Image-Space Illumination for Augmented Reality in Dynamic Environments

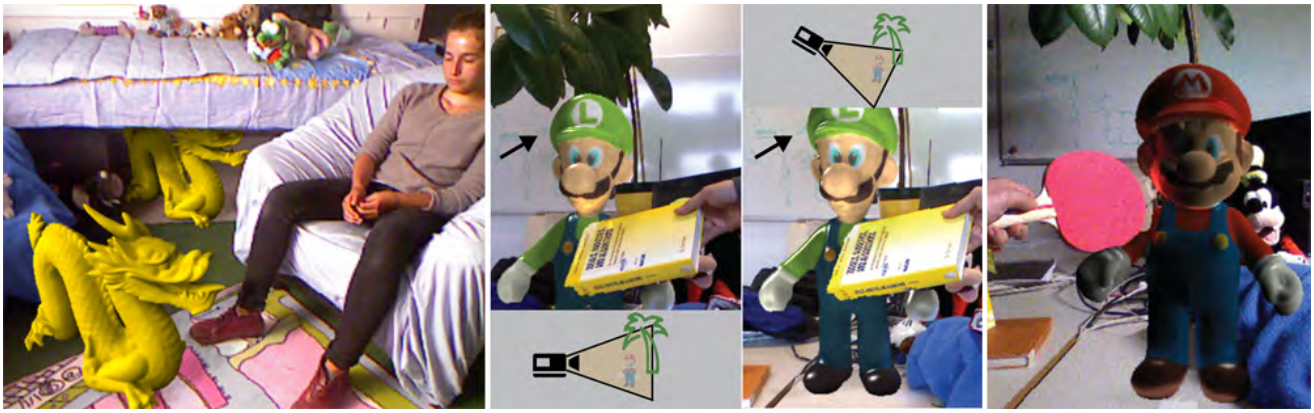


Figure 1: Augmented Reality lighting key features: (Left) Probeless light estimation and coherent Augemented Reality rendering in mid-sized dynamic environments. (Middle) Consistent shadowing from real and virtual occluders with objects moving out of the field of view. The character's head stays in shadow, although the plant, which casts a shadow on it, moves out of the field of view. (Right) User interaction supported by fast updating geometry (hand with paddle) and static geometry with indirect illumination. This work is the first than can deliver all illumination effects shown in the three images simultaneously.

Abstract

We present an efficient approach for probeless light estimation and coherent rendering of Augmented Reality in dynamic scenes. This approach can handle dynamically changing scene geometry and dynamically changing light sources in real time with a single mobile RGB-D sensor and without relying on an invasive lightprobe. We jointly filter both in-view dynamic geometry and outside-view static geometry. The resulting reconstruction provides the input for efficient global illumination computation in image-space. We demonstrate that our approach can deliver state-of-the-art Augmented Reality rendering effects for scenes that are more scalable and more dynamic than previous work.

Keywords: Augmented reality, photometric registration, radiance transfer

Index Terms: H.5.1 [Information Interfaces and Presentation]: Artificial, Augmented,Virtual Realities—;I.4.8 [Image Processing and Computer Vision]: Photometric registration—3D Reconstruction;I.3.3 [Computer Graphics]: Image Generation—Ray Tracing-Spherical Harmonics;

1 Introduction

Estimating real-world lighting and applying it to virtual objects is a key element of visually coherent rendering in Augmented Reality (AR). In the real world, shadows change, as people and objects move, and lighting changes, as lamps are switched on and off.

1.1 Common limitations of AR lighting

Previous work on AR lighting has made unrealistic assumptions about how dynamic changes in scene geometry and illumination are handled. These assumptions, which usually do not receive much discussion in the AR literature, impose rather severe restrictions on real-world AR applications such as games or home shopping.

Limitation to small rigid scenes A common approach is to limit AR lighting experiments to a small, carefully prepared and registered "desktop" scene with just a few tracked objects [17, 11]. This distracts from the fact that expensive rendering techniques such as recursive raytracing do not scale to larger scenes. Moreover, arbitrary scene changes are not supported, and the geometry outside the field of view is ignored.

Artificial lightprobes Lightprobes placed in the scene can capture dynamic lighting effects, but they clutter the scene, require additional equipment and capture only a small scene per lightprobe. AR constrained to a small scene after lots of preparations is rather impractical for end-users. With recent inexpensive RGB-D sensors, it is now feasible to rapidly obtain models of large static scenes [15] and also of smaller dynamic scenes [24], while simultaneouly tracking camera movements. The combination of reconstruction, tracking and imaging puts coherent rendering for AR in dynamic scenes within reach. However, previous approaches [18] have neither considered geometry outside the sensor's current field of view (FOV) nor dynamically changing light sources.

1.2 Practical AR light estimation requirements

Practical light estimation for mobile AR imposes several strong requirements, which must all be addressed simultaneously:

Probeless light estimation Mobile AR affords only a single RGB-D sensor.

Moderate computational requirements Since mobile devices are limited, the computation must scale well with scene size and complexity. Computational effort should be concentrated where a noticeable visual effect can be expected.

Incremental reconstruction of scene geometry The user must be free to explore new viewpoints. Requiring extensive scanning before starting the AR application is not practical.

Support for changing geometry and lighting While it is not possible to handle arbitrary changes that the RGB-D sensor cannot see, we can at least make optimal use of current and past observations, to address the widest possible variety of dynamic events.

1.3 Contribution

In this paper, we describe the first AR approach that handles all the requirements described in Section 1.2. We build on the idea that the real world inside the sensor's FOV, i. e., in image-space, should be treated differently to the world outside the FOV:

Joint object-space and image-space reconstruction We retain both a static geometry representation in object-space and a dynamic geometry representation in image-space, and jointly filter both representations to minimize sensor artifacts. Noise and holes in depth measurements are still a problem for inexpensive depth sensors targeting consumer markets. Our pipeline is independent to the data representation of the geometry reconstruction (e.g., volume or triangle meshes) and is open for many different reconstruction algorithm as long as a depth map is provided.

Light estimation from direct and indirect observations We maximize the opportunity for global light interaction by combining information from three sources in real-time: (1) The static geometric model outside the FOV, (2) the dynamic geometric model inside the FOV, (3) dynamic estimation of environment lighting from reflections observed in the scene inside the FOV [8]. This omits dynamic objects outside the FOV, which cannot be observed directly. However, small changes outside the FOV, which are relatively far away from the observer, will not be very noticeable. Large changes will likely have an influence on the observed reflections.

Scalable rendering High computational efficiency is achieved by approximating global illumination in image-space. We consider both the geometry inside and outside the FOV, so that radiance can be transfered across the FOV boundary. However, the efficiency is independent of the global scene complexity.

We show that this approach can handle complex and dynamic scenes, as encountered in real life. We demonstrate real-time performance on a desktop GPU and near real-time performance on a mobile GPU. Thus, we believe our work shows the first time fully dynamic illumination in AR.

2 Related work

An AR pipeline such as the one considered here consists of three main stages: geometry reconstruction, light estimation (including radiance transfer computation) and rendering. We discuss these topics in the following sections.

2.1 Geometry reconstruction

Real-time reconstruction using depth sensors has become a popular topic of research. One approach is object-space filtering, which continuously improves the reconstructed scene model by integrating depth images over time and space into a volumetric model [21], a 3D point cloud [15] or a 3D map composed of depth keyframes [16]. The main advantage of object-space filtering is the global scene model, which can represent geometry outside the current camera FOV. The main disadvantage is the inability to pick up fast scene changes, because robust integration requires time. Fast scene changes are classified as outliers and disregarded.

Image-space filtering processes only the current depth image and uses the result directly as a scene model for light estimation [18] and rendering [24]. The advantage of image-space filtering is that it instantaneously captures the current state of the scene. However, image-space filtering is more susceptible to sensor artifacts, cannot capture the scene outside the current FOV and requires additional effort for 3D camera tracking.

In this work, we combine object-space and image-space filtering to produce a scene model which is both smooth and dynamic. This idea is related to scene change detection from depth sensors, which has, for example, been proposed for model-based object tracking [9] and telepresence [19]. However, these works aim at object-space results, while we use the object-space only as an intermediate step and compute global illumination in image-space.

2.2 AR light estimation

AR lighting focuses on real-time applications requiring only minimal additional user input. This places them apart from post-production approaches, which require manual scene annotation [13] or extensive offline processing [14].

Real-time AR lighting often uses passive lightprobes, such as reflective spheres [3, 1], or active lightprobes, such as fisheye cameras [17, 6, 11]. While these methods provide excellent results, inserting lightprobes is a major hindrance.

Therefore, recent research has investigated probeless light estimation. For outdoor applications, knowledge of the user's geospatial position and trajectory of the sun allows to infer dominant lighting. Indoors, probeless approaches may assume static geometry and illumination of the real scene, and use a handheld RGB or RGB-D sensor to capture this information offline as a surface light-field. This approach can support environment light maps [4], non-diffuse materials [10] and even high-dynamic range [20]. While offline capturing can deal with noisy sensors through integration and smoothing, dynamic changes to geometry and light conditions cannot not be supported. Furthermore, the capturing is delicate and not end-user friendly.

Most related to our work are approaches which continuously estimate the environment light without lightprobes. This is more difficult than offline estimation, because light estimation is an ill-posed inverse rendering problem [22] and thus rather susceptible to sensor noise. Light estimation from the actual scene geometry at interactive rates has been shown by Gruber et al. [8]. Although their system supports a priori unknown geometry, the amount of dynamic scene changes is limited by the slowly updating reconstruction. Yao et al. [26] investigated a similar approach for estimating the environment light from a depth map, but do not deal with global illumination at all. Lensing et al [18] estimate indirect illumination from live depth images, but only for virtual light sources and only for geometry inside the current FOV.

2.3 AR rendering

A key concept for realistic illumination effects in AR is differential rendering, made popular by Debevec [2]. This technique computes global illumination for real and real+virtual separately, and applies the difference as a light map. Differential rendering can be combined with any popular global illumination technique, such as ray-tracing [11], reflective shadow maps [17, 18], irradiance caching on surfaces [12] or light propagation volumes [5].

However, all these techniques assume a small scene and artifact-free geometry. Exceptions are limited: Lensing et al. [18] filter geometry in image-space, but do not estimate real lighting, while Gruber et al. [8, 7] filter geometry in object-space, but do not compute indirect illumination.

To robustly support dynamic scenes with significant geometric artifacts, we adopt a deferred rendering approach. It has been shown that with deferred rendering, global illumination can be

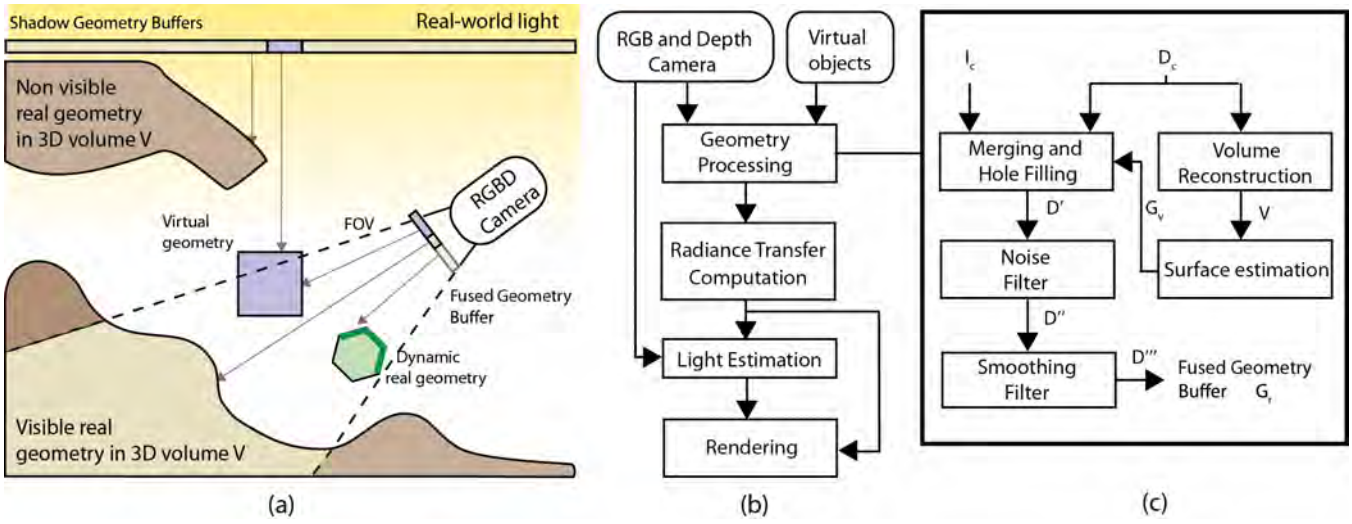


Figure 2: Working environment and data flow diagram. (a) Environment with geometry inside (light brown) and outside (dark brown) the camera FOV. Static real geometry (brown), dynamic real geometry (green) and virtual geometry (purple) are all projected into geometry buffers for occlusion computation. Additional shadow geometry buffers around the camera frustum provide further occlusion information. (b) Data flow of the overall AR pipeline. (c) Detailed data flow of the geometry reconstruction stage. The environment is captured in volume V, from which a geometry buffer (depth + normals) G_v is extracted and fused with the raw depth map D_c to capture fast geometry changes. The output of the final stage is the fused and filterd geometry buffer G_r.

computed by using a depth buffer instead of the scene geometry [25]. While this computation can only approximate true global illumination, it can be performed at high frame rates even for strong scene changes, as may occur when the depth buffer is obtained from an RGB-D sensor in an AR application.

3 Algorithm Overview

An overview of our pipeline is shown in Figure 2. The geometry processing stage (Section 4) represents the real-world as a volume V, which captures static or slowly updated global geometry. Filtering in object-space allows building a global scene model with high quality, and also conveniently provides camera tracking and re-localization abilities. However, fast object motion is suppressed in the filtering.

To react to moving objects, we extend the volumetric integration with a more agile filter. Rather than directly extracting surfaces from the volume, we use it only as a prior for image-space filtering of the raw depth image D_c. Dynamic information is taken from D_c, while static information is taken from V.

This approach can be seen as a variant of spatio-temporal depth filtering, where the temporal prior is taken from the volume rather than from the previous depth image [24]. The volumetric prior has multiple advantages over storing only a single depth image. First, the camera motion is already determined during the volumetric integration, so costly optical flow computation for motion compensation is not necessary. Second, in those regions where the image-space filtering relies on information from the volume, higher quality normals can be extracted. Third, the weights in the volume provide an additional trust measure for the subsequent image-space filtering. The result of the filtering is stored as a geometry buffer G_r.

The illumination stage (Section 5) performs image-space occlusion detection and first bounce estimation on G_r to compute direct and indirect radiance transfer. In addition to the main geometry buffer, auxiliary geometry buffers are strategically placed around the camera frustum to incorporate occlusion information from the entire environment. The result is stored in spherical harmonics (SH) form.

Image-space occlusion detection is fast and independent of global scene complexity, since it can use the filtered depth image directly without sampling any object-space geometry. The approximation error introduced by computing occlusion in image-space rather than in object-space is negligible compared to geometric reconstruction errors.

Radiance transfer computation is the most expensive step of the pipeline, but has two uses: First, distant environment illumination is estimated from observed reflections in the scene. Second, SH shading is used in the final differential rendering stage.

4 Geometry Reconstruction

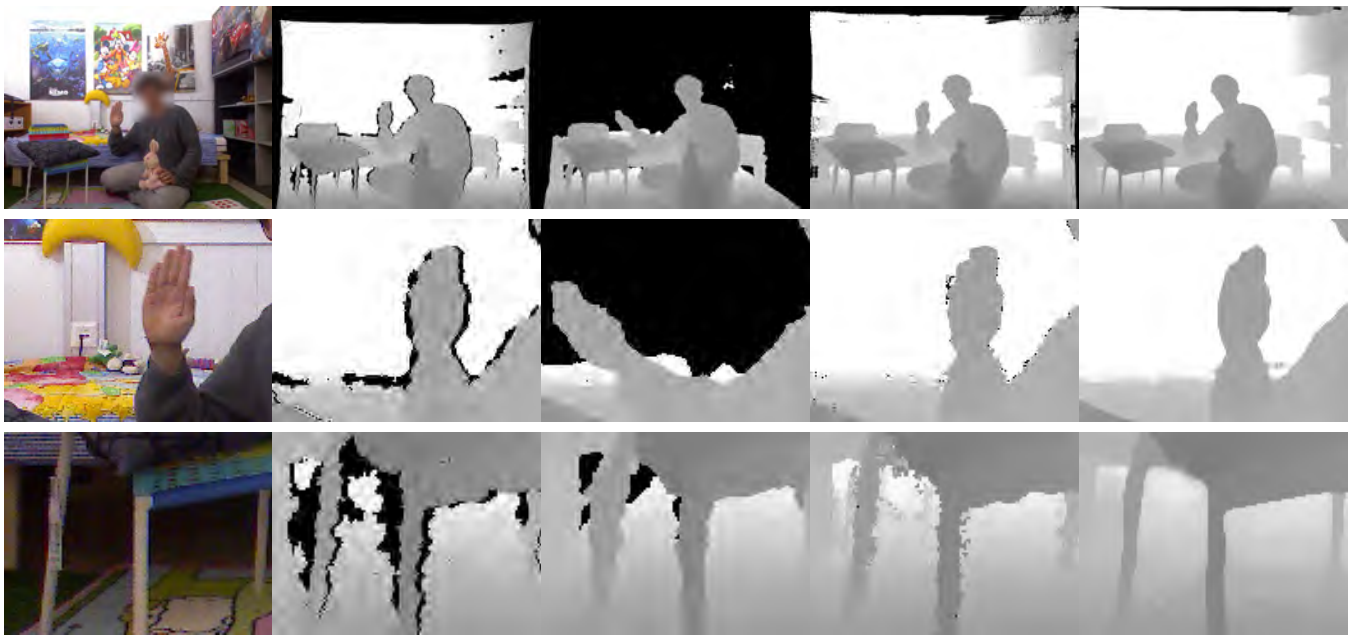
In the geometry reconstruction step, the raw data from the RGB-D sensor is processed into a global volume V and a filtered geometry buffer G_r, which represents the real scene in the current camera FOV. The geometry buffer contains vertex positions, surface normals and color. G_r and V are used as input to the following light estimation and rendering steps.

The raw depth image D_c contains artifacts, such as holes from missing depth measurements, noise, and poor alignment between color and depth channels. We address these problems by a filter chain designed to separate static and dynamic geometry, while suppressing undesired artifacts. The static part of the geometry, V, is obtained by conventional volumetric filtering [9]. This type of filtering is very robust, but discards observations of fast moving objects as outliers.

While updating V with D_c, we can extract a geometry buffer $G_v = (D_v, N_v)$ corresponding to the projection of V into the current view. The differences between D_c and D_v will be used to identify where the scene has changed with respect to V. In these regions, the depth information has to be taken from the raw image, even if it has imperfections.

To this aim, three filter passes in image-space are applied. A merging pass fuses D_c and D_v, a smoothing pass aligns depth and color edges, and a denoising pass cleans up erroneous measurements.

The merging pass D' selects among D_c and D_v according to a depth difference threshold λ_D. Non-missing pixels in D_c are directly compared to D_v; if the difference is within the threshold, then



(a) Color (b) Depth (c) Volume (d) Merged (e) Smoothed

Figure 3: Depth map filtering examples. Black pixels indicate missing depth measurements. Images (a), (b), (c) are inputs. Images (d) and (e) are outputs of the merging and smoothing filtering passes, respectively. The top row shows results of the specific case where the volume reconstruction lags behind the actual depth image from the sensor. This can be noticed by comparing the position of the hand in (b) and (c). The middle and bottom rows show close-up views of the top row. Note the significant improvements in depth edges along the borders of objects, such as the chair legs.

the more trustworthy depth D_v replaces D_c. To fill in missing depth pixels, we look at the valid pixels $\Omega_k(p)$ in a $k \times k$ region around a pixel p. We first determine a subset $\Omega_I(p) \subseteq \Omega_k(p)$ of pixels, for which the intensity difference to p lies within a threshold λ_I.

$$\Omega_I(p) = \{q_I \in \Omega_k(p), |I_c(p) - I_c(q_I)| < \lambda_I\} \quad (1)$$

This ensures that our support region does not cross an object boundary. We then find the subset $\Omega_D(p) \subseteq \Omega_I(p)$ of pixels for which the depth difference to the volume lies within a threshold λ_D.

$$\Omega_D(p) = \{q_D \in \Omega_I(p), |D_c(q_D) - D_v(q_D)| < \lambda_D\} \quad (2)$$

If the majority of inspected pixels in the support region are close in depth to the volume, then the depth value from the volume replaces the depth map input. Otherwise, the median depth of the support region is used.

$$D'(p) = \begin{cases} D_c(p), \text{ if } \exists D_c(p) \text{ and } |D_c(p) - D_v(p)| > \lambda_D \\ D_v(p), \text{ if } \exists D_c(p) \text{ and } |D_c(p) - D_v(p)| < \lambda_D \\ D_v(p), \text{ if } \nexists D_c(p) \text{ and } |\Omega_D(p)| \geq |\Omega_I(p)|/2 \\ \text{median}(\{D_c(q) : q \in \Omega_I(p)\}), \text{otherwise} \end{cases} \quad (3)$$

The denoising pass D'' is a joint bilaterial median filter with a smaller window, which operates on all pixels. This pass corrects registration errors between the depth image and intensity image.

$$D''(p) = \text{median}(\{D'(q) : q \in \Omega_I(p)\}) \quad (4)$$

Because computing a median over a large window is generally costly, we subsample from $k \times k$ to 5×5 before computing the median. This subsampling introduces some noise-related errors, which are cleaned up in the final pass D''' using a small filter window without subsampling. The intensity difference threshold is not needed in this last pass, because we assume that the depth edges are now correct and match the color edges.

$$D'''(p) = \text{median}(\{D''(q) : q \in \Omega_k(p)\}) \quad (5)$$

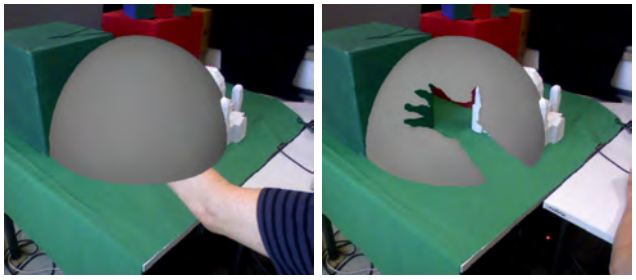
These filter passes are illustrated in Figure 3. A person is waving a hand in front of the camera. Because volumetric integration takes time, the hand is in the wrong place in the volume. In this case, the merging pass chooses samples from D_c for the hand. Joint color and depth filtering significantly improves depth edges from both the volume and the depth map, as can be observed on the thin chair legs, which are neither well represented in D_c nor in D_v.

Figure 4 demonstrates our algorithm for occlusion handling. The two top rows explain the possible errors created by a slower updating geometry reconstruction. In Figure 4 (c), the result of applying our merge and filtering algorithm is shown. Figure 4 (d) demonstrates the case where only the volumetric reconstruction is used.

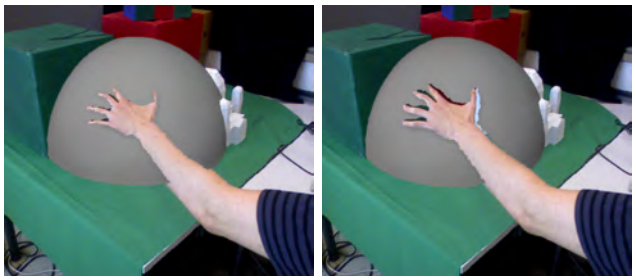
5 Illumination

Global illumination for large scenes is a challenging task, because every surface point could interact with every other surface point. Apart from the computational cost, incremental reconstruction may present situations where important surfaces have not even been scanned yet. Real-time global illumination techniques which rely on pre-computation cannot be used, since we allow both light changes and non-rigid geometry changes. Overall, we need to compute the possible radiance transfer across the scene's surfaces at real-time rates.

Computing radiance transfer in image-space improves performance, which is necessary to keep up with the dynamically changing environment. We need the radiance transfer for two purposes:



(a) (b)



(c) (d)

Figure 4: Top row figures show erroneous occlusion handling, where the volumetric reconstruction lagged behind. (a) The inserted arm is not reconstructed in time. (b) After the arm has been removed, the lagging reconstruction still creates wrong occlusions. (c) Our depth merging and filtering algorithm creates correct occlusions. (d) The result of a converged volumetric reconstruction is shown for comparison.

First, we combine observed reflections in the RGB image with per-pixel radiance transfer to estimate the real-world environment light via inverse rendering. Second, when combining first order and second order radiance transfer, a fast deferred shading pass is sufficient to produce the most significant global illumination effects.

As will be shown in the results section, this approach can produce appealing soft shadows and color bleeding. Nevertheless, the approximate light estimation and SH compression support only diffuse light transport (no mirrors), only directional light sources (no desktop lamps) and only white light sources (no dancefloors). We found these limitations acceptable for practical AR applications.

5.1 Radiance transfer computation

Diffuse reflection R_d on a surface point x with normal $n(x)$ and albedo $A(x)$ is an integral over the incoming radiance L from all possible directions ω_j, weighted by incident angle and the visibility $K(x,\omega_j)$. With discrete sampling in N directions, we write:

$$R_d(x) = A(x) \sum_{j=1}^{N} K(x,\omega_j) L(\omega_j)(\omega_j \cdot n(x))\, d\omega_j \tag{6}$$

Instead of computing the visibility term K by expensive raytracing, we approximate it with a variant of screen-space directional occlusion (SSDO) [25], where visibility is computed by spherically sampling the space near x. A sample point y_j in object space is computed by adding a pseudo-random displacement to x along the specific direction ω_j. The point y_j is then projected into the image-space aligned geometry buffer to look up the actual position s_j on the surface. If s_j is closer to the camera than x, we assume that light is blocked towards x along direction ω_j. Unlike standard SSDO [25], we do not instantly compute the lighting for each ray direction based on a known incident light source, but rather project the visibility result into per-pixel SH for deferred light estimation and rendering. Such an approach is faithful to the SSDO idea of *coupled* ambient occlusion and directional lighting, and results in more realistic rendering results.

For projecting radiance transfer into spherical harmonics functions efficiently, we pre-compute the SH weights for all N visibility directions. A naive visibility test delivers a binary result, which is 0 only if a sample does not pass the depth test. Wrong visibility quantization can lead to over-estimation of occlusion, making shadows too dark. We account for visibility quantization errors by modeling K as a continuous value, which rises from 0 to 1 with the angle between ω_j and the normalized vector t_j from x to s_j. Figure 5(c) and (d) show how false self-occlusion can be successfully reduced.

$$t_j = (s_j - x)/|s_j - x| \tag{7}$$

$$K(x,\omega_j) = min(\chi, 1-(t_j \cdot \omega_j))/\chi \tag{8}$$

The threshold χ in Equation 8 is related to the angle between adjacent sampling directions. We account for the jittered random sampling by using the median of all angles σ_{ij} between neighboring samples ω_i and ω_j, as determined from a spherical Delauney triangulation Ψ of the sampling. To account for the Nyquist limit, we determine χ from half of this angle:

$$\chi = \cos(\text{median}(\alpha_{ij} \in \Psi)/2) \tag{9}$$

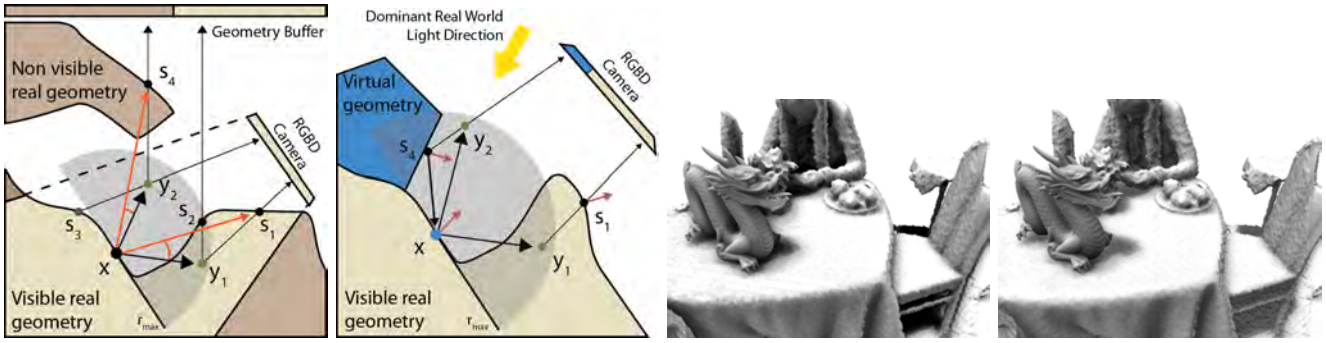
Radiance transfer is computed *from both* visible and invisible geometry, but *only to* visible geometry. This optimization is enabled by creating additional geometry buffers extracted from V with orthographic projection. We approximate a partial cubic map with the main buffer and three additional buffers for the left, right and top parts of the scene. Additionally we add a buffer from the dominant light direction directly extracted from the real-world light estimation represented in SH coefficients. All camera positions are assumed in world coordinates, so the sample points y_j for can be tested for visibility in each buffer directly.

This variant of visibility approximation may miss thin occluding objects in the scene and is only truly correct for the rays which are facing towards the camera. However, our evaluation shows that this approximation suffices for our purpose. Compressing the resulting radiance transfer into SH reduces storage and computation requirements for the remaining pipeline, and also conveniently enforces low-pass filtering on the light estimation, which is necessary for robust extraction from unreliable scene reflections. Visibility sampling is improved by precomputing random offsets for an area of 2×2 pixels rather than a single one.

To meet mobile hardware requirements we implemented a variation of our approach (OursFast) accelerating the radiance transfer computation by sub-sampling in image space. In the differential rendering step we use bicubic interpolation and standard Gauss filtering with a kernel radius of eight for up-sampling. In Figure 9 we compare the visual results of Gruber et al. [7] based on volume raytracing against our approach with full radiance transfer sampling and our fast approach with 4×4 sub-sampling.

5.2 Light estimation

With the pixel intensities of the RGB image $I_c(x)$ and the radiance transfer $R_d(x,z)$ in SH form (where z denotes the index of the SH coefficient), we can now recover the incoming light L in SH form [8], denoted $l(z)$. We set up a radiance transfer matrix $T = [R_d(x,z)]$ and evaluate an overdetermined linear equation system $T \cdot l = I_c$ for l in the least squares sense. In Figure 6, we visualize different stages of our light estimation pipeline.



(a) Visibility (b) Indirect illumination (c) No correction (d) Corrected visibility

Figure 5: (a) Visibility testing for point x. A sample point y_1 is tested by comparing its depth to the projection onto the surface, s_1. s_1 is closer to the camera image plane than x, so x is occluded along this direction. A second sample y_2 passes the depth test in the main camera buffer, but is occluded in the auxiliary geometry buffer covering the top view. The red arrows visualize the actual direction t_j from x to s_j and the angular distance to the visibility direction ω_j, which has to lie inside the threshold χ. (b) Indirect lighting is computed, if an occlusion is detected. For example, s_4 reflects light towards x, but s_1 faces away from x and hence does not reflect light. (c) Without corrected visibility testing, occluded areas are too dark. (d) Correcting for visibility testing results in more realistic shadows and allows light to reach small corners.

Compared to volume ray-tracing [8], the light estimation quality remains essentially the same, but our overall approach enables more convincing shadowing effects. This is mainly due to the ray length limitation of volume ray-tracing imposed by the high computational cost. Our approach is less accurate, but can cover the entire scene and therefore capture all relevant shadowing effects.

5.3 Indirect illumination

Indirect illumination R_i from one light bounce can easily be incorporated into SSDO. When a sample y_j near x is found invisible, the light transport from the corresponding surface point s_j to x is approximated. R_i is computed as a diffuse reflection from a single dominant light direction L_D without further visibility testing.

$$R_i(x) = \sum_j (1 - K(x,\omega_j))(n(s_j)\cdot L_D)F(x,j)G(x,j) \qquad (10)$$

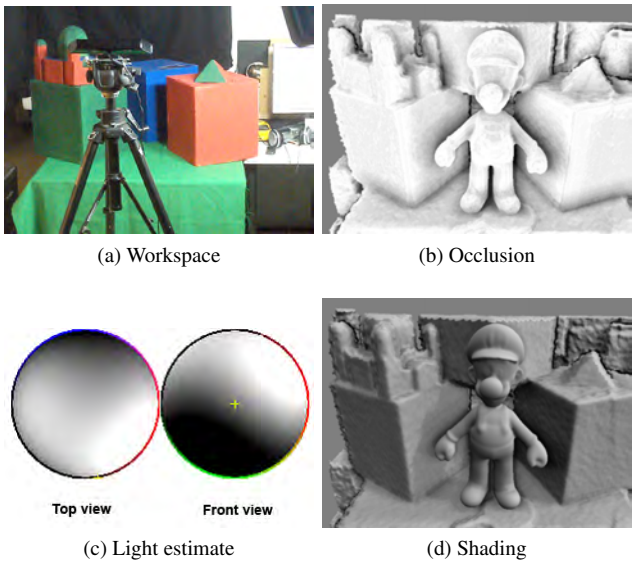
The indirect illumination is weighted by a "form factor" term $F(x,j)$, which computes Lambertian weights for the angles between the normalized transmittance direction $-t_j$ and the normals $n(x)$ and $n(s_j)$, respectively, and corrects for the distance from x to s_j. The term $G(x,j)$ suppresses light transfer from points facing away from x:

$$F(x,j) = \frac{(n(x)\cdot t_j)(n(s_j)\cdot(-t_j))}{(x-s_j)^2\pi} \qquad (11)$$

$$G(x,j) = \max(0, sgn(n(x)\cdot(-t_j))) \qquad (12)$$

5.4 Differential rendering

Differential rendering requires computing radiance transfer two times, once for the real world and once for the combination of the real world and the virtual world. In our approach, we compute differential rendering for directional lighting with occlusions and first bounce indirect lighting separately. For directional lighting with occlusions, we exploit the assumption that our estimated real-world light is white, computing two monochromatic differential rendering buffers R_d for real and R'_d for real+virtual. Geometry data for occlusion processing of real+virtual is obtained by rasterizing virtual objects with a standard depth test into the filtered geometry buffers. Using radiance transfer in SH form lets us evaluate the shading R_d



(a) Workspace (b) Occlusion (c) Light estimate (d) Shading

Figure 6: Different stages of the light estimation. (a) Overview of the setup. (b) Occlusions in image-space. (c) Top and front view of the light estimation projected onto a sphere. (d) Direct diffuse lighting from light estimation and surface radiance transfer

and R'_d as a simple dot product of l and T. The directional differential rendering result I_d is computed as follows:

$$I_d = I_c(1 + T(R'_d) - T(R_d)) \qquad (13)$$

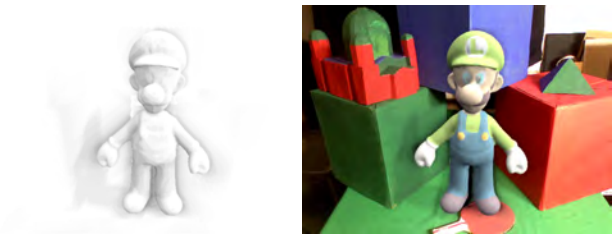
In Equation 13, we add the difference between the two differential rendering buffers (conf. 7(a) and (b)) to the color camera input image I_c. To support all frequencies, we have to multiply the intensity buffers R_d and R'_d with the color information of I_c first. Note that we apply a tone mapper [23] T to R_d and R'_d based on the average intensity of each buffer. This is necessary because our differential rendering buffers are computed in high dynamic range, while the input data I_c originates from a low dynamic range camera.

For adding indirect illumination to differential rendering, we compute analogue to the previous pass (see Section 5.3) the indirect illumination buffers R_i and R'_i. Extracting the dominant light direction L_D, as discussed in Section 5, is solved by using the second, third and fourth SH coefficient of l. While l is monochromatic, indirect illumination considers colored light transport based on the albedo $A(s_j)$ of the sample point. However, for differential rendering, we only want to add those pixels to the final result which come from the light interaction between real and virtual, R'_i, and exclude pixels which are affected from the real-world only. Therefore, in Equation 14, we select the right pixels from R'_i.

$$I_i = \begin{cases} R'_i, & \text{if } R'_i \neq R_i \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

The final AR image I_{final} is computed by adding the differential rendering result from the combination of direct lighting I_d and indirect lighting I_i (see Figure 7(c, d, e)).

$$I_{final} = I_d + I_i \tag{15}$$



(a) Differential buffer (b) Final rendering

(c) No ind. illumination (d) Ind. illum. only (e) Applied ind. illum.

Figure 7: Top row: (a) Differential occlusion buffer, (b) differential rendering applied to camera image. Bottom row: (c) Scene without indirect illumination, (d) Indirect illumination added to the diffusely shaded buffer. (e) Result with indirect illumination. Note the subtle effects of indirect illumination on natural diffuse surfaces.

6 Evaluation

6.1 Implementation parameters

The majority of the pipeline is executed on the GPU using CUDA. The Kinect delivers color and depth frames at 640x480, which is also the final rendering resolution. Our standard workspace size measures 2x2x2m and is reconstructed into a 256^3 voxel volume. For the light estimation, we employed SH with four bands (16 coefficients). Filter parameters for geometry fusion were chosen as follows: D': $k = 20, \lambda_I = 7, \lambda_D = 5$cm, D'': $k = 8, \lambda_I = 7$, D''': $k = 2$.

6.2 Light estimation

In this work we evaluate the light estimation quality comparing the dominant light directions of each method. The dominant light direction is extracted from the environment light estimation given in SH and has a great impact on the final rendering. We compared two configurations our image-space approach (Ours: full sampling and OursFast: quarter sub-sampling in image space) against a standard AR light estimation method (reference method) based on a diffuse spherical light probe. Additionally we evaluated the approach of Gruber et al. [8] (GR12). Since the reference method based on passive light probes naturally can suffer from light probe tracking, we additionally captured the entire testing setup with an omnidirectional HDR camera (PointGrey Ladybug3). We placed the omnidirectional camera at the same position of the passive light probe and manually registered it to the common world coordinate system. We obtained environment images of our testing setup directly sensing the light sources (see top row of Figure 10). Our test data set consists of six real-world scenes with increasing complexity of geometry and textural elements (Figure 10 bottom row). We installed four area light sources arranged in a half circle around the center of the test setup. We recorded RGB-D sequences with a Kinect V1 for each scene and light source with repeating camera movements, where each sequence has 1000 frames. In total, we recorded and evaluated 24000 frames.

In Figure 10, we show two types of results. The first results are the median value of all measured dominant light directions of all three evaluated methods and the reference method (yellow). We visualized the median points on the panoramic images from the omnidirectional cameras. As can be seen, the standard reference method is more accurate, but overall, all methods estimate the dominant light direction quite successfully. Note that the results of Ours and OursFast are very similar and therefor overlap. For the case of Light 4, the reference method clearly provided better results. This is due to the fact that the reference method has perfect surface normals as input, compared to the methods based on geometry reconstruction. If the camera does not sense enough information from the scene geometry, it is possible that certain surface normals are not sampled, which leads to biased light estimations.

The box plots in Figure 8 show the angular distance of our methods and [8] to the reference method. We visualized the mean (red bar) for each scene over all light sources In scene 1 we have a sphere as input geometry (same geometry as a light probe) and directly compare results between perfect input data (reference model) and input data from the geometry reconstruction. In scenes where the geometry is rather limited and contains different colors, such as Scene 3 in Figure 8, the light estimation algorithms deviate more. The best results are obtained in Scenes 5 and 6, where the surface geometry provides sufficient input for the light estimation. Overall, our image based methods have the same quality of light estimation results as the volume raytracing method.

6.3 Performance

We measured the performance of time-critical passes from the entire system. The major passes are labeled and described as follows: *Input*: The input data processing consists of capturing RGB and depth data and applying a noise filter to the RGB image. *Reconstruct*: With Kinect Fusion [21], we compute the geometry reconstruction and estimate the 6DOF camera pose. *Surface*: The implicit surface from the volume modeled as a truncated signed distance function (TSDF) is extracted by ray-casting the volume, which also includes the surface extraction for the occlusion buffers. *DynGeo*: Our dynamic geometry processing algorithm. *RadTransfer*: Radiance transfer computation (RT) differs on the principal method. These are based on ray-tracing or image space directional occlusion. *LightEst*: Light estimation depends on the number of samples consisting of the per pixel radiance transfer in SH and the

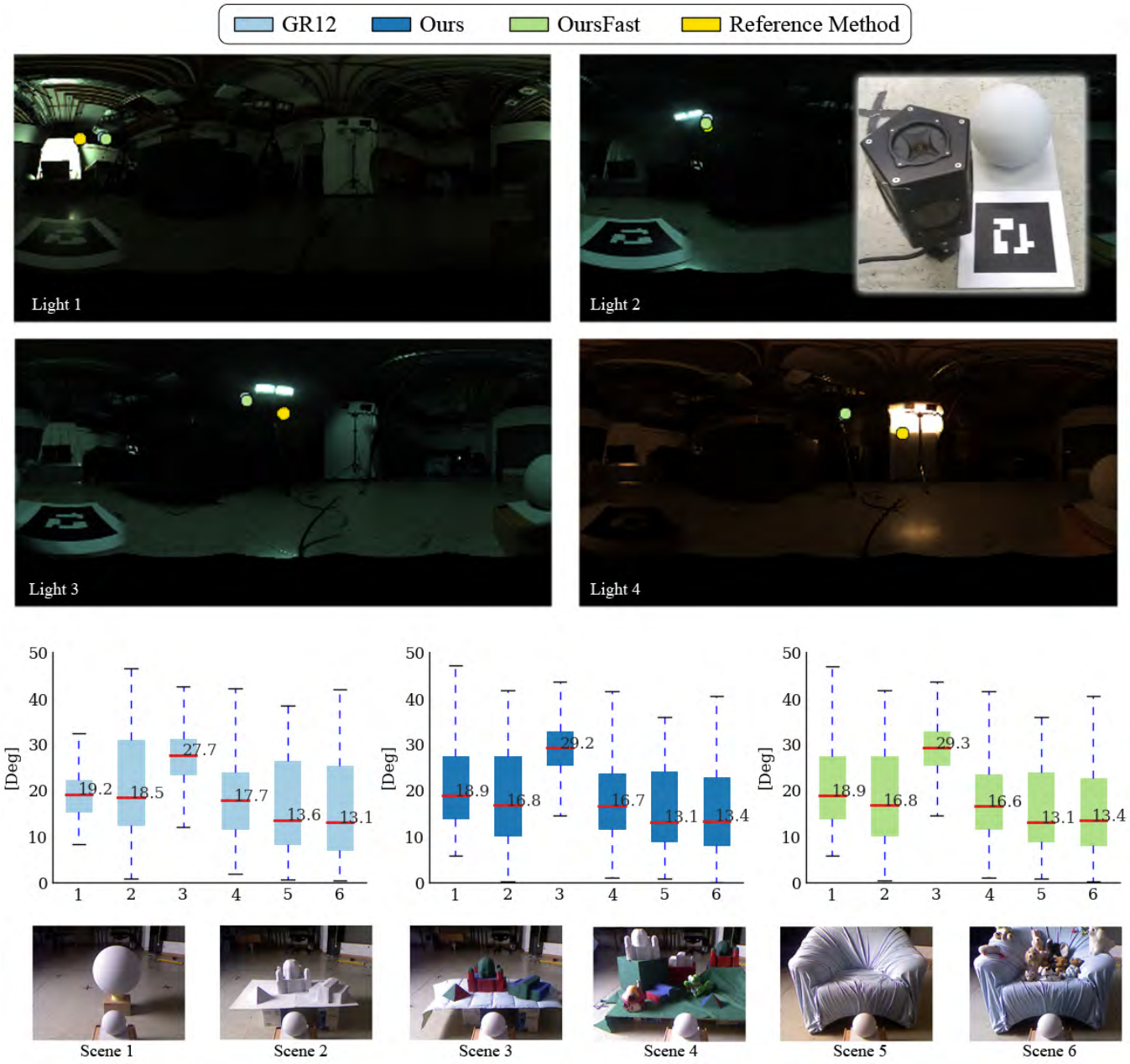


Figure 8: The top row images show environment maps of the testing setup with each light source. The environment maps are created with the omnidirectional Ladybug3 camera shown in the inset at the right. Each colored dot visualizes the median of the angular distance of the dominant light direction of each method to the reference method (yellow), drawn on top of the environment maps. For the reference method we used a diffuse gray painted sphere, which meets best our diffuse gray world assumption. The registration of the light probe was solved by using an ARToolkit marker, which also defines the origin of our test setup. The middle row shows the median of the angular distance of the dominant light directions of each method to the reference method. Each column in the plots correspond to the scenes in the bottom row.

intensity image from the camera. *Rendering*: The rendering pass includes rasterization of the virtual objects, differential rendering and AR compositing.

In a first test (Table 1), we compared GR12, which is based on volume ray-tracing, against our method, which is based on image-based occlusion techniques, on a desktop computer (PC) with an NVIDIA GeForce 780. Both methods regularly sample the entire image space. Besides GR14, we compared GR14 [7], which is based on a 4×4 regular sub-sampling in image space and adaptive edge refinement, against our variant of our method, which also uses a 4×4 regular sub-sampling in image space (OursFast). The size of the working volume is $2m^3$ with a 256^3 voxels.

In a second test (Table 2), we compared the performance of the PC to a notebook (NB) with an NVIDIA Quadro K2100M and a tablet (M) with an NVIDIA GT640M LE. To meet the hardware limitations of NB and M, we reduced the reconstruction to 64^3 voxels and omitted D''. The methods based on volume ray-tracing (GR12 and GR14) did not achieve frame rates above 1Hz and have

been left out in this evaluation. All timings, except from frames per seconds (FPS) are in milliseconds.

	GR12	Ours	GR14	OursFast
FPS	5.8	13.98	12.29	22.46
Update	172.45	71.5	81.36	44.53
Input	7.3	7.06	7.44	7.02
Reconstruct	11.1	10.9	10.24	10.64
Surface	6.69	12.57	6.69	12.6
DynGeo	0	1.19	0	1.02
RadTransfer	130.48	36.22	50.54	10.5
LightEst	3.86	2.67	5.27	1.8
Rendering	0.92	0.86	0.98	0.87

Table 1: Performance evaluation of our light estimation and rendering system against previous work based on volume ray-tracing. The results show strong performance improvements for the radiance transfer computation.

	Ours			OursFast		
	PC	NB	M	PC	NB	M
FPS	18.24	3.23	2.19	33.60	7.34	5.0
Update	54.8	309.86	456.78	29.76	136.24	200.48
Input	0.8	0.86	1.29	0.79	0.81	0.9
Reconstruct	9.42	37.68	66.0	9.42	36.86	66.16
Surface	5.28	30.73	45.68	5.28	30.02	44.99
DynGeo	1.27	18.25	25.77	1.12	17.89	25.75
RadTransfer	34.57	214.27	314.68	9.96	38.69	59.17
Light Est.	2.56	4.56	1.98	1.93	8.52	2.24
Rendering	0.89	3.52	1.36	1.24	3.43	1.26

Table 2: In this table, we show the performance measurements on a mobile device (M) compared to the values measured from a Notebook (NB) and a PC. We achieve interactive frame rates with the performance optimized variant (OursFast) of our algorithm. A major bottleneck for the mobile device is volume processing including reconstruction and surface evaluation. However, this could be, for example, substituted by a state of the art mobile visual SLAM system. The lower resolution of the volume does not affect camera tracking or light estimation, but can create visual artifacts at shadows.

6.4 Limitations

Obvious limitations of the geometry reconstruction are depth range and quality of the sensor and system memory. Visual artifacts can arise because, the entire scene has not been sensed and important occlusions cannot be computed. For example, when only the front of a table is reconstructed, this would create missing shadows under the table. However, this is a fundamental limitation of any single-camera approach. Moreover, the geometry from the depth image captures only the visible surface and not the entire volume of an object, which can also cause shadowing errors.

We do not account explicitly for the material color in the scene and assume that the real-world lighting is white. A real-world scene configuration with difficult material color distributions – for example, consider a couch with black-and-white stripes – could cause poor light estimation results. However, in our experience, these configurations are rather rare. We do not estimate the exact BRDF of the real-world geometry and are thus restricted to rather diffuse materials.

7 Results

Figure 9 compares the two variants of our approach against related work. In Figure 10 and

8 Conclusion and Future Work

Our work enables light estimation and photorealistic AR rendering from dynamic scenes. It also supports user interactions with the scene, such as moving hands and movable objects.

By combining object-space and image-space filtering, we produce a consistent reconstruction of dynamic geometry for AR light estimation and rendering. We show that we can successfully estimate environment light using a fast global illumination approximation in image-space. Our approach supports fast visibility determination covering the entire scene, and, therefore, produces consistent shadowing effects, even with geometry outside the FOV. The light estimation is stable, despite dynamic scenes and real-time operation. Furthermore, we use the real-world light estimation to add indirect illumination between real and virtual objects.

In future work, we would like to adapt our approach to use other scene reconstruction methods, such as point cloud fusion [15], which promises more lightweight and scalable reconstruction. Beyond recovering the geometry and lighting, we are also interested in real-time estimation of surface material parameters.

References

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

[2] 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. ACM, 1998. ACM ID: 280864.

[3] P. Debevec, A. Wenger, C. Tchou, A. Gardner, J. Waese, and T. Hawkins. A lighting reproduction approach to Live-Action compositing. Technical report, July 2002.

[4] S. DiVerdi, J. Wither, and T. Hollerer. Envisor: Online environment map construction for mixed reality. In *Virtual Reality Conference, 2008. VR'08. IEEE*, pages 19–26. IEEE, 2008.

[5] T. Franke. Delta light propagation volumes for mixed reality. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 125–132, Oct 2013.

[6] T. Grosch, S. Mueller, and W. Kresse. Goniometric light reconstruction for augmented reality image synthesis. In *Proc. GI Jahrestagung*, Frankfurt, Germany, 2003.

[7] L. Gruber, T. Langlotz, P. Sen, T. Hoherer, and D. Schmalstieg. Efficient and robust radiance transfer for probeless photorealistic augmented reality. In *2014 IEEE Virtual Reality, VR 2014, Minneapolis, MN, USA, March 29 - April 2, 2014*, pages 15–20, 2014.

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

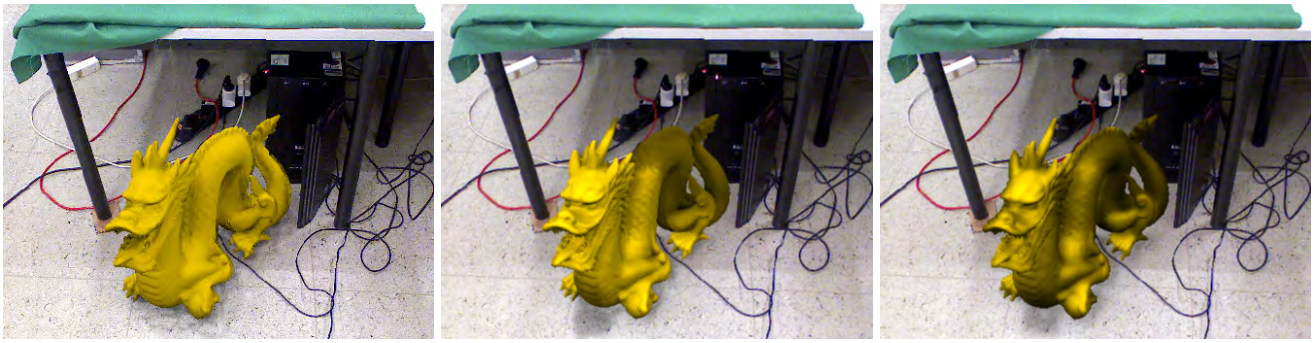
[9] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *UIST'11*, 2011.

[10] 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, 2012.

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

[12] P. Kan and H. Kaufmann. Differential irradiance caching for fast high-quality light transport between virtual and real worlds. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 133–141, Oct 2013.

[13] 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.



(a) GR14 (b) Ours (c) OursFast

Figure 9: Differential rendering results: (a) Gruber et al. [7] (b) Our approach (c) Fast variant of our approach based on sub-sampling in image space. Due to the limitations of expensive raytracing in (a), rays have a limited length and do not reach all surfaces. This is in contrast to (b) and (c) which capture lighting and shadowing effects from more distant occluders.

Figure 10: This figure shows a sequence of images from a video. The main light of the real world lighting comes from the left. The virtual character has a shiny and reflective material, as indicated by the specular highlights and rather glossy interreflections. From left to right: The yellow book sends reflections to the right hand of the character. The book is removed. The right hand receives indirect illumination from the couch. The book is now waved through the character to demonstrate the dynamic geometry reconstruction capabilities of our approach, providing correct occlusions between real and virtual and supporting indirect illumination effects.

[14] K. Karsch, K. Sunkavalli, S. Hadap, N. Carr, H. Jin, R. Fonte, M. Sittig, and D. Forsyth. Automatic scene inference for 3d object compositing. *ACM Trans. Graph.*, 33(3), June 2014.

[15] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-Time 3D Reconstruction in Dynamic Scenes Using Point-Based Fusion. In *2013 International Conference on 3D Vision (3DV)*, 2013.

[16] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*, 2013.

[17] M. Knecht, C. Traxler, O. Mattausch, W. Purgathofer, and M. Wimmer. Differential instant radiosity for mixed reality. In *2010 9th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 99–107. IEEE, Oct. 2010.

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

[19] A. Maimone and H. Fuchs. Encumbrance-free telepresence system with real-time 3D capture and display using commodity depth cameras. In *ISMAR*, 2011.

[20] M. Meilland, C. Barat, and A. I. Comport. 3d high dynamic range dense visual slam and its application to real-time object re-lighting. In *ISMAR*, pages 143–152, 2013.

[21] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, page 127136, Washington, DC, USA, 2011. IEEE Computer Society.

[22] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 117–128. ACM, 2001. ACM ID: 383271.

[23] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. *ACM Trans. Graph.*, 21(3):267–276, July 2002.

[24] C. Richardt, C. Stoll, N. A. Dodgson, H.-P. Seidel, and C. Theobalt. Coherent spatiotemporal filtering, upsampling and rendering of RGBZ videos. In *Eurographics*, 2012.

[25] T. Ritschel, T. Grosch, and H.-P. Seidel. Approximating Dynamic Global Illumination in Screen Space. In *Proceedings ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2009.

[26] Y. Yao, H. Kawamura, and A. Kojima. Shading derivation from an unspecified object for augmented reality. In *2012 21st International Conference on Pattern Recognition (ICPR)*, pages 57–60, 2012.