Differential Irradiance Caching for Fast High-Quality Light Transport Between Virtual and Real Worlds

Peter Kán* Hannes Kaufmann†

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

ABSTRACT

Fast and realistic synthesis of real videos with computer generated content has been a challenging problem in computer graphics. It involves computationally expensive light transport calculations. We present a novel and efficient algorithm for diffuse light transport calculation between virtual and real worlds called Differential Irradiance Caching. Our algorithm produces a high-quality result while preserving interactivity and allowing dynamic geometry, materials, lighting, and camera movement. The problem of expensive differential irradiance evaluation is solved by exploiting the spatial coherence in indirect illumination using irradiance caching. We enable multiple bounces of global illumination by using Monte Carlo integration in GPU ray-tracing to evaluate differential irradiance at irradiance cache records in one pass. The combination of ray-tracing and rasterization is used in an extended irradiance cache splatting algorithm to provide a fast GPU-based solution of indirect illumination. Limited information stored in the irradiance splat buffer causes errors for pixels on edges in case of depth of field rendering. We propose a solution to this problem using a reprojection technique to access the irradiance splat buffer. A novel cache miss detection technique is introduced which allows for a linear irradiance cache data structure. We demonstrate the integration of differential irradiance caching into a rendering framework for Mixed Reality applications capable of simulating complex global illumination effects.

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

1 INTRODUCTION

Accurate light transport simulation in real-time applications has been a topic of research for many years. It involves the calculation of Global Illumination (GI), which is computationally expensive. Rendering synthetic objects into real scenes usually doubles the effort since two solutions of light transport are needed [7, 5]. Therefore high-quality rendering techniques have not been suitable for real-time applications, and were used for offline production rendering only. However many real-time applications can benefit from accurate rendering of synthetic objects into real scenes. Visual coherence between real and virtual worlds is an important property of Mixed Reality (MR), and specifically Augmented Reality, because it can enhance users perception of spatial relationships, radiometric and geometric properties of inserted virtual objects. Furthermore realistic appearance of virtual objects and the proper light interaction with the real world is of high interest in many application areas

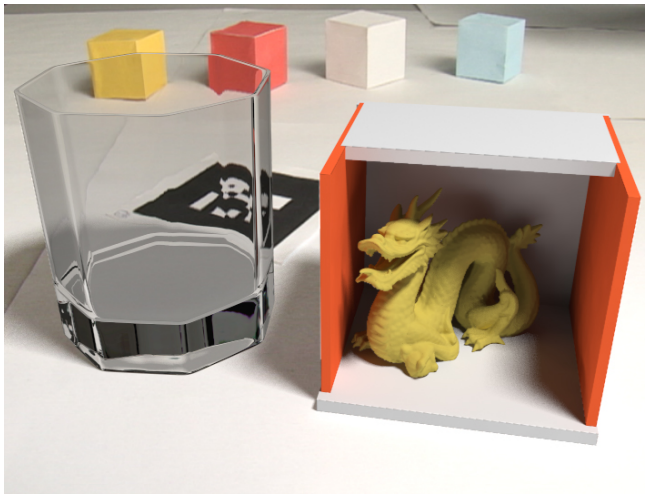

Figure 1: Virtual objects are placed into a real scene. Rendering is performed at interactive frame rates using differential irradiance caching in combination with ray-tracing. The virtual glass correctly refracts the real cubes behind it. Diffuse indirect illumination on the virtual box with the dragon is calculated by the differential irradiance caching algorithm.

e.g. entertainment, design, medicine (therapy, rehabilitation, preoperative visualization), education and others. Fast and high-quality light transport between real and virtual worlds can be especially beneficial for movie pre-visualization and cinematic relighting.

We propose a method utilizing Irradiance Caching (IC) [38] often used for high-quality production rendering in an offline process. Our novel algorithm based on irradiance caching and Differential Rendering [7, 5] runs at interactive to real-time frame rates and produces a high-quality result of diffuse light transport in mixed reality scenes. Our algorithm utilizes the massive parallel power of GPUs and combines ray-tracing and rasterization to achieve high-quality results while preserving interactivity. The proposed method takes a frame of a real-time video stream and superimposes virtual geometry onto it, introducing light interreflections between real and virtual worlds on-the-fly (Figures 1, 2). Direct illumination is calculated by ray-tracing from the camera. The indirect light is calculated by random hemisphere sampling using ray-tracing at a sparse set of points (cache records) in the scene. Rasterization is used to interpolate the indirect light between cache records by the irradiance cache splatting algorithm [10]. The limitation of single-bounce global illumination in irradiance cache splatting is overcome by using recursive path-tracing to evaluate the irradiance at the locations of cache records. This naturally enables multi-bounce global illumination and the irradiance in cache records is evaluated in an unbiased fashion. The problem of splatting approaches is the limited information stored in screen space. No data is available when informa-

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

tion from other locations is required, for example when computing depth of field or refraction, which require information behind the objects. We solve this problem by using a reprojection technique to access the indirect illumination splat buffer in the ray-tracing pipeline. This allows for a high-quality depth of field effect (Figure 5) and screen space information reuse on refractive and reflective surfaces. Our differential irradiance cache is a linear data structure, which fits well to the memory capabilities of modern GPU hardware. We avoid any search in the irradiance cache by using irradiance cache splatting for rendering and a novel miss detection technique to find places of new irradiance cache records. The importance of HDR rendering was highlighted by Debevec [5], therefore our system calculates the result in HDR and finally tonemaps it for display on standard display devices.

The main contributions of our work are:

- A novel algorithm for high-quality diffuse light transport in mixed reality using a combination of GPU ray-tracing and rasterization. The irradiance in cache records is evaluated using Monte Carlo integration by shooting recursive rays into random directions above the surface point. This method naturally enables the calculation of multiple bounces of indirect illumination.

- A reprojection technique which enables the screen space information generated by splatting approaches to be reused in arbitrary scene points. The reprojection technique is especially useful when calculating depth of field, refraction and reflection with ray-tracing.

- A novel miss detection method using a linear irradiance cache data structure. The irradiance splat buffer is traversed and points with missing indirect illumination are searched. New irradiance cache records are initialized and positioned where required. A greedy approach is used and the first position of a missing irradiance value is used.

- A global illumination framework for mixed reality capable of rendering complex GI effects including diffuse light transport, refractive/reflective media, depth of field, and caustics created by interactive photon mapping. This framework is using the combination of ray-tracing and rasterization to reach high-quality output at interactive frame rates.

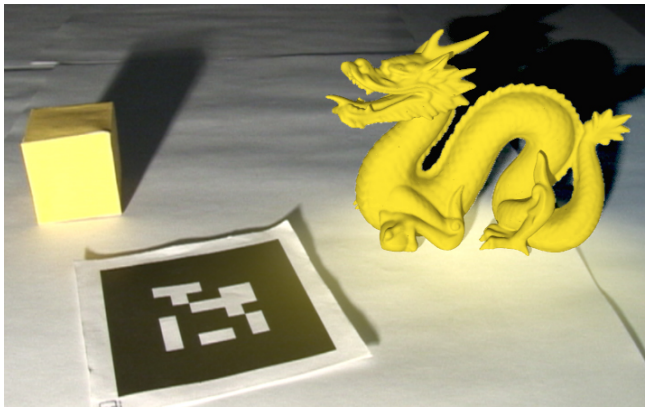

Figure 2: Indirect light transport between real and virtual worlds. The virtual dragon causes color bleeding on the real table similarly to the real cube.

2 Related Work

Visual coherence between virtual and real worlds is an important problem. Previous works proposed offline computationally expensive solutions. These solutions can produce a final image in high-quality with the cost of offline processing. Recent research introduced fast approximative solutions running interactively, but suffering from quality limitations. This section describes previous work in the area of light transport in MR, irradiance caching, and real-time ray-tracing in MR.

Global Illumination in Mixed Reality An important work bridging virtual and real objects in mixed scenes was presented by Debevec [5]. In his work global light transport in a mixed scene was addressed in a high-quality approach. He highlighted the importance of HDR rendering for visual realism, and used spherical light probes to capture real light and render the virtual objects with natural lighting. This concept was later followed by Grosch [12], who proposed Differential Photon Mapping to increase the efficiency of light transport calculation and to enable specific effects like refractions and caustics. Recent work for high-quality augmentations of virtual objects into legacy photographs was introduced by Karsch et al. [17]. The authors used a user-driven approach for light estimation and quick scene modeling. Then they used offline light transport to calculate the final mixed image. These methods are capable of creating high-quality final results, but do not run in real-time. Interactivity is often required by mixed reality applications. Our solution runs at interactive frame rates while providing a high-quality final result. A tradeoff between quality and speed is achieved.

Real-time approximative solutions for light transport in MR were proposed in recent research. Usually GPU rasterization is used in these approaches and real-time frame rates are achieved with the cost of introducing error. The work of Knecht et al. demonstrates diffuse global illumination in mixed reality by using Differential Instant Radiosity [18]. The authors use the Imperfect Shadow Maps [31] algorithm and situate it into the context of differential rendering [7]. This approach achieves real-time performance, but is limited to diffuse global illumination. Recently Lensing and Broll [24] proposed a method for fast indirect illumination calculation in mixed scenes with automatic 3D scene reconstruction. They used the Reflective Shadow Maps [4] algorithm for indirect illumination calculation. The disadvantages of both approaches are that they introduce approximation errors and are limited to diffuse global illumination. Our approach can be extended by radiance caching [20] to calculate both diffuse and glossy reflections. In our solution the error is introduced by interpolation between irradiance samples, and it can be reduced by decreasing the allowed interpolation error a. The illumination in irradiance samples is physically correct and unbiased. A real environment map for image-based lighting of virtual objects was used in [27, 1, 28]. The disadvantage of those techniques is that they ignore the visibility in irradiance calculation and therefore introduce a big amount of approximation error. Franke and Jung [8] used ambient occlusion to approximate visibility in irradiance calculation. In our solution we calculate interreflections between virtual and real worlds in both ways. We solve the visibility problem accurately by ray-tracing.

Grosch et al. [13] proposed to use an Irradiance Volume [11] to precalculate indirect illumination inside the scene for different directions of direct illumination. They stored the irradiance in Spherical Harmonics. High numbers of irradiance volumes were calculated for possible directions of incoming direct light. Indirect light could then be rendered dynamically in real-time depending on incoming direct light. The authors correctly simulated near-field illumination. The disadvantage of their method is that a vast precomputation of irradiance volumes is required and big objects have to be static to not change the irradiance. The advantage of our method is that no precomputation is required and we support dynamic geometry, materials and lighting.

Irradiance Caching Since its inception in 1988, Irradiance Caching proposed by Ward [38] has become a widely used algorithm for computing global illumination in production rendering. Numerous improvements were proposed in research, which made irradiance caching practical and efficient for high-quality offline rendering. The algorithm calculates the accurate irradiance in sparse scene locations and then exploits spatial coherence by interpolating between the samples. It is based on the assumption that indirect irradiance tends to change slowly over a surface. The contribution of an irradiance sample to an arbitrary scene point is determined by the weight w_i. We use the weight function proposed by Tabellion and Lamorlette [36]. The quality of irradiance interpolation can be improved by using irradiance gradients proposed in [37] and [22]. Křivánek et al. proposed the improvement of irradiance interpolation by introducing Neighbor Clamping [19]. The authors clamp the average distance to nearby geometry R_i according to the R_i values of neighboring cache records. Arikan et al. [2] proposed a method decompositing the radiance field into near-field and far-field components, that can be approximated separately. Improvements of the error metric for IC, to allow for efficient placement of cache records, were proposed in [14, 34].

The sampling of irradiance records is done in screen space and is therefore dependent on the camera pose. This dependence can cause temporal flickering in animations because new samples are permanently introduced. The temporal coherence in irradiance caching was addressed in [35] by introducing anchor points and in [9] by proposing temporal irradiance gradients. The work of Brouillat et al. [3] proposes the combination of IC and photon mapping. They compute a view-independent irradiance cache from a photon map. Important research on interactive GPU-based rendering of irradiance stored at cache locations was proposed by Gautron et al. [10]. The authors used the rasterization pipeline of GPUs to draw the irradiance contribution at every pixel by splatting and accumulating the weighted irradiance of a record to the nearby pixels. This extension enabled the interactive rendering of precalculated irradiance cache records and progressive irradiance evaluation. An extension of irradiance caching to simulate glossy reflections by radiance caching was done in [20]. A detailed overview of irradiance caching can be found in [21] and [29].

Ray-tracing in Mixed Reality Ray-tracing is an efficient algorithm for high-quality rendering. Its simplicity and robustness make it a preferable rendering method in many scenarios. Realistic rendering in mixed reality by ray-tracing was described by Scheer et al. [32]. Pomi et al. [30] demonstrated the use of ray-tracing in a TV studio application where the video of a real character is inserted into the virtual world. Recently an interactive method for high-quality specular global illumination in MR scenes including reflections, refractions, and caustics was proposed [15]. The use of real-time ray-tracing for physically correct depth of field calculation in MR was shown in [16]. A method for occlusion calculation in mixed reality environments rendered by ray-tracing was proposed by Santos et al. [6]. Former ray-tracing based solutions for mixed reality calculate only the specular global illumination or direct illumination. Our solution extends ray-tracing based rendering for MR with an innovative approach for high-quality diffuse global illumination.

3 Differential Irradiance Caching

High-quality rendering of virtual objects into the video of a real scene requires two solutions of global illumination. First a synthetic rendering of the reconstructed model of the real scene (I_r) is required. Second the model of the real scene is rendered together with the virtual models (I_m). As described by the differential rendering equation (Eq. 1) [5, 17] , these solutions can be combined with the real camera image (I_c) to obtain the final composited image (I_f):

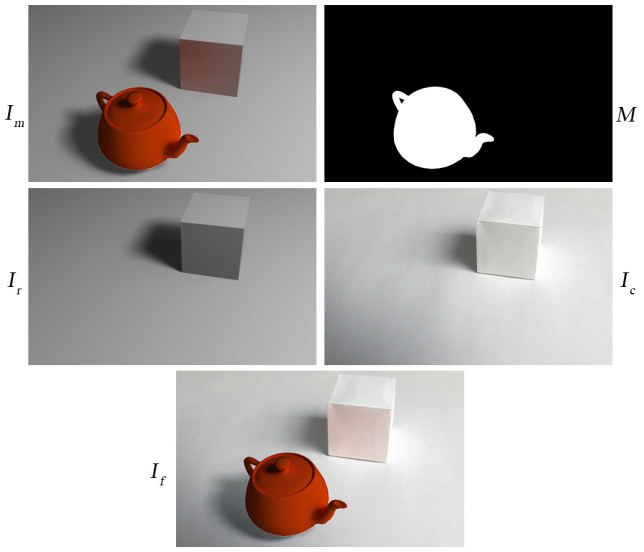

Figure 3: Terms of differential rendering from Equation 1. There is a real white cube on a white table and virtual teapot. Note the indirect red light introduced on the real white cube.

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

M is a mask which controls the amount of blending between differential radiance ($I_m - I_r$) and mixed radiance (I_m). Differential radiance is only used on real objects to introduce light changes caused by inserted virtual objects. Therefore the values of M are 1 at pixel locations of virtual objects and 0 elsewhere. However at object edges values between 0 and 1 indicate blending between virtual and real content (Figure 3, top right). Such a blending appears especially in case of depth of field rendering and for anti-aliasing. The terms of Equation 1 are depicted in Figure 3.

In our solution we calculate the direct light component of I_m and I_r together by ray-tracing using one-pass differential rendering proposed in [16]. The advantage of using ray-tracing for direct illumination is efficient rendering of specular surfaces and possibility of using advanced sampling and reconstruction techniques. The indirect light component of I_m and I_r is calculated together by differential irradiance caching. This algorithm evaluates the accurate differential and mixed irradiances at sparse scene locations and then interpolates the results in screen space to introduce diffuse indirect illumination. The calculation of irradiances at irradiance cache records is performed by Monte Carlo numerical integration. The hemisphere above the location of the cache record is sampled by random rays and the light integral is evaluated by tracing the random light paths. Calculation of differential and mixed radiance in one step and its integration into irradiance is described in Section 3.1. The steps of differential irradiance caching are depicted in Algorithm 1, which runs every frame. It updates the irradiances in the cache records and splats them to image-space.

Procedure $UpdateTemporalData$ updates the irradiance cache and removes old records. For this purpose irradiance record uses the variable t, which denotes the number of frames since it was last updated. If $t = t_{max}$, the record is reevaluated or removed from cache. In our implementation we use a constant t_{max} value, however it can be changed in the future for an adaptive value inversely proportional to the speed of changes in geometry and lighting. The procedure $RecalculateIrradiance$ recalculates the irradiance for outdated cache records. The same sequence of random ray parameters is used if the record is recalculated to achieve temporal

Algorithm 1 Calculate Indirect Light

Input: Vector <IrradRecord> IrradCache, virtual and real scene
Output: Buffer IndirectLight, Vector<IrradRecord> IrradCache

```
UpdateTemporalData(IrradCache)
RecalculateIrradiance(IrradCache)
IndirectLight = SplatIrradiance(IrradCache)

Vector <IrradRecord> NewRecords
NewRecords = DetectCacheMiss(IndirectLight)

while NewRecords.IsNotEmpty() do
   EvaluateIrradiance(NewRecords)
   IndirectLight += SplatIrradiance(NewRecords)
   IrradCache.Append(NewRecords)
   NewRecords.Clear()
   NewRecords = DetectCacheMiss(IndirectLight)
end while
```

coherence. The function *SplatIrradiance* splats all the valid irradiance cache records to the *IndirectLight* buffer by using rasterization. In our method the irradiance cache is a linear data structure. Our algorithm does not search in the irradiance cache. Therefore we can avoid complex non-linear data structures. In addition a linear irradiance cache fits well to the GPU architecture which provides slow GPU memory access and has a small GPU cache. Function *DetectCacheMiss* processes the resulting splat buffer and finds places where the irradiance information is missing. The new cache records are initialized at these locations. This function runs on the CPU and an asynchronous memory transfer between GPU and CPU is used. Cache miss detection is described in detail in Section 3.3. The procedure *EvaluateIrradiance* calculates the differential and mixed irradiances at locations of cache records in parallel on the GPU. The differential irradiance calculation is described in detail in Section 3.1. Algorithm 1 can iteratively calculate indirect illumination for any camera pose by repeating the *while* cycle.

3.1 Differential Irradiance Calculation

The differential irradiance in irradiance cache records is evaluated in parallel by utilizing GPU ray-tracing. The light interaction between the real and virtual objects is calculated in both ways. Both mixed and real irradiances are evaluated in a single pass. Real irradiance is computed as the integral over the product of real incoming light from a hemisphere above a surface point and the cosine function. The resulting vector contains a single value for each spectral component (color channel). This calculation takes only real objects into account:

$$E_r(\mathrm{p}) = \int_{H^+} L_{ir}(\mathrm{p},\omega)\cos\theta\,\mathrm{d}\omega \tag{2}$$

$E_r(\mathrm{p})$ is the real irradiance in point p calculated by integrating the incoming real radiance L_{ir}. This radiance is calculated using only the geometry of the real scene. θ is the angle between incident light direction ω and the surface normal. Analogously the mixed irradiance is the integrated product of incoming light with the $\cos\theta$. However, in this case both virtual and real objects are taken into calculation. The mixed radiance L_{im} is integrated:

$$E_m(\mathrm{p}) = \int_{H^+} L_{im}(\mathrm{p},\omega)\cos\theta\,\mathrm{d}\omega \tag{3}$$

We can calculate the differential irradiance $E_d(\mathrm{p})$ by subtracting the real irradiance $E_r(\mathrm{p})$ from the mixed irradiance $E_m(\mathrm{p})$:

$$E_d(\mathrm{p}) = E_m(\mathrm{p}) - E_r(\mathrm{p}) \tag{4}$$

This results in a difference in indirect lighting caused by adding virtual objects. After multiplication with diffuse albedo the result can be directly added to the differential solution of direct lighting to calculate the differential solution of global lighting. We add this difference to the video image to introduce light changes.

To evaluate the integral from Equations 2 and 3 we sample the integration domain H^+ by shooting random rays. We use Monte Carlo numerical integration to estimate the irradiance [29]:

$$E(\mathrm{p}) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{L_i(\mathrm{p},\omega)\cos\theta}{p(\omega)} \tag{5}$$

where ω_i is the random direction on the hemisphere, $p(\omega)$ is the probability of selecting ω in our sampling process, and N is the number of samples. When we draw our samples from the probability density function (pdf) proportional to $\cos\theta$ [21], Equation 5 can be rewritten as

$$E(\mathrm{p}) \approx \frac{\pi}{N}\sum_{i=1}^{N} L_i \tag{6}$$

Mixed and real radiance can be evaluated together in one sample by using multiple ray types. We use four different ray types similarly to [16]. The four ray types are: **mixed radiance ray, real radiance ray, mixed shadow ray, and real shadow ray**.

Mixed radiance and mixed shadow rays can intersect both real and virtual geometry while real radiance and real shadow rays can intersect real geometry only. Mixed radiance rays return both mixed and real radiances while real radiance rays return only real radiance. The per-ray-data structure contains three variables: mixed radiance, real radiance, and R_i, which is the minimum distance to objects hit by cast rays. R_i is later used in irradiance splatting to specify the area of influence of the irradiance cache record. Using the minimum distance to nearby objects was proposed by Tabellion and Lamorlette [36] to not miss important geometric details.

Primary rays shot from the locations of irradiance cache records are always mixed radiance ray types, because both mixed and real radiances are required. When a ray hits a surface four different cases can happen:

1. **If real geometry is hit by a mixed radiance ray** then a reflected mixed radiance ray is shot in a random direction on a hemisphere above the surface, mixed shadow rays and real shadow rays are shot towards all light sources to evaluate visibility. If lights are visible, light contribution is calculated to both mixed and real radiances and the contribution from the reflected ray is added.

2. **If a virtual object is hit by a mixed radiance ray** then a real radiance ray which continues in the same direction is shot. This ray will evaluate the real radiance ignoring the virtual object at the current location. Additionally the reflected mixed radiance ray is shot in a random direction towards the hemisphere above the surface. This ray will evaluate the reflected mixed radiance. The mixed shadow rays are shot towards positions of light sources and the direct contribution of mixed radiance is calculated.

3. **If a real object is hit by a real radiance ray** then real shadow rays are cast and the direct light contribution is calculated. The reflected real radiance ray is shot in a random direction on a hemisphere above the surface and the reflected real radiance is evaluated.

4. In the fourth case **virtual geometry is hit by a real radiance ray**, however this case is prevented in the ray-triangle intersection routine. It would not contribute to the result since real radiance is not influenced by virtual objects.

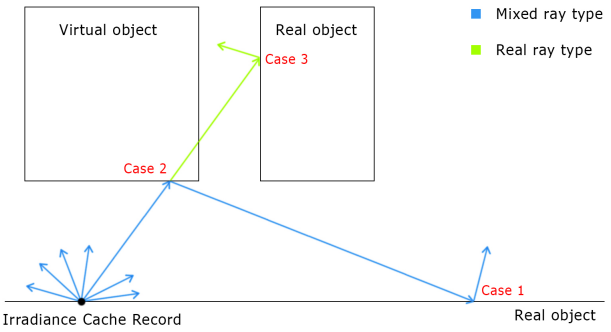

Figure 4: Differential irradiance calculation in the irradiance cache record. Mixed rays are shot in random directions towards the hemisphere above the surface. The mixed rays calculate both real and mixed radiance coming from specified directions. Mixed and real radiances are integrated to calculate corresponding irradiances.

The evaluation of differential irradiance in a cache record by shooting a high number of mixed rays is depicted in Figure 4. Since both real and mixed irradiances are calculated by applying Equation 6 to evaluate both Equations 2 and 3, the differential irradiance can be calculated by Equation 4. Differential and mixed irradiances are stored in each irradiance cache record. These irradiances are later used in irradiance cache splatting. Differential irradiance is used at image locations where real objects appear. It introduces the change of indirect lighting created by adding virtual objects. Mixed irradiance is used at image locations where virtual objects appear. It is the indirect illumination reflected from virtual and real objects. The Monte Carlo evaluation of irradiance by using recursive ray-tracing with random hemisphere sampling has the advantage of calculating multiple bounces of indirect illumination in a single pass. In all scenes used in this paper we shot from 100 to 4000 mixed rays for the evaluation of irradiance in each cache record.

3.2 Differential Irradiance Splatting

The indirect illumination calculated in cache records is splatted by using the irradiance cache splatting algorithm [10]. The differential and mixed irradiances are splatted in a fragment shader in the GPU rasterization pipeline. The quads centered at irradiance cache record locations are sent to the vertex shader. Quads are scaled proportionally to the minimum distance to surrounding objects R_i in a vertex shader. In every fragment a weight w_i is used to accumulate the differential or mixed irradiances. The weight contribution of each record to each pixel is given by the Equation proposed in [36]. If a pixel belongs to a real object, differential irradiance is added. Mixed irradiance is added for pixels belonging to virtual objects. Finally the accumulated irradiances are addressed in the ray-tracing pipeline by a reprojection technique (Section 4). The accumulated values are divided by accumulated weights and the final indirect irradiance contribution is calculated by Equation 7:

$$E(\mathrm{p}) = \frac{\sum_{i \in pixel_{xy}} E_i(\mathrm{p}) w_i(\mathrm{p})}{\sum_{i \in pixel_{xy}} w_i(\mathrm{p})} \tag{7}$$

This irradiance is multiplied by surface albedo in ray-tracing to calculate the reflected indirect light.

3.3 Miss Detection

Irradiance caching approaches typically store cache records in a nonlinear data structure, e.g. in a tree. The set of contributing irradiance cache records is searched for every pixel. If no record is contributing to a pixel, a cache miss is detected and a new record has to be calculated. The search for a set of contributing records has the complexity $O(log(n))$, where n is the number of irradiance records. However this search is invoked many times and non-linear memory access is not efficient on current GPU architectures. Therefore we propose an irradiance cache as a linear vector of cache records without any spatial relationships. Cache records are added sequentially whenever they are created. This cache can easily be rendered. The irradiance cache splatting algorithm splats all records directly to the framebuffer independent of the order of cache entries. In case of linear cache organization the problem of cache miss detection arises. Linear search in a cache has complexity $O(n)$ and is therefore slow if invoked for every pixel.

To make cache miss detection independent of the number of cache records, we propose to use the irradiance splat buffer to detect cache misses. Splat buffer locations with zero weights can be used to select the positions of new cache records and the image can be filled in an iterative manner. The irradiance splat buffer is asynchronously transferred from GPU to CPU using direct memory access (DMA) in our cache miss detection algorithm. The splat buffer is divided into tiled rectangular portions and pixels are traversed within each tile. If at any point the accumulated weight, calculated by the irradiance splatting algorithm, is zero, a new cache record is created. In this case the search for cache misses stops at a tile where a cache record was created to avoid oversampling by multiple neighboring cache records. The rectangular portions of the splat buffer are processed and pixels with zero weight w_i are found in a linear fashion. Iterative refinement is needed to cover an image by cache records, but it can be performed in a progressive fashion by exploiting time coherence. In order to avoid repetitive search in the same pixels within a tile, the position of the last checked pixel can be stored and used in the next iteration. A speedup of miss detection can be achieved by running the search for each tile in parallel.

4 Reprojection to the Splat Buffer

Splatting approaches have been useful in computer graphics for fast energy projection from world positions to the specified buffer. Different problems were solved by introducing splatting [10, 23]. The irradiance cache splatting method allows fast projection of cache records' energy into the framebuffer. The drawback of splatting methods is that the information is stored per pixel in screen space and therefore the depth of field, anti-aliasing, or refraction of a splatted value in refractive objects cannot be calculated from this one-pixel information.

We propose to use a reprojection method and reuse the information from different locations of the splat buffer when needed. This method was previously used by Kán and Kaufmann [15] and Grosch [12] to introduce information from real scenes on reflective and refractive objects. We changed the reprojection to operate on a splat buffer and to access irradiance information of a splat buffer by multiple rays. If indirect lighting information is needed, we reproject the hitpoint of a ray with geometry back to the splat buffer and then read the information from the reprojected position. If the reprojected position is outside of the splat buffer, the splat buffer is repetitively tiled in the 2D plane to provide the information. Reprojection allows addressing more pixels in the irradiance splat buffer by shooting rays with different aperture samples. These multiple values of indirect illumination are finally blurred by filtering ray samples to create the depth of field effect. Visually plausible results of depth of field and refraction can be created. A comparison of rendering with reprojection and without reprojection to the splat buffer can be seen in Figure 5. Our method is the first using the reprojection to access splat buffer data and it can be used for any splatting approach. Furthermore the reprojection method can be generalized to improve access to data in any rendering approach that operates in screen space.

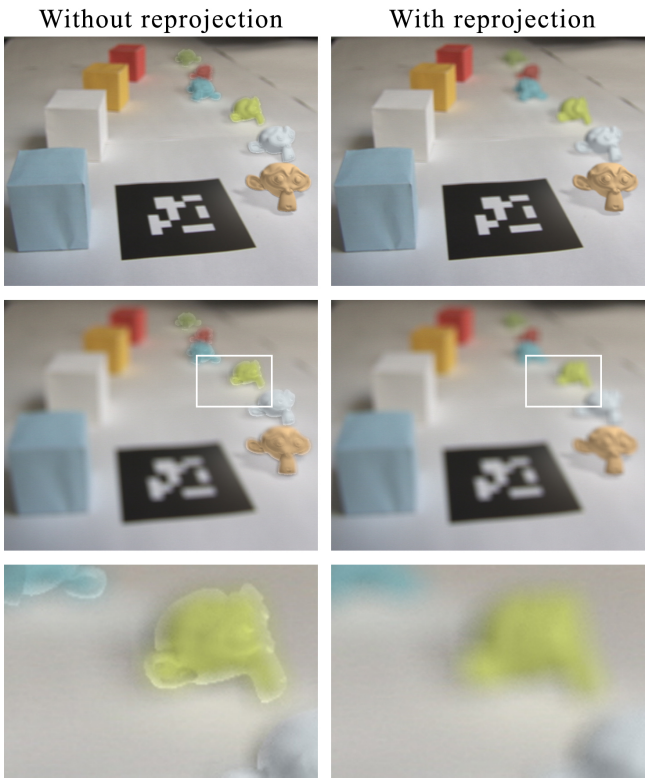

Figure 5: Rendering without (left) reprojection to the splat buffer and with (right) reprojection. Virtual monkeys are rendered in the scene together with real cubes, while both real and virtual cameras have opened aperture causing a depth of field effect. Artifacts caused by direct reuse of irradiance information from the splat buffer in every pixel can be seen in the left column. Properly smoothed edges are achieved by introducing reprojection (right column). In the top row the camera is focused to the frontmost monkey. Middle row shows defocused camera. Bottom row is the close-up from rectangles in the middle row. Note the incorrect sharp ghosting edges appearing in images without reprojection.

5 Global Illumination Framework

The differential irradiance caching method is especially useful for diffuse light transport in mixed reality scenes. It can be extended to differential radiance caching by storing directionally dependent radiance instead of irradiance in every cache record. Radiance caching approaches [20, 19] use spherical harmonics functions to store the radiance in an efficient way. The extension to differential radiance caching can lead to fast and high-quality light transport in glossy scenes.

With the goal of simulating a wide range of global illumination effects in mixed scenes we incorporated differential irradiance caching into a framework for ray-tracing based rendering for mixed reality [15]. This system can calculate the specular light transport in mixed reality scenes including effects like specular reflection, refraction, and caustics (Figure 6) at interactive frame rates. The caustics are calculated by interactive photon mapping. Differential irradiance caching calculates diffuse indirect illumination and stores it in the splat buffer. This buffer is later addressed in the ray-tracing pipeline using reprojection. Difference of indirect mixed and indirect real illumination is added to the result of differential direct illumination.

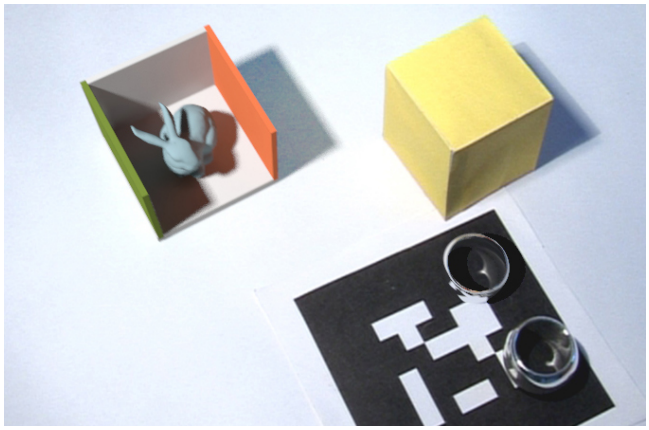

Figure 6: Result of the global illumination framework for mixed reality calculating diffuse and specular GI. The virtual ring (top) casts a caustic to the real table. The real ring (bottom) casts a similar caustic. Diffuse GI in the virtual box with the bunny is calculated by differential irradiance caching.

6 Implementation

Our implementation of differential irradiance caching utilizes the parallel power of modern GPUs. The core of our algorithm is GPU ray-tracing where differential irradiance is evaluated at locations of cache records. The direct illumination and specular indirect illumination are calculated by ray-tracing. The OptiX [26] ray-tracing engine is used in our system. We implemented the irradiance cache splatting in OpenGL using GLSL.

The rendering system uses an additional camera with a fish eye lens to obtain lighting information from the real world. The images from both cameras are converted to HDR by using inverse tone mapping. The environment map is processed and positions of light sources are detected. Our system can use point light sources and area light sources. Lighting of the real scene is approximated by extracting positions, orientations, and sizes of light sources from the environment image. Thresholding and blob detection is used to detect light sources [15]. The light source estimation step is performed in each frame and therefore dynamic lighting is supported in our system. As the Optix engine allows for dynamic Kd-tree rebuild we also support dynamic geometry and materials. The geometry of the real scene is required in differential rendering approaches. Lensing and Broll [24] proposed a method for real-time scene reconstruction and its application to AR. We use a predefined model of the real scene, however since our method fully supports dynamic geometry and materials, the implementation can be extended with a real-time 3D scene reconstruction approach.

An essential part of MR systems is camera pose estimation. Any available tracking system can be used in combination with our rendering method. We track the location and orientation of the camera by using the marker-based ARToolKitPlus library. Whenever we render a frame, the same camera image is used for computing the camera pose by marker tracking and for rendering into the image. Therefore we avoid synchronization of tracking and rendering.

7 Results

To evaluate quality and speed of our differential irradiance caching approach we compared it to real-time progressive path tracing and to a differential instant radiosity solution [18], which is state of the art in real-time diffuse light transport in mixed reality. We implemented a reference solution using offline rendering by differential path tracing on the GPU based on Debevec [5]. We performed a per-pixel comparison of the final images to calculate differences

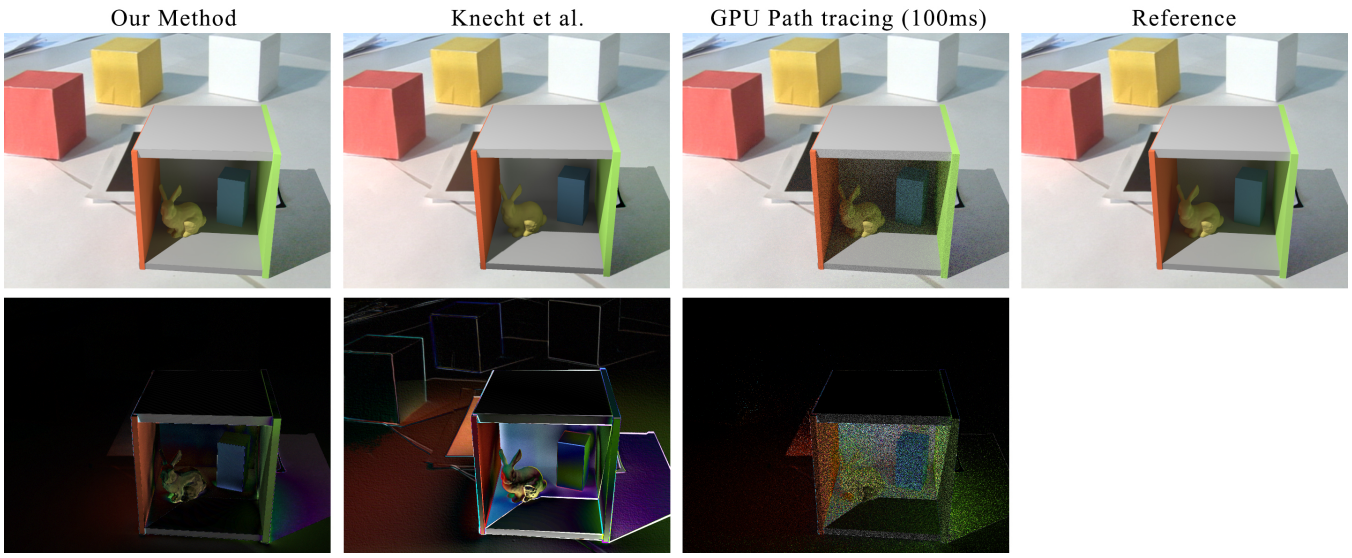

Figure 7: Comparison of rendering techniques for mixed reality. First row from left: Our Method (10fps), Differential Instant Radiosity (28fps), Progressive Path Tracing (10fps), Reference solution calculated by Differential Path Tracing in 2 minutes. 4000 samples were used per cache record to evaluate differential irradiance in our method. 256 VPLs were used in DIR and antialiasing was enabled. The scene contains a virtual box including a bunny and a small box placed on a real table together with real cubes (16K triangles). Indirect illumination is visible especially in shadow areas inside the box. The second row shows the differences of results to the reference. All differences are multiplied by the factor of 5.

of all three methods to the reference solution (Figures 7, and 8). All measurements were taken at PAL resolution of 720x576 on one Nvidia GeForce GTX 690 graphics card.

We compared two different scenes. The same time was given to our method and progressive path tracing to render a frame (100ms in Figure 7 and 200ms in Figure 8). Differential instant radiosity (DIR) computes a solution with 256 Virtual Point Lights (VPLs) in 35-40ms. We did not use more VPLs for DIR because the solution is already practically converged. The remaining error in DIR is due to a bias caused by the visibility approximations and clamping inherent in imperfect shadow maps [31], and a biased light-path distribution. The first scene (Figure 7) consists of real paper cubes on a table and a virtual box with a bunny and a small box inside. Our method introduces a small error due to interpolation between irradiance cache records. The error is rather visible on curved surfaces, like the bunny, where the incorrect illumination is interpolated. The indirect shadow on the left wall of the box exhibits an error caused by missing geometric details in the visibility evaluation. This error is smooth and therefore not visually disturbing. In contrast to that, real-time progressive GPU path-tracing introduces perceptually visible noise. DIR introduces a large error in shadow areas where only indirect light is present due to its light transport approximation. Moreover the bright corners are caused by the visibility approximation in DIR. The second scene (Figure 8) contains real cubes on the table and two virtual dragons with a big virtual box. Color bleeding from virtual objects onto real geometry can be seen. The results show that our method is close to the reference solution. Our solution introduces a smooth error in shadowed areas on the virtual dragon, where only indirect illumination is present. This error is caused by interpolation between irradiance cache records on curved surfaces of a model. Differential instant radiosity suffers from numerical error especially near the corners of the box. The results show that our method can calculate high-quality global light transport between virtual and real objects at interactive frame rates. Our results outperform fast methods in terms of quality and predictive solutions in terms of speed, while achieving quality close to the reference. The limitation of differential irradiance caching is bias in form of interpolation between cache records.

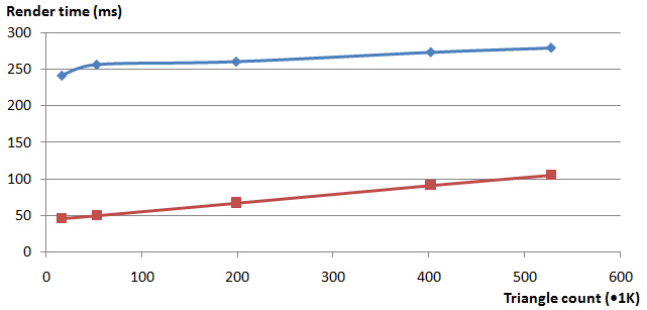

Figure 9: Dependence of rendering time on triangle count. Red curve (bottom) illustrates the rendering with precomputed irradiance cache, the blue curve shows rendering with irradiance cache recalculation. The triangle count is stated in thousands.

Figures 9 and 10 show the dependence of rendering time on the triangle count and resolution. In Figure 9 we can see the logarithmic dependence of rendering time on triangle count caused by the nature of ray-tracing. The rendering time without irradiance cache rebuild has sublinear behavior caused by the combination of ray-tracing and rasterization. In Figure 10 the sublinear dependence, caused by the parallelization of our algorithm, can be observed. The visible linear jumps are caused by exceeding the number of CUDA cores by the number of tasks leading to additional processing.

A performance evaluation can be seen in table 1. Our method provides interactivity even for complex scenes. The processing times of different rendering steps are displayed in Figure 11. We can see that the GPU ray-tracing from the camera for direct light calculation and rasterization-based irradiance cache splatting are the fastest parts of our algorithm. These parts are running in real-time even for complex scenes. The most computationally expensive part is irradiance evaluation. The speed and quality of irradiance evaluation can be controlled by the number of samples on the hemisphere in Monte Carlo numerical integration. The inter-

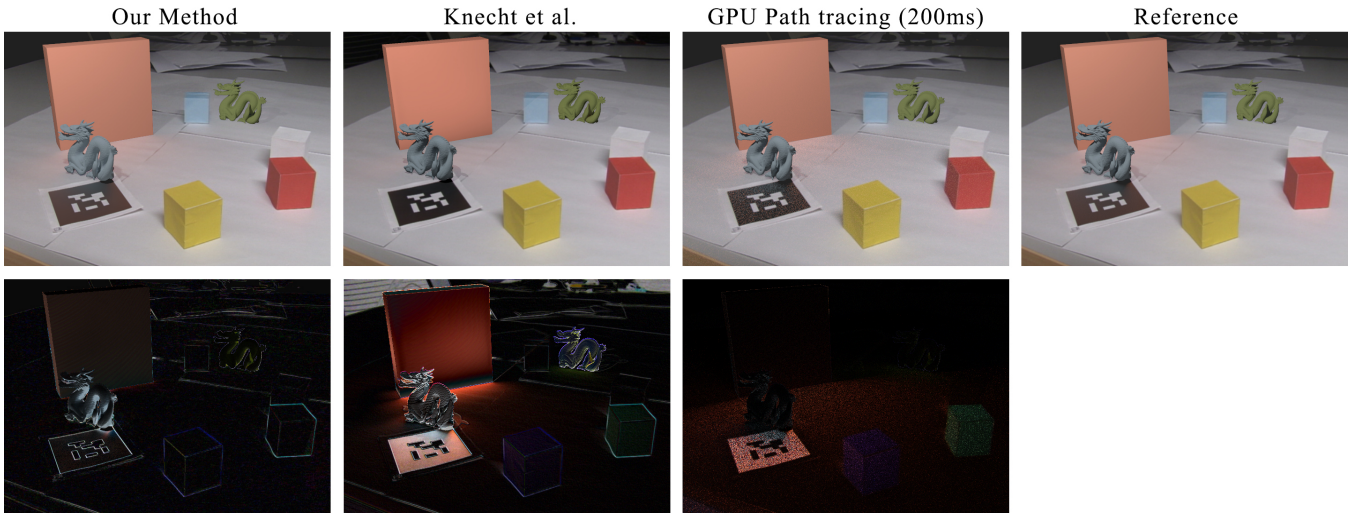

Figure 8: Rendering of a scene (402K triangles) with virtual dragons and a big box by different methods. In the first row from left: Our Method (5fps), Differential Instant Radiosity (25fps), Progressive Path Tracing (5fps), Reference solution calculated by Differential Path Tracing in 2 minutes. 1000 samples were used per cache record to evaluate differential irradiance in our method. 256 VPLs were used in differential instant radiosity. The second row shows the differences of results to the reference . All differences are multiplied by the factor of 5. Sharp edges in background of the difference images are caused by different sampling in the reference solution and are not caused by the evaluated methods.

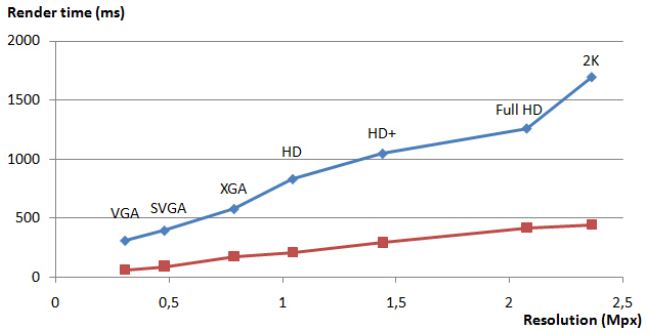

Figure 10: Dependence of rendering time on output resolution. Red curve (bottom) represents rendering with precomputed irradiance cache, and the blue curve shows rendering with irradiance cache recalculation. The measurements were taken with a model consisting of 15K triangles. 256 sample rays were used to evaluate irradiance.

activity of this step is achieved by parallel GPU implementation in Optix. There is a tradeoff between performance and quality in our algorithm. The user can set the allowed interpolation error a. This value determines the radius of interpolation around a cache record and the maximum allowed normal difference. A higher allowed interpolation error will lead to fewer cache records and then to higher performance. Other parameters controlling the quality and performance are the count of ray samples for irradiance evaluation and the number of the splat buffer rectangular search regions in cache miss detection. Our method can be used both in real-time and offline scenarios resulting in different quality settings.

8 Conclusion and Future Work

We present a novel method for high-quality diffuse indirect illumination calculation for mixed reality scenes running at interactive frame rates. Our method outperforms state of the art real-time methods in terms of quality and physically-based methods in terms of speed. We introduce a path-tracing based parallel differential irradiance calculation in radiance cache records. Differential irradiance splatting addressed by a reprojection technique is presented.

Table 1: Performance measurements. The resolution of 720x576 and 1 ray per pixel was used. fps_r denotes the frame rate with irradiance cache recalculation but no new cache records are added (miss detection turned off). The fps_m stands for frame rate with miss detection enabled, and fps_n is the frame rate with no irradiance recalculation and disabled miss detection. 128 samples were used per each cache record to evaluate differential irradiance.

Scene	Triangle count	fps_r	fps_m	fps_n
Bunny	16K	19	10	**31**
Dragon	201K	14	7	**19**
Happy Buddha	886K	7	5	**8**

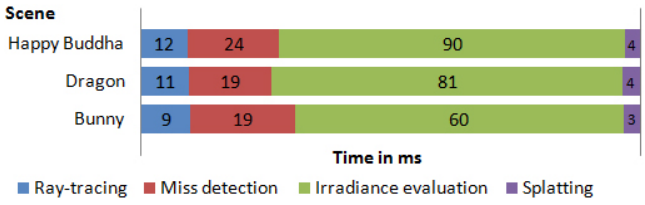

Figure 11: Duration of different rendering steps.

In the future a natural extension of differential irradiance caching is differential radiance caching, where the directional radiance can be stored in spherical harmonics [21]. This extension can simulate light transport on glossy and diffuse surfaces. Together with specular light transport by ray-tracing and photon mapping it will cover the full range of global light transport between surfaces in mixed scenes. Recent research demonstrates novel ways of storing and handling directional radiance information [33], which can also be used for mixed reality scenarios. The aliasing artifacts caused by rasterization in irradiance splatting can be reduced by supersampling the irradiance splat buffer in future work. Our future research will focus on using real-time ray-tracing to create high-quality MR scenes that are indistinguishable from reality.

ACKNOWLEDGEMENTS

We thank Martin Knecht, Christoph Traxler, and Michael Wimmer for their help with the evaluation of results. The 3D models of the dragon and bunny are courtesy of Stanford University. The 3D teapot was created by Martin Newell. The monkey model is courtesy of Blender. We used data from the McGuire Graphics Data archive [25]. Peter Kán is financially supported by the Vienna PhD School of Informatics and the IMS group.

REFERENCES

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

[2] O. Arikan, D. A. Forsyth, and J. F. O'Brien. Fast and detailed approximate global illumination by irradiance decomposition. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 1108–1114, New York, NY, USA, 2005. ACM.

[3] J. Brouillat, P. Gautron, and K. Bouatouch. Photon-driven irradiance cache. *Comput. Graph. Forum*, 27(7):1971–1978, 2008.

[4] 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. ACM, 2005.

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

[6] A. dos Santos, D. Lemos, J. Lindoso, and V. Teichrieb. Real time ray tracing for augmented reality. In *Virtual and Augmented Reality (SVR), 2012 14th Symposium on*, pages 131 –140, may 2012.

[7] 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, Canada, 1993.

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

[9] P. Gautron, K. Bouatouch, and S. Pattanaik. Temporal radiance caching. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):891–901, 2007.

[10] P. Gautron, J. Křivánek, K. Bouatouch, and S. N. Pattanaik. Radiance cache splatting: A GPU-friendly global illumination algorithm. In *Rendering Techniques 2005, Eurographics Symposium on Rendering, Proceedings*, pages 55–64. Eurographics Association, 2005.

[11] G. Greger, P. Shirley, P. M. Hubbard, and D. P. Greenberg. The irradiance volume. *IEEE Comput. Graph. Appl.*, 18(2):32–43, Mar. 1998.

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

[13] 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.

[14] W. Jarosz, V. Schönefeld, L. Kobbelt, and H. W. Jensen. Theory, analysis and applications of 2d global illumination. *ACM Transactions on Graphics (Presented at ACM SIGGRAPH 2012)*, 31(5):1–21, 2012.

[15] 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*, pages 99–108. ACM Press, 2012.

[16] P. Kán and H. Kaufmann. Physically-based depth of field in augmented reality. In *EG 2012*, pages 89–92, 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, and M. Wimmer. Reciprocal shading for mixed reality. *Computers & Graphics*, 36(7):846–856, Nov. 2012.

[19] J. Křivánek, K. Bouatouch, S. N. Pattanaik, and J. Žára. Making radiance and irradiance caching practical: Adaptive caching and neighbor clamping. In *Rendering Techniques 2006, Eurographics Symposium on Rendering*, pages 127–138, Nicosia, Cyprus, June 2006. Eurographics Association.

[20] J. Křivánek. *Radiance Caching for Global Illumination Computation on Glossy Surfaces*. Ph.d. thesis, Université de Rennes 1 and Czech Technical University in Prague, December 2005.

[21] J. Křivánek and P. Gautron. *Practical Global Illumination with Irradiance Caching*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers, 2009.

[22] J. Křivánek, P. Gautron, K. Bouatouch, and S. Pattanaik. Improved radiance gradient computation. In *Proceedings of the 21st spring conference on Computer graphics*, SCCG '05, pages 155–159, New York, NY, USA, 2005. ACM.

[23] 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, 2003. ACM.

[24] P. Lensing and W. Broll. Instant indirect illumination for dynamic mixed reality scenes. In *Proceedings of the 11th IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2012)*, ISMAR2012, pages 109–118. IEEE, 2012.

[25] M. McGuire. Computer graphics archive, August 2011.

[26] 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, 2010.

[27] 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.

[28] S. A. Pessoa, G. de S. Moura, J. P. S. M. Lima, V. Teichrieb, and J. Kelner. Rpr-sors: Real-time photorealistic rendering of synthetic objects into real scenes. *Computers & Graphics*, 36(2):50–69, 2012.

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

[30] A. Pomi and P. Slusallek. Interactive Ray Tracing for Virtual TV Studio Applications. *Journal of Virtual Reality and Broadcasting*, 2(1), Dec. 2005.

[31] 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.

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

[33] D. Scherzer, C. H. Nguyen, T. Ritschel, and H.-P. Seidel. Pre-convolved Radiance Caching. *Computer Graphics Forum (Proc. EGSR 2012)*, 4(31), 2012.

[34] J. Schwarzhaupt, H. W. Jensen, and W. Jarosz. Practical hessian-based error control for irradiance caching. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2012)*, 31(6), Nov. 2012.

[35] M. Smyk, S. Kinuwaki, R. Ďurikovič, and K. Myszkowski. Temporally coherent irradiance caching for high quality animation rendering. *Computer Graphics Forum*, 24(3):401–412, 2005.

[36] E. Tabellion and A. Lamorlette. An approximate global illumination system for computer generated films. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 469–476, New York, NY, USA, 2004. ACM.

[37] G. Ward and P. Heckbert. Irradiance gradients. In *Eurographics Rendering Workshop*, pages 85–98, May 1992.

[38] G. J. Ward, F. M. Rubinstein, and R. D. Clear. A ray tracing solution for diffuse interreflection. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '88, pages 85–92, New York, 1988. ACM.