Reflective and Refractive Objects for Mixed Reality

Martin Knecht, Christoph Traxler, Christoph Winklhofer, and Michael Wimmer

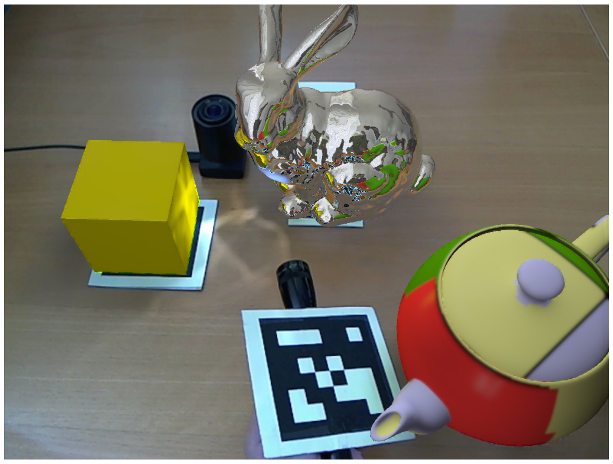

Fig. 1. This image shows the Stanford bunny with reflective material illuminated by a virtual spotlight, causing reflections and caustics onto the real table and the virtual yellow cube (rendered at 19 fps).

Abstract—In this paper, we present a novel rendering method which integrates reflective or refractive objects into a differential instant radiosity (DIR) framework usable for mixed-reality (MR) applications. This kind of objects are very special from the light interaction point of view, as they reflect and refract incident rays. Therefore they may cause high-frequency lighting effects known as caustics. Using instant-radiosity (IR) methods to approximate these high-frequency lighting effects would require a large amount of virtual point lights (VPLs) and is therefore not desirable due to real-time constraints. Instead, our approach combines differential instant radiosity with three other methods. One method handles more accurate reflections compared to simple cubemaps by using impostors. Another method is able to calculate two refractions in real-time, and the third method uses small quads to create caustic effects. Our proposed method replaces parts in light paths that belong to reflective or refractive objects using these three methods and thus tightly integrates into DIR. In contrast to previous methods which introduce reflective or refractive objects into MR scenarios, our method produces caustics that also emit additional indirect light. The method runs at real-time frame rates, and the results show that reflective and refractive objects with caustics improve the overall impression for MR scenarios.

Index Terms—Mixed reality, reflections, refractions, caustics

1 Introduction

Real-time rendering for mixed-reality (MR) applications is expected to achieve a high degree of realism. Virtual objects should smoothly blend with real ones and produce a plausible illusion. This contrasts with illustrative scenarios, where virtual objects intentionally look artificial to visualize meta-information of real objects. Mixed reality is an increasingly attractive technology applied in diverse areas such as virtual design and prototyping, architecture, entertainment & edutainment, training simulations and cultural heritage. In such scenarios, users should get the impression that the virtual objects are part of the real world. This implies that mutual shading effects must be considered for real-time rendering [7].

- *Martin Knecht is with Vienna University of Technology. E-mail: knecht@cg.tuwien.ac.at.*
- *Christoph Traxler is with VRVis - Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH. E-mail: traxler@vrvis.at.*
- *Christoph Winklhofer is with Vienna University of Technology. E-mail: christoph.winklhofer@student.tuwien.ac.at.*
- *Michael Wimmer is with Vienna University of Technology. E-mail: wimmer@cg.tuwien.ac.at.*

Manuscript received 13 September 2012; accepted 10 January 2013; posted online 16 March 2013; mailed on 16 May 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

Two important aspects for realism are reflection and refraction, which are mostly neglected in mixed-reality applications. Especially in the scenarios listed above, ignoring glassy and mirroring objects would significantly restrict the potential of the technology. Reflection and refraction have already been addressed in real-time rendering solutions [16, 22, 24]. However, only limited results have been achieved so far for mixed-reality applications.

A mixed-reality solution has to consider two major requirements. First, rendering performance must be high enough to allow smooth interaction, which is a key aspect of MR. Second, the mutual shading effects between real and virtual objects must be considered, otherwise the resulting artifacts would destroy the illusion. Concerning reflection and refraction, this means that real objects should be reflected in and refracted through virtual ones and vice versa. Caustics should also be cast between real and virtual objects. Furthermore, virtual or real light should be appropriately attenuated when passing through translucent materials. All these effects associated with glassy and mirroring objects have so far not been considered for mixed reality in their entirety.

In this paper, we present a novel method that simulates the mutual reflection and refraction effects between real and virtual objects. Figure 1 shows a reflective Stanford bunny illuminated by a virtual spotlight, causing caustics on the real table and the virtual yellow box. A popular method to embed virtual objects into a real scene is Differential Rendering [4]. Differential Instant Radiosity (DIR) [11] is a global-illumination solution for MR based on this method. It simulates mutual shading effects between real and virtual objects and thereby considers direct and indirect light transport and casting of shadows. Our reflection and refraction method works together with DIR and can therefore be seen as an extension that improves the overall plausibility of the MR illusion by considering two additional important material properties.

Our method adds differential effects to reflected or refracted objects in the same way differential rendering adds these effects to diffuse or glossy objects. Furthermore, spotlight sources introduce caustics when they illuminate reflective or refractive objects. The method is based on differential instant radiosity and therefore spotlight sources place virtual point lights at illuminated surface points to cause an additional indirect light bounce. However, for reflective or refractive objects, these VPLs are first reflected or refracted. In this way, the generated caustic effects are able to indirectly illuminate the surrounding scene.

Our main contributions are:

- Reflective and refractive objects are integrated into DIR using multiple geometry buffers.
- Specular effects like reflections, refractions and caustics caused by real and virtual objects are simulated.
- Caustics from spot lights emit indirect light.

The main advantage of our method is its high performance and its tight integration with DIR, so that this global-illumination solution for MR is extended with light coming from reflections and refractions.

2 Related Work

One of the first approaches to handle reflective and refractive objects in a mixed-reality setting was introduced by State et al. [18] in 1996. They used an outside-in approach where a chrome or glass sphere was placed at a known position. Then the image of the sphere was extracted from the video feed and remapped onto reflective or refractive objects.

Debevec introduced differential rendering [4], a method which is also able to render reflective and refractive objects into images of a real scene. However, the reflected objects visible in reflections belonged to the rendered representation of the real objects, which means that the effects of differential rendering where not applied to reflective or refractive objects. Grosch [5] extended differential rendering with photon mapping, which makes it possible to consider additional light paths caused by reflection and refraction. His so-called differential photon map contains photons with positive and negative energy, which in combination yield caustic patterns on real surfaces. Reflection and refraction of real objects through virtual ones are simulated by image back-projection. Although this approach produces impressive results, it is an off-line method applied to single images.

A recent method by Karsch et al. [9] is able to add reflective and refractive objects including caustics to images. However, the method is mainly designed for easy editing of the images and therefore does not fulfill the real-time requirement.

Reflective and refractive objects also appeared in a master thesis by Pirk [15]. Refractions are rendered on the GPU by using the image of the see-through camera as background texture or a static cube map, where appropriate lookup functions approximate distortion effects. Pessoa et al. [14] also use static environment maps to render reflections and refractions for mixed-reality applications. They combine this with spatial BRDFs including Fresnel factors and sophisticated texture sampling on the GPU. In both approaches, only virtual objects can have refractive materials – neither caustics nor light attenuation are considered. The static environment maps used are an even more severe limitation for the highly dynamic nature of mixed-reality scenarios. Uranishi et al. [21] used a box marker to estimate the reflectance and roughness of the floor to simulate its reflection properties. Another method proposed by Tawara and Ono [19] showed caustics cast by virtual water in an augmented-reality setup. However, the method does not take surrounding illumination into account. Our method, in contrast, uses a hemispherical environment map from a live video stream coming from a camera with a fish-eye lens.

A recent rendering method was proposed by Kan and Kaufmann [8] in 2012. They use a real-time ray-tracing framework instead of a rasterization approach to render reflections, refractions and caustics in mixed-reality scenarios. They are able to produce high-quality results, but the frame rate is still low for completely dynamic scenes with caustics enabled.

3 Background

Our proposed method for reflective and refractive objects, including caustics, is a combination of four different methods, which we shortly describe in this section.

3.1 Differential Instant Radiosity

The first one is differential instant radiosity, introduced by Knecht et al. [11, 12]. It is used to generate global illumination effects in mixed-reality scenarios and combines differential rendering (DR), proposed by Debevec [4], with instant radiosity (IR), introduced by Keller [10]. Instant radiosity is a method to simulate diffuse to glossy global illumination effects by placing virtual point lights (VPLs) at surfaces illuminated by primary light sources. These VPLs are then used to illuminate the scene, causing one additional indirect illumination. In order to have more light bounces, an existing VPL can be used to spawn new VPLs.

Differential rendering is a convenient method to add virtual objects into images of real scenes. Furthermore it allows taking the mutual lighting effects, such as shadowing or indirect illumination, between real and virtual objects into account. To do so, two global-illumination (GI) solutions are computed. One GI solution (L_{rv}) is calculated by taking the real *and* the virtual objects into account. The second GI solution (L_r) only takes the real objects into account. Note that this also implies that the real scene must be pre-modeled. These two buffers are normally calculated in high dynamic range (HDR) and need to be tone-mapped first before the final composed image can be calculated. For the rest of the paper we assume that these buffers are already tone-mapped and thus the final buffer (L_{final}) can be calculated as follows:

$$\Delta L = L_{rv} - L_r \quad (1)$$

$$L_{final} = CI_m + \alpha \Delta L \quad (2)$$

where CI_m is the see-through camera image (CI) masked to black where virtual objects are placed and $\alpha = CI_m / L_r$ is a scaling factor to compensate for the relative error between the camera image CI and the L_r buffer. In this way the absolute values in the differential buffer ΔL are relative to CI instead to L_r, resulting in more visually pleasing images (see Debevec [4]). Note that Equations 1 and 2 only apply for diffuse objects, reflective and refractive objects are described in Section 4.

In DIR [11], the buffers L_{rv} and L_r are calculated using VPLs, and decision flags are used to determine whether the light contribution by a VPL should be added to L_{rv} and L_r or only to L_{rv}. Although instant radiosity causes indirect lighting, effects like caustics are difficult to simulate since the technique is mainly applicable for low-frequency lighting effects. That said, Dachsbacher and Stamminger [3] showed that simulating caustics is possible with instant radiosity. However, a lot of VPLs are needed just for a single caustic, which results in high computation costs. Therefore, in our method we avoid rendering caustics using virtual point lights and instead use more efficient methods.

3.2 Reflections, Refractions and Caustics

Since specular objects should have high-quality reflections, the second method used in this work was introduced by Popescu et al. [16]. It introduces a convenient way to create realistically looking reflections. A simple approach is to just render a cube map from the point of view of a reflective or refractive object similar to Pirk [15]. In this way artifacts appear if a reflected object is close to the reflective or refractive one. The method from Popescu et al. [16] avoids these artifacts by using impostors. For each object that is reflective or refractive, a list of billboards is created, where each billboard belongs to a surrounding object that should appear in the reflection/refraction. When the reflecting or refracting object is rendered, the refracted and reflected rays are intersected with each billboard. Since the billboards do not just store the color but also the normal and the depth at each pixel, a ray hit position can be refined iteratively, leading to higher quality reflections.

The third method, which introduces realistic-looking refractions, was proposed by Wyman [22]. It is able to compute two refractions for an incident ray using a two-pass approach. In the first pass, the back-facing triangles of the refractive object are rendered into a depth and normal buffer. In the second pass, the front faces are rendered. At each pixel, Snell's law is used to calculate the refracted ray direction. With the stored depth from the previous pass, a new position at which the refracted ray approximately exits the object can be calculated. Using this point and the associated normal, a new refracted outgoing ray is computed.

The fourth method was proposed by Wyman and Davis [24] and is used to compute caustics in real time. For that, a scene gets rendered from the light source position. Besides the standard shadow map, three additional buffers are generated. One contains the hit positions from the refracted rays. The second buffer stores the photon intensity, and the third one stores the incident photon direction. In the second pass, the photons are rendered from the point of view of the camera or the spot light source using quads, where each quad approximates one photon. These quads are then blended into a caustic buffer. Using this buffer, realistically-looking caustics can be rendered.

4 Extending Differential Instant Radiosity

The original DIR method only supported diffuse to medium-glossy light bounces. In terms of Heckbert's [6] classification of light paths, and considering only one indirect light bounce, this means that the system was limited to *LDE* and *LDDE* paths, where *L* is the light source, *D* is a diffuse reflection at an object in the scene, and E is the eye. However, reflective and refractive objects, which generate caustics, were not supported. In the proposed extension to DIR, these reflections (*S*) and (double) refractions (*SS*) are added. Furthermore, for any primary spotlight source, the reflective or refractive objects produce caustics that emit indirect light.

Section 4.1 will introduce reflective and refractive objects to support *LDSE* and *LDSSE* light paths in DIR. Then Section 4.2 describes how caustic effects (*LSDE* and *LSSDE* light paths) are added, including those with an additional indirect light bounce (*LSDDE* and *LSSDDE*), using VPLs.

4.1 Reflective and Refractive Objects in DIR

Imagine a virtual reflecting and refracting sphere that is surrounded by several real objects as illustrated in Figure 2. At the sphere's surface point p, the incoming reflected and refracted light from points p_{refl} and p_{refr} is composed according to Fresnel's law. In the original DIR approach, reflections and refractions at such a point are always rendered objects because the sphere is set to be virtual. In other words, only L_{rv} stores the rendered representations of the real objects (points p_{refl} and p_{refr}) and L_r is set to zero. However, there are no DR effects applied to real reflected or refracted objects, which is not desirable, leading to less accurate and therefore less immersive appearance of real reflected and refracted objects.

Grosch [5] described the idea that points p_{refl} and p_{refr} could be back-projected into image space to lookup the color in the see-through video frame if they belonged to real objects. In this way, the reflected and refracted colors are taken from the see-through video frame itself instead of the rendered representations, leading to a higher degree of immersion. Point p_{refl} (ray two) lies inside the camera frustum (dashed yellow lines) and can be used for back-projection if the occluding green box is a virtual object, otherwise not. On the other hand, p_{refr} (ray three) lies outside of the camera frustum and therefore the rendered representation must be used. Please note that the back-projection is only valid if the real surfaces are assumed to be diffuse.

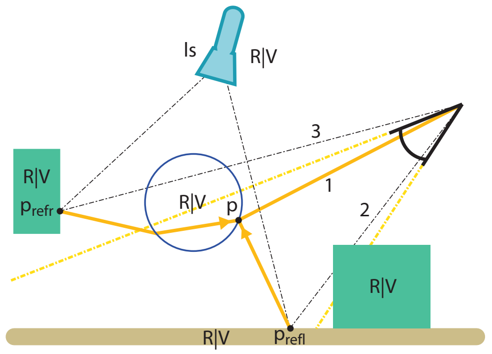

Fig. 2. This figure illustrates the possible different cases. Each light interacting object (light source, reflected object, refracted object, reflective or refractive object (blue sphere)) can be a real or a virtual object. The dashed yellow lines illustrate the view frustum of the video see-through camera. Rays one, two and three show possible back-projection cases where rays one and two are inside and ray three is outside of the view frustum (see Section 4.1.3).

The previous example described only one out of several cases that need to be handled. Each of the light-interacting elements (light source, reflected object, refracted object, reflective or refractive object) could be real or virtual (see Figure 2). In order to handle all possible cases, two steps must be altered in the DIR pipeline.

First, we describe in Section 4.1.1 how the light-accumulation step must be adapted for reflective or refractive objects. Then we describe how the final image gets composed when reflective or refractive objects are involved in Section 4.1.2. Please note that the GI solution buffers are computed in screen space and on a per-pixel basis. Each pixel belongs either to a diffuse object or a reflective or refractive object. This means that Equations 1 and 2 are only applied if a diffuse object is visible at a pixel. On the other hand, equations marked with primes are only applied if a reflective or refractive object is visible at a pixel. For readability we therefore introduce two additional GI solution buffers called L'_r and L'_{rv} although in practice L_r and L'_r are one buffer and so are the buffers L_{rv} and L'_{rv}.

4.1.1 GI Solution Buffer Computation

As in the previous example, let us assume that p_{refl} and p_{refr} are points on real diffuse objects and that the reflective and refractive object is a virtual one. Then, in contrast to the original DIR approach, the reflected and refracted real objects have to be written into the L'_r buffer even though point p belongs to a virtual object (see Equation 4).

The outgoing radiance at point p towards the view direction (ray one) is a composition, according to Fresnel's law, of the incident radiances from the reflective and refractive ray directions illustrated by the orange arrows in Figure 2. The functions $L_{refl}(ls)$ and $L_{refr}(ls)$ calculate the incident radiance from the reflected and refracted ray directions at point p for a given light source ls with the methods described in Section 3.2. Depending on Equation 5 the contribution of p_{refl} or p_{refr} due to a light light source ls (all primary light sources and all VPLs) is added to the L'_r buffer or not. Let $r(x)$ be a function that returns *true* if element x is associated with a real object and *false* if not. x can be a light source (ls), a light path ($\bar{x}$) or a surface point (p). Then the two GI solutions for reflective or refractive objects are computed as follows:

$$L'_{rv} = \sum_{ls} \left(L_{refl}(ls) + L_{refr}(ls)\right) \tag{3}$$

$$\begin{aligned} L'_r = & \sum_{ls} \big(L_{refl}(ls) \cdot real(ls, p_{refl}) + \\ & L_{refr}(ls) \cdot real(ls, p_{refr})\big) \end{aligned} \tag{4}$$

$$real(ls, p) = \begin{cases} 1 & \text{if } r(ls) \wedge r(p) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

4.1.2 Compositing Final Image

Since the reflected and refracted light contributions of real objects are also written into the L'_r buffer, the contribution of real objects gets canceled out when the difference buffer $\Delta L'$ is calculated (see Equation 6). What remains in $\Delta L'$ are only the differential effects, caused by the virtual objects. The next step is to add information available in the video see-through image (see Equation 7). However, in order to get the correct color values from the video see-through image, back-projection as proposed by Grosch [5] must be applied.

$$\Delta L' = L'_{rv} - L'_r \tag{6}$$

$$L'_{final} = CI_{std}(p, p_{refl}, p_{refr}) + \alpha' \Delta L' \tag{7}$$

Equation 8 first checks whether all points are on real surfaces. If so, we can take the color at the pixel of point p straight away, since it already shows the composed real reflections and refractions of p_{refl} and p_{refr}. This is done by function $CI(x)$. It simply takes a 3D point x, projects it into screen space and returns the color value from the see-through video feed.

L_{env} is a screen-space buffer that contains data from the environment map (fish-eye lens) that has been reflected or refracted through the object at a pixel, but only for those cases where no ray hit points for p_{refl} or p_{refr} can be found (e.g., when the ray leaves the modeled scene without intersection, see Section 4.1.3 and Figure 3). Equation 9 does the compositing and back-projection of the points p_{refl} and p_{refr}. Therefore in cases where not all points are on real surfaces, the returned color value of Equation 8 is the sum of the L_{env} buffer and the CI_{bp} function.

$$CI_{std}(p, p_{refl}, p_{refr}) = \begin{cases} CI(p) & \text{if } r(p) \wedge \\ & \quad r(p_{refl}) \wedge \\ & \quad r(p_{refr}) \\ L_{env} & \\ +CI_{bp}(p, p_{refl}, p_{refr}) & \text{otherwise} \end{cases} \tag{8}$$

In Equation 9, η_1 and η_2 are the Fresnel factors for the reflected and refracted radiances and T_c is the transmittance color for refractive objects, which gets attenuated according to the distance the light travelled through the object. These translucent effects were implemented as described in [1]. $CI_r(x)$ is a helper function that checks whether point x is real or virtual and thus returns $CI(x)$ or zero. The data from the back-projection, however, is only used if the validation function $v(p, p_{refl}, p_{refr})$ returns *true* (see Section 4.1.3) – otherwise the L'_r buffer is used. Returning the L'_r buffer cancels out the substracted L'_r part from Equation 6 and thus only L'_{rv} contributes to the final image.

$$CI_{bp}(p, p_{refl}, p_{refr}) = \begin{cases} \eta_1 CI_r(p_{refl}) & \text{if } v(p, p_{refl}, p_{refr}) \\ +\eta_2 T_c CI_r(p_{refr}) & \\ L'_r & \text{otherwise} \end{cases} \tag{9}$$

$$CI_r(p) = \begin{cases} CI(p) & \text{if } r(p) \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Similar to Equation 2, an α factor is computed to compensate for the relative error between the CI and the L'_r buffers. However, for pixels belonging to reflective or refractive objects, the computation of α' must be altered as follows:

$$\alpha'(p, p_{refl}, p_{refr}) = \begin{cases} \frac{CI(p)}{L'_r} & \text{if } r(p) \wedge \\ & \quad r(p_{refl}) \wedge \\ & \quad r(p_{refr}) \\ \frac{CI_{bp}(p, p_{refl}, p_{refr})}{L'_r} & \text{otherwise} \end{cases} \tag{11}$$

Note that due to Equation 9, α' results to one if the points p, p_{refl} and p_{refr} are not valid.

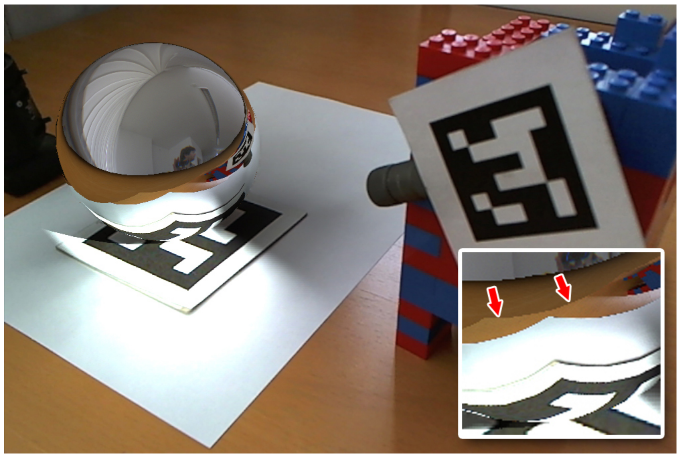

Fig. 3. The image shows a virtual reflective sphere illuminated by a virtual spot light. The see-through video image is also visible in the lower part of the sphere. At pixels where no information is available from the video image, the information from the L'_{rv} buffer is used. If there is also no information available in the L'_{rv} buffer then the image from the L_{env} buffer generated out of the fish-eye lens camera is used. The bottom right cut out shows a zoom-in on the sphere. Note the color change due to differences in the back-projected camera image and the L'_{rv} buffer.

4.1.3 Dealing with Missing Information

So far we have assumed that there is always enough data for the reflected and refracted rays available, meaning that the validation function $v(p, p_{refl}, p_{refr})$ always returns *true*. However, in common scenarios this is usually not the case, as shown in Figure 3. There are two stages where data might not be valid.

The first one is at the reflection and refraction stage (see Section 4.1.1). For each reflective or refractive object, there are two so-called geometry buffers (G-Buffers) that store all information necessary to shade a point. These G-buffers contain the surface information of the hit points of reflected or refracted rays (see Section 5.1). However, it might happen that a reflected or refracted ray does not hit any of the pre-modeled real or virtual geometry as illustrated by the leftward going ray in Figure 4. In such cases, no illumination computation is possible since the corresponding G-buffers cannot be filled. Our strategy here is to use the information available in the environment map from the fish-eye lens camera and write the color information into a separate buffer L_{env} at the corresponding screen space position that only contains color information from the environment map. The buffers L'_{rv} and L'_r are left untouched.

The second problem may arise at the compositing stage in Equation 9, where the ray hit positions of reflected or refracted rays are back-projected to look up the video see-through image. First, if the lookup position is inside the video image (as illustrated by rays number one and two in Figure 2), we have to check if the color information is actually what we are looking for. It is possible that another real object occludes the hit point in the video feed, as happens with ray two in Figure 2. Therefore the depths of the back-projected point and the depth stored in the primary G-Buffer are compared, and if they are within an epsilon range, the color information from the video image is assumed to be valid. However, if the looked up color information is not valid due to the depth test, the color information from the L'_{rv} buffer at the original (not back-projected) point p is used. Please note that the not back-projected pixel of the L'_{rv} buffer contains the reflected or refracted objects visible on the surface of the reflective or refractive object.

In case the back-projected look up is outside of the video image, the data from the L'_{rv} buffer is used. This case is illustrated by ray three in Figure 2. Furthermore note that artifacts may appear at the border line between the back-projected camera image data and the L'_{rv} buffer data if the colors differ too much (see zoom-in in Figure 3).

An exception is made for virtual refractive objects. In most cases the refractive component is far more important than the reflected one ($\eta_1 \ll \eta_2$) and only at the borders of an object, where the surface is seen from a very flat angle, the reflection component gets more important. However, according to the equations introduced in Section 4.1.2, the L'_{rv} buffer would be used if the reflected ray does not hit any object ($v(p, p_{refl}, p_{refr}) = false$). Therefore the validation function returns *true* if point p is virtual and p_{refr} is valid. In this way we can nicely refract the see-through video even though the reflected component is not valid. This leads to visually more pleasing results, however note that this may introduce wrong coloring information – a limitation of our approach.

4.2 Caustics in DIR

Reflective and refractive objects cause high-frequency caustic effects that cannot be approximated with instant-radiosity techniques within a reasonable amount of computation time. Therefore we integrated a faster method from Wyman and Davis [24] to compute the caustic illumination effects. In Section 4.2.1 we describe how caustics can be added to the GI solution buffers L_{rv} and L_r and Section 4.2.2 introduces reflected and refracted VPLs to simulate light paths with additional diffuse bounces, like *LSDDE* or *LSSDDE*.

4.2.1 GI Solution Buffer Computation

Caustic effects are integrated for real or virtual spotlight sources and, similar to reflective or refractive objects (see Section 4.1.1), different cases have to be handled to decide whether a caustic quad should be added to the L_r buffer or not:

$$L_{rv} = \sum_{ls} C_{quad}(ls) \qquad (12)$$

$$L_r = \sum_{ls} C_{quad}(ls) \cdot real(ls, p_{hit}, p) \qquad (13)$$

$$real(ls, p_{hit}, p) = \begin{cases} 1 & \text{if } r(ls) \wedge r(p_{hit}) \wedge r(p) \\ 0 & \text{otherwise} \end{cases} \qquad (14)$$

where C_{quad} is the caustic quad (see Section 5 for more details) on its hit position p_{hit}, emitted from surface p, and ls are all spotlight sources. The L_{rv} buffer is straightforward and does not need any special conditions. However, a caustic quad is only written into the L_r buffer if all light-interacting elements belong to real objects. Note that these quads are additively added to the L_{rv} and L_r buffers, which results in caustics, introducing *LSDE* and *LSSDE* light paths. Visible caustics in reflective or refractive objects are computed in a similar way, except that their contribution is added to the L'_{rv} and L'_r buffers and that the Fresnel factors and the transmittance color for refracted caustics is multiplied to them. To simulate *LSDDE* and *LSSDDE* paths, caustics need to emit additional light. At that point, VPLs can be used again.

4.2.2 Creating Reflected and Refracted VPLs

In differential instant radiosity, so-called primary light sources are used to create VPLs. One of them is the spotlight source, which behaves like standard spotlights, except that it creates so-called reflective shadow maps (RSM) [2]. In previous approaches, these RSMs are used to create virtual point lights at surface points illuminated by the spot light. However, since reflective or refractive objects cause high-frequency illumination effects, placing VPLs directly on their surface is not a good idea.

Instead, we reflect or refract the position of a VPL that should originally have been placed on a reflective or refractive object as shown in Figure 4. In this way, the VPLs are placed where caustics appear, and additional indirect light gets emitted, simulating reflective *LSDDE* and refractive *LSSDDE* light paths.

Although the final color of a reflective and refractive object is composed of the incident refracted and reflected radiance, only one light path can be used to create the additional VPL. This is a limitation due

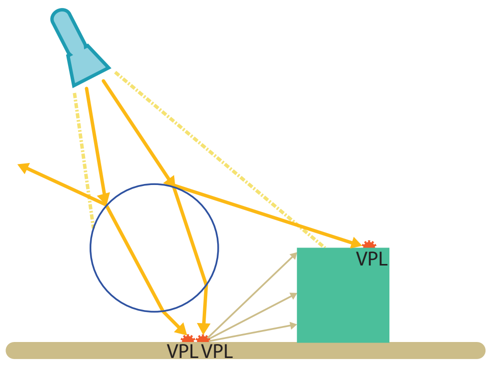

Fig. 4. This figure illustrates the placement of the reflected and refracted VPLs. Depending on whether ray hits can be found, either the reflected or refracted ray hit is used to place a VPL. This VPL then approximates indirect illumination due to caustics.

to the way VPLs are created on the GPU, since all VPLs are generated in parallel. Therefore, a decision has to be made whether to follow the reflected or the refracted ray. Furthermore, the availability of hit information must be taken into account (see Section 4.1.3). Although the real scene geometry needs to be pre-modeled for DR, not every reflected or refracted ray will hit this geometry, since commonly only nearby objects are pre-modeled. Therefore, if the reflected or refracted ray does not hit any surface, no valid data is available. Our method takes these cases into account by first checking if both rays have valid hit data available. If only one of them has valid data, then this data is used to create a VPL on the surface that the ray hits. In case of two valid hit points for the reflected and refracted ray, a quasirandom Halton value is used to choose the final VPL position. Both ray hits have the same probability, and no weighting is applied. Since each hit position is chosen every second frame on average, the VPL's intensity is doubled. Similar to [11], temporal smoothing of the illumination buffers L_{rv}, L_r, L'_{rv} and L'_r avoids flickering artifacts.

5 Implementation

5.1 Rendering Reflective and Refractive Objects

The differential instant radiosity method is based on a deferred rendering system. This means that a G-buffer is created from the point of view of the camera. The G-buffer provides all information necessary for later illumination computation. Using a G-buffer has the advantage that the illumination is only calculated for pixels which are visible to the camera. Since multiple G-buffers are introduced in this paper, we will call this G-buffer the primary G-buffer $GBuf_P$.

For reflective and refractive objects, this G-buffer cannot be used straight away, as the surface information on these objects does not provide the information necessary to calculate its final color. The final color consists of the incident radiance from the reflected and the refracted ray directions. Therefore, we create two additional G-buffers ($GBuf_{refl}$ and $GBuf_{refr}$). Compared to the primary G-Buffer ($GBuf_P$), these G-Buffers contain the geometry information from the reflective and refractive ray hits of a reflective or refractive object. Note that they are also rendered from the point of view of the camera. They are created using the methods from Popescu et al. [16] and Wyman [22], with the difference that they store the geometric information from the ray hits.

In the next step, the additional G-Buffers are used to calculate the illumination from all primary light sources and all VPLs. The result is added to the L'_{rv} and L'_r buffers according to the equations mentioned in Sections 4.1.1 and 4.1.2. Note that the resulting incident radiance is pre-multiplied according to Fresnel's law.

5.2 Rendering Caustics

To render caustics, two so-called photon G-Buffers ($PGBuf_{refl}$ and $PGBuf_{refr}$) are created for each spotlight. In contrast to the previous

G-Buffers, these G-Buffers are rendered from the point of view of the spotlight. Similar to Shah et al.[17], we count the number of pixels to estimate the photon intensities. Note that the final photon intensity furthermore depends on the spotlight characteristics, the distance to the light source itself, the attenuation and the transmittance color for refracted photons. Once $PGBuf_{refl}$ and $PGBuf_{refr}$ are created, they can be utilized to splat photon quads in the screen space into a so-called caustic buffer $CBuf$. The photon's quad size is calculated similar to Wyman [23]. The caustic buffer stores the photon intensity, number of photon hits, the depth and average incident direction of the photons. With the caustic buffer $CBuf$ and the primary G-Buffer $GBuf_P$, the contribution to the L_{rv} and L_r buffers is computed according to Section 4.2.1. Since caustics should also be visible in reflections and refractions (L'_{rv} and L'_r buffers), the caustic buffer $CBuf$ and the G-buffers $GBuf_{refl}$ and $GBuf_{refr}$ are used to compute the reflected and refracted caustics.

6 Limitations and Future Work

The proposed method inherits some limitations from the differential instant radiosity method. These are double shadowing artifacts, inconsistent color bleeding and the need to pre-model the real environment. These limitations are solved in [12] for the basic DIR method, but left for future work for this method. While double shadowing artifacts should be straight forward to implement as proposed in [12], inconsistent color bleeding introduces challenges for VPL placement in case of reflection or refraction. Currently real caustics which are visible in the see-through video image can only be canceled out if there is no occluding virtual object between the caustic emitting object and the hit point of the caustic. In Figure 7(a) at the table in front of the yellow cube, the real caustic is still visible. Furthermore, the illumination computation time on the G-Buffers for reflections and refractions could be improved by applying the method proposed by Nichols and Wyman [13].

Currently, the system does not support multiple reflections and refractions between several reflective or refractive objects. This could be done with the method proposed by Umenhoffer et al. [20]. However, note that multiple reflections and refractions add complexity to the computation equations due to multiple ray splits on reflective and refractive objects.

Another minor limitation is that VPLs do not cause any caustics for now, only spot lights are able to generate them. In the future, the rendering quality of the caustics could be further improved by implementing the method from Wyman and Nichols [25].

7 Results

For the results shown in this paper, we used a PC with an Intel Core2 Quad CPU at 2.8Ghz, 8GB of memory, and an NVIDIA Geforce GTX 580 with 1.5GB of video memory. All results were rendered at a resolution of 1024x768 pixels. A uEye camera from IDS with a fish-eye lens was used to acquire the environment map, and a Logitech HD Pro 910 webcam was used for the video see-through input.

For all test scenarios, we used a total of 256 VPLs with an imperfect shadow-map size of 128x128 pixels, using a total of 1024 points per VPL to represent the scene. Both the caustic maps for the pocket light source and the impostor texture maps had a resolution of 512x512.

In Figures 5 and 6, a spot light illuminates a virtual Stanford bunny that has green-colored glass. The light travels through the Stanford bunny and introduces green caustics on the white box behind it. Figure 5 shows caustics which do not introduce additional indirect illumination. In contrast, Figure 6 shows one added indirect light bounce by placing VPLs at the appearance of caustics. These green caustics then introduce indirect color bleeding on the white paper and on the desk. This image was rendered at 13 frames per second (polygon count: 69680).

Figures 7(a) – (d) show a comparison between real and virtual caustic illumination effects. A small pocket light illuminates a metallic pocket bottle from the right side, causing a caustic on the real table and on the virtual yellow box. In the top row 7(a) and 7(b), the pocket

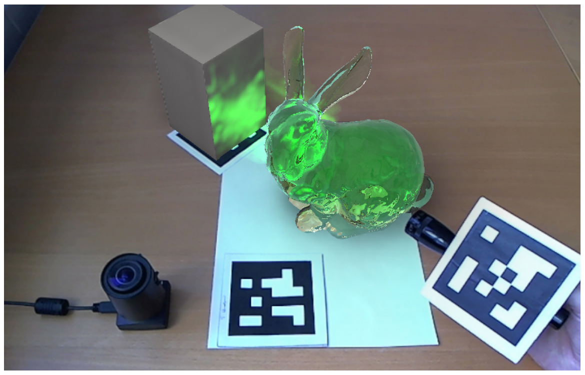
Fig. 5. In this figure, a spotlight creates caustics on a white box without any indirect illumination effects.

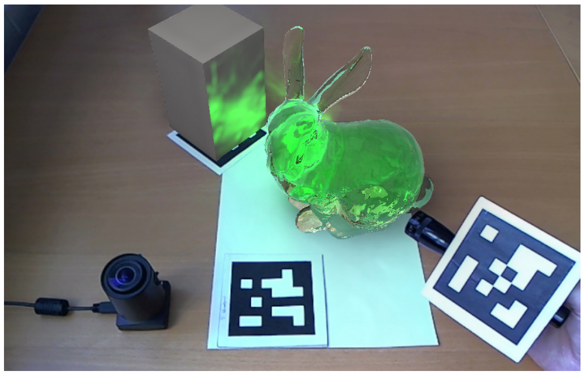
Fig. 6. In this figure, a spotlight creates caustics on the white box. VPLs placed on the caustics cause additional indirect illumination on the white paper.

bottle is a real object, whereas in the bottom row 7(c) and 7(d), a virtual pocket bottle is placed into the scene. In the left column 7(a) and 7(c), a real pocket lamp is used, and in the right column 7(b) and 7(d), a virtual one (real pocket lamp is switched off). Please note how the yellow box is reflected in all images on the pocket bottle. All images were rendered at 20 frames per second (polygon count: 3028).

8 Conclusion

In this paper, we presented an extension to the differential instant radiosity method that tightly integrates reflective and refractive objects. The original differential rendering method was not able to apply the differential effects to reflected or refracted objects. The proposed method adds differential effects to those objects and back-projects color information from the see-through video image if possible. In this way, we are able to simulate reflective and refractive objects in mixed-reality scenarios.

Furthermore, spotlights are able to generate caustics from those reflective or refractive objects. By using a fast splatting method, the resulting caustics have a higher quality than caustics produced with virtual point lights. In addition, these caustics emit light at their hit points, causing one additional indirect light bounce. With the proposed method, we are able to achieve 15 to 25 frames per second.

Acknowledgments

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.

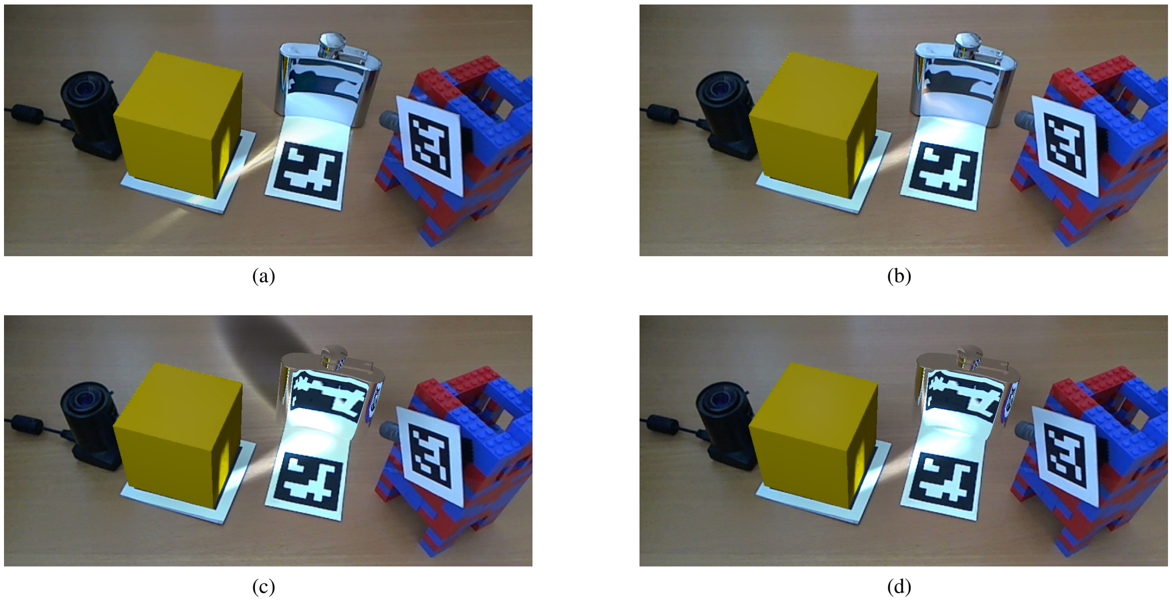

Fig. 7. (a) Real spotlight illuminates real pocket bottle. (b) Virtual spotlight illuminates real pocket bottle. (c) Real spotlight illuminates virtual pocket bottle. (d) Virtual spot light illuminates virtual pocket bottle.

References

[1] T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008.

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

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

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

[5] T. Grosch. Differential photon mapping: Consistent augmentation of photographs with correction of all light paths. In *Eurographics 2005 Short Papers*, pages 53–56, Trinity College, Dublin, Ireland, 2005. Eurographics Association.

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

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

[8] P. Kán and H. Kaufmann. High-quality reflections, refractions, and caustics in augmented reality and their contribution to visual coherence. In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR)*. ACM Press, 2012.

[9] K. Karsch, V. Hedau, D. Forsyth, and D. Hoiem. Rendering synthetic objects into legacy photographs. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, pages 157:1–157:12, New York, NY, USA, 2011. ACM.

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

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

[12] M. Knecht, C. Traxler, O. Mattausch, and M. Wimmer. Reciprocal shading for mixed reality. *Computers & Graphics*, 36(7):846–856, Nov. 2012.

[13] G. Nichols and C. Wyman. Interactive indirect illumination using adaptive multiresolution splatting. *IEEE Transactions on Visualization and Computer Graphics*, 16(5):729–741, Sept. 2010.

[14] S. A. Pessoa, G. de S. Moura, J. P. S. M. Lima, V. Teichrieb, and J. Kelner. Photorealistic rendering for augmented reality: A global illumination and brdf solution. In *VR'10*, pages 3–10, 2010.

[15] S. Pirk. Gpu-based rendering of reflective and refractive objects in augmented reality environments. Master's thesis, University of Applied Sciences, Oldenburg, Germany, 2007.

[16] V. Popescu, C. Mei, J. Dauble, and E. Sacks. Reflected-scene impostors for realistic reflections at interactive rates. *Computer Graphics Forum*, 25(3):313–322, 2006.

[17] M. A. Shah, J. Konttinen, and S. Pattanaik. Caustics mapping: An image-space technique for real-time caustics. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):272–280, Mar. 2007.

[18] A. State, G. Hirota, D. T. Chen, W. F. Garrett, and M. A. Livingston. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 429–438, New York, NY, USA, 1996. ACM.

[19] T. Tawara and K. Ono. An application of photo realistic water surface interaction using mixed reality. In H. Prautzsch, A. A. Schmitt, J. Bender, and M. Teschner, editors, *VRIPHYS*, pages 59–65. Eurographics Association, 2009.

[20] T. Umenhoffer, G. Patow, and L. Szirmay-Kalos. Robust multiple specular reflections and refractions. In H. Nguyen, editor, *GPU Gems 3*, pages 387–407. Addison-Wesley, 2008.

[21] Y. Uranishi, A. Ihara, H. Sasaki, Y. Manabe, and K. Chihara. Real-time representation of inter-reflection for cubic marker. In *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '09, pages 217–218, Washington, DC, USA, 2009. IEEE Computer Society.

[22] C. Wyman. Interactive image-space refraction of nearby geometry. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '05, pages 205–211, New York, NY, USA, 2005. ACM.

[23] C. Wyman. Hierarchical caustic maps. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, I3D '08, pages 163–171, New York, NY, USA, 2008. ACM.

[24] C. Wyman and S. Davis. Interactive image-space techniques for approximating caustics. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, I3D '06, pages 153–160, New York, NY, USA, 2006. ACM.

[25] C. Wyman and G. Nichols. Adaptive caustic maps using deferred shading. *Computer Graphics Forum*, 28(2):309–318, 2009.