High-Quality Reflections, Refractions, and Caustics in Augmented Reality and their Contribution to Visual Coherence

P. Kán* H. Kaufmann†

Institute of Software Technology and Interactive Systems,
Vienna University of Technology, Vienna, Austria

ABSTRACT

In this paper we present a novel high-quality rendering system for Augmented Reality (AR). We study ray-tracing based rendering techniques in AR with the goal of achieving real-time performance and improving visual quality as well as visual coherence between real and virtual objects in a final composited image. A number of realistic and physically correct rendering effects are demonstrated, that have not been presented in real-time AR environments before. Examples are high-quality specular effects such as caustics, refraction, reflection, together with a depth of field effect and anti-aliasing.

We present a new GPU implementation of photon mapping and its application for the calculation of caustics in environments where real and virtual objects are combined. The composited image is produced on-the-fly without the need of any preprocessing step. A main contribution of our work is the achievement of interactive rendering speed for high-quality ray-tracing algorithms in AR setups.

Finally we performed an evaluation to study how users perceive visual quality and visual coherence with different realistic rendering effects. The results of our user study show that in 40.1% cases users mistakenly judged virtual objects as real ones. Moreover we show that high-quality rendering positively affects the perceived visual coherence.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems —Artificial, augmented, and virtual realities; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism —Raytracing

1 INTRODUCTION

By definition Augmented Reality (AR) is the combination of real and virtual to augment the real world in order to improve people's senses and skills. A requirement in most AR applications is visual coherence between real and virtual objects. Visual coherence may provide precise information about spatial location, radiometric and geometric properties of inserted virtual objects. Furthermore realistic appearance of virtual objects and their proper interaction with the real world is of high interest in many applications areas e.g. entertainment (movie production, previsualisation), design, medicine (therapy, rehabilitation, surgery), education and others.

The majority of AR applications use simple rendering and shading algorithms, thus achieving only a low level of visual coherence. A simplified illumination model and the lack of lighting interchange between virtual objects and the real world together with the pinhole camera model are insufficient for achieving effects created by a physical lens. They cause a low degree of visual coherence between virtual and real objects. A complete light interaction simulation between real and virtual objects involves global illumination (GI) calculation. Calculation of accurate GI in AR in interactive frame rates is a challenging task. Especially high-quality global light simulation on specular surfaces in interactive AR environments is a difficult problem.

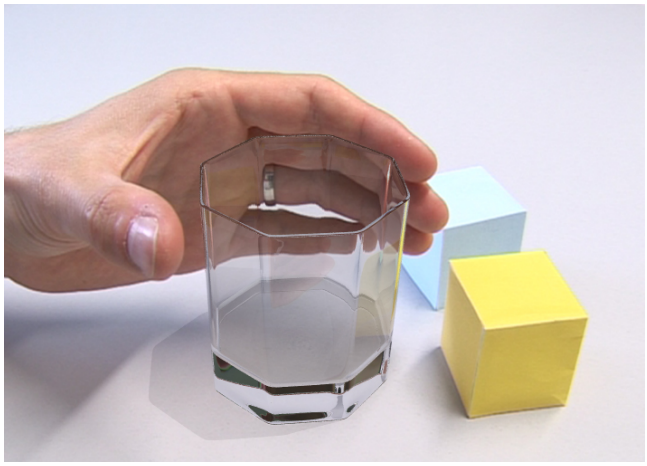

Figure 1: Refractive virtual glass surrounded by the real environment. Note the correct refraction of the hand in the glass obtained by the reprojection method.

In this paper we propose a novel high-quality rendering system for AR using ray-tracing based algorithms which achieves a high degree of visual realism and visual coherence. For proper simulation of light interreflections between real and virtual worlds we use a new interactive GPU implementation of photon mapping [15]. We use a physically-based camera model to achieve coherent visual effects like those caused by a real camera. One-pass differential rendering [16] is utilized to increase rendering performance. We demonstrate high-quality interactive rendering of various realistic effects such as correct light reflection, refraction, anti-aliasing, depth of field (DoF) and caustics. We have implemented realistic glass material rendering (Figure 1) by ray-tracing using Fresnel reflection [24] and proper refraction. To overcome the problem of getting incoming radiance of real light refracted in glass objects, we use a camera reprojection method similar to [9] thus finding the correct radiance in the camera image. In order to specify the correct light source positions we incorporate a light source estimation algorithm in our system. In this algorithm the environment map is processed by thresholding and blob detection and the positions of the light sources are found. Furthermore the real environment map is used to render proper reflection/refraction of the real world in specular virtual objects. In our implementation we exploit the parallel nature of ray-tracing algorithms and the massive parallel power of modern GPUs. Therefore our solution provides from interactive to real-time frame rates depending on the required quality.

In order to study the impact of ray-tracing based rendering on visual coherence in AR, we evaluated users' perception of our ren-

*e-mail: peterkan@peterkan.com
†e-mail: kaufmann@ims.tuwien.ac.at

dering solution using quantitative methods. Visual coherence was evaluated in two different scenarios. Furthermore we asked participants to state which objects in the shown video are real and which are virtual, in order to see if the visual quality and coherence is good enough to make people believe that AR content is real. The results of the user study show that our system is able to achieve a very high degree of visual coherence between real and virtual objects.

The main contributions of our work are:

- New high-quality ray-tracing based rendering and compositing system for interactive AR providing realistic rendering effects.
- Interactive photon mapping on the GPU for caustics rendering in augmented reality introducing global light transport between virtual and real objects.
- Real-time physically based rendering of specular reflection and refraction in AR.
- User studies confirming the positive impact of ray-tracing based rendering in AR to visual coherence.

The advantage of our method over previous work is that we can very naturally render specular surfaces like glass or mirrors in high quality. Moreover we propose a method for calculating caustics in AR in interactive frame rates. Caustics can be created by reflecting or refracting light on both real and virtual specular objects. Our rendering and compositing method runs on-the-fly and no preprocessing is needed while high rendering quality is achieved.

The rest of the paper is organized as follows. In section 2 previous research is discussed. Section 3 describes the main algorithms that we developed in detail: simulation of ray-tracing based refraction and reflection, rendering of caustics by photon mapping and the light source estimation technique. Moreover the used image reprojection technique and details about density estimation are discussed there. Section 4 details the implementation of all proposed algorithms. In section 5 the evaluation of the system is presented. User study examining the influence of ray-tracing based rendering on the human perception is described and results are discussed. Furthermore performance measurements are summarized.

2 Related Work

In this section we give an overview of previous research that focused on high-quality rendering in AR, comparing it to our approach.

High-quality rendering in AR The majority of techniques that aim to achieve high rendering quality in AR use rasterization-based rendering. Usually they focus on special effects like material simulation, self-shadowing simulation by ambient occlusion, or diffuse global illumination and color bleeding. These approaches usually do not simulate the light paths reflected from specular surfaces and they use a high amount of approximation in order to achieve high rendering speed.

Believable material simulation can increase the amount of realism in AR. Pessoa et al. [23] created an approach for photorealistic rendering in AR using rasterization. They proposed an extended version of Lafortune Spatial BRDF model to properly simulate material properties. Additionally they used the Irradiance Environment Mapping [29] in order to simulate global illumination coming from the real world to virtual objects. Hover their solution cannot simulate global illumination coming from virtual objects to real ones. Furthermore a static environment image was used in their solution and therefore no dynamic movement of reflected real objects or lighting change could be simulated.

Diffuse global illumination in real-time augmented reality was proposed by Knecht at al. [18] in their Differential Instant Radiosity approach. They extended Imperfect Shadow Maps [30] to work in AR and used single-pass differential rendering to create the final composite image. Their system is capable of simulating diffuse global illumination including color bleeding between real and virtual objects at interactive frame rates. However they do not simulate the specular light paths that create caustics and light reflection or refraction. Another approach simulating high-quality diffuse global illumination in AR was proposed by Grosch et al. [10]. Authors used Spherical Harmonics arranged in a grid to simulate diffuse light transport. Moreover a correct near-field illumination was calculated using the predefined model of surrounding geometry (Room) in combination with a fish-eye lens camera. They achieved real-time frame rates, however their solution does not simulate specular effects like caustics, refraction or reflection. An approximation of self-shadowing in AR calculated by Ambient Occlusion was used by Franke and Jung [7]. They proposed a material reconstruction technique using genetic algorithms. Multipass rendering for AR was proposed by Agusanto et al. [1]. They used irradiance environment maps to simulate global illumination coming from real light to virtual objects. However their system did not simulate light reflected from virtual objects affecting the real scene.

An advantage of our system in comparison to a rasterization-based AR rendering systems is that we can calculate specular effects like reflection, refraction and caustics in high-quality, while specular effects are very difficult to calculate in rasterization-based rendering. We can also simulate the lighting change on real objects caused by inserting virtual ones. Moreover ray-tracing rendering systems can usually provide higher quality of the final result.

AR systems usually use the simulation of real lighting in rendering. In some systems an image-based lighting approach is used [18, 3] by sampling the environment map according to the probability density function (pdf) similar to the intensities of the pixels. Other approaches use light reconstruction to precisely specify the light positions and intensities. Frahm et al. [6] used an image processing approach to obtain information about light sources. They search for regions with high saturation in all channels and then apply a segmentation. They approximate the real light by point light sources and calculate their positions by processing two images from two fish-eye lens cameras. High-quality light reconstruction from a single-camera image is proposed in [17]. Authors used a user guided algorithm, which processes the light sources marked by the user to find the optimal positions, shapes and sizes of them.

A composited image of real and virtual objects in augmented reality is usually created using differential rendering which was originally proposed by Fournier et al. [5] and later extended by Debevec [3]. This algorithm requires two solutions of lighting simulation, doubling the rendering time. To overcome this problem we use our one-pass differential rendering algorithm proposed in [16].

A good overview of reflection and refraction simulation in AR can be found in [26]. The author describes rasterization-based techniques. However ray-tracing can provide physically-based simulation of refractive and reflective material while rasterization based techniques provide solutions at lower quality.

Ray-tracing in AR Physically-based algorithms developed in computer graphics often use ray-tracing to achieve accurate light transport calculation. However the problem of high-quality ray-tracing based rendering systems is performance. A good image can take time to render. Offline rendering systems for mixing virtual objects to simulate full global illumination were proposed in [17, 3]. Important research for specular effects simulation in AR was done by Grosch [9]. He proposed Differential Photon Mapping in order to simulate caustics and shadows more efficiently. Moreover he proposed an image reprojection technique to obtain real radiance coming through refractive surfaces. His system achieves high rendering quality, however it runs offline and not in real-time. Advantage of our method over differential photon mapping is that we achieve interactive to real-time frame rates and our system is capa-

ble to simulate more visual features, for example physically-based depth of field.

The use of the ray-tracing in AR applications was examined by Scheer et al. [31] who used it for realistic rendering and Pomi et al. [27] who studied the insertion of the real video of characters into a virtual environment in TV Studio applications. The disadvantage of Pomi's implementation was the requirement of a PC cluster to achieve interactive frame rates.

Photon mapping Photon mapping proposed by Jensen [15] is an effective two pass algorithm for the calculation of global illumination. It traces photons from the light source into the scene. When a photon hits a surface it is either reflected, refracted, or absorbed. The photons are stored at their hit positions and later sorted into a Kd-tree data structure called photon map. In the second step the radiance information is reconstructed from the photon map in the points visible from the camera. The Kd-tree is traversed in order to find the photons contributing to a specified location. Photon mapping can simulate full global illumination and it is especially efficient in specular global illumination effects like caustics, which are difficult for other GI algorithms.

Since photon mapping was invented many improvements have been published. Several solutions for fast photon mapping were proposed. An interactive solution was created by Fabianowski and Dingliana [4], who generated the footprints of all photon hits and stored them in the improved BVH optimized for fast search. A very efficient photon mapping implementation combining GPU rasterization and CPU ray-tracing was proposed by McGuire and Luebke [20]. They used photon volumes rasterization in order to avoid costly KNN queries. An efficient GPU-based approach for full global illumination calculation was proposed by Wang et al. [39]. The authors used the GPU Kd-tree construction proposed by Zhou [40] and they implemented the whole global illumination calculation only on the GPU, achieving interactive frame rates. Purcell et al. [28] proposed another GPU implementation of photon mapping using shader programs. A grid-based photon map was used in their approach. Gupte [12] proposed an interactive photon mapping implementation using the OptiX ray-tracing engine that we use as well. The main difference compared to our method is that they used a Spatial Hashing Method instead of the Kd-tree for storing and searching photons. An interactive photon mapping implementation running on multiple computers is shown in [11].

A special case of the photon mapping algorithm reformulation is shown in splatting approaches. Usually the energy carried by photons is splat to the specified buffer and the photon's contribution and weight are added to already stored values in the area of contribution. A scalable photon splatting algorithm was proposed in [19] and photon ray splatting was proposed in [13].

Another kind of algorithmic improvement can be achieved by increasing the quality of the produced result. A high-quality photon mapping improvement was proposed by Spencer and Jones [35]. The authors focused mainly on the density estimation part of caustics rendering. They proposed the novel method of relaxing the initial distribution into one with a blue noise spectral signature. This improvement enables the use of low bandwidth kernels and increases efficiency of the rendering. The same authors also proposed Hierarchical Photon Mapping [34] to increase the quality of the final gathering.

3 High Quality Rendering System for AR

In order to solve the problem of visual coherence between virtual and real objects, a high-quality composition of the final image is required. As shown in previous research, some realistic rendering effects can be achieved using the rasterization pipeline on the GPU. However it is still difficult to render high-quality specular effects like realistic refraction, or caustics. The natural way of rendering more complex lighting effects is by using ray-tracing, which is also used in an extended form for physically based rendering. With the recent development of graphics hardware, ray-tracing becomes available for use in real-time applications. We propose the adaptation and application of ray-tracing based algorithms into AR in order to achieve visually appealing rendering results.

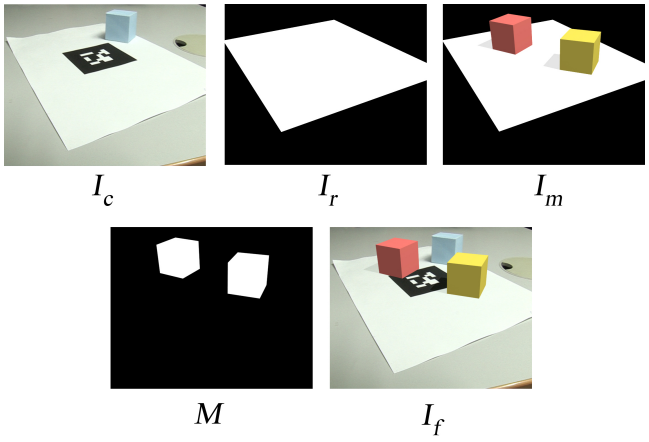

Figure 2: The images obtained with differential rendering. I_c is the video camera image, I_r is the rendering of the real scene, I_m is the rendering of the mixed scene, M is the mask of virtual objects and I_f is the final result.

3.1 Ray-tracing

In our research we developed a new ray-tracing based rendering system for AR scenarios. The core of our system is based on one-pass differential rendering proposed in [16]. This technique is capable to render the final composited image in a single ray-tracing step instead of running two separate ray-tracing solutions, which are required in the standard differential rendering. In order to calculate mixed and real radiances together four ray types are used: the mixed radiance ray, the real radiance ray, the mixed shadow ray, and the real shadow ray. The final image is then obtained by combining the mixed and the real radiance results together in a ray-generation program using the differential rendering equation [17]:

$$I_f = M \odot I_m + (1 - M) \odot (I_c + I_m - I_r) \quad (1)$$

where I_f is the final composited image, M stands for the average of the mask values of the rays shot through pixels. I_m is the rendered result of both real and virtual objects (mixed radiance), while I_r is the rendering result of only real objects (real radiance). The I_c is the source image captured by the camera. Images used in the differential rendering are shown in figure 2. In the one-pass differential rendering I_r and I_m are produced together.

Specular refraction and reflection Ray-tracing allows us to properly simulate reflection and refraction on specular surfaces. When a ray hits a reflective surface, the reflected ray direction is calculated and a new ray is shot. When a ray hits a refractive surface, both reflective and refractive rays are shot in order to properly simulate the behaviour described by Fresnel equations [24]. We use Schlick's approximation of the Fresnel term described by the following equation [32, 37]:

$$F(\theta) = F_\perp + (1 - F_\perp)(1 - \cos\theta)^5 \quad (2)$$

where θ is the angle between the incident ray direction and the surface normal, and $F_\perp$ is the Fresnel term at perpendicular direction. The refractive ray direction is calculated according to Snell's law [24].

Figure 3: The resulting image produced by our AR rendering system. The image was rendered shooting 9 rays per pixel to obtain a high-quality result and to reduce aliasing artifacts. 1M photons were shot in the photon shooting phase. The scene was rendered at 1fps, and the rendering rate can achieve 15 fps by decreasing the number of samples per pixel. There is one virtual glass monkey casting caustics onto real objects and three real cubes in the image.

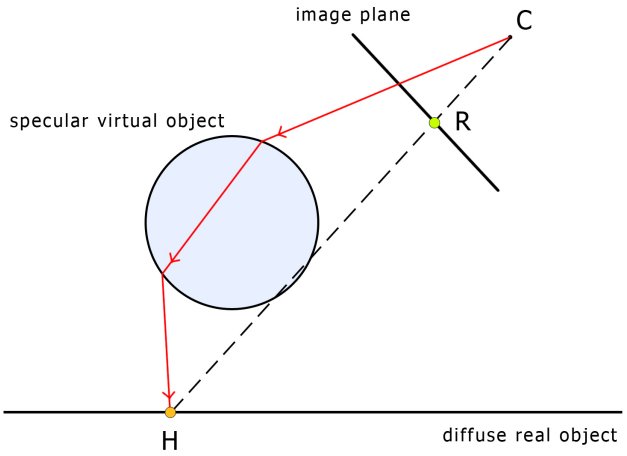

Figure 4: The reprojection method for obtaining the outgoing radiance from the diffuse real surface seen through the refractive virtual object. Red rays are the rays sent from the camera to calculate the radiance coming from a certain direction. Point C is the center of projection. Point H is the hitpoint of the refracted ray with the diffuse real surface. In order to retrieve the outgoing radiance from point H, it is reprojected to the image plane to the point R.

A problematic situation arises if a refracted ray hits a real surface; because if the pure result of the rendering is used, information about the refracted real world image is missing. We solve this problem by using the image reprojection method similar to [9]. The per-ray-data structure contains a flag *wasSpecular*. If the ray hits a specular object, this flag is set to true. If diffuse real geometry is hit by a mixed radiance ray, *wasSpecular* is checked. If the flag is set to true the diffuse object was hit after the specular reflection/refraction. In this case we need to use the outgoing radiance from the real object. In the video image obtained from the camera the measured radiances per sensor pixel are stored. If we assume a diffuse surface, the outgoing radiance is the same for every outgoing direction. This fact allows us to use the radiance measured by the camera as the outgoing radiance from the real diffuse surface to the virtual refractive one. To obtain the correct measured radiance, we reproject the hitpoint of the diffuse surface onto the image plane and calculate the reprojected point position in image space coordinates. We can then directly access the video image, which was previously sent to GPU memory as a texture. If the hitpoint contains glossy material, the radiance obtained by the reprojection is still a good approximation of outgoing radiance. The reprojection method is depicted in the figure 4 and the result of the refractive material rendering is shown in the figures 1, and 3.

In order to properly display the reflected/refracted environment in reflective specular objects, we use the hemispherical environment map obtained by the fish-eye camera. We approximate the surrounding environment as a distant light coming from infinity when reflected or refracted radiance is calculated. This assumption allows us to access the environment texture according to the ray direction. The environment texture is used as incoming radiance only in cases when no real or virtual geometry has been hit. The texture is accessed directly in the miss program in the ray-tracing pipeline. We use asynchronous image capturing independent from the rendering thread and then update the image synchronously sending it as a texture to the GPU. Usually the rays pointing to the lower hemisphere hit the surface of the scene, however it can happen that these rays also miss any geometry. In this case we reuse the captured environment image of the upper hemisphere in the lower hemisphere. This mirroring provides visually acceptable results of the radiance from missed rays.

We use a physically-based camera model with finite-sized aperture described in [16]. This model enables a high-quality depth of field effect rendering in augmented reality. Another advantage of using ray-tracing in augmented reality is that it can naturally simulate specular reflection also on real specular surfaces. Moreover caustics can be reflected on those surfaces. We can see the reflection of the virtual object and its caustic reflected on the real mirror surface in figure 5.

Anti-aliasing As we show in our evaluation, an important feature for visual realism in AR is the anti-aliasing of rendered objects. It reduces artifacts caused by insufficient sampling density in high-frequency parts of the image function, such as discontinuities on the edges of virtual geometry. Aliasing artifacts can immediately tell users that objects are virtual and therefore decrease the overall realism of the composited video. Distributed Ray-tracing [2] offers a very elegant and natural method for anti-aliasing by just supersampling the pixel area. Supersampling appears to be a suitable method since various random variables can be distributed over multiple rays shot per pixel. For example it can be used to sample the 2D domain of the pixel area together with 2D aperture to get the DoF effect [16]. We use stratified jittered sampling [21] to achieve a good distribution of samples.

In order to reduce aliasing we increase the number of rays shot through each pixel. The final pixel color is then calculated by filtering the calculated irradiance values obtained by shooting the rays. In the compositing equation the reduction of aliasing on edges of virtual objects can be achieved by filtering the mask value as well. The mask can have a value between 0 and 1 and controls the blending of virtual and real objects. Moreover this blending strategy can also be used with the DoF calculation, where blurred edges of out-of-focus objects should be blended with the real background.

3.2 Caustics

Important visual features that increase the amount of visual realism are caustics. They are created by light reflecting from specular surfaces to diffuse surfaces and then to the camera. In order to create high-quality caustics, a photon mapping [15] algorithm is usually used. We created a new GPU implementation of photon mapping

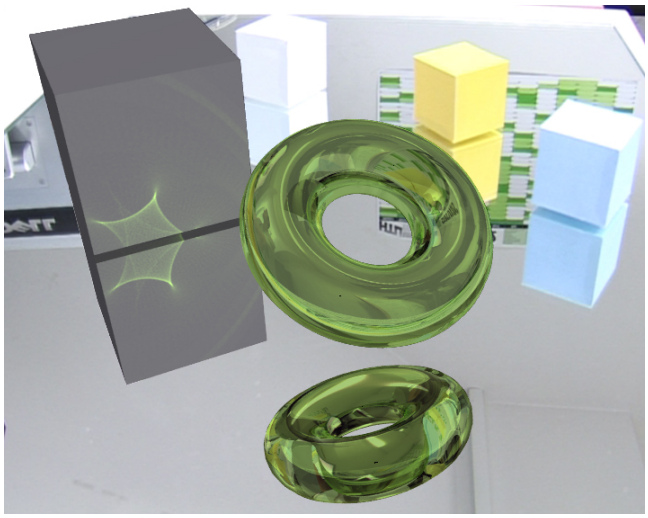

Figure 5: Caustics can be created by both virtual and real specular objects. The green caustic was created by the virtual torus. Part of it was created by the reflection of light from the real mirror. The diffuse cube on which the caustic is drawn is also virtual. Note the correct reflection of the generated caustic in the real mirror and also the correct refraction in the torus.

using the OptiX ray-tracing engine [22] in order to achieve interactive frame rates while keeping quality of the created caustics high (Figure 6).

In our implementation we use a two-pass caustic generation algorithm. In the first pass photons are emitted from the light source into the scene. When photons hit a specular virtual surface they are reflected or refracted in the direction of specular reflection or refraction. If a photon hits a diffuse surface after reflection from a specular object, it is recorded in an array of photons. This array is later processed on the CPU, and a Kd-tree is created to allow faster search for near-by photons. In the next step rays are traced from the camera through the image plane to obtain the radiance incoming from the scene. If a ray hits a surface in the scene, direct illumination is calculated and indirect caustic illumination is reconstructed from the photon map (Figure 3).

In order to reconstruct indirect illumination at a certain point of the scene from point samples that are stored in the photon map, density estimation techniques are applied. There are three main approaches for density estimation: using a histogram, nearest neighbour search or kernel density estimation [33, 36].

The K-nearest neighbour (KNN) search is the method that is often used in combination with photon maps. This technique reduces the variance while keeping the bias low, however a high number of K has to be used in order to obtain accurate results. Because of many samples required, the KNN search is often a bottleneck of radiance estimation from photon maps.

We use a kernel method to estimate illumination based on the photon map, which allows us to perform a fast calculation of visually correct results. With kernel methods there is always a tradeoff between bias and noise. A standard kernel method estimates the probability density function (pdf) $p(x)$ given N samples x_i by the equation [33, 36]:

$$\hat{p}(t) = \frac{1}{Nh^d} \sum_{i=1}^{N} \mathcal{K}(\frac{t - x_i}{h}) \quad (3)$$

$\mathcal{K}$ is a kernel function, h is the kernel bandwidth, d is the dimension of the domain of p, and t is the position of estimation. The accuracy of the kernel density estimation technique depends on shape and bandwidth of a kernel. A kernel bandwidth selection is an important step. If the kernel is too wide, more bias is produced and if it is too narrow, more variance can be observed. Kernel estimation techniques with adaptive bandwidth were proposed in previous work [36, 13]. They use a different bandwidth for every sample according to its correctness, previously estimated density, or the number of surrounding samples. Those techniques are often iterative and require additional computational time to find a good bandwidth.

We decided to use density estimation with a fixed kernel size. This approach can potentionally produce bias and blur the discontinuities in caustics. However, we solved this problem by selecting a narrow kernel width. The variance is then reduced by increasing the number of shot photons. Using a fixed kernel size with a narrow kernel enables very fast photon search in a Kd-tree as well as fast density estimation. We use the Epanechnikov kernel, which is a standard in density estimation [36]. A comparison of rendering with and without caustics can be seen in figure 8.

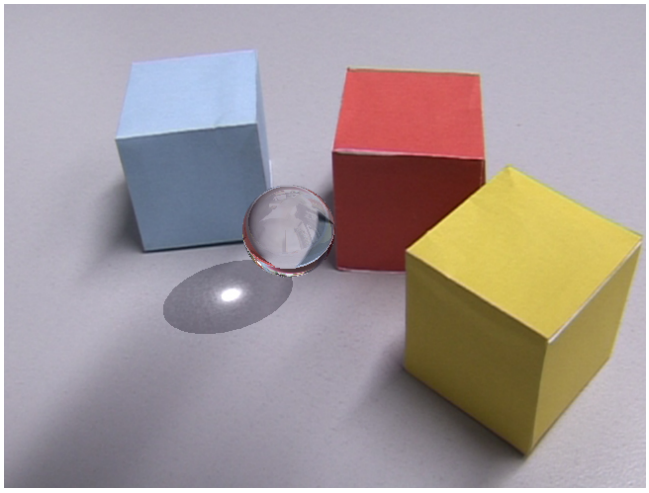

Figure 6: Interactive caustic rendering. There is a virtual glass sphere inserted into the real scene. The image was rendered at 10 fps.

A problem when emitting photons for caustics creation is the selection of good directions in which a specular surface will be hit. Jensen proposed a solution to rasterize specular objects to hemisphere to obtain only directions where they are seen from the light position [15]. Another approach is to approximate the geometry of specular surfaces in a form of bounding volumes and to shoot rays into the resulting primitives. In our system specular objects are approximated by bounding spheres for the purpose of photon shooting. In the scene loading phase all geometry is traversed with the goal of finding caustic generators. If a caustic generator is found, a bounding sphere, including all geometry of this object, is created. In the photon shooting program the center and the radius of each bounding sphere is used in the following way. First, one of the caustic generators is randomly selected according to the projected area of the bounding sphere. Then a disk perpendicular to the direction from the center of the object to the light source is sampled using stratified jittered sampling. A direction of photon is then calculated as the direction from light source position to the sampled point. By this sampling process some photons may miss a caustic generator, however a high numbers of hitpoints are achieved.

3.3 Light Source Estimation

We are using a fish-eye camera to capture the light situation in the real environment which allows us to estimate the positions and in-

tensities of the light source(s). Generally two different approaches have been reported in literature. The first approach builds a cumulative distribution function from the environment image, allowing for random sampling from the probability distribution, which equals the intensity of the incoming light from different directions on the hemisphere [24].

The second approach is to apply image processing techniques to the environment image and extract the positions of light sources. This approach was used for example in [6].

We use image processing in our system, because random environment light sampling requires a lot of samples and creates an additional overhead. In our implementation first thresholding is applied to the captured environment image. In the next step we use blob detection on the binary image to detect the biggest sources of high incoming radiance. Connected component analysis provided by OpenCV is utilized here and a contour tracing approach [8] is used to find the contours of radiance blobs. The area of every blob is calculated and the biggest blobs are selected as light sources. The direction of incoming light is estimated according to the blob center position in the environment image by reprojecting it to the hemisphere. The exact position is then estimated with the user supplied average room size constant and the light is positioned in that distance in the reconstructed direction.

An arbitrary number of light sources can be extracted and bigger area light sources can be sampled by more point light sources. We usually use single point light source extraction in our experiments. Point light sources produce sharper caustics than area light sources do.

We run environment image capturing and light source estimation asynchronously in a separate thread. Rendering speed is then almost independent of the light source estimation and the last calculated values are always read.

4 Implementation

We implemented our AR rendering system by using the OptiX ray-tracing engine [22]. It is a powerful tool for running ray-tracing based algorithms on modern parallel GPUs. To composite the final image with the real video captured by the camera we use one-pass differential rendering directly in the ray-generation program. The photon shooting pass is also implemented using OptiX.

The main steps of our rendering and compositing method are depicted in figure 7. For every frame the images from video camera and fish-eye camera are captured and sent to the GPU. The environment image is processed and light source positions are found. A visual marker is detected in the video image and position and orientation of the camera are estimated. The first GPU step is photon mapping. Photons are then processed on the CPU and a Kd-tree is created. The final rendering step is GPU ray-tracing from the camera position. All calculated results together with geometry and materials of real and virtual scenes are used here. Density estimation is performed on every diffuse surface hit in the ray-tracing step to estimate the indirect illumination from the photon map. The result of the rendering is composited with the real video image and directly displayed on the output device.

A computer with hexa-core CPU and a GeForce GTX 590 graphics card was used for high-speed rendering. The upper hemisphere of the scene's illumination was captured with an environment camera with a fish-eye lens. The captured image is used as a distant source of real illumination for reflected/refracted light and as a point light source for direct light calculation. This allows us to reuse the environment map in every point of the scene. Using a light source as a distant light in reflection and refraction completely omits the spatial variation, however it is a very good approximation for the directional variation in the scene. Therefore the environment image is accessed with the ray direction in the miss program.

Our main camera to capture images and augment them with virtual objects in real time is the Sony HVR-Z1E. This camera has a lens that allows big aperture size and is therefore suitable for the creation of a good depth of field effect by the optics of the real lens. The camera parameters can then be used in the rendering system to simulate physically correct depth of field for the rendering of virtual objects.

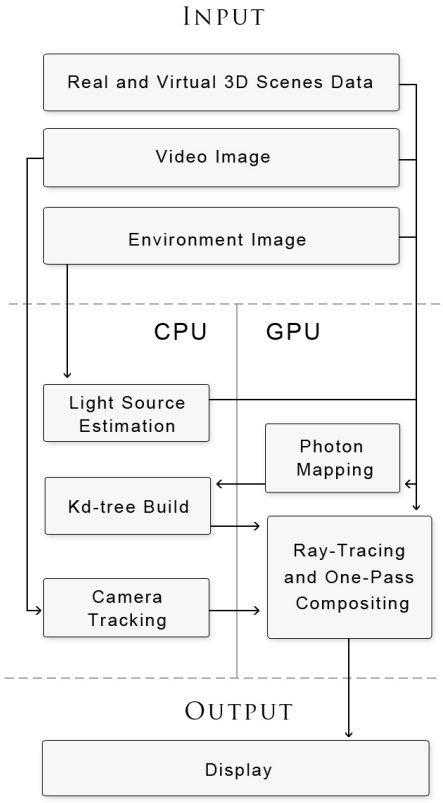

Figure 7: Overview of our rendering and compositing method. Arrows indicate the data flow between different components.

To track the camera's position and orientation, the ARToolkitPlus [38] marker-based tracking system was used. The actual camera image is utilized both for the camera pose calculation and for rendering. By reconstructing the camera position and orientation directly out of the camera image that is also used for rendering, no synchronization problems between tracking and rendering appear. The orientation of the marker determines the rotation of the world coordinate space and therefore determines where virtual objects are positioned. The environment image retrieved by the fish-eye camera is orientation-dependent, therefore the fish-eye camera has to be aligned with the marker. Proper alignment ensures correct reflections and refractions on specular virtual surfaces.

Reconstructed geometry and materials of the real scene are required in order to properly create the composited image by differential rendering. We use a predefined model of the real scene for this purpose.

5 Evaluation and Results

In order to evaluate the impact of interactive ray-tracing on visual coherence in AR we designed and performed a user study. Quantitative methods were used to evaluate our hypotheses. Our hypotheses were:

- Realistic features of ray-tracing based rendering have a positive impact on the user's perception of visual coherence in augmented reality.
- Every realistic rendering effect used in our system positively influences the realistic appearance of the composited video.

5.1 Study Design

At the beginning of the user study we showed a single video of a table containing diffuse real cubes, diffuse virtual cubes, a refractive virtual glass sphere and a reflective virtual ring. All effects that we describe in this paper (simulation of refraction/reflection, anti-aliasing, caustics, and DoF) were enabled in this video. We asked users to rate how realistic the video was on a linear visual analogue scale from -3 to 3. The value of -3 means that the video is not realistic at all and the value of 3 means that the video is completely realistic. Moreover we asked people to indicate which objects were real and which were virtual.

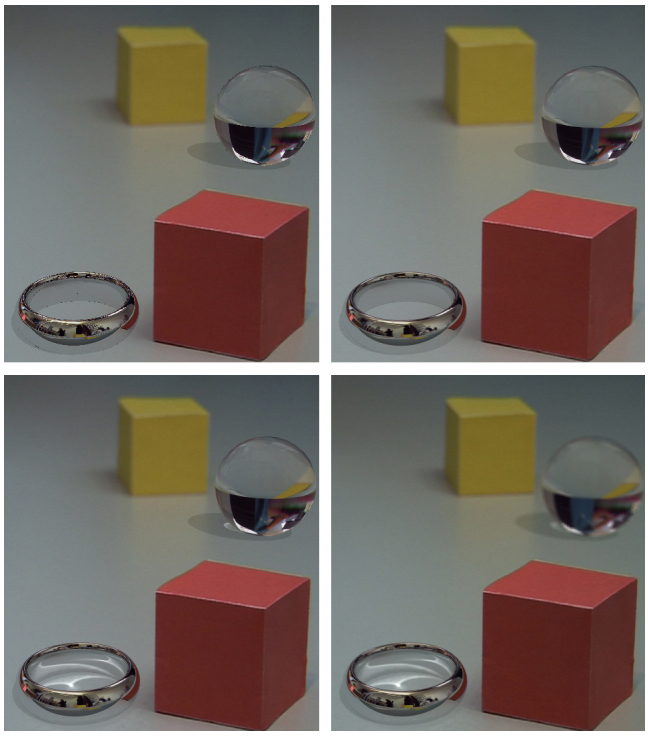

Figure 8: Comparison of rendering with different features. The scene contains a real red and yellow cube, a virtual refractive sphere and a virtual metal ring. (Top left) Rendering refraction and reflection using 1 ray per pixel and no caustic simulation at 27 fps. (Top right) Anti-aliasing added using 25 rays per pixel - 7 fps. (Bottom left) Caustics are enabled using 150K photons. Rendering speed is 3 fps. (Bottom right) Depth of Field effect is enabled. Frame rate is 2.5 fps. Differences in images can better be seen in closeup.

In the next stage of the user study we used sequences consisting of five videos each. The same scene was shown in each clip but in each successive video in a sequence a new realistic feature was added. Figure 8 is an example of enabling different effects one after another. In order to show that the evaluation results are independent of the scene layout and scene content, two different scenes were used. In the first scene a virtual refractive glass sphere on a real table with real diffuse cubes was rendered. In a second scene a virtual metal ring with diffuse real cubes was shown. Each video, including the first general video, was played to users only once. Two sequences of videos were shown to every user in order to avoid the dependence of answers on order of effects and rendered scenes. Each sequence introduced the effects in a different order. Therefore the order was randomized for all participants in a standardized way. In total four different sequences were created and two of them were selected for every user. This selection made sequences randomized to avoid the bias in results.

In the first sequence of videos the realistic effects were added in the following order: no effect, refraction/reflection, anti-aliasing, caustics, depth of field. The virtual refractive sphere was rendered in this sequence.

In the second sequence the effects were added in a different order: no effect, caustics, refraction/reflection, anti-aliasing, depth of field. The virtual metal ring was rendered in this sequence.

In a third sequence the effects were added in the same order as in sequence 1, however the metal ring scene was used. And finally in the fourth sequence the effects were added in the same order as in sequence 2 with a refractive glass sphere used.

Users were asked to compare each pair of successive videos in the sequence. They had to answer the question: Which video in the pair (previous, current) looks more realistic? Users evaluated the increase of realism in successive videos by using a linear visual analogue scale in range from -3 to 3. -3 means that the previous video was much more realistic. 0 has the meaning of a similar, comparable amount of realism in both videos. And 3 means the current video is much more realistic. The decision to evaluate the difference between two successive clips was made because we wanted users to concentrate on desired rendering effect and to evaluate the increase of realism when this effect is enabled.

5.2 Results

The participants of the user study were selected randomly. 43 users participated in our evaluation. 12 of them were men and 31 were women. 26% of the test subjects had previous knowledge in computer graphics and 74% did not. The general video showing all effects was shown to all participants. The first and second video sequences were shown to 21 users; the third and fourth sequences were shown to the other 22 users.

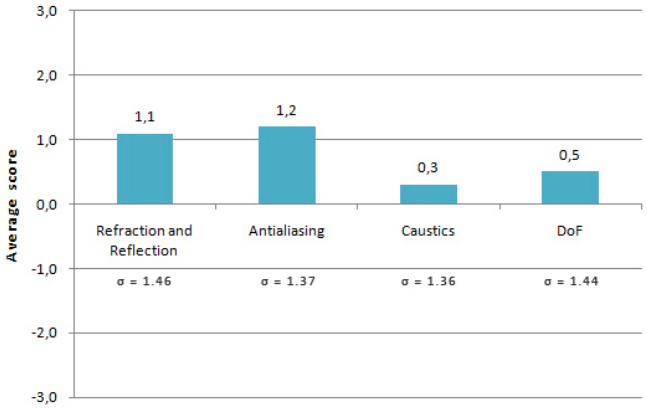

Figure 9: The influence of single realistic rendering effects on perceived realism in AR videos. (σ = std. dev.).

First we evaluated the increase of realism caused by enabling certain effects. Figure 9 shows the results of pair-wise comparison of all effects. The addition of each effect contributed positively to the perception of realism in AR. We can see that the highest contribution is caused by anti-aliasing. It reduces disturbing alias on the edges of virtual objects, making them appear more natural. Realistic refraction and reflection is also an important feature to enhance realism. We can see that caustics are not so important for users,

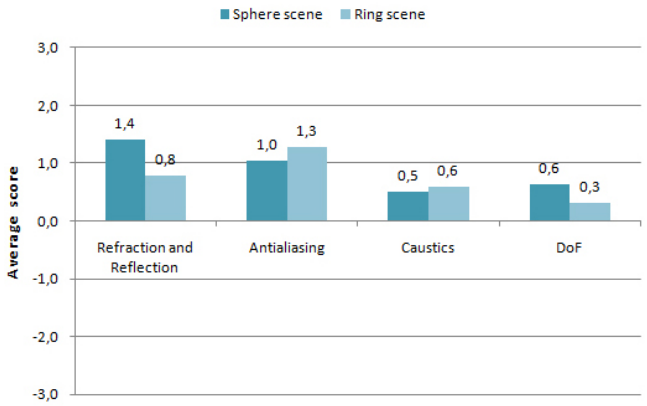

Figure 10: Difference in perceived realism with single realistic effects using different virtual content.

but also contribute to increase the amount of realism in AR. However, the result on caustics could be influenced by the fact that in the second sequence caustics were added to an object before reflection/refraction. This made caustics look unnatural because the respective objects were rendered just semitransparently. In summary our second hypothesis was confirmed because the results indicate that all added rendering features increased the amount of realism in AR. Influence of virtual content to realism perceived by users in video sequences 1 and 3 can be seen in figure 10.

We were interested to see how users perceived realism of the overall video. The results can be seen in figure 11. The average of 0.8 indicates that the scene was perceived as realistic and the result confirms our first hypothesis. However, there is room for improvement in future work. In addition we analyzed the perception of realism separately for selected groups within our participants. We analyzed the results depending on gender and computer graphics experience. The results are shown in figure 11. Men with CG knowledge report higher values of realism.

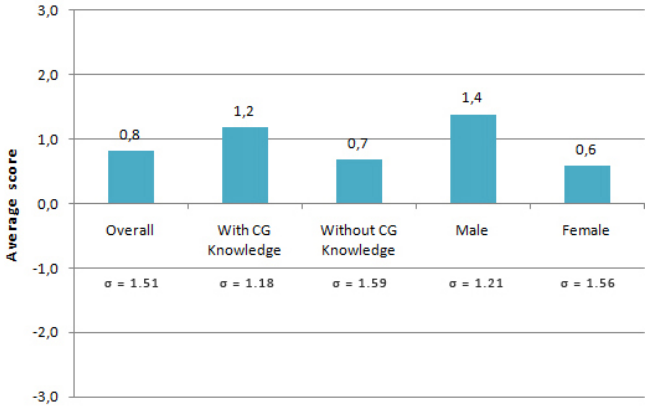

Figure 11: Average results of perceived realism of the overall AR video. (σ = std. dev.).

Another result was more surprising. We asked users to judge which objects in the overall video are real and which are virtual. The intention of this question was to see if the visual quality and coherence is good enough to make people believe that AR content is real. One goal is to make virtual objects indistinguishable from real ones. In average 40.1% of the virtual objects were mistakenly marked as real ones. The most realistic appearance according to this evaluation had the virtual metal ring reflecting the real environment and casting caustics. 53.5% of users marked this virtual object as a real one. It is important to say that in 36% of all cases users mistakenly thought that real objects were virtual ones.

In summary the evaluation demonstrates that each developed rendering effect has a positive impact to visual coherence in AR. The results of enabling different effects can be seen in figure 8. In the top left image of the figure, the rendering of specular refraction and reflection is enabled. However in this image only 1 ray per pixel was shot and we can observe aliasing artifacts. The top right image shows how aliasing artifacts disappear when supersampling is enabled. In the bottom images caustics rendering was added. Note the small caustic created under the glass sphere. In the lower left image the real yellow cube is blurred by DoF of the real lens. Therefore the virtual sphere is visually incoherent because it is sharp. The DoF effect is enabled in the bottom right image.

The second part of our evaluation concerned rendering performance. We measured the rendering speed of different effects and at different sampling rates. As we can see in table 1 our system is capable to produce a high-quality result in interactive frame rates. The table also shows the computation times for different steps of rendering: photon emitting, Kd-tree build, and ray-tracing with density estimation. Texture copying to the GPU memory usually took 20 ms in our experiments. Our system is capable to preserve interactivity even with complex scenes like the glass happy Budha consisting of 855K triangles. Increasing the sampling rate improves the quality of the produced result, however as we see in table 1 it decreases rendering speed, thus offering a tradeoff between quality and speed.

6 Conclusion and Future Work

In this paper we present a novel high-quality rendering and compositing system for augmented reality. Our system is able to render various realistic effects by using ray-tracing at interactive or real-time frame rates. A novelty of our work was achieved by using interactive ray-tracing in AR on a single PC while simulating specular effects like refraction, reflection and caustics. Our method is capable to produce believable interactive augmentations. Moreover we designed and performed a user study confirming the positive impact of realistic rendering on user's perception of visual coherence.

There are several possible extensions of the proposed system which can increase the amount of realism in AR. We plan to incorporate full global illumination into rendering in future work to allow effects like color bleeding and diffuse indirect lighting to be produced. Possible performance improvements can be achieved by using a GPU implementation of Kd-tree construction [40] for photon mapping. Accurate camera tracking is an important part of an AR system, because even little jitter in tracking data can be immediately observed by users as unnatural and unstable behavior of virtual objects. We plan to extend our system using a more stable tracking solution like the ioTracker [25]. Automatic 3D reconstruction of the real scene would be of high benefit in our system. A possible solution to this problem can be KinectFusion [14] which can provide both 3D reconstruction of the real scene and a stable tracking solution.

In our work we demonstrate the possibility of using high-quality ray-tracing based rendering techniques in interactive AR and we expect GPU ray-tracing to be the future of rendering in AR.

Acknowledgements

We thank the anonymous reviewers for their valuable comments and suggestions. We would like to thank to Iana Podkosova who helped us with reformulation of some parts of the paper. This work was supported in part by the PhD school of informatics at TU Vienna.

Scene	Triangle count	Emitted photons	Primary rays per pixel	t_p	t_b	t_r	fps_r	fps_n
Metal Ring	9K	280K	1	45 ms	196 ms	21 ms	4	**20**
		280K	4	47 ms	196 ms	94 ms	3.3	**10**
		280K	9	47 ms	196 ms	206 ms	2.7	**5.7**
		No caustics	1	0 ms	0 ms	6 ms	-	**31**
Glass Monkey	15K	100K	1	20 ms	31 ms	24 ms	11	**20**
		1M	1	167 ms	619 ms	45 ms	1.6	**17**
		1M	4	167 ms	618 ms	138 ms	1.4	**6**
Glass Dragon	201K	240K	1	28 ms	47 ms	36 ms	10	**17**
		240K	9	29 ms	48 ms	269 ms	2.8	**3.6**
Glass Happy Budha	855K	240K	1	16 ms	17 ms	42 ms	7.5	**14**
		240K	4	16 ms	17 ms	165 ms	4.1	**6.5**

Table 1: Performance of our ray-tracing based rendering system. t_p is the duration of the photon shooting phase in ms, t_b is the time for Kd-tree building, and t_r is the time of ray-tracing from the camera including the density estimation on hitpoints. fps_r is the frame rate with photon map rebuild enabled and fps_n is the frame rate with photon map rebuild disabled. All measurements were taken at a resolution of 720x576. A GeForce 590 GTX graphics card was used to render all measured scenes.

REFERENCES

[1] K. Agusanto, L. Li, Z. Chuangui, and N. W. Sing. Photorealistic rendering for augmented reality using environment illumination. In *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '03, pages 208–218, Washington, DC, USA, 2003. IEEE Computer Society.

[2] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '84, pages 137–145, New York, NY, USA, 1984. ACM.

[3] 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.

[4] B. Fabianowski and J. Dingliana. Interactive global photon mapping. *Computer Graphics Forum*, 28(4):1151–1159, 2009.

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

[6] J. Frahm, K. Koeser, D. Grest, and R. Koch. Markerless augmented reality with light source estimation for direct illumination. In *Visual Media Production, 2005. CVMP 2005. The 2nd IEE European Conference on*, pages 211 – 220, nov. - 1 dec. 2005.

[7] T. Franke and Y. Jung. Real-time mixed reality with gpu techniques. In *GRAPP*, pages 249–252, 2008.

[8] C. Grana, D. Borghesani, and R. Cucchiara. Connected component labeling techniques on modern architectures. In *Proceedings of the 15th International Conference on Image Analysis and Processing*, ICIAP '09, pages 816–824, Berlin, Heidelberg, 2009. Springer-Verlag.

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

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

[11] J. Günther, I. Wald, and P. Slusallek. Realtime caustics using distributed photon mapping. In *Rendering Techniques*, pages 111–121, June 2004. (Proceedings of the 15th Eurographics Symposium on Rendering).

[12] S. Gupte. Real-time photon mapping on gpu. 2011.

[13] R. Herzog, V. Havran, S. Kinuwaki, K. Myszkowski, and H.-P. Seidel. Global illumination using photon ray splatting. In D. Cohen-Or and P. Slavik, editors, *Computer Graphics Forum (Proceedings of Eurographics)*, volume 26(3), pages 503–513, Prague, Czech Republic, 2007. Blackwell.

[14] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. 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, pages 559–568, New York, NY, USA, 2011. ACM.

[15] H. Jensen. *Realistic Image Synthesis Using Photon Mapping*. Ak Peters Series. A K Peters, Limited, 2009.

[16] P. Kán and H. Kaufmann. Physically-based depth of field in augmented reality. In *EG 2012*, Cagliari, Italy, 2012. Eurographics Association.

[17] 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.

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

[19] F. Lavignotte and M. Paulin. Scalable photon splatting for global illumination. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '03, pages 203–ff, New York, NY, USA, 2003. ACM.

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

[21] D. P. Mitchell. Consequences of stratified sampling in graphics. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 277–280, New York, NY, USA, 1996. ACM.

[22] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich. Optix: a general purpose ray tracing engine. *ACM Trans. Graph.*, 29:66:1–66:13, July 2010.

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

[24] M. Pharr and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann. Elsevier Science, 2010.

[25] T. Pintaric and H. Kaufmann. Affordable Infrared-Optical Pose Tracking for Virtual and Augmented Reality. In *IEEE VR Workshop on Trends and Issues in Tracking for Virtual Environments*, pages 44–51, 2007.

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

[27] A. Pomi and P. Slusallek. Interactive Ray Tracing for Virtual TV Studio Applications. *Journal of Virtual Reality and Broadcasting*, 2(1),

Dec. 2005.

[28] T. J. Purcell, C. Donner, M. Cammarano, H. W. Jensen, and P. Hanrahan. Photon mapping on programmable graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, HWWS '03, pages 41–50, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[29] R. Ramamoorthi and P. Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 497–500, New York, NY, USA, 2001. ACM.

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

[31] F. Scheer, O. Abert, and S. Müller. Towards using realistic ray tracing in augmented reality applications with natural lighting. *GI Workshop ARVR 07*, 2007.

[32] C. Schlick. An inexpensive brdf model for physically-based rendering. *Computer Graphics Forum*, 13:233–246, 1994.

[33] B. W. Silverman. *Density estimation for statistics and data analysis*. Chapmann and Hall, New York, 1986.

[34] B. Spencer and M. W. Jones. Hierarchical photon mapping. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):49–61, Jan-Feb 2009.

[35] B. Spencer and M. W. Jones. Into the blue: Better caustics through photon relaxation. *Comput. Graph. Forum*, 28(2):319–328, 2009.

[36] F. Suykens and Y. D. Willems. Adaptive Filtering for Progressive Monte Carlo Image Rendering. In *WSCG*, 2000.

[37] L. Szirmay-Kalos, L. Szécsi, and M. Sbert. *GPU-Based Techniques for Global Illumination Effects*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers, 2008.

[38] D. Wagner and D. Schmalstieg. ARToolKitPlus for Pose Tracking on Mobile Devices. Technical report, Institute for Computer Graphics and Vision, Graz University of Technology, Feb. 2007.

[39] R. Wang, R. Wang, K. Zhou, M. Pan, and H. Bao. An efficient gpu-based approach for interactive global illumination. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 91:1–91:8, New York, NY, USA, 2009. ACM.

[40] K. Zhou, Q. Hou, R. Wang, and B. Guo. Real-time kd-tree construction on graphics hardware. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 126:1–126:11, New York, NY, USA, 2008. ACM.