Reciprocal Shading for Mixed Reality

Martin Knecht[a,*], Christoph Traxler[b], Oliver Mattausch[c], Michael Wimmer[a]

[a] *Vienna University of Technology, Institute of Computer Graphics and Algorithms, Favoritenstrasse 9-11, A-1040 Vienna, Austria*
[b] *VRVis - Center for Virtual Reality and Visualization Research, Ltd., Donau-City-Strasse 1, A-1220 Vienna, Austria*
[c] *University of Zurich, Department of Informatics, Binzmuehlestrasse 14, CH-8050 Zurich, Switzerland*

Abstract

In this paper we present a novel plausible 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. 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. In addition, we avoid artifacts like double shadows or inconsistent color bleeding which appear in previous work. The dynamic real illumination is derived from the image stream of a fish-eye lens camera. The scene gets illuminated by virtual point lights, which use imperfect shadow maps to calculate visibility. A sufficiently fast scene reconstruction is done at run-time with Microsoft's Kinect sensor. Thus a time-consuming manual pre-modeling step of the real scene is not necessary. 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, reconstruction, Microsoft Kinect

1. Introduction

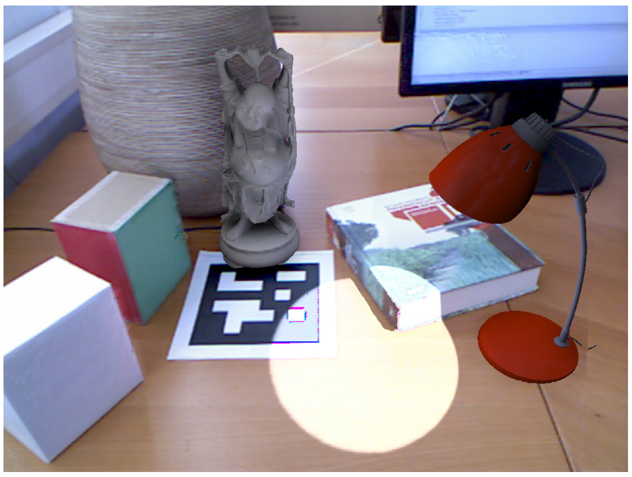

Figure 1: This figure shows a mixed reality scenario where a virtual table lamp illuminates the real and virtual objects causing indirect shadows from the Buddha on a real vase. The scene is reconstructed during run-time and rendered at 17.5 fps.

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 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. In the comprehensive classification of Jacobs and Loscos [1] 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.

There are some previous methods [2, 3] 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

*Corresponding author, Tel.: +43 (1) 234462413, Fax: +43 (1) 23446299
Email addresses: `knecht@cg.tuwien.ac.at` (Martin Knecht), `traxler@vrvis.at` (Christoph Traxler), `mattausch@ifi.uzh.ch` (Oliver Mattausch), `wimmer@cg.tuwien.ac.at` (Michael Wimmer)

camera and the Microsoft Kinect Sensor to create a representation of the real scene on the fly. Figure 1 gives an impression on how our method handles indirect illumination in mixed reality scenarios.

In this paper, we extend a previous Differential Instant Radiosity system [4] with the following novel contributions:

- Inconsistent color bleeding is avoided due to a new type of virtual point lights (VPLs).
- Double shadowing artifacts due to unsupported light paths are eliminated.
- The geometry of the real scene is reconstructed at run-time.

2. Related Work

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

2.1. 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 [5, 6, 7, 8]. The inside-out methods use a camera to capture the surrounding illumination. Ritschel and Grosch [9] used a HDR video camera to capture the surrounding illumination. Sato et al. [10] as well as Korn et al. [11] used a stereo vision inside-out approach to calculate the environmental radiance distribution and to reconstruct the environment.

Once the environment is captured, a fast method is needed to extract light sources from the environment map. Several methods exist to detect light sources in the environment map efficiently according to some importance sampling [12, 13, 14]. To our knowledge these methods use the CPU to generate the samples. We use hierarchical warping from Clarberg et al. [15], since the algorithm works with mipmap levels of the luminance probability map and thus allows us to perform importance sampling directly on the GPU.

2.2. 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 [17] 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. 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. [19] introduced the concept of imperfect shadow maps (ISMs). The idea is to represent the scene as a sparse 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.

There exist several other methods to compute interactive global illumination. We only covered those related to Instant Radiosity as this is the method used in our approach. For a comprehensive overview we refer to the STAR of Ritschel et al. [20].

2.3. 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. [22] 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. Drettakis et al. [23] extended this method to dynamic virtual objects. Debevec [5] introduced Differential Rendering, which is based on the work of Fournier et al. [22]. 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 [2] 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] used 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 got very impressive results for simple scenes and were able to simulate many illumination effects.

2.4. Real-time scene reconstruction

The Microsoft Kinect sensor was released in the year 2010. Since then a large number of researchers focused on various ways to use this very cheap commodity device. The Kinect sensor is capable to deliver a color and a depth stream at a resolution of 640x480 pixel at 30 Hz.

Recently a method called Kinect Fusion was proposed by Shahram et al. [25] and Newcombe et al. [26]. Their method is able to track the camera pose and to reconstruct the environment during run-time. By using a tight feedback loop system, new depth values are constantly added to a voxel volume used for reconstruction and tracking. Lieberknecht et al. [27] proposed another tracking and meshing method. Their approach produces a polygon mesh reconstruction of the real scene. Since our framework is based on a deferred rendering system we do not necessarily need a triangle mesh, as long as we can feed the data from the Kinect sensor into the geometry buffer (see Section 4.3).

Lensing and Broll [28] proposed a method to enhance the quality of the raw depth map from the Kinect sensor. To our knowledge some parts of their algorithm are done on the CPU, while our reconstruction method runs entirely on the GPU. They reach an update rate of 11 frames per second.

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 reconstruct the real scene during run-time (see Section 4). The idea of *Differential Rendering* [22, 5] is to minimize the BRDF estimation 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 (L_{rv} & L_r): one using both virtual and real objects, and one using only the real objects. 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 direct illumination and one-bounce global illumination, which in terms of Heckbert's [29] classification of light paths corresponds to LDE- and $LDDE$-paths, where L is the light source, D is a (potentially) glossy reflection at an object in the scene, and E is the eye. In Section 3.6 we extend the method to be able to calculate multiple bounce global illumination.

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}.

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.

3.2. 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 (previous LD or DD segments) are reused multiple times.

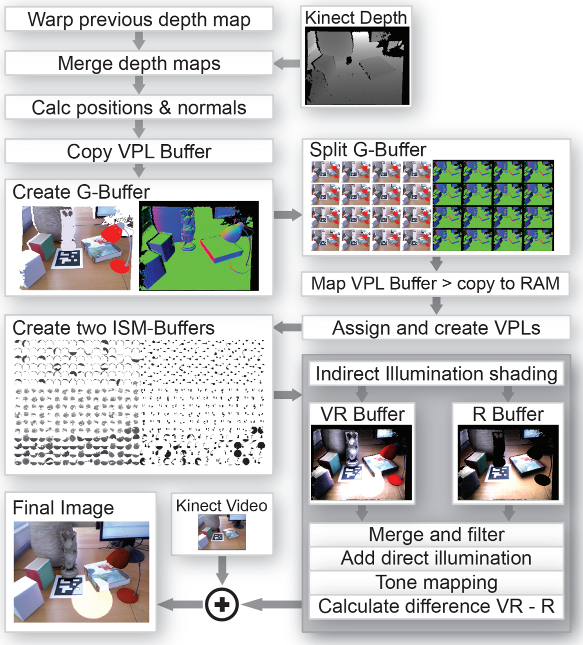

Figure 2: This figure shows the algorithm outline. The raw depth values from the Kinect sensor are preprocessed to improve the quality of the real scene reconstruction. Furthermore a G-Buffer is created and afterwards split to reduce shading costs. The primary light sources generate VPLs from which ISMs 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 2 shows the main steps of a differential instant radiosity rendering system. The renderer is set up as a deferred rendering system and therefore one of the first steps 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 3.7).

After the two ISMs are created and illumination from the VPLs is calculated, the remaining steps are to perform tone mapping and add the result to the video frame using differential rendering.

3.3. Differential Instant Radiosity

In Instant Radiosity, the *DD*-path from a VPL to a surface point (and likewise the *LD*-path from a light source to a surface point for direct illumination) is potentially blocked, and therefore, some of these paths have to be discarded after a visibility test carried out with shadow mapping. In a mixed-reality setting, the type of blocker for these bounces plays an important role when deciding to which GI solution a given light path should be added. Suppose we have light paths as shown in Figure 3. Illustration a) shows an $L_r b_v D_r E$ path and illustration b) an $L_r D_r b_v D_r E$ path. In this notation, b_v is some virtual blocking geometry. So for L_r, both light paths are valid, since virtual objects, including virtual blocking geometries, are not taken into account. However, the result is invalid for L_{rv} since virtual objects are taken into account and therefore the light path does not exist.

If we substitute the virtual blocking geometry b_v with a real blocking geometry b_r, then the light paths cannot exist in both solutions and are simply ignored. So far a simple *visibility test* using shadow mapping and an additional flag whether the blocking geometry belongs to a real or a virtual object would be sufficient. However, if we have a path setup as shown in Figure 4, naive shadow mapping does not deliver enough information to correctly discard both light paths since only the virtual blocking geometry is visible to the system. In a previous Differential Instant Radiosity system [4], double shadowing artifacts, as shown in Figure 5, occurred because the method could only handle either a real or a virtual blocking geometry. Two or more blocking geometry parts behind each other were not supported.

To overcome this limitation, we propose to use two shadow maps for real primary light sources and for the VPLs. Real primary light sources create two reflective shadow maps (RSMs) [17]. One RSM always stores only the real objects and the other RSM only the virtual objects. By having two RSMs, the visibility test needs two shadow map lookups, but double shadowing artifacts are avoided. Note that for performance reasons, two RSMs are only created for real primary light sources like a spot light. If the spot light is set to be virtual, no double shadowing may occur anyway and using one RSM as described in Knecht et al. [4] is sufficient.

Since VPLs can be placed on real and on virtual objects, we always need two imperfect shadow maps [19].

So far only blocking geometry was taken into account to decide to which GI solution the light path should be added. However, it also depends on whether the intersections of a light path belong to real or to virtual objects.

Suppose we have a real spot light that illuminates a virtual green cube and a real blue cube placed on a real desk as shown in Figure 6.

Here the inserted virtual green cube causes four types of light interaction. First, light that hits the real desk causes color

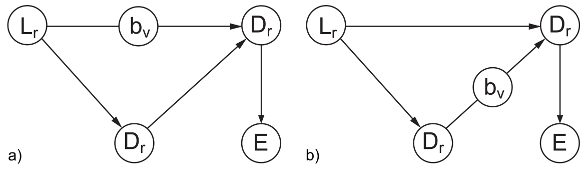

Figure 3: a) Illustrates a virtual blocker in the $L_r D_r$ segment. The virtual blocker casts a shadow from direct illumination. b) Illustrates a path that has a virtual blocker between the $D_r D_r$ segment. Here the shadow results from one-bounce indirect illumination.

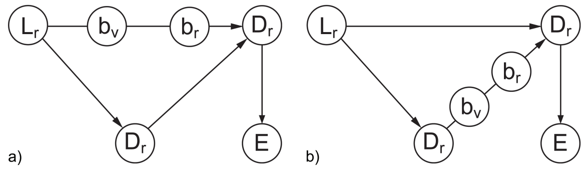

Figure 4: Illustration a) and b) show situations where naive shadow mapping cannot handle correct visibility. In contrast to [4], our method is able to correctly evaluate these paths.

bleeding on the virtual green cube (VPL 1, $L_r D_r D_v E$). Second, light that hits the virtual green cube causes color bleeding on the real desk (VPL 2, $L_r D_v D_r E$). Third, the virtual green cube casts a shadow on the desk illustrated by ray R ($L_r b_v D_r E$). Fourth, the real light arriving along ray R in the real scene causes real color bleeding on the real desk, thus a new VPL is placed on the real blue cube (VPL3 $L_r b_v D_r E$). This will eliminate the real color bleeding effect during the differential rendering step.

Suppose we shade a point on the virtual green cube, illuminated by VPL 1. We have to decide if the path contribution 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 ($L_r D_r D_v E$). 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 again be added to L_{rv} but not to L_r, because VPL 2 is placed on a virtual object ($L_r D_v D_r E$).

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

VPL3 ($L_r b_v D_r E$) is a special new type of virtual point light source. VPL3 will only be used to illuminate real objects. For virtual objects, this light source is completely ignored. The placement and special treatment of VPL3 overcomes the artifact of inconsistent color bleeding, a limitation of [4]. The light caused by VPL3 will be added only to the solution L_r. After differential rendering is applied, that light will cancel out the real color bleeding which is visible in the video frame.

3.4. Direct Bounce Computation

The direct lighting contribution (*LDE* paths) is calculated for primary light sources, i.e., using a spot light.

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 or a light path (see Section 3.6).

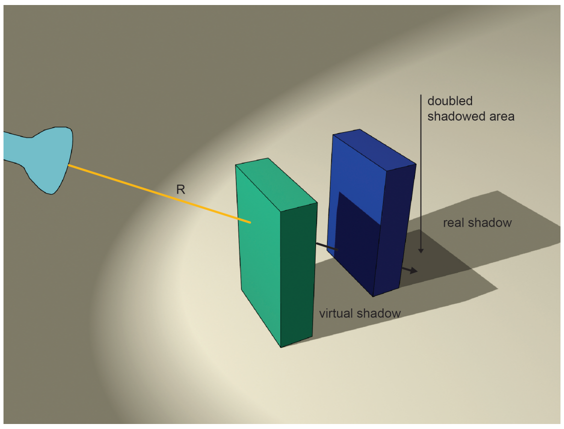

Figure 5: The spotlight and the blue box are real objects and therefore a real shadow is cast onto the desk. In Knecht et al. [4] the reflective shadow map of the light source only had information about the front most object – the green virtual cube, which resulted in double shadowing artifacts.

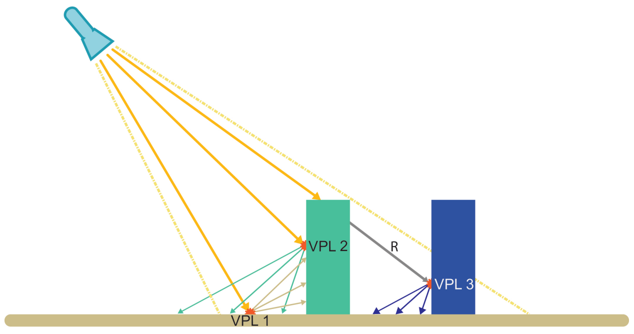

Figure 6: Illustration of an augmented scene with a real desk, a virtual green cube, a real blue cube and a real spot light illuminating the scene. VPL 1 will cause color bleeding from the desk onto the green cube - VPL 2 from the green cube onto the desk. Ray R illustrates the shadowing caused by the virtual green cube. Light arriving at ray R causes real color bleeding from the real blue cube onto the real desk. VPL 3 cancels that color bleeding out. Note that VPL 3 only illuminates real objects.

- For a primary light source pl, $r(pl)$ is stored with the light source.
- For a VPL, $r(\text{VPL})$ is set when the VPL is created (see Section 3.7).
- For the point to be shaded p, $r(p)$ is taken from the object definition.

Depending on visibility, direct light paths *LDE* need to be added to L_r and/or L_{rv}. To perform the *visibility test*, we define the shadow test $st_r(pl, p)$ as a shadow map lookup against real shadow casters and $st_v(pl, p)$ as a shadow map lookup against virtual shadow casters for a given primary light source. L_o is the unobstructed radiance from the surface point p in viewing direction. For a given surface point p in the G-Buffer, the illumination is calculated as follows:

$$L_{rv}^{pl} = L_o^{pl} st_{rv}(pl, p) \tag{1}$$
$$L_r^{pl} = L_o^{pl} r(pl) r(p) st_r(pl, p) \tag{2}$$
$$st_{rv}(pl, p) = min(st_r(pl, p), st_v(pl, p)) \tag{3}$$

The contributions for *LDE* paths are simply summed up as follows for each primary light source:

$$L_{rv} = \sum_{pl} L_{rv}^{pl} \tag{4}$$
$$L_r = \sum_{pl} L_r^{pl} \tag{5}$$

3.5. Indirect Bounce Computation

For indirect light bounces, we need to calculate L_r and L_{rv} for *DDE* paths, i.e., for a point p illuminated by the i^{th} VPL which is placed at the end of light path segment *LD*. To perform the *visibility test* for the i^{th} VPL, the shadow test functions are used in a similar way as in the previous section, but the imperfect shadow maps related to the i^{th} VPL are used. L_o is the unobstructed radiance, i.e., without taking visibility for the VPL into account. $ir(\text{VPL})$ returns 1 if the VPL should only illuminate real objects and 0 otherwise (see Section 3.3). Then, for the i^{th} VPL,

$$L_{rv}^i = L_o^i (1 - ir(\text{VPL}^i)) st_{rv}(\text{VPL}^i, p) \tag{6}$$
$$L_r^i = L_o^i r(pl) r(\text{VPL}^i) r(p) st_r(\text{VPL}^i, p) \tag{7}$$
$$st_{rv}(vpl, p) = min(st_r(vpl, p), st_v(vpl, p)). \tag{8}$$

The contributions for *DDE* paths are summed up in a similar way as shown in Equations 4 & 5.

3.6. Multiple Bounces

Extending Differential Instant Radiosity to multiple bounces is straightforward. Since in Instant Radiosity, each light bounce is simulated by one VPL, we have to forward the information of the previous light path to the next VPL. In this way, it is possible to calculate the influence on the L_{rv} and L_r solutions at the last bounce correctly.

There are two flags that are of interest: First, a flag $r(x)$ whether the previous light path only contained real objects and second, $ir(\text{VPL})$, whether only real objects should be illuminated by the given light path. Imagine that light bounces off several times from real surfaces coming from a real primary light source illuminating a real object. In this case the whole light path interacts only with real objects and the same amount of light will be added to L_{rv} and L_r. However, if there is only one bounce on a virtual object, as illustrated by the green object in Figure 7, the final influence on L_{rv} and L_r differs. The forwarding of the changed flag $r(x)$ is illustrated by the change of the light path color after VPL 1.

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:

$$\bar{x} = r(\bar{x}_{prev}) r(p_{VPL}) \tag{9}$$
$$\bar{x}_0 = r(pl) \tag{10}$$

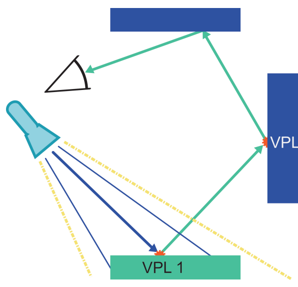

Figure 7: After encountering the first virtual object (shown in green), the light path is flagged to contain virtual objects (also visualized in green). Note that this information is forwarded to subsequent VPLs.

where $\bar{x}_{prev}$ indicates if the path to the preceding VPL only belonged to real objects and p_{VPL} indicates whether the new VPL is created on the surface of a real or a virtual object. 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.

Furthermore the flag ir(VPL), which indicates whether only real objects should be illuminated, is directly copied without modification to the next VPL. Note that virtual objects on the path to the next VPL are ignored if the previous VPL has the flag set ($r(p_{VPL}) = 1$), since it must be placed on a real object.

3.7. Creating VPLs

So far we have assumed that the light paths are already given and thus the VPLs are already placed in the scene. In our system all VPLs are created by the primary light sources and there are three types currently supported: A spotlight, an environment light and a special primary light source that performs one light bounce.

The spot light source behaves like a standard spot light except that it can be set to be a real or virtual light source. If the spot light is set to be virtual, it stores a single 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 (RSM) [17]. When VPLs are created, the importance factor is used to perform importance sampling on the RSM as proposed by Clarberg et al. [15]. After a proper sample position has been found, the information from the RSM is used to create a new VPL in the VPL-Buffer.

However, if the spot light is set to be real, we have to be aware that we need to cancel out inconsistent color bleeding. Figure 6 illustrates this effect with VPL 3. In this case a virtual object is in front of the real one and it would not be possible to place VPL 3 there if only one RSM were used. So if the spot light is set to be real, two RSMs are created each frame. The spot light however has only a certain amount of VPL slots available in the VPL Buffer. Therefore we need a proper way to distribute the VPLs over the two RSMs. As a metric we use the number of pixels covered by real or respectively virtual objects in the RSMs. Their sum is normalized to have a probability value. A pseudo-random Halton value is then used to decide which of the RSMs should be used to create a new VPL. The flag $r(VPL)$ is set with respect to the chosen RSM. Note that since the VPLs have a given intensity, but only a subset of VPLs are applied on either of the RSMs, we must compensate their intensity according to the distribution of VPLs between the two RSMs.

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 8 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 using hierarchical warping [15]. 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.

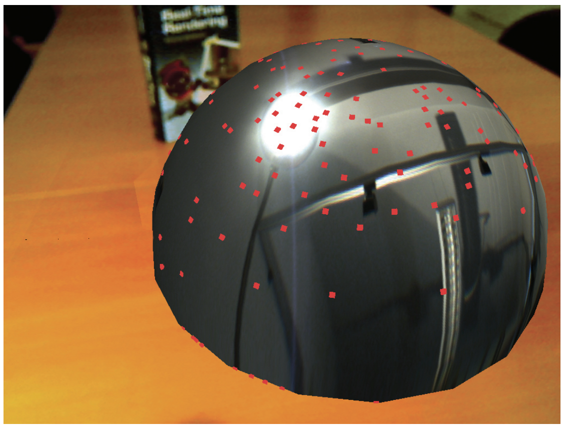

Figure 8: Illustration of incident illumination from the surrounding environment captured with the fish-eye lens camera. The red dots show the positions of the VPLs

The third primary light source is a special type since it performs one light bounce. The idea behind it is to use the geometry itself as light sources, i.e., the VPLs already placed in a previous bounce.

In order to do so, the VPLs from the VPL-Buffer of the previous frame are used to generate new VPLs, and importance sampling ensures that stronger VPLs are used more often than weaker ones. The new VPLs to be created must be placed on the surface of objects in the scene. Therefore we use the point clouds normally used for ISM creation to find possible new VPL positions. For each point in the point cloud representation, three steps are performed: First, the point is assigned to a VPL slot in the new VPL-Buffer. Second, a VPL from the previous frame that illuminates the point is chosen as source VPL. We do this by using pseudo-random Halton values and the previously mentioned importance sampling. The third step is to calculate a glossy light bounce from the source VPL onto the selected point. For visibility testing, the ISM textures from the previous frame are used, since the source VPL is also from that frame.

All three steps are performed on the graphics card, and only a limited number of VPL slots are available to simulate a light bounce. Therefore several points from the point cloud representation are mapped into the same VPL slot. However only one VPL can be stored per slot and we are only interested in the VPL candidate which would have the highest influence respectively the highest outgoing radiance compared to all other VPLs assigned to the same VPL slot. We can ensure this by writing out a depth value for each new VPL candidate that is related to its maximum outgoing radiance. Due to z-buffering, only the strongest VPL candidate will survive. Note that the compare function of the depth buffer must be reversed.

4. Real Scene Reconstruction

Differential rendering is a good method to add virtual objects into a real scene, because the errors of the BRDFs only appear in the difference and not in the absolute values of the final image. However, two GI solutions (L_r and L_{vr}) are needed and to calculate those the geometry and the BRDFs of the real scene must be known. In previous work [4], the real geometry was modeled in a preprocessing step. This is very time consuming and as soon as the real scene changes, the model needs to be updated. Therefore we propose a method that reconstructs the real scene during runtime, which highly increases its usefulness and ease of use.

4.1. Microsoft Kinect

The Microsoft Kinect device is a very low-cost video and depth sensor. Figure 9 shows the two input streams, an *RGB* color image and the depth image. Both streams are delivered in a resolution of 640×480 @ 30 Hz. The input of the depth image is very noisy, and at depth discontinuities, no values are acquired at all. Therefore the unprocessed depth values cannot be used directly to estimate positions and normals.

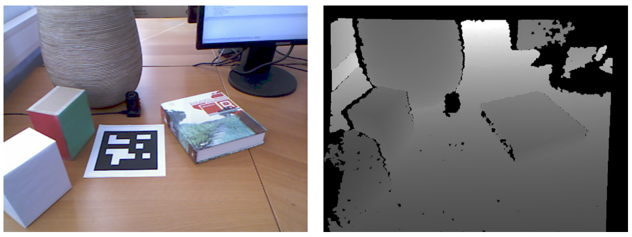
Figure 9: The left image shows the *RGB* color input stream and the right image shows the depth values of the Microsoft Kinect sensor.

4.2. Position and Normal Estimation

In our approach, we perform a warping and merging step before the positions and normals are estimated. In order to reduce the noise and fill in holes, we need more information than available from just one frame. Therefore the information from previous frames is reused. The following steps are performed to get better position and normal estimations:

- Warp depth map from previous frame into new camera view using forward projection
- Merge the new depth data with the warped depth
- Estimate normals based on new depth map

For the warping step, a triangle mesh with 640×480 vertices is created. The vertices store texture coordinates (u, v) for lookup into the depth/weight buffer. The camera pose $T_{current}$ is estimated with Studierstube Tracker and the pose of the last frame T_{prev} is simply cached.

First, for each vertex, the according depth and weight values from the depth/weight buffer of the previous frame are looked up. Then the $(u, v, depth_{prev})$ triple must be back-projected to homogenized view-space coordinates p using the inverse projection function ϕ^{-1}.

$$p = \phi^{-1}(u, v, depth_{prev}) \tag{11}$$

These positions are in the view space of the previous frame (T_{prev}) and therefore need to be transformed into the viewing coordinate system of the current frame ($T_{current}$). Equation 12 calculates the warping transformation T_{warp} and Equation 13 warps a given point p from the old viewing coordinate system into the current viewing coordinate system.

$$T_{warp} = T_{prev}^{-1} T_{current} \tag{12}$$
$$\hat{p} = T_{warp}(p) \tag{13}$$

Finally the depth value is calculated by applying the projection function ϕ on the warped point $\hat{p}$ as shown in Equation 14.

$$depth = \phi(\hat{p}).z \tag{14}$$

Note that parts of the warped depth mesh may overlap, but visibility will be correctly resolved in the z-buffer with the depth test enabled. However, certain vertices or triangles may have invalid depth values or be degenerated (edge length larger than a certain threshold) and are therefore discarded.

The render target for the warping step stores two float values per pixel. One for the warped depth value and one for a weighting factor that gets applied in the merging step. The weight values which were looked up from the depth/weight buffer of the previous frame are simply copied into the render target.

The second step is to merge the warped depth values with the newly available depth values from the Kinect. Our approach for merging the new depth values and weighting values is similar to Newcombe et al. [26]. If both depth values are valid but their difference exceeds a certain threshold (which indicates that they belong to different surfaces), only the new depth value will be written into the render target. After the merging step a depth map is available that has lower noise and less holes than the raw depth input of the Kinect sensor.

In the last step, the normals are estimated. For this, the world-space positions are calculated with respect to the projection matrix of the Studierstube Tracker. Those positions are stored in a separate position map. Then the normals are calculated as follows:

$$N(u, v) = (V(u + 1, v) - V(u, v)) \times (V(u, v + 1) - V(u, v))$$

where (u, v) are the screen-space coordinates and $V(u, v)$ the world space position stored in the position map at (u, v). Due to the reuse of previous data, the normal estimation is far better than just using the current input of the Kinect sensor (see Section 6 for a comparison).

4.3. Kinect Data Integration

The acquired data of the Kinect sensor must be tightly integrated into the differential rendering system. It must be added into the G-Buffer and should be usable for depth-map creation. Adding it into the G-Buffer is relatively straight forward. The world-space position map and the normal map are directly rendered into the G-Buffer. Furthermore the G-Buffer must be filled with Phong BRDF estimation parameters of the real scene's materials. However, we have not performed any Phong BRDF estimation so we must set the diffuse intensity (D_I), specular intensity (S_I) and specular power (S_P) to values that are at least a usable approximation. The specular component is set to zero and the diffuse intensity is set to the input image color from the camera stream of the Kinect sensor. Although these RGB values are a convolution with the incident light, they can be used to a certain degree for our purposes. Note that the error in the BRDF estimation for real objects only influences the differential effect and not the absolute values in differential rendering. However, the error can still get arbitrarily large.

As described in Section 3.3, imperfect shadow maps are used for visibility tests with the VPLs. Therefore all the objects need to have a point cloud representation available. In the case of the Kinect sensor data, we use a vertex buffer with the doubled size of 1280×960 vertices to have enough splats for the ISMs. For each vertex lookup, coordinates are stored that are used to lookup the position map. Furthermore, each vertex gets assigned a unique id by the graphics card while rendering. During creation of the ISMs, the vertex buffer is rendered. In the vertex shader the world space position is looked up and based on the unique vertex id a VPL is chosen to be the target mapping position. According to the chosen VPL and the world space position, the vertex is mapped onto the ISM. A geometry shader is used to create a splat that aligns it in tangent space to the surface (see Knecht et al. [4] for more details). By using a sufficient splat size, the imperfect shadow maps are tightly filled with depth values from the real scene.

Finally the filtered Kinect sensor data must be integrated into the reflective shadow maps of the spot light. For that, a 640×480 vertex buffer is used with an index buffer that connects the vertices together to render small triangles in a similar way as in the depth warping pass. In the geometry shader, the vertex positions of a triangle are looked up using the position map in the same way as it is done for the ISM creation step. However, this time degenerated triangles must be deleted before they are rasterized. They may occur if the positions are invalid. Such cases are easily recognized since those vertices end up at $(0, 0, 0)$ in our framework. So if any vertex in the triangle has a position at the origin the triangle gets discarded. Note that this kind of shadow-map creation can lead to artifacts during visibility tests since the vertices have a limited validity for a different point of view than the position of the camera.

5. Limitations

Our method has a couple of limitations that either lower the quality of the final results or limit the range for which the system can be applied for.

First, since we reduce the reconstruction problem to screen-space rather than to a volume-based approach like in Newcombe et al. [26], the system is not aware of any geometry that is not visible to the camera. While this results in fast execution, the user might observe artifacts when interacting with virtual spot light sources. Imagine that the spot light illuminates a scene from approximately 90 degrees of the observers view. Then shadow calculations may lead to wrong results since a wall facing the spotlight is not necessarily visible to the camera.

The reconstruction method reuses data from previous frames and therefore temporal artifacts appear when interacting with the scene. Although outliers are rejected during the reprojection phase, they cannot be filtered out completely.

In the current integration of the Kinect sensor data, the geometry data is rendered into the G-Buffer. Beside geometry data like depth or normals, the G-Buffer also stores the material parameter (D_I, S_I and S_P) needed for the Phong BRDF model. However, there is no BRDF estimation performed for the real scene geometry. As an approximation for the parameters, the specular component is set to zero. For the diffuse intensity D_I, the current RGB video frame of the Kinect sensor is used. This is a very rough approximation and a better estimation method should be targeted in future work.

In Newcombe et al. [26], the depth data from the Kinect device is also used to estimate the new pose of the camera. In comparison we are using BCH markers that are tracked using Studierstube Tracker. It would be interesting to have a markerless tracking system so that every kind of scenario could be used as an augmented reality setup. Furthermore it would be interesting to have object tracking as proposed by Park et al. [31].

6. Results

All 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 a NVIDIA Geforce GTX 580 with 1.5GB of dedicated 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 the Microsoft Kinect sensor for see-through video and depth acquisition and an uEye camera from IDS with a fish-eye lens to acquire the environment map. Our tests took place in an office with some incident light through a window. Furthermore we have a small pocket lamp to simulate some direct incident light. We use Studierstube Tracker for tracking the Kinect 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 shadow map size for one VPL is 128x128 and we split the G-Buffer into 4x4 tiles.

We also want to mention that the Kinect sensor is too big and heavy for practical usage on an HMD. However, we think that this device could get much smaller in the future and therefore will become feasible in real usage scenarios.

6.1. *Kinect Data Filtering*

In Section 4 we described how the depth map is filtered to gain higher quality position and normal maps. Figure 10 compares the normal estimations. The filtered normals on the right have lower noise and fewer holes on flat areas and therefore the final illumination results are of higher quality. The drawback of the screen-space approach is that areas which just got disoccluded from the previous frame will have low-quality depth values, i.e., due to interpolation at a depth discontinuity, and therefore also wrong normal estimations. For these areas it takes some frames until the correct normal is estimated again.

Depending on the shutter settings of the Kinect sensor and the intensity of the real spotlight, it sometimes happened that no depth values could be evaluated in the center of the spot. Furthermore if the sun shines directly onto the scene, large areas cannot be measured by the Kinect sensor since it is too bright. This results in visual artifacts, since there is simply no geometry rendered into the G-Buffer in those areas.

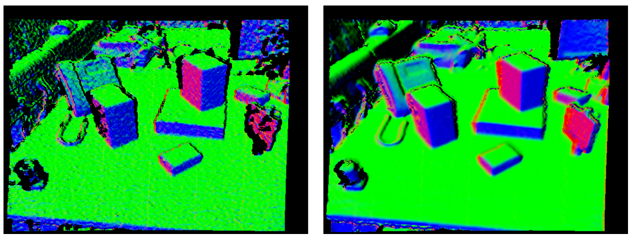
Figure 10: Left: Shows the normal estimation based on the raw depth values from the Kinect sensor. Note that the normal map suffers from noise and holes, since no depth values are available. Right: Shows the normal estimation based on the filtered depth values, resulting in higher quality normal estimations.

6.2. *Rendering Results*

Figure 11 shows a virtual Cornell box and a real white box illuminated by the captured environment. The Cornell box shadows the real box. The image is rendered at 18.0 fps with multiple bounces enabled.

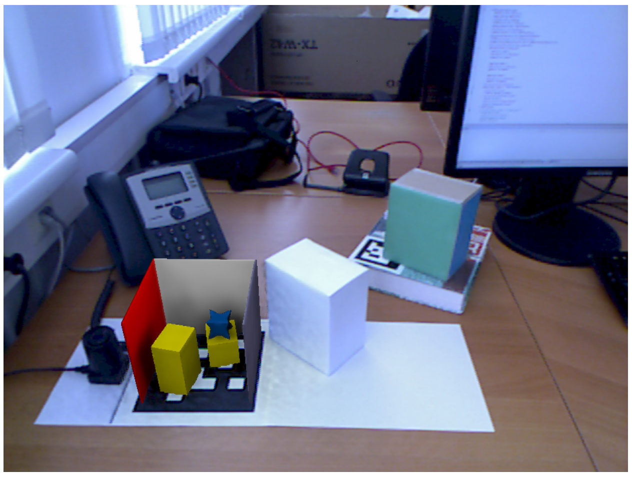
Figure 11: Virtual object shadows a real box. Image rendered at 18.0 fps.

Figure 12 shows the same scene with additional light from 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 13. Here the pocket lamp illuminates the real box, again causing indirect illumination. Our system is capable of handling these different cases in a general way. Both images are rendered at 18.0 fps.

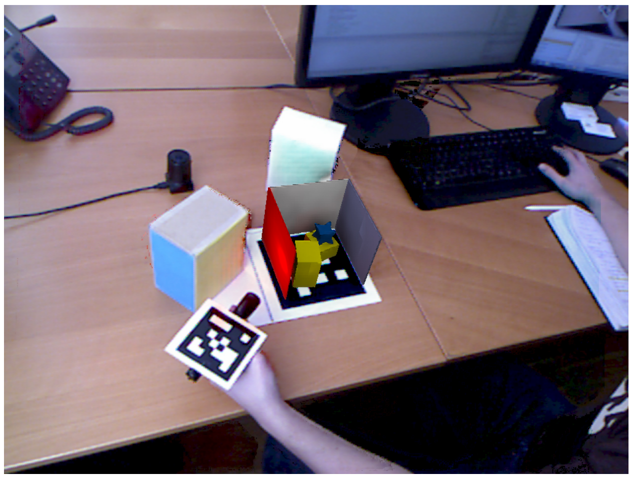
Figure 12: The real pocket lamp illuminates the virtual Cornell Box and causes red color bleeding on the desk and the real box.

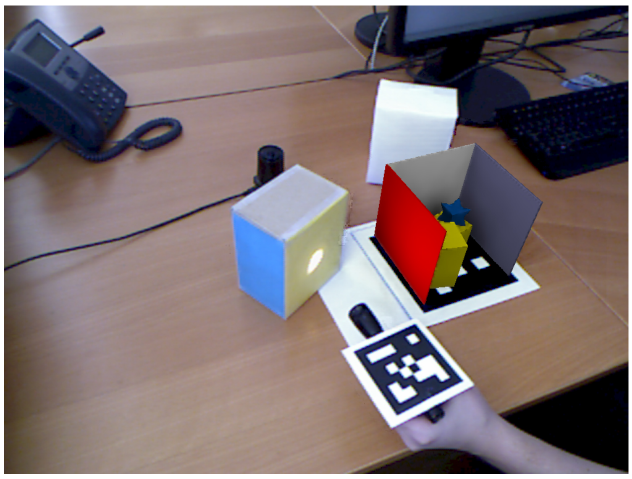
Figure 13: The real pocket lamp illuminates the real box causing indirect illumination onto the virtual Cornell Box.

In Knecht et al. [4], double shadowing and inconsistent color bleeding artifacts occurred as shown in Figure 14. The proposed extensions to Differential Instant Radiosity overcome these issues, resulting in images as shown in Figure 15. Note that the dark shadow behind the real box is removed. Furthermore note the slight darkening of the area near the bright spot of the pocket lamp. Our framework cancels out indirect color bleeding caused by the real box illuminated by the real pocket lamp. Compared to the previous method, the same amount of objects is rendered for shadow map generation, they are just rendered into different render targets. Hence our new approach causes only a small overhead due to the additional shadow map lookup. In numbers this means that the average frame rate for Figure 14 (with artifacts) is at 18.4 fps and for Figure 15 (no artifacts) is at 18.2 fps.

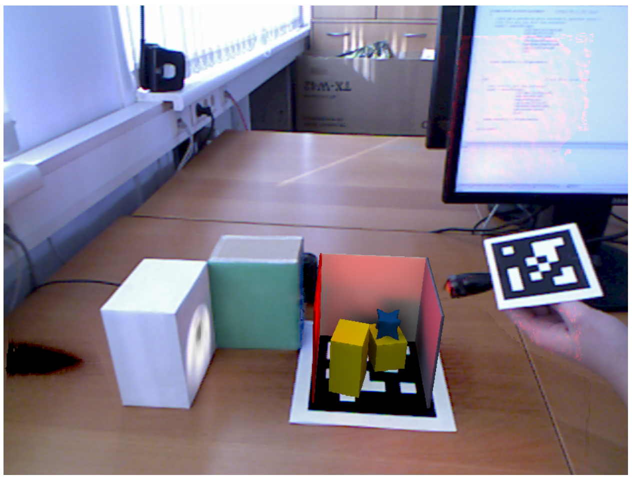
Figure 14: Double shadowing and inconsistent color bleeding artifacts. The dark spot on the left is due to insufficient visibility tests. Note the slight indirect illumination of the bright spot onto the desk and the green wall of the real box. The method from Knecht et al. [4] could not avoid these artifacts.

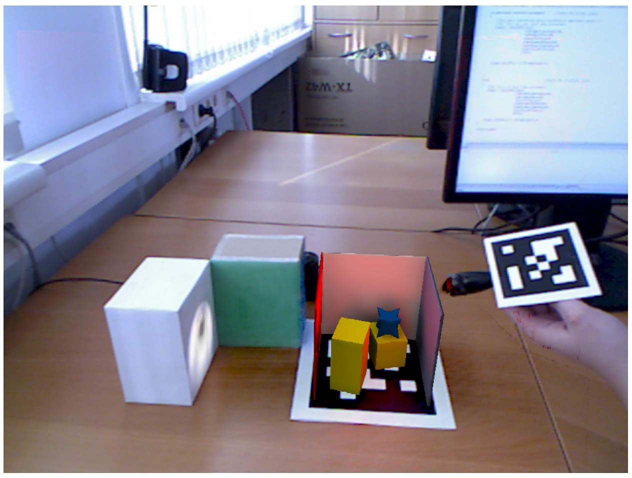
Figure 15: Our method avoids double shadowing and inconsistent color bleeding artifacts. Compared to Figure 14, there is no dark spot in the left part of the image. Furthermore the inconsistent color bleeding on the desk and the real green box is canceled out.

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 supporting more complex light-path combinations. 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.

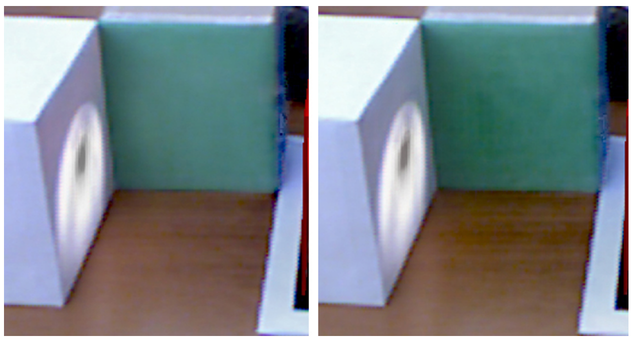
Figure 16: Left: Zoom-in of Figure 14 on incorrect color bleeding on the green box (due to illumination from a spot light which is occluded by a virtual object). Right: Zoom-in of Figure 15. Our new method correctly accounts for occlusion and cancels out the color bleeding.

Thus, our method is capable of relighting the real scene and illuminating the virtual objects in a general way by either using real or virtual light sources. The Microsoft Kinect sensor is used to reconstruct the real scene during run-time, and therefore no time-consuming pre-modeling step is necessary anymore. The results show that our method is able to simulate the mutual influence between real and virtual objects.

In the future, we want to improve the overall quality by estimating the BRDFs of the real scene instead of using an approximation. Furthermore we want to support reflective or refractive objects in the current framework.

8. Acknowledgements

The authors thank Wolfgang Knecht for his support during screenshot and video creation. 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] Jacobs, K., Loscos, C.. Classification of illumination methods for mixed-reality. Computer Graphics Forum 2006;25:29–51.

[2] Grosch, T.. Differential Photon Mapping - Consistent Augmentation of Photographs with Correction of all Light Paths. Trinity College, Dublin, Ireland: Eurographics Association; 2005, p. 53–56.

[3] Grosch, T., Eble, T., Mueller, S.. 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; New York, NY, USA: ACM; 2007, p. 125–132.

[4] Knecht, M., Traxler, C., Mattausch, O., Purgathofer, W., Wimmer, M.. Differential instant radiosity for mixed reality. In: Proceedings of the 9th IEEE International Symposium on Mixed and Augmented Reality. ISMAR ’10; Washington, DC, USA: IEEE Computer Society; 2010, p. 99–108.

[5] Debevec, P.. 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; New York, NY, USA: ACM; 1998, p. 189–198.

[6] Agusanto, K., Li, L., Chuangui, Z., Sing, N.W.. Photorealistic rendering for augmented reality using environment illumination. In: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented

Reality. ISMAR '03; Washington, DC, USA: IEEE Computer Society; 2003, p. 208–216.

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

[8] Supan, P., Stuppacher, I., Haller, M.. Image based shadowing in real-time augmented reality. International Journal of Virtual Reality 2006;5(3):1–7.

[9] Ritschel, T., Grosch, T.. On-line estimation of diffuse materials. In: Dritter Workshop Virtuelle und Erweiterte Realitaet der GI-Fachgruppe VR/AR; vol. 3. 2006, p. 95–106.

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

[11] Korn, M., Stange, M., von Arb, A., Blum, L., Kreil, M., Kunze, K.J., et al. Interactive augmentation of live images using a hdr stereo camera. Journal of Virtual Reality and Broadcasting 2007;4(12):107–118.

[12] Dachuri, N., Kim, S.M., Lee, K.H.. Estimation of few light sources from environment maps for fast realistic rendering. In: Proceedings of the 2005 international conference on Augmented tele-existence. ICAT '05; New York, NY, USA: ACM; 2005, p. 265–266.

[13] Havran, V., Smyk, M., Krawczyk, G., Myszkowski, K., Seidel, H.P.. Importance sampling for video environment maps. In: ACM SIGGRAPH 2005 Sketches. SIGGRAPH '05; New York, NY, USA: ACM; 2005,.

[14] Debevec, P.. A median cut algorithm for light probe sampling. In: ACM SIGGRAPH 2005 Posters. SIGGRAPH '05; New York, NY, USA: ACM; 2005,.

[15] Clarberg, P., Jarosz, W., Akenine-Möller, T., Jensen, H.W.. Wavelet importance sampling: efficiently evaluating products of complex functions. In: ACM SIGGRAPH 2005 Papers. SIGGRAPH '05; New York, NY, USA: ACM; 2005, p. 1166–1175.

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

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

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

[19] Ritschel, T., Grosch, T., Kim, M.H., Seidel, H.P., Dachsbacher, C., Kautz, J.. Imperfect shadow maps for efficient computation of indirect illumination. In: ACM SIGGRAPH Asia 2008 papers. SIGGRAPH Asia '08; New York, NY, USA: ACM; 2008, p. 129:1–129:8.

[20] Ritschel, T., Grosch, T., Dachsbacher, C., Kautz, J.. The state of the art in interactive global illumination. Computer Graphics Forum 2012;31:160–188.

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

[22] Fournier, A., Gunawan, A.S., Romanzin, C.. Common illumination between real and computer generated scenes. In: Proceedings of Graphics Interface '93. Toronto, ON, Canada; 1993, p. 254–262.

[23] Drettakis, G., Robert, L., Bougnoux, S.. Interactive common illumination for computer augmented reality. In: Proceedings of the Eurographics Workshop on Rendering Techniques '97. 1997, p. 45–56.

[24] Pessoa, S., Moura, G., Lima, J., Teichrieb, V., Kelner, J.. Photorealistic rendering for augmented reality: A global illumination and brdf solution. In: 2010 IEEE Virtual Reality Conference (VR). Washington, DC, USA: IEEE Computer Society; 2010, p. 3–10.

[25] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In: Proceedings of the 24th annual ACM symposium on User interface software and technology. UIST '11; New York, NY, USA: ACM; 2011, p. 559–568.

[26] Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., et al. Kinectfusion: Real-time dense surface mapping and tracking. In: Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality. ISMAR '11; Washington, DC, USA: IEEE Computer Society; 2011, p. 127–136.

[27] Lieberknecht, S., Huber, A., Ilic, S., Benhimane, S.. Rgb-d camera-based parallel tracking and meshing. In: Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality. ISMAR '11; Washington, DC, USA: IEEE Computer Society; 2011, p. 147–155.

[28] Lensing, P., Broll, W.. Fusing the real and the virtual: A depth-camera based approach to mixed reality. In: Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality. ISMAR '11; Washington, DC, USA: IEEE Computer Society; 2011, p. 261–262.

[29] Heckbert, P.S.. Simulating global illumination using adaptive meshing. Ph.D. thesis; EECS Department, University of California; 1991.

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

[31] Park, Y., Lepetit, V., Woo, W.. Texture-less object tracking with on-line training using an rgb-d camera. In: Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality. ISMAR '11; Washington, DC, USA: IEEE Computer Society; 2011, p. 121–126.