This section will give a brief overview on current developments.

Instant Radiosity was introduced by Keller [16] in 1997. The idea is to place so-called virtual point lights (VPLs) in the scene to approximate global illumination. This method is particularly suitable for RTGI on current graphics hardware, as it does not need any complex pre-computations of the scene. Dachsbacher and Stamminger [3] extended standard shadow maps to so-called reflective shadow maps, where every pixel was treated as a light source. By adaptively sampling the shadow map to create VPLs, they were able to calculate indirect illumination. However, the indirect illumination computation did not contain any visibility calculation since generating shadow maps for each VPL is too costly. Laine et al. [18] developed a real-time Instant Radiosity method that caches the shadow map for each VPL over several frames. This way only a few shadow maps need to be recreated every frame, thus achieving real-time frame rates. However, moving objects cannot influence indirect visibility calculation. In 2008, Ritschel et al. [27] introduced the concept of imperfect shadow maps (ISMs). The idea is to represent the scene as a large point cloud and use this point cloud to generate a shadow map for every VPL. Using this approach it is possible to create hundreds of shadow maps per frame, allowing for completely dynamic scenes.

A recent method introduced by Ritschel [25] uses CUDA to perform fast final gathering on a hierarchical scene representation of splats. The images look very impressive but currently the frame rates are too low for mixed reality applications. Kaplanyan [15] uses a light propagating volume where radiance is propagated from source voxels based on spherical harmonics coefficients. The method does not include visibility but delivers a good GI approximation at very low costs. McGuire and Luebke [20] extend photon mapping in a way that photon volumes are splat in screen-space to compute indirect illumination. Wang et al. [31] are able to simulate several illumination effects by clustering the scene into coherent shading clusters on the GPU. Final gathering is then performed only on those clusters that result in a significant speedup. However, performance is still too low for mixed reality applications. Nichols et al. [22] perform screen-space hierarchical radiosity using the multi-resolution splatting technique from Nichols et al. [23], which greatly reduces shading costs. Ritschel et al. [28] introduce a method similar to screen-space ambient occlusion called screen-space directional occlusion (SSDO). It samples the neighboring screen-space pixels and uses these to calculate indirect illumination.

Merging Real and Virtual Scenes Nakamae et al. [21] were the first to concentrate on merging real and virtual scenes. They had a simple geometry for the real scene and information about the date and time when the background picture was taken. From this information, they could place the sun to calculate shadows cast from virtual objects. Fournier et al. [8] used a progressive radiosity algorithm to compute global illumination. Depending on whether the object of a patch belongs to a virtual or real object they changed the calculation behavior. For virtual objects, the full global illumination solution is used. For real objects only the additional radiosity caused by virtual objects or light sources is added. Drettakis et al. [7] extended this method to dynamic virtual objects. They were able to render new images every 2.8 seconds. Debevec [5] introduced Differential Rendering, which is based on the work of Fournier et al. [8]. Differential Rendering greatly reduces the error that is introduced by BRDF estimation, at the cost of having to compute the global illumination solution twice.

Grosch [9] used a variation of photon mapping in combination with Differential Rendering to merge virtual and real scenes. However, the proposed method does not achieve real-time frame rates. Pessoa et al. [24] use an environment map for each object in an augmented reality scene to simulate mutual light interaction. While performance scaling is an issue when more objects are placed in the scene, they get very impressive results for simple scenes and are able to simulate many illumination effects.

3 Differential Instant Radiosity

3.1 Differential Rendering

Mixing real with virtual objects requires the calculation of a global illumination solution that takes both virtual and real objects into account. In our approach, we assume that a geometric model of the real scene is available, and correctly registered to the virtual scene. We also assume that basic material properties of real objects are known. However, BRDF estimations for the real scene are usually not accurate, which leads to strong artifacts in the display of real objects. The idea of *Differential Rendering* [8, 5] is to minimize

this error by calculating only the "differential" effect of virtual objects, leaving the effect of the original BRDFs in the scene intact. This requires calculating two global illumination solutions: one using both virtual and real objects (L_{rv}), and one using only the real objects (L_r). The final image is created by adding the difference ΔL between these two solutions to the captured camera image.

One might be inclined to calculate the difference between L_{rv} and L_r directly and add it to the captured real image. However, since the captured image has only low dynamic range, we first need to apply tone mapping (performed by function TM), which requires a complete image as input. Therefore two complete images, including both direct and indirect illumination, need to be tone mapped before calculating the difference:

$$\Delta L = TM(L_{vr}) - TM(L_r)$$

We first focus on one-bounce global illumination, which in terms of Heckbert's [12] classification of light paths corresponds to $LDDE$-paths, where L is the light source, D is a (potentially) glossy reflection at an object in the scene, and E is the eye.

For a mixed reality setting, the idea is that the illumination of the real scene L_r is calculated by only considering the (outgoing) contribution L_o of those paths where all elements are real, i.e., $L_rD_rD_rE$ paths. The contributions of all other paths, including those with a virtual light source (L_vDDE), a bounce on a virtual object (LD_vDE) or a virtual object to be shaded (LDD_vE), only count for the combined solution L_{rv}.

3.2 "Single-Pass" Differential Rendering

The solutions L_r and L_{rv} could be calculated separately using $L_rD_rD_rE$ and general $LDDE$ paths respectively. However, depending on how paths are generated, this might be very inefficient, and might also lead to artifacts if different paths are chosen for the two solutions. Instead, we assume that the paths are already given, and for each path, we decide whether it should count for L_r or L_{rv} or both.

Furthermore, depending on which paths the used global illumination algorithm considers, we need to take visibility into account: some $L_rD_rD_rE$ paths that are valid for L_r might be blocked by virtual objects, so that they should not be added to L_{rv}. This can happen for the segments L_rD_r or D_rD_r. In our approach, we can handle the cases when a D_rD_r segment in an $L_rD_rD_rE$ path, or an L_rD_r segment in an L_rD_rE path, is blocked by virtual blockers b_v (only). The latter corresponds to shadows cast by a virtual object. Figure 2 illustrates the different cases.

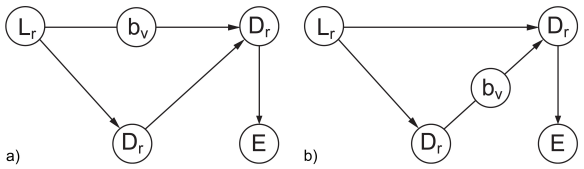


Figure 2: a) Illustrates a virtual blocker in the L_rD_r segment. The virtual blocker casts a shadow from direct illumination. b) Illustrates a path that has a virtual blocker between the D_rD_r segment. Here the shadow results from one-bounce indirect illumination.

3.3 Instant Radiosity

In order to achieve real-time performance, we use Instant Radiosity to approximate global illumination. Instant Radiosity uses virtual point lights (VPLs) to propagate light from the light sources. VPLs are created as endpoints of LD (for the first bounce) or DD (for subsequent bounces) path segments. The VPLs are used to shade every visible pixel on screen using a DD path, so the paths creating them are reused multiple times.

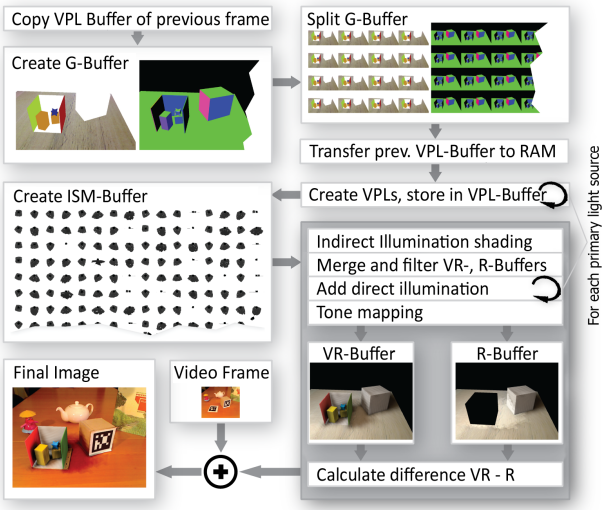


Figure 3: This figure shows the algorithm outline. A G-Buffer is created and afterwards split to reduce shading costs. The primary light sources generate VPLs from which ISMs are created. The VR- and R-Buffers are created in parallel, while the shading computation is performed only once. After some finishing steps like merging, adding direct illumination and tone mapping, the video frame is added to the difference between the VR- and R-Buffers.

Figure 3 shows the main steps of an instant radiosity rendering system. The first step is to render the scene from the camera and store all necessary information like position and normals in the so-called geometry buffer (G-buffer). The G-Buffer is then split to save shading costs (see Segovia et al. [30] for more details). The next step is to create VPLs and store them in a so-called VPL-Buffer. Depending on the primary light source type, the VPLs are created in different ways (see Section 5.2). For example, the spotlight uses reflective shadow maps (RSM) [3]. An RSM is a shadow map that also stores the light intensity propagated to the hit surface. From this RSM, a number of virtual point lights (VPL) are sampled using importance sampling.

In order to be able to calculate visibility, for each VPL an imperfect shadow map is created, which is a low-resolution shadow map created from a point-based representation of the scene. The last steps are to perform tone mapping and add the result to the video frame using differential rendering.

3.4 Differential Instant Radiosity

In our differential rendering method, we modify instant radiosity so that we can keep track where real and virtual objects are involved. We create reflective shadow maps (RSM) considering the combined real and virtual scene, but note the type of object for each pixel of the RSM.

Afterwards lighting is calculated from each VPL and the primary light source. The light contribution from a VPL corresponds to a DD path segment, whereas the light contribution from the primary light source corresponds to an LD path segment in an LDE path.

In Differential Instant Radiosity, visibility is handled in the following way: for the LD-segment in $LDDE$-paths, there can be no blockers because virtual point lights are sampled directly from the RSM, which already takes visibility into account. For the LD-segment in LDE-paths, visibility can be determined using a shadow lookup into the RSM. The visibility for DD-segments in $LDDE$-paths is calculated using a shadow map created for each individual virtual point light.

Suppose we have a real spot light that illuminates a virtual green cube placed on a real desk as shown in Figure 4.

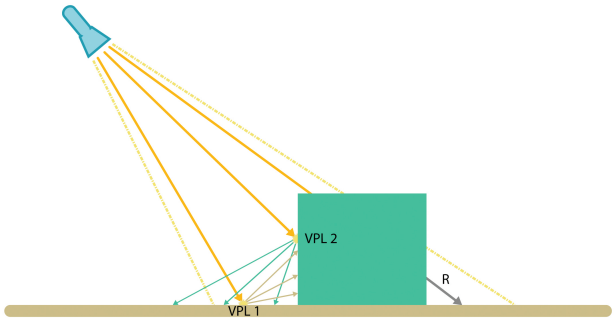


Figure 4: Illustration of an augmented scene with a real desk, a virtual green cube and a real spot light illuminating the scene. VPL 1 will cause color bleeding from the desk onto the green cube - VPL 2 from the green cube onto the desk. Ray R illustrates the shadowing caused by the virtual green cube.

Here the inserted virtual green cube causes three types of light interaction. First, light that hits the real desk causes color bleeding on the virtual green cube (VPL 1, $L_rD_rD_vE$). Second, light that hits the virtual green cube causes color bleeding on the real desk (VPL 2, $L_rD_vD_rE$). Third, the virtual green cube casts a shadow on the desk illustrated by ray R ($L_rb_vD_rE$).

Suppose we shade a point on the virtual green cube, illuminated by VPL 1. We have to decide if the path contribution L_o should be added to L_{rv} and/or to L_r. The spotlight is real, VPL 1 is placed on the real desk and the shaded point corresponds to a virtual object ($L_rD_rD_vE$). In this case the result gets added to L_{rv} but not to L_r as there is a virtual object in the light path. When shading a point on the real desk, which gets illuminated by VPL 2, the result must be again added to L_{rv} but not to L_r. This time, because VPL 2 is placed on a virtual object ($L_rD_vD_rE$).

On the other hand when a point on the real desk is shadowed by the virtual green cube ($L_rb_vD_rE$), the outgoing radiance must be added to L_r but not to L_{rv}.

3.5 Single Bounce Computation

Let $r(x)$ be a function that returns 1 if element x is associated to a real object and 0 if not, where x can be one of the following: A primary light source, a VPL, a surface point, blocking geometry or a light path (see Section 3.8). During creation of the RSM and for each shadow map of a VPL we store a flag $r(x)$ denoting whether the texel corresponds to a real object. We compute $r(x)$ for other elements of the renderer in the following way:

- For a primary light source pl, $r(pl)$ is stored with the light source.
- For a VPL, $r(\text{VPL})$ is taken from the RSM during sampling and stored with the VPL.
- For the point to be shaded p, $r(p)$ is taken from the object definition.
- For blockers b, $r(b)$ is taken from the RSM or the VPL shadow map depending on which path b blocks.

We want to calculate L_r and L_{rv} for $LDDE$ paths, i.e, for a point p illuminated by the ith VPL. For resolving visibility, we define the shadow $st(\text{VPL}, p)$ as a shadow map lookup into the shadow map associated with the VPL, while L_o is the unobstructed radiance, i.e., without taking visibility for the VPL into account. Then, for the ith VPL,

$$L^i_{rv} = L^i_o st(\text{VPL}^i, p) \quad (1)$$

$$L^i_r = L^i_o r(pl) r(\text{VPL}^i) r(p) rB(b) \quad (2)$$

$$rB(b) = \begin{cases} st(\text{VPL}^i, p) & \text{if } r(b) \text{ is 1} \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

The function $rB(b)$ ensures that only real blockers are taken into account when calculating the real scene. The GI solutions for $LDDE$ paths are simply summed up as follows:

$$L_{rv} = \sum_{i=0}^{N} L^i_{rv} \quad (4)$$

$$L_r = \sum_{i=0}^{N} L^i_r \quad (5)$$

where N is the total number of VPLs.

3.6 Direct and Environment Lights

The direct lighting contribution (LDE paths) is basically calculated in the same way as single bounce illumination, with the following differences: $r(\text{VPL}^i) = 1$, as no VPL is involved, and the shadow term is calculated from the RSM as $st(RSM, p)$.

We can also handle environment lights: these can be thought of as virtual point lights generated by a "virtual" primary light source that only sees an environment map, which in our framework directly comes from a camera with a fish-eye lens attached. The environment acts as RSM with no depth information, since the environment is infinitely far away. Sampling VPLs from the environment is done exactly the same way as sampling the RSM.

3.7 Limitations

There are three cases which cannot be handled by this approach, which are shown in Figure 5. Case a) and b) suffer from the fact that only the front-most objects are stored either in the RSM or VPL shadow map. The real blocker b_r behind the virtual one b_v is not considered in the algorithm and thus wrong double shadowing occurs. This is because the contribution of the path counts for L_r but not for L_{rv}, and thus is subtracted from the final image. On the other hand, in case c), if there is a virtual blocker between the segment L_rD_r, the subsequent segment D_rD_r cannot be canceled out, since no VPL will be created for the first segment. This leads to inconsistent color bleeding.

Figure 6 graphically illustrates these problems. The spot light is real, the green box is virtual and the blue one is real, casting a real shadow onto the desk. The blue box also causes indirect illumination onto the desk. Since we use reflective shadow maps to create new VPLs no VPLs will be created on the blue box as they are not visible in the reflective shadow map. Double shadowing happens because the desk is already shadowed and the virtual green box should not cast another shadow from the same light source. However, in practical situations these artifacts are hardly noticeable and we therefore will address them in our future work.

3.8 Multiple Bounces

Extending Differential Instant Radiosity to multiple bounces is straightforward. Imagine that light bounces off several times from real surfaces coming from a real primary light source illuminating a real object. In this case nothing has to be changed. However, if there is only one bounce on a virtual object, the final illumination must be changed. In Instant Radiosity, each bounce is simulated by one VPL. This way we are able to forward the information from one VPL to the next.

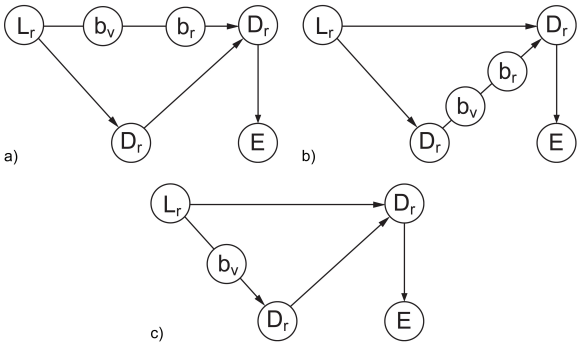


Figure 5: Illustration a) and b) are both responsible for wrong double shadowing since the real occluder b_r is not considered in the computation. In c) our system is not able to cancel out the segment D_rD_r which leads to inconsistent color bleeding.

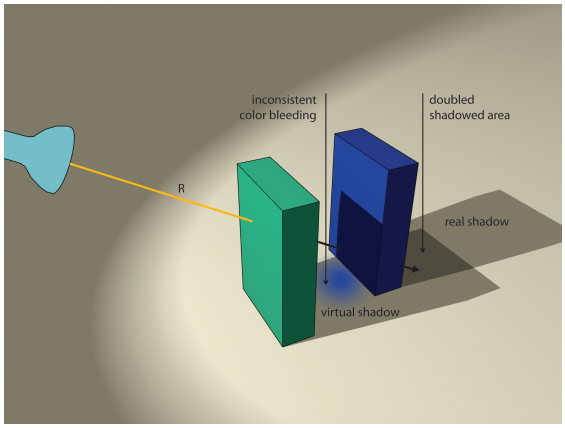


Figure 6: The spotlight and the blue box are real objects and therefore a real shadow is cast onto the desk. Furthermore indirect illumination causes blue color bleeding. However, in the reflective shadow map of the light source we only have information about the front most object – the green virtual cube, which results in wrong indirect illumination and double shadowing.

The flag about the primary light source pl will be substituted by a flag that indicates if the previous path $\bar{x}$ included any virtual objects or a virtual primary light source. When a new VPL is created from a previous one the flag is calculated as follows:

$$\bar{x} = r(\bar{x}_{prev})r(p_{\text{VPL}}) \quad (6)$$

$$\bar{x_0} = r(pl) \quad (7)$$

where $\bar{x}_{prev}$ indicates if the path to the preceding VPL only belonged to real objects. Once a virtual object was in the light path, the path flag will always be 0. The new illumination equation just uses $\bar{x}$ instead of pl. Figure 7 shows the difference between a single bounce and multiple bounces. Note that the back plane gets brighter due to multiple light bounces.

4 Improving Instant Radiosity

The Instant Radiosity approach we use is based on imperfect shadow maps (ISM) introduced by Ritschel et al. [27]. The idea of imperfect shadow maps is to use very low resolution shadow maps for each virtual point light, and use a sub-sampled version of the scene, represented as a point cloud, to create these shadow

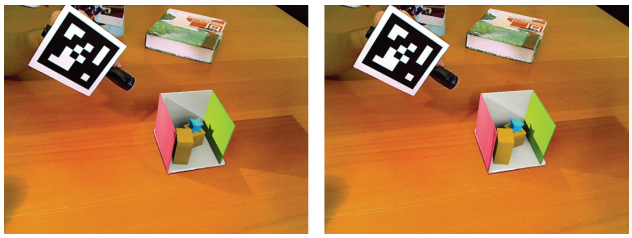


Figure 7: The image on the left shows a scene illuminated via an environment light and spot light using a single light bounce. The right image shows the same scene using multiple bounces enabled. Note the brighter back plane when using multiple bounces.

maps. Here we describe a number of improvements to the Instant Radiosity solver that improve quality and performance.

4.1 Geometry-Aligned Point Splats for ISMs

In the original ISM approach, the point sprites were splatted into ISMs as billboards. Their size was scaled by the inverse squared distance, so that a splat nearer to the view point would have a larger area of influence. However, this is not a good geometrical approximation of the original geometry and may lead to self-shadowing artifacts. We propose a method where the point splats are aligned to the corresponding surface normal and thus greatly reduce wrong self occlusion.

Instead of using screen-space splats, we define the splat in tangent space of the point sample, and transform all four corner vertices of the splat into the ISM coordinate system. To cover the complete upper hemisphere of a VPL for shadow mapping, Ritschel et al. use parabolic mapping. In the original method only the center point was subjected to parabolic mapping. Since we perform this step for each corner vertex, the size of the point splat is implicitly increased or decreased depending on the relative position of the point splat to the VPL. Figure 8 shows a comparison of the shadowing results. In this scene 256 VPLs, shown in red, are all placed at the same position on top of the Cornell box. In cases where the VPL direction is normal to the surface normal, a lot of self occlusion occurs with the standard method (left). With our approach, the depth values in the ISM are more accurate and thus less self occlusion occurs (right).

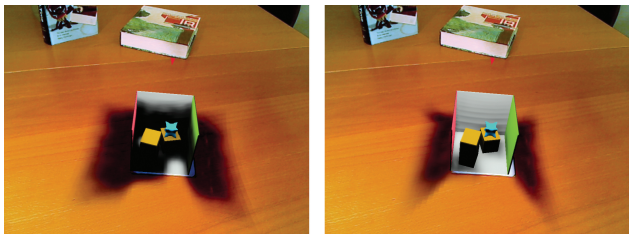


Figure 8: The image on the left shows the result with standard imperfect shadow maps, when 256 VPLs, shown in red, are placed at the same position on top of the Cornell box. Here, point splats are rendered as billboards into the ISM resulting in a lot of wrong self occlusions (rendered at 24 fps). As shown in the right image, our new method, which splats the points aligned to the surface normal into the ISM, removes most of the self shadowing artifacts (rendered at 22 fps).

4.2 VPL Assignment

One of our contributions is an adaptive way to assign VPLs to a given primary light source. The framework is capable of handling

multiple primary light sources, without reducing the indirect illumination computation performance. This is achieved by keeping a constant number of VPLs in a so-called VPL-Buffer. For each primary light source, an appropriate number of VPLs must be assigned. Our idea is to take into account the intensity that is emitted by the VPLs generated by that light source. This intensity depends on the material properties of the point a VPL is placed on. Thus, more VPLs will be generated for primary light sources that shine into more reflective areas. We use the VPL-Buffer of the last frame to calculate the new amount of VPLs to be assigned. Each primary light source sums up the radiant flux for the previously assigned VPLs stored in the old VPL-Buffer and divides it through the area of the primary light source. The renderer distributes the number of VPLs accordingly. In this way, primary lights that do not contribute much to the illumination get fewer VPLs. Note that in the first frame, all primary light sources get assigned the same amount of VPLs. A special case is the primary bounce light source described in Section 5.2. Here we do not have any information about the area of the light source. For this case we just set the percentage of VPLs that should be assigned to it.

4.3 Multiple Bounce Optimization

For multiple bounces, we developed a new method to calculate secondary VPLs, which greatly reduces fill rate compared to the method proposed by Ritschel et al. [27]. Their approach was to extend imperfect shadow maps to reflective shadow maps and afterwards use importance sampling to sample the RSM for each VPL. However, this method has a very large fill rate, while only very little information is finally used.

Our method instead renders the point cloud representing the scene directly into the VPL-Buffer and thus largely reduces the fill rate. The idea is to assign each point of the point cloud to a given VPL slot in the new VPL-Buffer. Then for every slot a so-called source VPL is selected from the previous VPL-Buffer. Every assigned point will be illuminated by this source VPL and the maximum outgoing radiance from that point is used to decide if a new VPL should be placed. All assigned points compete against each other, but only the strongest one will survive. Note that the selection of the source VPL is importance driven and dependent on the VPL slot. Furthermore it would be better to select the new position according to a probability that depends on the outgoing radiance instead of just selecting the best position. However, by reusing the VPLs from the previous buffer, several consecutive light bounces are possible in an efficient way.

4.4 Reducing Temporal Flickering

To get high frame rates, we are forced to keep the number of VPLs as low as possible, while at the same time the image quality should not be decreased too much. For this reason we temporally smooth the illumination results caused by the VPLs, exploiting the temporal coherence between adjacent frames. We do this by storing the illumination buffer from the last frame and reuse the calculated illumination. However, since the objects are moving, and the illumination and the camera may change, we have to calculate how confident an indirect illumination value from the previous frame is. We therefore take three different parameters into account.

1. The relative position change from one frame to the next. For performance reasons we divide the position change into two parts. The difference in depth per screen-pixel and the difference in 2d screen space.

2. The difference between the surface normals. If the normal changes, illumination will also change and hence, the previous values are not as confident as the new ones. We calculate the difference using the dot product between the normals.

3. Global illumination is a global process. So it may happen that a point at a given pixel does not move (no change in position and normal) but the confidence should still be low, because due to other moving objects, the illumination changes a lot. Therefore the difference of the previously calculated illumination and the current illumination also reduces the confidence. We use the cubed difference, so that low differences have little impact and higher differences have more.

The confidence for each pixel is calculated as follows:

$$\varepsilon_{pos} = ||(x_s - x_{prev}, y_s - y_{prev}, d_s - d_{prev})w_p|| \quad (8)$$

$$\varepsilon_{normal} = (1 - (n \cdot n_{prev}))w_n \quad (9)$$

$$\varepsilon_{ill} = \text{saturate}(||I_{new} - I_{prev}||^3)w_i \quad (10)$$

$$confidence = \text{saturate}(1 - \max(\varepsilon_{pos}, \varepsilon_{normal}, \varepsilon_{ill}))c_B \quad (11)$$

where $[*]_{prev}$ always directs to values from the previous frame, (x_s, y_s) is the screen position, d_s the screen depth, n the normal of the screen pixel and c_B is the base confidence. I_{new} is the indirect illumination calculated in this frame. The weighting factors w_* depend heavily on the scene characteristics. For the Cornell box, we have chosen the following weights: $w_p = (10, 10, 10000)$, $w_n = 1.0$ and $w_i = 0.025$.

Note that w_p is a vector so that we are able to weight depth movement independent of screen space movement. If w_p and w_n have smaller values, ghosting artifacts start to show up. Parameter w_i is a little bit more difficult to choose. If it is too high, the indirect illumination difference between adjacent frames tends to be too high, so the confidence will be low and thus flickering artifacts start to show up again. On the other hand if it is too low, illumination updates may take too long.

We also tested the confidence value computation using different equations like multiplication of ε_{pos}, ε_{normal} and ε_{ill}. However, we got best results when using the maximum of these values. Furthermore, the saturation function ensures that the confidence value is between zero and one.

The final per-pixel indirect illumination for the current frame is:

$$I = confidence I_{prev} + (1 - confidence) I_{new} \quad (12)$$

5 Implementation

The framework is designed as a deferred rendering system, which enables us to decouple scene complexity from illumination computation by using a so called G-Buffer. The diagram in Figure 3 shows an overview of the necessary steps that are performed every frame. We also use single-pass interleaved shading from Segovia et al. [30] to reduce shading costs.

5.1 System Overview

The first step is to copy the VPL-Buffer of the previous frame to mappable textures. This is necessary because the VPL-Buffer itself is used as a render target (during VPL creation) and current graphics hardware does not allow for CPU read back of those kind of textures.

The G-Buffer is created afterwards. It stores the color, diffuse intensity, specular intensity and specular power. Furthermore it stores the normal at a screen pixel and the movement in screen-space compared to the last frame (see Section 4.4). To calculate the world space position in every pixel the depth is also stored. The G-Buffer also needs an indicator if the current pixel belongs to a real or virtual object. In our implementation the sign of the depth value is used for this. For real objects, the depth value is positive and for virtual ones it is negative.

Ritschel et al. [27] as well as Laine et al. [18] use interleaved shading by Segovia et al. [30] to only assign a subset of VPLs to a given pixel. We therefore split the G-Buffer. After splitting, the data

in the textures which were copied in the first step must be copied to system memory. Then, for each primary light source, the radiant flux of the assigned VPLs is estimated using the copied data, and the number of VPLs, which get assigned to each primary light source, is calculated. Each primary light source is responsible for creating VPLs on its own by directly rendering into the VPL-Buffer.

After the new VPLs have been created, the imperfect shadow maps introduced by Ritschel et al. [27] are rendered. Each object in the scene has a corresponding point cloud, and these point clouds are used to create the ISMs. The point clouds itself are created at loading time by simply distributing a given number of points over the total surface area of all objects. Each point gets a unique id which is used to assign the point to a VPL.

During ISM creation, the information whether the depth value belongs to a real or virtual object is again stored using the sign of the depth.

Once the VPL-Buffer is created and the imperfect shadow maps are available, illumination computation can start. In Figure 3, these steps are surrounded by a grey box. The output of the grey box is a difference image that will be applied to the video frame. All other steps in the grey box render into two buffers simultaneously, the *VR-Buffer* and the *R-Buffer*. The VR-Buffer stores the complete GI solution including real and virtual objects. The R-Buffer only stores the GI solution computed from the real objects. Note that these buffers are double buffered because simultaneous read and write operations are not possible.

First indirect illumination is computed for both buffers. This is done by drawing a mesh consisting of quads that corresponds to the split G-Buffer in screen space. The mesh has the same amount of quads as there are VPLs in the VPL-Buffer. Each quad has a unique id attached that is used to lookup the corresponding VPL in the VPL-Buffer. Note that both buffers are manipulated in parallel, and indirect illumination is accumulated in one single draw call. In this shading process the flags are used to determine how to calculate the illumination on a per-pixel basis (see Section 3).

Afterwards the split buffers are merged. During the merging step, the results from the previous frames are reused to temporally smooth the indirect illumination calculation (see Section 4.4) and filtered. Then for each primary light source direct illumination is added. The resulting buffers must be tone mapped and for that the tone mapper calculates the average world luminance based on the VR-Buffer. Then both buffers are mapped to low dynamic range. The resulting VR-Buffer and R-Buffer contain illumination caused by the VPLs and primary light sources as shown in Figure 3. In the last step the background image is added to the difference between the tone mapped VR- and R-Buffers. Note that the background image is masked and only added where there are no virtual objects.

5.2 Primary Light Sources

The rendering system currently supports three types of primary light sources. A spotlight, an environment light and a special primary light source that performs one light bounce.

Spot Light The spot light source behaves like a standard spot light except that it can be set to be a real or virtual light source. It stores a reflective shadow map that is rendered from the point of view of the light source. Beside the depth, which is used for standard shadow mapping, it stores the surface color, the material parameter, the normal of the surface and an importance factor similar to the reflective shadow maps from Dachsbacher et al. [3]. When VPLs are created, the importance factor is used to perform importance sampling as proposed by Clarberg et al. [2] on the RSM. After a proper sample position has been found, the information from the RSM is used to create a new VPL in the VPL-Buffer.

Environment Light The environment light source uses the input image from a fish-eye lens camera to capture the surrounding illumination. It does this by placing virtual point lights on a hemisphere around the scene center. Figure 9 shows the virtual point lights placed on the hemisphere. To get a better approximation, the VPLs are first importance sampled according to the illumination intensity. This is again done using the method from Clarberg et al. [2]. Since the environment light source uses the image from the camera, it is set to be a real light source. Note that the environment light is different to the spot light as it uses the VPLs for direct illumination.

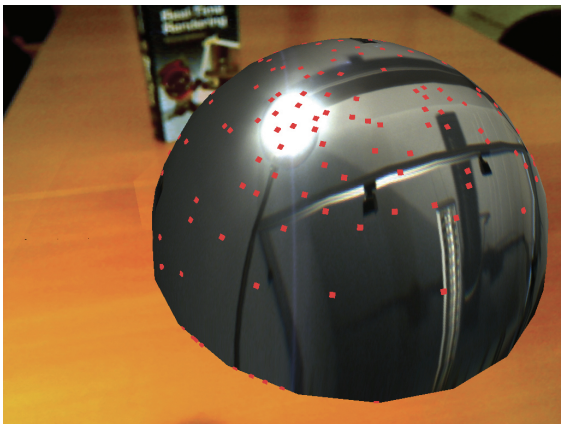
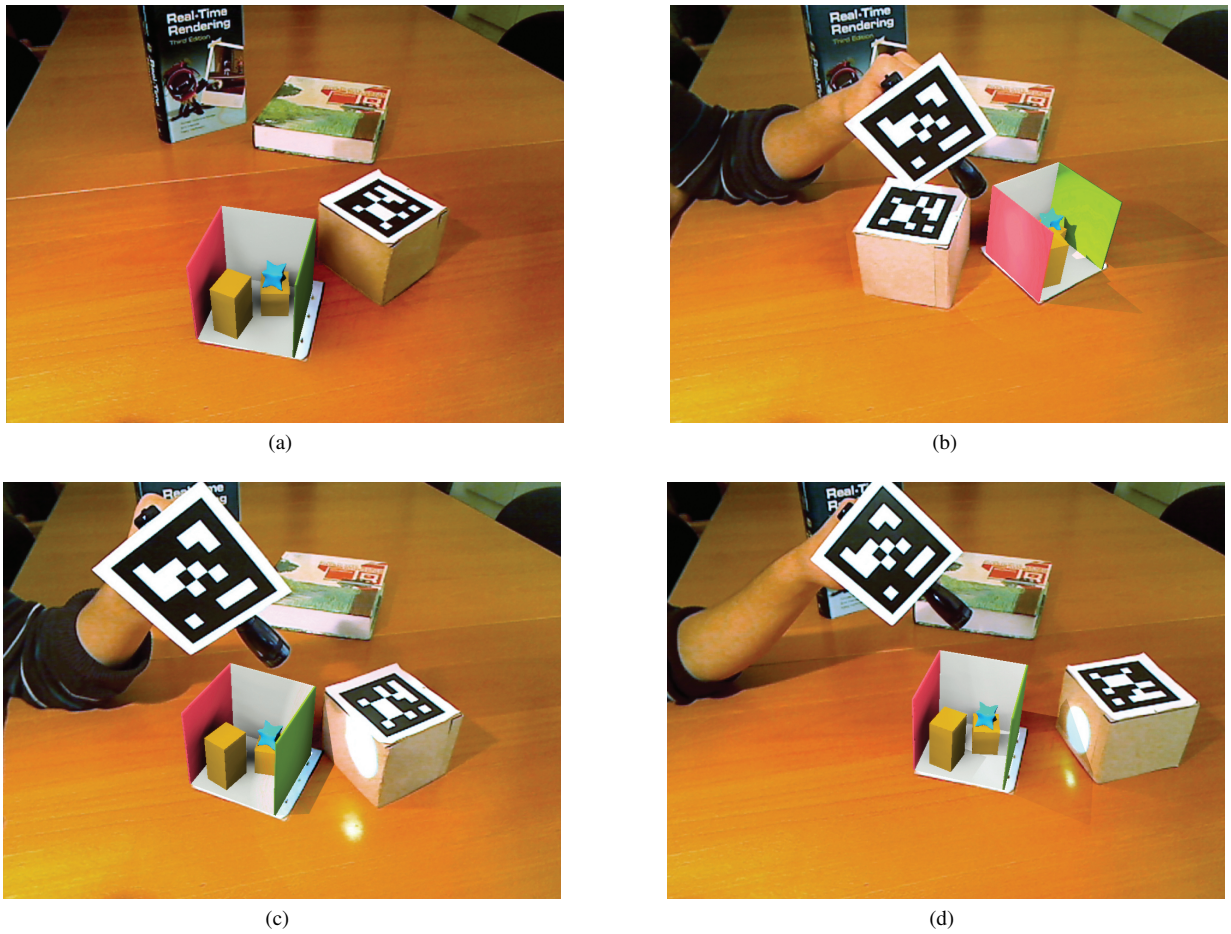


Figure 9: Illustrates incident illumination from the surrounding environment captured with the fish-eye lens camera. The red dots show the positions of the VPLs

Multiple Bounces The multiple bounce light is a special kind of primary light source. The idea behind it is to see the geometry as light source itself. The sources of light are the already placed virtual point lights and from that new ones can be created (see Section 3.8). All VPLs in the VPL-Buffer of the previous frame are used to generate new VPLs, but importance sampling ensures that stronger VPLs are used more often than weaker ones. In the first step, each sample point is assigned to a VPL slot. Then a source VPL dependent on the VPL slot is selected via a lookup texture. Afterwards a glossy light bounce from the source VPL is calculated. For visibility testing the ISM texture from the last frame is used. In the last step we use the depth buffer to find the most important sample point. This is simply done by writing out a depth value for each sample point that is related to the maximum outgoing radiance from the new VPL candidate. The compare function of the depth buffer must be reversed and the final VPL slot will contain only the VPL which has the highest contribution to the scene.

6 Results

All the results were rendered at a resolution of 1024x768 pixels on an Intel Core2 Quad CPU Q9550 at 2.8GHz with 8GB of memory. As graphics card we used an NVIDIA Geforce GTX 285 with 1GB video memory. The operating system is Microsoft Windows 7 64-bit and the rendering framework is developed in C#. As graphics API we use DirectX 10 in conjunction with the SlimDX library. Our system uses a standard webcam from Logitech for see-through video and a Stingray F-125 camera with a fish-eye lens from Allied Vision to acquire the environment map. Our tests took place in an office with some incident light through the window and one spot light illuminating the scene directly. The surrounding illumination is captured using the fish-eye camera. Furthermore we have a small pocket lamp to simulate some direct incident light. We use Studierstube Tracker for tracking the camera and the position of the pocket lamp. Unless otherwise mentioned, we use 256 virtual point lights and represent the scene using 1024 points per VPL. The imperfect



(a) (b)

(c) (d)

Figure 10: (a) Virtual object shadows a real one. (b) The pocket lamp points towards the virtual Cornell box causing red color bleeding through indirect illumination towards the desk and the cardboard box. (c) The pocket lamp illuminates the real cardboard box. Indirect illumination via VPLs causes the green wall of the Cornell box to appear brighter. (d) The pocket lamp illuminates parts of the Cornell box and our system tries to cancel out the highlight on the cardboard box.

shadow map size for one VPL is 128x128 and we split the G-Buffer into 4x4 tiles.

Figure 10(a) shows a virtual Cornell box and a real cardboard box illuminated by the captured environment. The Cornell box shadows the real box. The image is rendered at 24 fps with multiple bounces enabled.

Figure 10(b) shows the same scene with additional light of a real pocket lamp. It points towards the virtual Cornell box causing indirect illumination towards the real box. Note the red color bleeding on the box and the desk. The same illumination effect but reversed is shown in Figure 10(c). Here the pocket lamp partly illuminates the real desk and the real cardboard box, again causing indirect illumination. Our system is capable of handling these different cases in a general way. Both images are rendered at 22 fps.

Figure 10(d) shows the scenario when the pocket lamp shines into the virtual Cornell box. In this image some artifacts are visible. The real spot light actually illuminates the area behind the virtual Cornell box. Our system tries to cancel out this highlight. However, our current method is insufficient, because the pocket lamp approximation is not accurate enough and tone mapping does not correspond with the response curves of the webcam.

We have also implemented a small game to test our method in an interactive environment. It consists of a UFO and several goodies that must be collected so that the UFO is able to land on top of the tower. Figure 11 shows the game environment and the UFO. Please note the illuminated area on the desk caused by the virtual spotlight of the UFO. Furthermore the UFO circles around the tower and our method is able to handle occlusions correctly.

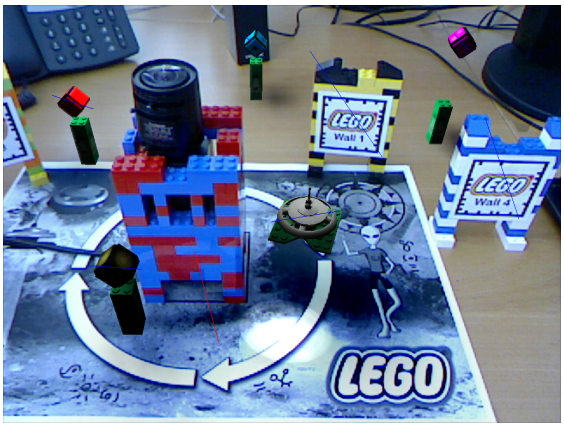


Figure 11: The UFO has its own spot light which illuminates the real desk.

7 Conclusion and Future Work

We introduced a novel method to render mixed reality scenarios with global illumination at real-time frame rates. The main contribution is a combination of the Instant Radiosity algorithm with Differential Rendering. By adding information in various locations of the rendering pipeline it is possible to distinguish between shading contributions from the real scene and from the combined real and virtual scene. Thus, our method is capable to relight the real scene and illuminate the virtual objects in a general way by either using real or virtual light sources.

To enhance image quality we have introduced a number of new techniques. First, we proposed a new way to align the point splats, used for imperfect shadow map creation, along the surface normal. This greatly reduces artifacts caused by wrong self occlusions. Second, to remove temporal flickering artifacts between adjacent frames, we reuse the information from the last frame and smooth indirect illumination computation over time. The results show that our method is able to simulate the mutual influence between real and virtual objects.

In the future, we intend to improve placement of the VPLs for multiple light bounces and address the limitations mentioned in Section 3.7. Furthermore we want to increase the image quality by using a shading method similar to Nichols et al. [23].

Acknowledgements

The authors wish to thank Ralf Habel, Reinhold Preiner and Wolfgang Knecht. This work was supported by a grant from the FFG-Austrian Research Promotion Agency under the program "FIT-IT Visual Computing" (project nr. 820916). Studierstube Tracker is kindly provided by Imagination Computer Services.

References

[1] K. Agusanto, L. Li, Z. Chuangui, and N. W. Sing. Photorealistic rendering for augmented reality using environment illumination. In *ISMAR '03: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, page 208, 2003.

[2] P. Clarberg, W. Jarosz, T. Akenine-Möller, and H. W. Jensen. Wavelet importance sampling: efficiently evaluating products of complex functions. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1166–1175, 2005.

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

[4] N. Dachuri, S. M. Kim, and K. H. Lee. Estimation of few light sources from environment maps for fast realistic rendering. In *ICAT '05: Proceedings of the 2005 international conference on Augmented teleexistence*, pages 265–266, 2005.

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

[6] P. Debevec. A median cut algorithm for light probe sampling. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Posters*, page 66, 2005.

[7] G. Drettakis, L. Robert, and S. Bugnoux. Interactive common illumination for computer augmented reality. In *8th Eurographics workshop on Rendering*, June 1997.

[8] A. Fournier, A. S. Gunawan, and C. Romanzin. Common illumination between real and computer generated scenes. In *Proceedings of Graphics Interface '93*, pages 254–262, May 1993.

[9] T. Grosch. Differential photon mapping: Consistent augmentation of photographs with correction of all light paths. In *Eurographics 2005 Short Papers, Trinity College*, 2005.

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

[11] V. Havran, M. Smyk, G. Krawczyk, K. Myszkowski, and H.-P. Seidel. Importance sampling for video environment maps. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, page 109, 2005.

[12] P. S. Heckbert. *Simulating Global Illumination Using Adaptive Meshing*. PhD thesis, EECS Department, University of California, June 1991.

[13] S. Heymann, A. Smolic, K. Müller, and B. Froehlich. Illumination reconstruction from real-time video for interactive augmented reality. In *International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'05)*, pages 1–4, Montreux, 2005.

[14] K. Jacobs and C. Loscos. Classification of illumination methods for mixed-reality. *Computer Graphics Forum*, 25:29–51, March 2006.

[15] A. Kaplanyan. Light propagation volumes in cryengine 3. In *SIGGRAPH 2009 Course*, 2009.

[16] A. Keller. Instant radiosity. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 49–56, 1997.

[17] M. Korn, M. Stange, A. von Arb, L. Blum, M. Kreil, K.-J. Kunze, J. Anhenn, T. Wallrath, and T. Grosch. Interactive augmentation of live images using a hdr stereo camera. *Journal of Virtual Reality and Broadcasting*, 4(12), January 2007.

[18] S. Laine, H. Saransaari, J. Kontkanen, J. Lehtinen, and T. Aila. Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering 2007*, pages 277–286, 2007.

[19] C. B. Madsen and M. Nielsen. Towards probe-less augmented reality - a position paper. In *GRAPP*, pages 255–261, 2008.

[20] M. McGuire and D. Luebke. Hardware-accelerated global illumination by image space photon mapping. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009*, pages 77–89, 2009.

[21] E. Nakamae, K. Harada, T. Ishizaki, and T. Nishita. A montage method: the overlaying of the computer generated images onto a background photograph. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 207–214, 1986.

[22] G. Nichols, J. Shopf, and C. Wyman. Hierarchical image-space radiosity for interactive global illumination. *Computer Graphics Forum*, 28:1141–1149, June 2009.

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

[24] S. Pessoa, G. Moura, J. Lima, V. Teichrieb, and J. Kelner. Photorealistic rendering for augmented reality: A global illumination and brdf solution. In *2010 IEEE Virtual Reality Conference (VR)*, pages 3–10. IEEE, March 2010.

[25] T. Ritschel, T. Engelhardt, T. Grosch, H.-P. Seidel, J. Kautz, and C. Dachsbacher. Micro-rendering for scalable, parallel final gathering. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers*, pages 1–8, 2009.

[26] T. Ritschel and T. Grosch. On-line estimation of diffuse materials. In *Dritter Workshop Virtuelle und Erweiterte Realitaet der GI-Fachgruppe VR/AR*, volume 3, pages 95–106, 2006.

[27] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect shadow maps for efficient computation of indirect illumination. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, pages 1–8, 2008.

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

[29] I. Sato, Y. Sato, and K. Ikeuchi. Acquiring a radiance distribution to superimpose virtual objects onto a real scene. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):1–12, 1999.

[30] B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche. Non-interleaved deferred shading of interleaved sample patterns. In *GH '06: Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, pages 53–60, 2006.

[31] R. Wang, R. Wang, K. Zhou, M. Pan, and H. Bao. An efficient gpu-based approach for interactive global illumination. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–8, 2009.