Interactive Near-Field Illumination for Photorealistic Augmented Reality on Mobile Devices

Kai Rohmer[*]
Computational Visualistics
University of Magdeburg

Wolfgang Büschel[†]
Interactive Media Lab
Technical University of Dresden

Raimund Dachselt[‡]
Interactive Media Lab
Technical University of Dresden

Thorsten Grosch[§]
Computational Visualistics
University of Magdeburg

Abstract

Mobile devices become more and more important today, especially for augmented reality (AR) applications in which the camera of the mobile device acts like a window into the mixed reality world. Up to now, no photorealistic augmentation is possible since the computational power of the mobile devices is still too weak. Even a streaming solution from a stationary PC would cause a latency that affects user interactions considerably. Therefore, we introduce a differential illumination method that allows for a consistent illumination of the inserted virtual objects on mobile devices, avoiding a delay. The necessary computation effort is shared between a stationary PC and the mobile devices to make use of the capacities available on both sides. The method is designed such that only a minimum amount of data has to be transferred asynchronously between the stationary PC and one or multiple mobile devices. This allows for an interactive illumination of virtual objects with a consistent appearance under both temporally and spatially varying real illumination conditions. To describe the complex near-field illumination in an indoor scenario, multiple HDR video cameras are used to capture the illumination from multiple directions. In this way, sources of illumination can be considered that are not directly visible to the mobile device because of occlusions and the limited field of view of built-in cameras.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism H.5.1 [Information Interfaces and Representation]: Artificial, Augmented and Virtual Realities

1 Introduction

Mobile devices like smartphones and tablet PCs are part of our everyday life. In combination with the integrated camera, the mobile device can act like a window into an augmented real world [11]. When virtual objects are inserted, their appearance is often inconsistent with the real environment, mainly because of wrong illumination. So far, a consistent illumination that handles both dynamic scenes and dynamic lighting conditions eluded the mobile platforms. Even though many sophisticated illumination methods allow for plausible global illumination at interactive rates on non-mobile platforms, several applications would benefit from such a mobile system. For instance, applications in interior planning and architecture can be enhanced by convincing in-place visualization of the designer's vision. Bringing back virtual versions of lost or destroyed artifacts would open up new possibilities in the field of cultural heritage. Mobile AR can also be used in movie productions or in the process of creating a theater play by providing previews of the final scene and the created mood to the director at early stages. Furthermore, it is relevant for mobile AR games, as users expect more and more enhanced graphics.

We primarily aim for applications like the augmentation of real prototypes, for which a correct illumination at any place in the scene is required and a perceptively plausible illumination is not sufficient.

To accomplish this, three problems need to be addressed:

- Mobile devices, such as tablets, do not yet have the computational resources necessary for computing interactive global illumination on their own.
- Streaming the rendered images from a powerful desktop PC causes a latency that is too high to meet the requirements for seamlessly integrated virtual objects, especially during user interactions and for multiple simultaneous views.
- Real lighting conditions in the dynamic scene need to be captured reliably and limitations caused by occlusions and the limited field of view have to be overcome.

In this paper, we present a novel, *distributed* illumination approach for AR with consistent illumination of virtual objects with direct light, indirect light (color bleeding) and shadows of primary and strong secondary lights. Due to the limitations in computational power of the mobile device and the lack of important information (see Figure 2, left), we split the illumination into two parts: In the first part, the existing radiance values are captured by a number of HDR video cameras that are placed at different locations in the scene. This acquisition process and the extraction of parameters for our lighting model is executed on a stationary PC. Based on the extracted information, we display augmentations with consistent illumination at interactive frame rate on the mobile device, as shown

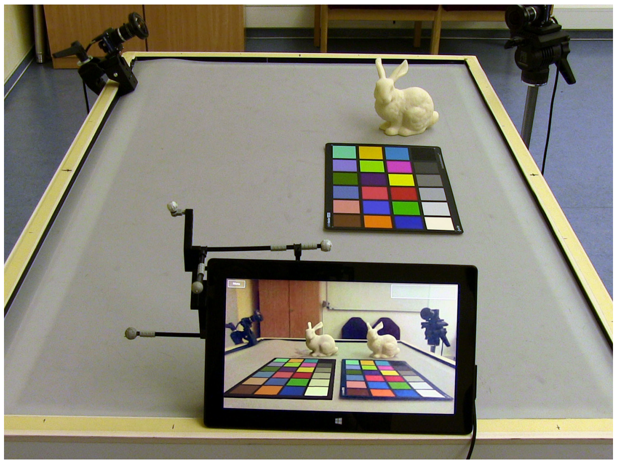
Figure 1: The tablet camera shows the real world, augmented by virtual objects with consistent illumination, displayed at 27 fps. By using a tracked device, the user can move the tablet freely in the augmented real world.

[*]e-mail: kai.rohmer@isg.cs.ovgu.de
[†]e-mail: bueschel@acm.org
[‡]e-mail: dachselt@acm.org
[§]e-mail: grosch@isg.cs.ovgu.de

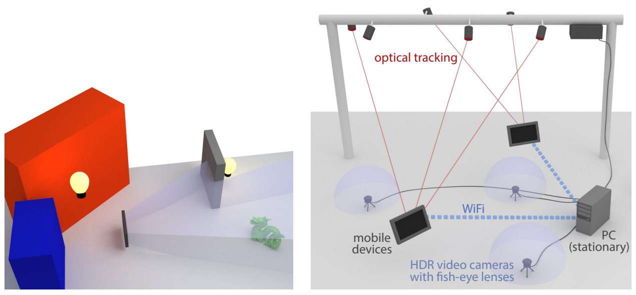

Figure 2: Left: The camera image of a mobile device does not see the important light sources required for a consistent illumination of a virtual object. Right: Our hardware setup.

in Figure 1. To avoid a potential bottleneck of the bandwidth between PC and mobile devices, our illumination model reduces the amount of transferred data that is required for the reconstruction of the environmental lighting condition and the illumination of virtual objects. In summary, our main contributions are:

1. A new distributed approach for interactive Augmented Reality under dynamic real-world environment lighting,
2. A lighting model for correct near-field illumination with compact parametrization to be transferred to one or multiple display devices.

2 Previous Work

AR with Consistent Illumination The first work for AR with consistent illumination was presented by Fournier et al. [12] who invented the differential rendering technique. Further extensions exist for hierarchical radiosity [10] and final gathering [36]. Debevec [8] introduced the high dynamic range (HDR) light probe to capture the distant real illumination. A similar approach was presented by Sato et al. [43] using a fish-eye lens. Grosch [17] introduced differential photon mapping to correctly display reflecting and refracting objects and their changes in illumination. To capture the existing near-field illumination, Corsini et al. [6] used a pair of light probes to estimate the distance of light sources. Using a movable HDR camera with a light probe, Unger et al. [45] captured the complete near-field illumination inside a small region. Even if no information of the surrounding illumination is available, a plausible augmentation can be implemented, as shown by Karsch et al. [30] for legacy photographs. A list of AR illumination techniques can be found in Jacobs and Loscos [25].

Interactive AR with Consistent Illumination For an interactive setting, the first works for AR with consistent illumination were presented by Kanbara and Yokoya [29], and Agusanto et al. [2]. They assume a distant and constant illumination, captured by a light probe. Several approaches simply define the existing illumination manually, e.g. Haller et al. [22] and Pessoa et al. [40]. Gibson and Murta [15] demonstrated how to implement differential rendering on the GPU for distant illumination. This was further extended in [14] for the augmentation of pictures of real rooms, taking near-field illumination into account by combining point lights and an irradiance volume. Grosch [18] showed how this technique can be further improved for the augmentation in a panoramic image viewer. For temporally varying illumination, Havran et al. [23] introduced a sampling approach for an HDR video camera. In Korn et al. [34], two such HDR video cameras were used to estimate the distance of moving light sources. In Grosch et al. [19], a near-field illumination approach was presented for augmentations under daylight in a real room. The idea of differential rendering has been applied to several interactive global illumination methods, including instant radiosity [32, 33], progressive path tracing [28] and light propagation volumes [13]. Kan and Kaufmann [27] presented a partial implementation of differential photon mapping, running at near-interactive frame rates on the GPU. Assuming a single dominant light direction, Nowrouzezahrai et al. [39] introduced a real-time light factorization method that allows soft and hard virtual shadows. Instead of special equipment, an approximate reconstruction of the environmental light is possible from simple objects. This can be a diffuse sphere [3], a special shading probe [5] or only the user's hand [46]. Madsen and Lal [37] demonstrated a photometric reconstruction from shadows, Jachnik et al. [24] used a specular surface. A first approach to augment live images based on geometry captured by a RGB-D camera was presented by Lensing and Broll [35]. The captured depth image was used here for a fast illumination based only on screen-space information. Meilland et al. [38] reconstructed both 3D geometry and HDR radiance values based on a moving RGB-D camera. For static illumination, a real-time rendering can be performed with correct near-field reflections and shadows of extracted light sources. Gruber et al. [21] demonstrated a probeless approach that displays a visually plausible augmentation. Here, a low-frequency environment map is reconstructed from the illumination of diffuse objects that allows soft shadows between virtual and real objects [20]. Recently, Csongei et al. [7] presented a progressive path tracing solution for AR on a mobile phone based on a pre-recorded environment map. Here, the simulation is distributed between a stationary PC and the mobile device.

We observe that so far there is no solution for augmented reality with consistent, spatially and temporally varying illumination on a mobile device. The first reason is the low computational power of current mobile devices. Secondly, none of the existing methods addresses an on-line capture process of the spatially and temporally varying near-field illumination. Our solution solves these problems.

3 Overview

Our goal is the consistent illumination of virtual objects on mobile devices in a real environment. Multiple users should be able to interact in the real world with photorealistic augmentations. We thereby focus on an indoor scenario with a difficult, spatially and temporally varying near-field illumination (Figure 2). This requires the knowledge of the plenoptic function [1], which includes the real radiance values at any point in the scene, viewed from any direction at any time. Based on this information, an interactive global illumination simulation can be computed.

3.1 Hardware Setup and Precomputations

Our hardware setup is shown in Figure 2 (right):

Multiple HDR video cameras are connected to a stationary PC. The cameras are equipped with fish-eye lenses and placed in the scene, such that all regions are visible in at least one camera image. To enable the measurement of radiance values on real environment surfaces, each camera has to be calibrated in a pre-process. Intrinsic parameters are estimated by using the OCamCalib Toolbox[1] by Davide Scaramuzza and stored in a lookup table to map the captured image on a perfect equidistant projection. To reconstruct the extrinsic parameters, we simply capture a tracked checker board and reconstruct the position of the camera. To acquire absolute real radiance values instead of arbitrary pixel colors, a photometric calibration is necessary. Therefore, we reconstruct the camera response curve using pfstools[2], leading to linear relative radiance values after applying the inverted response curve. By capturing images of an XRite ColorChecker[3] and a least-squares approximation, we obtain

[1]https://sites.google.com/site/scarabotix/ocamcalib-toolbox
[2]http://resources.mpi-inf.mpg.de/hdr/calibration/pfs.html
[3]http://xritephoto.com/ph_product_overview.aspx?id=1192

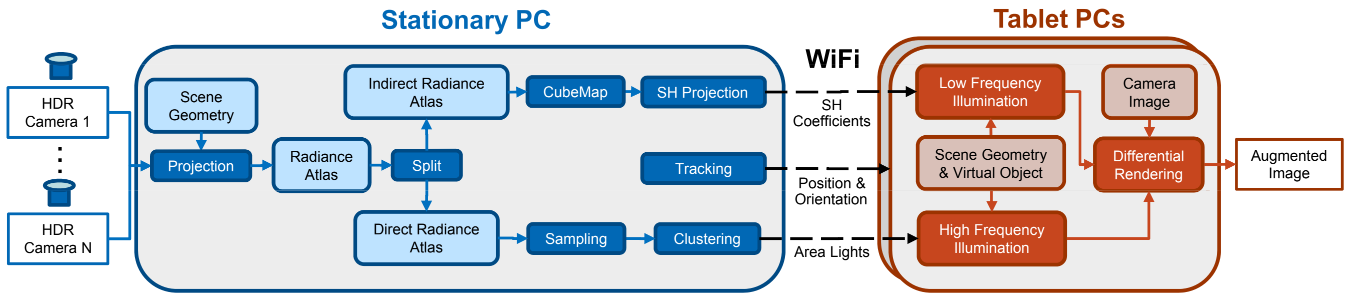

Figure 3: The whole pipeline for distributed illumination.

a matrix that maps the measured linear radiances onto absolute radiances known for each tile of the checker. A similar process is repeated for each mobile device. Additionally, we estimate another matrix to compensate the color shift introduced by the display. The current position and orientation of the mobile device are captured at runtime. For communication between the stationary PC, mobile devices and the tracking system we use WiFi.

In a pre-process, the geometry and the diffuse materials of the real environment are reconstructed manually using a common DCC tool. The resulting model is a very coarse representation with a simple uv-mapping that is later used as texture atlas (Sec. 3.3).

3.2 Distributed Illumination

Given the HDR information – real radiance at each position of the environment – in combination with the 3D model, we aim for a consistent illumination of virtual objects. Based on measured radiance values of the real environment, there are different choices for interactive global illumination [42]. The obvious solution is to use one of these methods to render on a static machine and stream the resulting images to all mobile devices. We do not follow this idea for several reasons: First of all, we need a different image for each mobile device which can lead to a performance break-down on the server side in case of many mobile devices. The main difficulty in direct user interaction is the in-time update of the displayed augmented scene. This is because there is a latency in sending notifications of user input to the stationary PC, as well as waiting for the generation, compression, and transmission of the rendered image that is eventually combined with the camera image. We therefore developed an illumination model that distributes the computation between the static PC and the mobile devices.

One option for interactive global illumination is the extraction of a set of virtual point lights (VPLs) [31]. This allows for a complete illumination from all directions with a shadow cast by each VPL. For good quality, at least a few hundred VPLs are required. Unfortunately, only a few VPLs are possible on a mobile device at interactive frame rates. On the other hand, precomputed radiance transfer (PRT) [44] techniques can be used, which allow real-time illumination with natural light. These techniques work well for low-frequency illumination and diffuse materials. This is especially useful for indirect illumination, such as color bleeding from real to virtual objects. However, high-frequency illumination and hard shadows are difficult to achieve. To solve this problem, we developed a *hybrid* solution that combines the best of both worlds. Our solution is based on the observation that most typical settings consist of a few bright light sources and large low-frequency indirect light regions. We follow the idea introduced in [14] and *split* the incoming light into a *high-frequency* and a *low-frequency* part. There are two reasons for this: First, it allows for an efficient illumination with the desired effects: The high-frequency illumination and shadows can be displayed with a small set of VPLs, while the low-frequency illumination such as color bleeding can be implemented with PRT. The second reason is that this combination *requires only a small amount of data* that needs to be transferred between the stationary PC and the mobile devices, enabling interactive update rates.

3.3 Pipeline Overview

The whole pipeline from capturing images of the real world to displaying the augmented image using the distributed illumination is summarized in Figure 3: The HDR video cameras with fish-eye lenses capture the existing radiance values. On the stationary PC, each image is then projected onto the reconstructed 3D geometry using a hemispherical projection and shadow mapping. The recorded radiance values are stored in a *radiance atlas* which describes a 1:1 mapping of 3D scene points to atlas texels (Sec. 4.1). To capture the illumination at as many surface positions as possible, we use multiple cameras. Areas seen by multiple cameras therefore receive multiple measurements. For an illumination at both interactive speed and high quality on a mobile device, we proceed as follows: The radiance atlas is split into two parts: A *direct* (high-frequency) radiance atlas and an *indirect* (low-frequency) radiance atlas (Sec. 4.2). The direct radiance atlas is transformed into a small set of area lights (Sec. 4.3), which is transferred to the mobile device. For the indirect radiance, PRT is used for the illumination. Thus, the indirect light is transformed into spherical harmonic (SH) basis (Sec. 4.4) and the resulting coefficients are transferred to the tablet PC as well. Based on this information, the illumination of virtual objects can be computed quickly on the tablet PC without streaming any images. Using differential rendering, the virtual object can then be inserted into the tablet camera image with correct appearance and shadows (Sec. 5).

4 Server Computations

We explain the server computations of our method based on a simple synthetic example shown in Figure 4 (left).

4.1 Acquiring the Radiance Atlas

We use a texture atlas to record radiance values for all points in the scene. To update the current lighting conditions, each HDR camera permanently projects its radiance values into the atlas. This is implemented by rendering the reconstructed scene with a vertex shader that replaces the vertex position by its texture coordinate and outputs the world position along with the vertex normal to the pixel shader stage. There, we project the world position of each fragment into the camera image space to get the corresponding image coordinate. Subsequently, we sample the camera image and a previously generated artificial depth image at this location to decide the visibility of the currently processed texel in the manner of shadow mapping (see Figure 4). Since triangles not facing the camera cannot be seen, they are rejected during the rendering into the atlas, depending on the dot product of normal and view direction. When

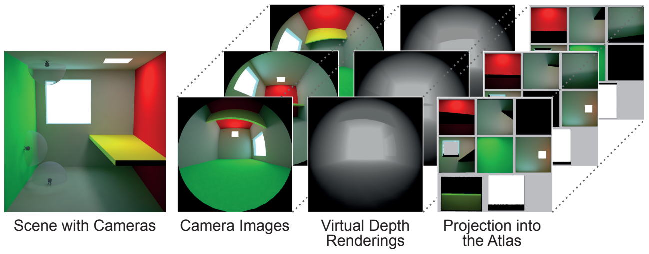

Figure 4: Acquiring the radiance atlas for a simple synthetic scene with three HDR cameras (left). Each HDR camera records a fish-eye image of the scene. Additionally, a depth buffer is rendered for each camera using the reconstructed scene. Using the depth buffer for visibility tests, the camera image is then projected into the radiance atlas (right). Note that each camera has only a partial information of the total scene radiance.

multiple cameras see the same region, we compute a weighted average of the camera images. To account for the low resolution in the border regions of a fish-eye projection, we use the angle to the main camera direction as a weighting. Since each texel in the atlas can become an indirect light source, we store both position and normal to correctly place and rotate the light. For photometric correctness, each texel stores both the radiance value and the spatially varying world-space area of the texel. To compensate artifacts at texture seams we, apply a dilation over the 8 neighbors with a range of 2 texels.

4.2 Splitting the Radiance Atlas

To determine the radiance L at a point $\mathbf{x}$, seen from direction ω, we integrate over the hemisphere Ω to solve the rendering equation [26]

$$L(\mathbf{x},\omega) = \int_{\Omega} f_r(\mathbf{x},\omega_i,\omega)\, L_{in}(\mathbf{x},\omega_i)\, \cos\theta_i \, d\omega_i \tag{1}$$

whereas L_{in} is the incoming radiance from direction ω_i, f_r is the bidirectional reflectance distribution function (BRDF), and θ_i is the angle between ω_i and the surface normal at $\mathbf{x}$. For efficiency reasons, we separate the reflected radiance in direct radiance L_{Dir} and indirect radiance L_{Ind}

$$L(\mathbf{x},\omega) = L_{Dir}(\mathbf{x},\omega) + L_{Ind}(\mathbf{x},\omega) \tag{2}$$

whereas L_{Dir} corresponds to the direct radiance caused by light sources and strong indirect lights and L_{Ind} is the remaining indirect radiance. This means that we decide for each direction of the hemisphere around $\mathbf{x}$ whether it corresponds to incoming direct radiance Ω_{Dir} or incoming indirect radiance Ω_{Ind}:

$$L_{Dir}(\mathbf{x},\omega) = \int_{\Omega_{Dir}} f_r(\mathbf{x},\omega_i,\omega)\, L_{in}(\mathbf{x},\omega_i)\, \cos\theta_i \, d\omega_i \tag{3}$$

$$L_{Ind}(\mathbf{x},\omega) = \int_{\Omega_{Ind}} f_r(\mathbf{x},\omega_i,\omega)\, L_{in}(\mathbf{x},\omega_i)\, \cos\theta_i \, d\omega_i \tag{4}$$

To implement this separation, we split the radiance atlas in a *direct radiance atlas* and an *indirect radiance atlas*. For this, we determine a *threshold value*: Texels in the atlas with a radiance larger than the threshold are assigned to the direct radiance atlas, the other texels are assigned to the indirect radiance atlas. To allow for varying lighting conditions, this threshold is adjusted *dynamically*. For this purpose, we provide a user-defined ratio that decides how much of the *total* amount of light in the scene should be assigned to the direct light. To determine the threshold value, we first compute the histogram of *all* radiant intensity values in the atlas. We then accumulate the radiant intensity values from high to low until the given

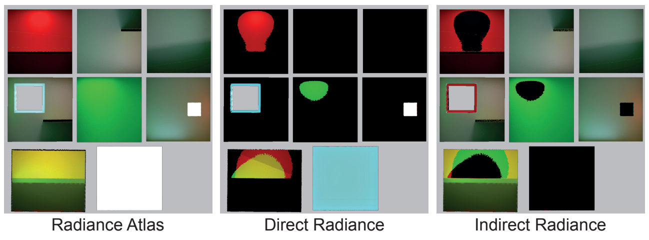

Figure 5: The radiance atlas (left) is split in direct radiance (middle) and indirect radiance (right). Note that the direct radiance atlas contains both the light sources and bright indirect regions. The separation is computed per color channel to allow sources in monochrome regions that would have a low gray scale brightness.

ratio is reached. In this way, we always extract a certain percentage of either direct lights or strong indirect lights. Figure 5 shows an example for this separation. In our examples, we used $75-98\,\%$ of the total radiant intensity for the direct light.

4.3 Finding Direct Light Sources

After splitting the atlas, the direct radiance at $\mathbf{x}$ is computed by summing up all N_{Dir} texels in the direct radiance atlas:

$$L_{Dir}(\mathbf{x},\omega) \approx \sum_{i=1}^{N_{Dir}} f_r(\mathbf{x},\omega_i,\omega)\, L_i\, V(\mathbf{x},\omega_i)\, \cos\theta_i\, \frac{\Delta A_i \cos\theta}{r^2} \tag{5}$$

whereas L_i and ΔA_i are the radiance and area of texel i, and V is the binary visibility function. The distance between sender and receiver is r and the angle to the sender normal is θ. To simplify the computation, we now group the N_{Dir} texels from the direct radiance atlas into a *low* number of $j = 1..M$ clusters. In accordance with [9] we denote the clusters as *virtual area lights* (VALs). The unoccluded direct radiance at $\mathbf{x}$ due to a VAL j is then given by

$$L_{VAL_j}(\mathbf{x},\omega) = f_r(\mathbf{x},\omega_j,\omega)\, \frac{I_j \cos\theta_j \, \cos\theta}{r^2} \tag{6}$$

whereas the radiant intensity I_j of VAL j is computed by summing up all N_j texels assigned to this cluster:

$$I_j = \sum_{k=1}^{N_j} L_k\, \Delta A_k \tag{7}$$

The direct radiance can therefore be approximated by summing up all M VALs:

$$L_{Dir}(\mathbf{x},\omega) \approx \sum_{j=1}^{M} L_{VAL_j}(\mathbf{x},\omega)\, V(\mathbf{x},\omega_j) \tag{8}$$

Note that for $M = N_{Dir}$, this yields Eq. (5) without $V(\mathbf{x},\omega_i)$.

To avoid flickering, these extracted virtual area lights have to be coherent under temporally varying illumination. To accomplish this, we follow the clustering method described in [9]: First, we use importance sampling by a prefix sum (scan) and an inverse cumulative density function to generate a set of samples in the direct radiance atlas. Secondly, we use k-means clustering with positions and normals as weight to generate clusters of the samples. To correctly compute the radiant intensity I_j of each VAL using Eq. (7), each texel in the direct radiance atlas is assigned to its closest cluster center, using the same distance metric. Finally, the data to be transferred to the mobile device for each VAL is the following: Position (12 bytes), normal (4 bytes, compressed), radiant intensity (12 bytes) and area (4 bytes). In total, these are only 32 bytes per

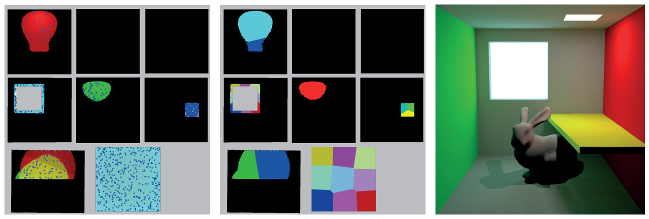

Figure 6: VAL extraction for direct light: Samples are placed on the direct radiance atlas (left). Using k-means clustering, these samples are grouped in M clusters (middle). Each texel is assigned to the closest cluster center. Integration over each cluster leads to M VALs that are used for direct illumination of a virtual object (right).

VAL. The total number of VALs is a time-quality tradeoff, in our experiments we use $8 \leq M \leq 64$.

Note that we only use the direct radiance atlas for the VAL extraction and ignore the current position of the virtual objects. As an alternative, an environment map could be rendered from the virtual object center position. The drawback of this option is that we might miss some important light sources which are not visible from the center of the virtual object. Additionally, the global VAL selection achieves a better temporal coherence, and in the case of multiple virtual objects, only one set of VALs is used which follows the concept of a consistent global light model.

4.4 Compressing Indirect Light

For the indirect light L_{Ind}, we assume that the remaining illumination in the indirect radiance atlas is of low frequency. In this case, a compression using spherical harmonics (SH) can be applied to the environment light around the virtual object. For a diffuse virtual object with reflection coefficient ρ, it is sufficient to use only the first $K = 9$ basis functions for an illumination with a barely visible error, as shown in [41, 44]. Given a vertex v at position $\mathbf{x}$, the indirect radiance is computed as a simple dot product:

$$L_{Ind}(\mathbf{x}, \omega) \approx \frac{\rho}{\pi} \sum_{k=0}^{K-1} C_k C_{vk} \tag{9}$$

whereas the coefficients C_k and C_{vk} are obtained by a projection onto the SH basis function Y_k.

$$C_k = \int_{\Omega_{Ind}} L_{in}(\mathbf{x}, \omega_i)\, Y_k(\omega_i)\, d\omega_i \tag{10}$$

$$C_{vk} = \int_{\Omega} V(\mathbf{x}, \omega_i)\, \cos\theta_i\, Y_k(\omega_i)\, d\omega_i \tag{11}$$

The C_k coefficients can be interpreted as the amount of light that is incident from *all* directions of the surrounding. Corresponding to that, the C_{vk} coefficients at a vertex v describe a set of directions from which incident light is *not occluded*. Due to the orthogonality of the SH basis functions, the dot product of both is then the amount of *incident unoccluded* light from all directions at the vertex. The coefficients C_{vk} of the transfer function $V \cos\theta_i$ are *static*, so they can be precomputed and stored per vertex at the virtual object. In contrast, the coefficients C_k of the environment map L_{in} *change* whenever the incoming illumination changes. We therefore render a low-resolution $(6 \times 32 \times 32)$ cube map from the virtual object's position with the indirect radiance atlas as texture of the surrounding scene. Then, this is projected to the first nine spherical harmonic basis functions Y_k and the resulting coefficients C_k are transferred to the mobile device. Using RGB float values, these are only $9 \times 3 \times 4 = 108$ bytes in total. In case of multiple mobile devices, the same coefficients can be reused. In case of multiple virtual objects, this process is repeated for each object. The computation cost for this step is small (see Sec. 6).

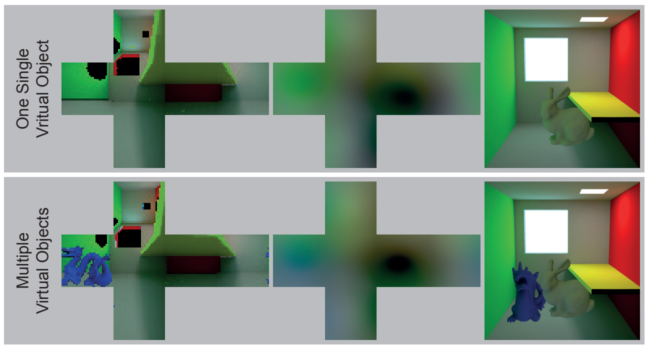

Figure 7: To estimate the indirect illumination of a virtual object, we first render a cube map with the indirect radiances from the object center (left). This is projected into the first nine SH basis functions (center) which allows a real-time illumination of a virtual object (right). The second row shows multiple virtual objects with mutual interreflections, like the blue color bleeding from DRAGON to BUNNY.

In fact, we are able to take account of indirect light transmission between virtual objects with a small overhead by including the other virtual objects during the indirect light estimation of the one that is updated. This is shown in Figure 7 (bottom) for the BUNNY interacting with a blue DRAGON. To keep the additional effort low, the objects are illuminated by PRT, too. Therefore, we render another cube map containing direct and indirect light to derive SH-coefficients with the correct amount of light. In summary, we are rendering $2n$ cube maps, each containing $n-1$ virtual objects and the reconstructed scene in order to achieve additional indirect light transmission between n virtual objects. Note that these interreflections cover $b-1$ bounces for objects that are static for b iterations.

5 RENDERING ON THE CLIENT

Due to the described separation of illumination, the final image generation on the client only requires lightweight operations for a mobile device with limited rendering capabilities: For each VAL, we compute the direct radiance and visibility using shadow mapping Eq. (8). This is added to the indirect illumination which is computed per vertex by applying Eq. (9) using the stored coefficients C_{vk} and the transferred coefficients C_k. To display virtual shadows with correct brightness, we use the differential rendering technique introduced by Gibson et al. [14] and subtract direct radiance in the virtual shadows.

To improve the rendering performance, we use a *tile-based deferred shading* based on Andersson [4]. Compared to simple forward rendering and non-tiled deferred rendering, a tile-based approach reduces overdraws to a minimum because each final screen texel is processed only once. Additionally, the G-Buffer (see Figure 8) needs to be read only once, which improves performance since memory accesses are expensive.

In the first pass, the reconstructed and the virtual scene are rendered into one G-Buffer containing projection space depth, world space normal, diffuse reflection coefficients, and indirect radiance as well as flags to distinguish virtual from real objects (see Figure 8). To avoid unnecessary geometry processing, we calculate the indirect radiance for virtual objects by PRT in the vertex shader. Hence, we need to render the scene only once, except for the shadow map generation described later in this section.

The second pass handles light calculations and the composition of the augmented image in one single compute shader program.

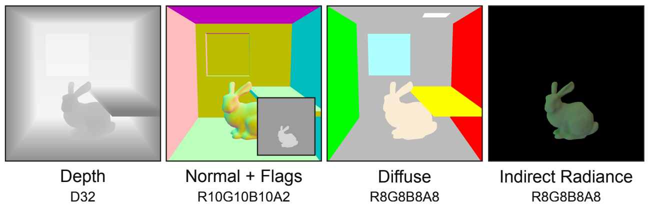

Figure 8: G-Buffer: an off-screen buffer containing geometry and material information of the reconstructed and virtual scene per pixel.

Thus, the screen is divided into tiles of 8×8 texels – which performed best in our tests – that are processed by a thread group of 64 threads per tile. Each thread then executes the following steps:

1. Read the background image and G-Buffer data for the corresponding texel and construct the view frustum around the tile. Near and far plane are determined by the minimum and maximum occurring depth value within the tile.
2. Cull the VAL assigned to the thread if the view frustum is entirely in the negative hemisphere of the VAL. Due to the Lambertian emission of the VALs, such a VAL does not contribute to the illumination of the tile. Because of the group size, 64 VALs can be treated simultaneously. If there are more VALs than threads per tile, this process is performed in a loop. Lights not culled are added to a shared list, containing $m \leq M$ visible VALs.
3. Perform visibility and shading operations to illuminate the surface position $\mathbf{x}$ at the texel by all remaining m VALs in the group shared VAL list. For differential rendering, we accumulate the radiance L_{VAL_j} of all VALs depending on the texels' flags. For texels marked virtual we store radiance that is not shadowed, neither by real nor by virtual objects. For non-virtual texels, we store radiance that is shadowed by virtual but not by real objects. In its essence, we estimate the light that should be missing because of new virtual shadows.
4. Combine the results of step 3, the background color, and the indirect radiance using Eq. (12) for texels marked as virtual and Eq. (13) otherwise. The visibility at $\mathbf{x}$ from $VAL\ j$ in the reconstructed scene is referred to as V_j, where $\hat{V}_j$ is the visibility in the virtual scene.

$$L = L_{Ind} + \sum_{j=1}^{m} V_j \cdot \hat{V}_j \cdot L_{VAL_j} \tag{12}$$

$$L = L_{background} - \sum_{j=1}^{m} V_j \cdot (1 - \hat{V}_j) \cdot L_{VAL_j} \tag{13}$$

As described above, we use two shadow maps per VAL to cover shadows from reconstructed and virtual objects [33, 15]. This is necessary to prevent virtual objects from casting shadows through real objects (see Figure 9, left). Using only one shadow map containing the closest distance in light space can lead to correct shadows (green). But without the distance of the closest reconstructed object we are not able to identify the correct shadow receiver and add wrong shadows (red) on every further surface. Hence, we need $2M$ shadow maps for direct illumination with M VALs, which is not feasible for large M in real-time. To reduce the geometry processing overhead we update 16 shadow maps at once by using a geometry shader for duplicating the primitives and rendering to multiple viewports simultaneously. Therefore, we organize our shadow buffer in a texture array containing 4×4 shadow maps per slice (see Figure 9, right). To adjust to the narrow time budget, we update only one slice of the virtual shadow buffer and one slice of the reconstructed shadows per frame. The update order for virtual shadows follows round robin, but currently updated VALs are preferred, while reconstructed shadows are only updated after receiving new VAL positions. To obtain good shadows with low resolution we construct each shadow frustum to closely fit all visible virtual objects and use this frustum for both virtual and reconstructed shadows maps. The reconstructed geometry between light and near plane is projected onto the near plane.

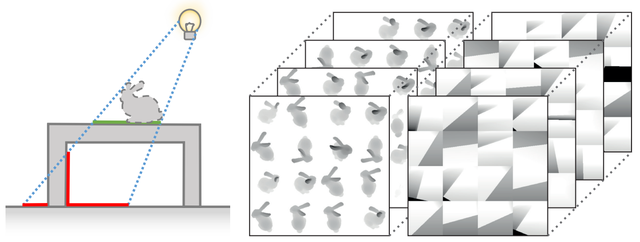

Figure 9: Two shadow maps per VAL are required to avoid double shadowing from virtual objects (dashed contour) in regions that are shadowed in the real environment.

6 RESULTS

In this section, we will report the results that are obtained by our distributed approach for augmenting live camera streams with virtual objects illuminated by the dynamically captured real world environment. All performance experiments for rendering were run on a Microsoft Surface Pro with Intel i5-3317U CPU, 1.7 GHz, 4 GB RAM and Intel HD Graphics 4000. The stationary PC used for image acquisition and calculation of the light model parameters was equipped with an AMD Phenom II X4 965 CPU, 3.4 GHz, 8 GB RAM and an NVIDIA GeForce 580 GTX.

For comparison in quality and performance we decided to evaluate the synthetic CORNELL SCENE used in Sec. 4 to avoid inaccuracies caused by the camera sensors and lenses as well as during the reconstruction process. Nevertheless, results of real world scenarios are demonstrated in the later part of this section. The scene was designed to be as simple as possible while showing the most important interactions between real and virtual objects. In particular, there are shadows from virtual on real objects and vice versa, virtual objects occluding real objects and vice versa, and there is a strong indirect light that causes color bleeding. The radiance of the small light at the ceiling is 15 W/srm^2 and thereby 5 times brighter than the window with 3 W/srm^2. The scene is augmented by a BUNNY with 2.5k triangles and the resolution of the G-Buffer is 960×540, where not otherwise stated.

6.1 Comparison

To evaluate our approach we compare it with a standard VPL-based lighting, PRT and different combinations of light clustering and splitting into direct and indirect light. To achieve fair results, we used the same renderer with all optimizations by tiled rendering and simplifications during shadow map updates for rendering VPLs as we do in our case. For PRT we were also using the G-Buffer to treat occlusions between real and virtual objects as well as the same calculation of indirect light used in our approach, but we disabled all direct light calculations and shadow map updates since they cannot be used with PRT. Figure 10 shows results of the different methods depending on the number of direct light sources. For the synthetic scene we created a path traced reference image, depicted in the lower right corner. Above this ground truth solution the result of simple PRT without any directional lights is shown. In the first row, the classic Monte Carlo-based VPL lighting is depicted. To generate VPLs, the radiance atlas was sampled using the gray scale intensity as density function p. While the position is directly read from the atlas at sample position s, the radiant intensity of the resulting VPL is calculated by Eq. (14), where N_{VPL} is the number of VPLs (assuming the surface is diffuse).

$$I_{VPL} = \frac{1}{N_{VPL}} \cdot \frac{I_s}{p_s} \tag{14}$$

The second row combines the classic VPL lighting with our light separation. Instead of sampling the complete radiance atlas, we just sample the direct light atlas (see Figure 6, left). The indirect light is compressed to spherical harmonics as we do in our approach. The third row shows clustered VPLs without light separation. In this case, we apply our direct light clustering step to the classical VPL method. Here we draw 4k samples, cluster them by k-Means and integrate the radiance atlas to result in one VPL for each cluster, as described in Sec. 4.3. The last row contains the final results of our approach with light separation and clustering.

6.2 Evaluation

In comparison with the ground truth image, the result of the PRT method shows significant differences. Besides the lack of shadows, there is a visible shift in the color of shading. The environment coefficients used for this image were derived from a cube map rendered at the object center from where the bright red wall is only slightly visible. This explains the cold tone of the image and why PRT alone is not a good choice for near-field illumination, even though the measured timings are best.

Evaluating the classical VPL approach confirms the expected behavior known from instant radiosity implementations [31, 32, 33]. A large number of lights is required to converge against the correct solution. We stopped at 512 sources, which produced a result close to the reference image in 503 ms.

By separating high-frequency from low-frequency light the regions to sample for VPL generation become smaller. In combination with PRT-based low-frequency illumination, the visual quality of the results increases especially for a low number of light sources. Because of the smaller sample regions, the point lights concentrate in bright areas, which leads to more plausible shadows as second benefit. The additional computation cost for PRT lighting is constant and rather low compared to the direct lighting. Note that these two methods without light clustering are not coherent over time for smaller light counts. This results in a distracting flickering and is not suitable in most scenarios. To provide an impression, we refer to the accompanying video.

The experiments with clustered VPLs showed an improved spatial coherence but did not lead to a temporal coherent illumination because of the large cluster sizes that need to cover the whole radiance atlas. For the client, there is no difference to classical VPLs in terms of calculations for lighting, which was confirmed by equal timings.

The results of our approach, depicted in the last row, contain features of both improvements. The separation of the radiance atlas leads to smaller areas to be sampled, hence the light sources concentrate in the brightest regions. The additional clustering leads to coherent light positions and thus coherent virtual shadows. It also allowed us to integrate the area per light to be approximated as disc. Hence, virtual objects close to light sources do not show the singularities of classic point lights. Considering the measured timings, there is no difference compared with the approach in row two, since there is no difference in rendering on the client side. Comparing the images created with varying numbers of VALs reveals only slight differences in the shading of the virtual object. The most obvious distinction can be found in the quality of the shadows, especially at the transition from the virtual to the real shadow cast by the yellow board. A drawback of our approach can be observed in the shadowed region of the BUNNY which is too bright in comparison with the reference image. One reason for that is the limited number of SH coefficients C_k and the lack of details in the reconstructed indirect light. Another influencing factor is that the cube map is only valid for the center of the virtual object. Other locations on the virtual object, e.g. below the yellow board, have a slightly different environment illumination. This problem could be addressed by evaluating the indirect light at multiple locations and interpolating the SH coefficients per vertex during rendering which leads to an approach similar to irradiance volumes [16]. Finally, there is another aspect that contributes to a too bright indirect light. The indirect radiance estimation at the center of the virtual object by cube map rasterization does not consider the shadows cast by any virtual objects. However, we are currently restricted to diffuse materials because of the diffuse PRT, which is also a limitation of the presented approach.

Table 1: Timing breakdown in ms for the stationary PC at an atlas resolution of 1024×1024, 4 HDR cameras, 4k direct light samples and 16 clusters with 20 iterations per clustering step.

Update Atlas (for dynamic scenes)	
Position, normals and area	0.5 ms
Dilatation	1.33 ms
Acquiring the Radiance Atlas (* per camera)	
Acquire color image *	2.35 ms
Render depth image *	0.26 ms
Project into atlas *	0.44 ms
Combine radiance atlas	0.58 ms
Splitting the Radiance Atlas	
Find separating threshold	9.1 ms
Split into direct and indirect atlas	0.53 ms
Finding Direct Light Sources	
Sampling (4k Samples)	6.67 ms
k-Means clustering (M=16)	7.3 ms
Integrating cluster radiances	17.5 ms

6.3 Light Extraction Performance

All tasks executed by the server are implemented on the GPU, but they are not well optimized in terms of performance, since the results are transmitted asynchronously and do not affect the rendering performance discussed in the next paragraph. However, an interactive update rate improves the visual quality of the rendering while moving virtual objects and reduces the time that is needed to respond to changes in the dynamic environment. Table 1 contains the timings measured with our current implementation. The single steps are not executed in the listed order. For example, the first block is only required if the tracking system reported a moving real object. The operations concerning the camera images of the second block are applied only if a camera captured and transferred a new image during the last iteration. The latter is influenced by the type and number of cameras used, their resolution, and the available bandwidth for the transfer to the GPU.

The time to update the indirect light coefficients depends on the number of virtual objects. Figure 11 illustrates the increasing effort with growing number of virtual objects. As described in Sec. 4.4, we include the other virtual objects to take account of indirect transmission between the objects. Hence, the time required for rendering a cube map increases with the number of objects while the duration for compressing the cube maps into SH-coefficients is constant. Note that the time to update the indirect light without interaction between virtual objects would be equal to the duration measured for one virtual object. The rendering of an extra cube map with direct and indirect light is not necessary in this case.

6.4 Rendering Performance

As noted in Sec. 5, the tile-based approach reduces the number of geometry processing passes, overdraw, and G-Buffer accesses per texel. Nevertheless, the G-Buffer resolution is the most dominant

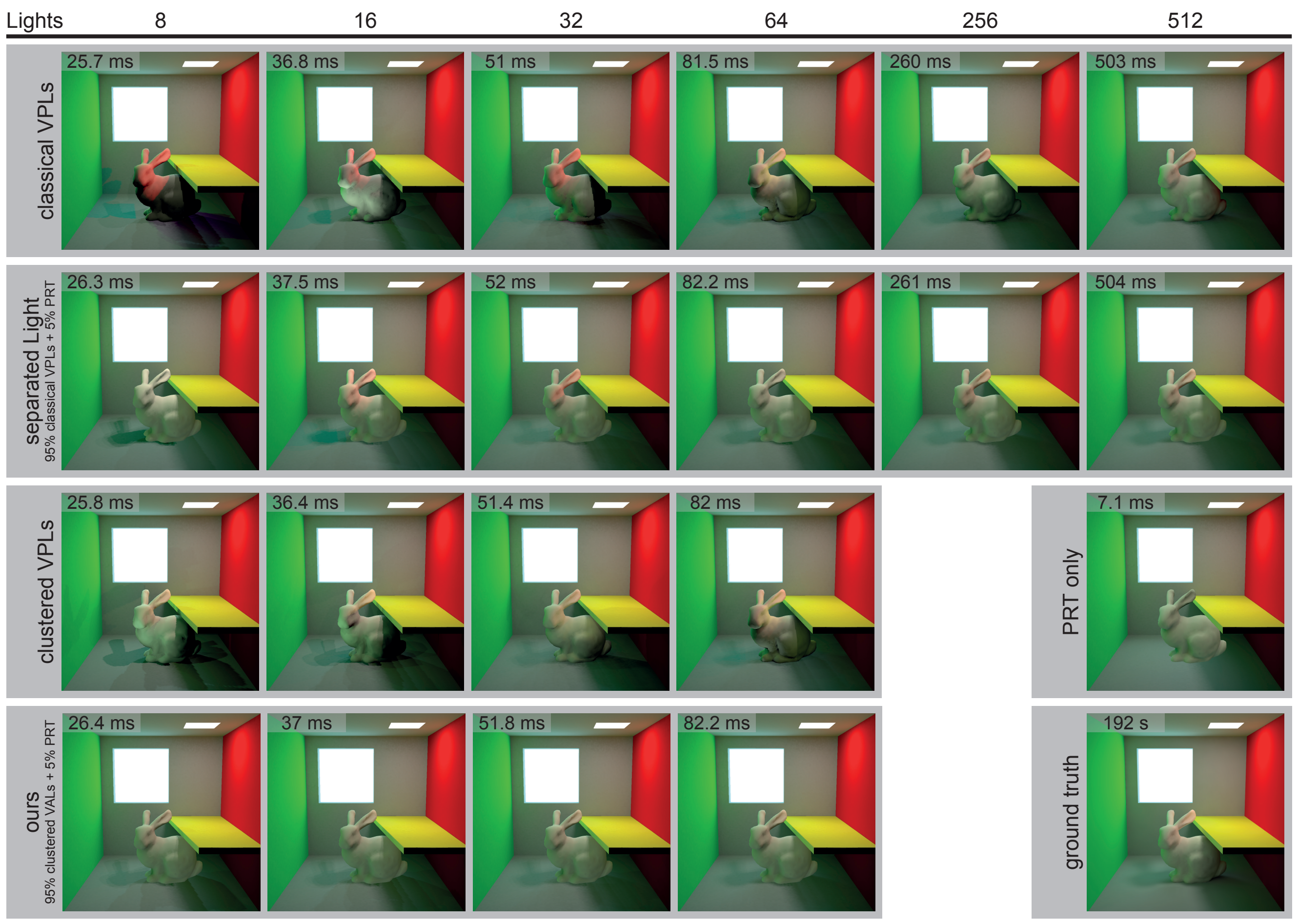

Figure 10: Comparison with classical VPL lighting, PRT and combinations

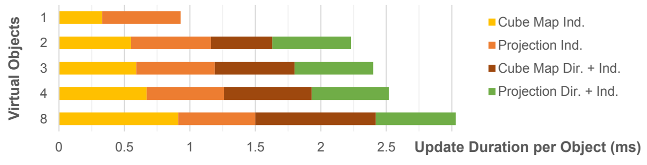

Figure 11: Timings for updating the indirect light coefficients per object. Measured in the CORNELL SCENE with multiple virtual BUNNIES.

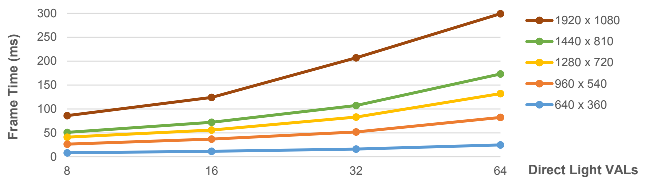

Figure 12: Influence of the G-Buffer resolution on the frame time. Timings measured for augmenting the CORNELL SCENE with the virtual BUNNY.

factor on the rendering performance. Figure 12 illustrates the increasing time per frame with growing light count and resolution. In consequence of the tiled rendering the timings grow almost linearly with the number of rendered tiles.

In Figure 13 the frame timings are broken down to five steps. The acquisition of the background image, the rendering of reconstructed shadows, and the generation of the G-Buffer are independent of the number of VALs. The update of 16 virtual shadow maps takes 2.2 ms if 16 or more lights are present. The largest part of the time is spent on calculating the direct illumination and visibility. After a constant offset, the required time increases linearly with the number of lights.

Because of the deferred rendering, the impact of geometry complexity on performance is assumed to be low as each model has to be rendered only once to generate the G-Buffer. The result of the evaluation with virtual models of different complexity is depicted in Figure 14. As anticipated, the time required to create the G-Buffer increases with the number of primitives, up to 10 ms for 260*k* triangles. The other step that needs to render the virtual geometry is the update of the virtual shadows. Since we are processing 16 shadow maps per iteration, the duration grows exponentially up to 216 ms for the largest model. Fortunately, this large number is not relevant in practice, since low-poly models can be used for rendering shadow maps of low resolution. Hence, a few hundred primitives are sufficient for the 128×128 shadow maps we used in our examples, and highly detailed models are only required during the G-Buffer generation.

6.5 Real World Scenarios

To measure the real-world radiance values, we use Matrix Vision mvBlueFOX-IGC200 HDR video cameras with 180 degrees fish-

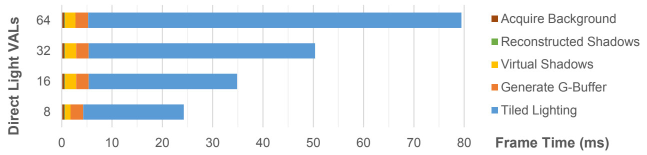

Figure 13: Timings with respect to the number of VALs measured for augmenting the CORNELL SCENE with the virtual BUNNY broken down and accumulated.

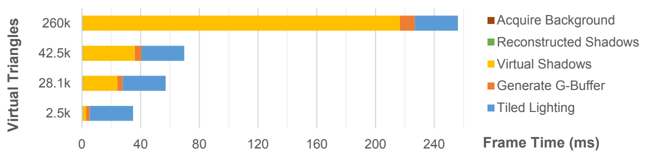

Figure 14: Timings with respect to vertex count measured for augmenting the CORNELL SCENE with different virtual models and 16 lights broken down and accumulated.

eye lenses. For tracking, we use OptiTrack with 12 infrared cameras, capturing a range of approximately 3×2 meters.

Figure 1 shows the consistent appearance of a 3D-printed and a virtual BUNNY side-by-side. We added a real and a virtual color checker to show the quality of the reproduced colors. To verify the correct capture process of the *near-field* illumination, we place the BUNNY close to a local light source and a strong indirect light, as shown in Figure 15. An interactive session from the accompanying video is shown in Figure 16, demonstrating both temporally and spatially varying illumination. The performance in real and synthetic scenes has been very similar in all our experiments. This is because the cost for transferring the mobile camera image to VRAM and the cost for rendering the synthetic background image compensate each other.

7 CONCLUSIONS AND FUTURE WORK

We demonstrated that augmented reality with consistent illumination is possible on current mobile devices at interactive frame rates. To achieve this, we developed a lighting method that shares the computation effort among a stationary PC and the participating mobile device. The amount of data to be exchanged between both is reduced, avoiding a bottleneck in transmission due to limited bandwidth. Multiple mobile devices are supported without additional overhead in terms of lighting calculation and transmission since the parameters of the light model are valid for all devices and can be broadcasted. We captured the near-field illumination of indoor scenarios with multiple HDR video cameras and use this as information for illumination of the virtual objects. The virtual objects can be moved freely with a consistent illumination at any position and adapt to temporal changes in the incident illumination, although the sources of light are not visible to the tablet camera.

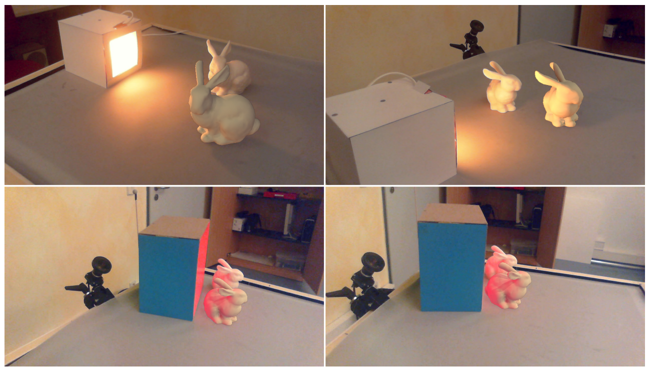

Figure 15: A virtual BUNNY in front of its real counterpart, illuminated by a local light source (top) and a strong indirect light (bottom). In both cases, the sender becomes invisible after user movement.

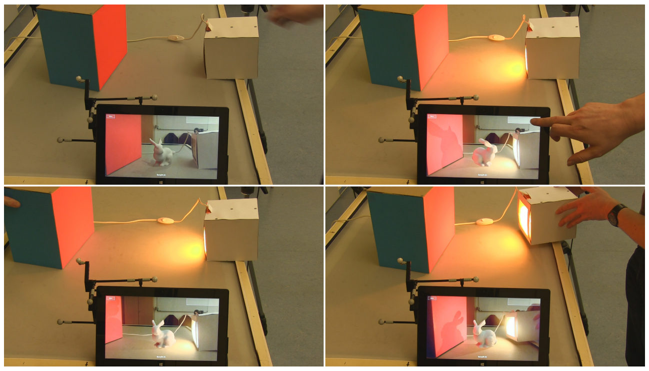

Figure 16: Moving tracked objects: Initial configuration (top left), user switches on the light and rotates the BUNNY at 27 fps (top right), red color bleeding disappears when the box is moved away (bottom left) and direct light changes after light movement (bottom right).

At present, we place the HDR cameras manually such that all relevant regions are visible in at least one of the cameras. However, if there are regions not visible to any camera, some of the illumination might be missing. To overcome this problem, we evaluate a dynamic, tracked HDR camera that can be moved to such invisible regions. This additional information also improves the quality of the captured light sources, since a few static cameras are not enough to capture a goniometric diagram of a complex light source. The same applies to the capture process of real objects with non-diffuse materials. Since portable 3D sensors are available, dynamic capturing of the geometry and materials is also interesting [35].

Currently, we do not include the indirect illumination which is reflected from the virtual object to the real scene. This could be added by analyzing the radiance distribution on the virtual object and the placement of virtual light sources onto the virtual object. Due to missing high-resolution environment images on the tablet, we cannot display highly glossy virtual objects. Instead of reflections of the real environment, only the highlights of the extracted area lights would be visible. Making more advanced BRDFs possible is also a topic for further investigation. To improve the shadow quality, a soft shadow could be displayed for each area light, similar to [9].

Our method supports manipulation of virtual objects with correct illumination at interactive rates, but (as visible in the accompanying video) the update rates of the direct light sources are lower since they are only updated after a *complete* iteration of the server pipeline. Additionally, we neither predict the VAL positions on the client side nor blend between updated and former VALs, which would both hide this latency.

Note that our hardware setup allows for working in a dynamic environment with moving real objects and under changing light conditions. For static environments one could use *only* the mobile devices to capture the surrounding in a pre-process and track visual features for estimating the device position. The presented approach works in this setting without the complex hardware setup, too.

ACKNOWLEDGEMENTS

This work was sponsored by grant no. GR 3833/2-1 and DA 1319/2-1 of the German Research Foundation (DFG).

References

[1] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*, pages 3–20. MIT Press, Cambridge, MA, 1991.

[2] K. Agusanto, L. Li, Z. Chuangui, and N. W. Sing. Photorealistic rendering for augmented reality using environment illumination. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 208–216, 2003.

[3] M. Aittala. Inverse lighting and photorealistic rendering for augmented reality. *The Visual Computer*, 26(6-8):669–678, 2010.

[4] J. Andersson. Parallel graphics in frostbite - current & future. SIGGRAPH Course: Beyond Programmable Shading, New Orleans, USA, 2009.

[5] D. A. Calian, K. Mitchell, D. Nowrouzezahrai, and J. Kautz. The shading probe: Fast appearance acquisition for mobile AR. *Proceedings of ACM SIGGRAPH Asia Technical Briefs*, 32(6), 2013.

[6] M. Corsini, M. Callieri, and P. Cignoni. Stereo light probe. *Comput. Graph. Forum*, 27(2):291–300, 2008.

[7] M. Csongei, L. Hoang, C. Sandor, and Y. B. Lee. Global illumination for augmented reality on mobile phones (Poster). In *IEEE VR*, 2014.

[8] P. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proc. SIGGRAPH '98*, pages 189–198, 1998.

[9] Z. Dong, T. Grosch, T. Ritschel, J. Kautz, and H.-P. Seidel. Real-time indirect illumination with clustered visibility. In *Vision, Modeling, and Visualization Workshop*, 2009.

[10] G. Drettakis, L. Robert, and S. Bougnoux. Interactive common illumination for computer augmented reality. In *Eighth Eurographics Workshop on Rendering*, pages 45–56, 1997.

[11] G. W. Fitzmaurice, S. Zhai, and M. H. Chignell. Virtual reality for palmtop computers. *ACM Trans. Inf. Syst.*, 11(3):197–218, 1993.

[12] A. Fournier, A. Gunavan, and C. Romanzin. Common illumination between real and computer generated scenes. In *Proc. of Graphics Interface*, pages 254–262, 1993.

[13] T. A. Franke. Delta light propagation volumes for mixed reality. *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2013.

[14] S. Gibson, J. Cook, T. Howard, and R. Hubbold. Rapid shadow generation in real-world lighting environments. In *Eurographics Symposium on Rendering*, pages 219–229, 2003.

[15] S. Gibson and A. Murta. Interactive rendering with real-world illumination. In *Eurographics Workshop on Rendering*, pages 365–376, 2000.

[16] G. Greger, P. Shirley, P. M. Hubbard, and D. P. Greenberg. The irradiance volume. *IEEE Computer Graphics and Applications*, 18(2):32–43, 1998.

[17] 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.

[18] T. Grosch. PanoAR: Interactive augmentation of omni-directional images with consistent lighting. In *Mirage, Computer Vision / Computer Graphics Collaboration Techniques and Applications*, pages 25–34, 2005.

[19] T. Grosch, T. Eble, and S. Mueller. Consistent interactive augmentation of live camera images with correct near-field illumination. In *ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 125–132, 2007.

[20] L. Gruber, T. Langlotz, P. Sen, T. Hoellerer, and D. Schmalstieg. Efficient and robust radiance transfer for probeless photorealistic augmented reality. In *IEEE VR*, pages 91–97, 2014.

[21] L. Gruber, T. Richter-Trummer, and D. Schmalstieg. Real-time photometric registration from arbitrary geometry. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 119–128, Washington, DC, USA, 2012.

[22] M. Haller, S. Drab, and W. Hartmann. A real-time shadow approach for an augmented reality application using shadow volumes. In *Proc. Symp. on Virtual reality software and technology*, pages 56–65, 2003.

[23] V. Havran, M. Smyk, G. Krawczyk, K. Myszkowski, and H.-P. Seidel. Importance Sampling for Video Environment Maps. In *Eurographics Symposium on Rendering*, pages 31–42, Konstanz, Germany, 2005.

[24] J. Jachnik, R. A. Newcombe, and A. J. Davison. Real-time surface light-field capture for augmentation of planar specular surfaces. In *ISMAR*, pages 91–97. IEEE Computer Society, 2012.

[25] K. Jacobs and C. Loscos. Classification of illumination methods for mixed reality. *Comput. Graph. Forum*, 25(1):29–51, 2006.

[26] J. T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, 1986.

[27] P. Kán and H. Kaufmann. High-quality reflections, refractions, and caustics in augmented reality and their contribution to visual coherence. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 99–108, 2012.

[28] P. Kán and H. Kaufmann. Differential progressive path tracing for high-quality previsualization and relighting in augmented reality. In G. Bebis, editor, *ISVC 2013, Part II, LNCS 8034*, pages 328–338. Springer-Verlag Berlin Heidelberg, 2013.

[29] M. Kanbara and N. Yokoya. Geometric and photometric registration for real-time augmented reality. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, page 279, 2002.

[30] K. Karsch, V. Hedau, D. Forsyth, and D. Hoiem. Rendering synthetic objects into legacy photographs. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 30(6):157:1–157:12, 2011.

[31] A. Keller. Instant Radiosity. In *SIGGRAPH '97*, pages 49–56, 1997.

[32] M. Knecht, C. Traxler, O. Mattausch, W. Purgathofer, and M. Wimmer. Differential instant radiosity for mixed reality. In *ISMAR '10: Proceedings of the 2010 9th IEEE International Symposium on Mixed and Augmented Reality*, 2010.

[33] M. Knecht, C. Traxler, O. Mattausch, and M. Wimmer. Reciprocal shading for mixed reality. *Comput. and Graph.*, 36(7):846–856, 2012.

[34] M. Korn, M. Stange, A. von Arb, L. Blum, M. Kreil, K. Kunze, J. Anhenn, T. Wallrath, and T. Grosch. Interactive augmentation of live images using a HDR stereo camera. In *Workshop Virtuelle und Erweiterte Realitaet der GI-Fachgruppe VR/AR*, volume 3, pages 107–118, 2006.

[35] P. Lensing and W. Broll. Instant indirect illumination for dynamic mixed reality scenes. *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 0:109–118, 2012.

[36] C. Loscos, M. Frasson, G. Drettakis, B. Walter, X. Granier, and P. Poulin. Interactive virtual relighting and remodeling of real scenes. In *Tenth Eurographics Workshop on Rendering*, pages 329–340, 1999.

[37] C. B. Madsen and B. B. Lal. Outdoor illumination estimation in image sequences for augmented reality. In *GRAPP*, pages 129–139, 2011.

[38] M. Meilland, C. Barat, and A. Comport. 3D High Dynamic Range Dense Visual SLAM and Its Application to Real-time Object Relighting. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2013.

[39] D. Nowrouzezahrai, S. Geiger, K. Mitchell, R. Sumner, W. Jarosz, and M. Gross. Light factorization for mixed-frequency shadows in augmented reality. In *10th IEEE International Symposium on Mixed and Augmented Reality (Proceedings of ISMAR 2011)*, 2011.

[40] S. A. Pessoa, G. de S. Moura, J. P. S. M. Lima, V. Teichrieb, and J. Kelner. Photorealistic rendering for augmented reality: A global illumination and BRDF solution. In *IEEE Virtual Reality Conference*, pages 3–10, Waltham, Massachusetts, USA, 2010.

[41] R. Ramamoorthi and P. Hanrahan. An efficient representation for irradiance environment maps. In *SIGGRAPH*, pages 497–500, 2001.

[42] T. Ritschel, C. Dachsbacher, T. Grosch, and J. Kautz. The state of the art in interactive global illumination. *Computer Graphics Forum*, 31(1):160–188, 2012.

[43] I. Sato, Y. Sato, and K. Ikeuchi. Acquiring a radiance distribution to superimpose virtual objects onto a real scene. *IEEE Trans. Vis. Comput. Graph.*, 5(1):1–12, 1999.

[44] P.-P. J. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.*, 21(3):527–536, 2002.

[45] J. Unger, S. Gustavson, P. Larsson, and A. Ynnerman. Free form incident light fields. *Comput. Graph. Forum*, 27(4):1293–1301, 2008.

[46] Y. Yao, H. Kawamura, and A. Kojima. The hand as a shading probe. In *ACM SIGGRAPH Posters*, pages 108:1–108:1, 2013.