Real-Time Photometric Registration from Arbitrary Geometry

Lukas Gruber* Thomas Richter-Trummer† Dieter Schmalstieg‡

Graz University of Technology, Austria

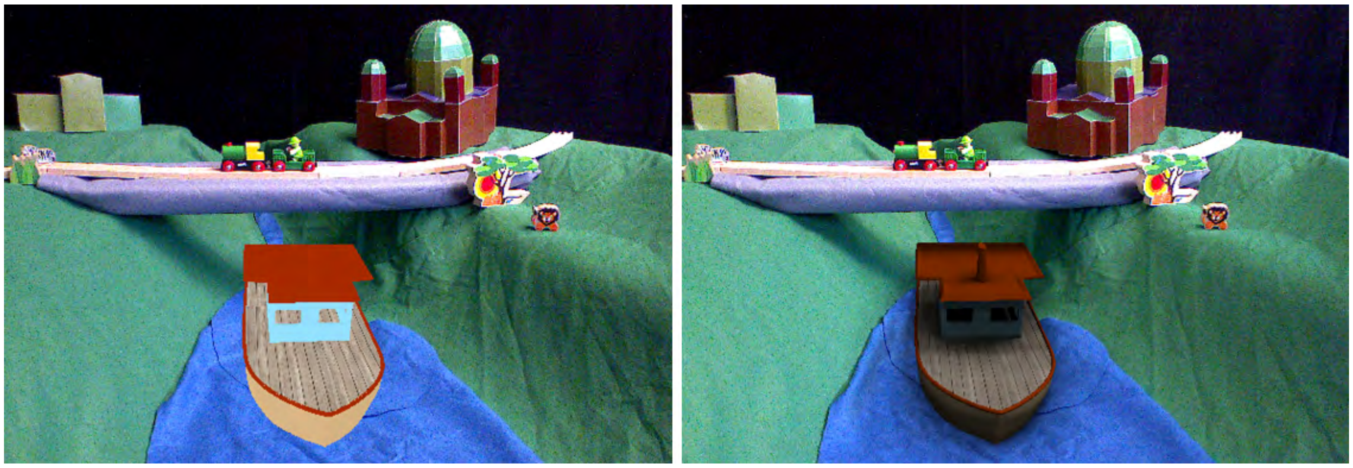

Figure 1: Our system estimates dynamically changing environment lighting from an unknown and also dynamically changing scene using a moving RGBD camera without any additional manual user input or artificial light probes. The left figure above shows a virtual ship without lighting. The right figure shows the same object lit by our real-world lighting estimation.

Abstract

Visually coherent rendering for augmented reality is concerned with seamlessly blending the virtual world and the real world in real-time. One challenge in achieving this is the correct handling of lighting. We are interested in applying real-world light to virtual objects, and compute the interaction of light between virtual and real. This implies the measurement of the real-world lighting, also known as photometric registration. So far, photometric registration has mainly been done through capturing images with artificial light probes, such as mirror balls or planar markers, or by using high dynamic range cameras with fish-eye lenses. In this paper, we present a novel non-invasive system, using arbitrary scene geometry as a light probe for photometric registration, and a general AR rendering pipeline supporting real-time global illumination techniques. Based on state of the art real-time geometric reconstruction, we show how to robustly extract data for photometric registration to compute a realistic representation of the real-world diffuse lighting. Our approach estimates the light from observations of the reconstructed model and is based on spherical harmonics, enabling plausible illumination such as soft shadows, in a mixed virtual-real rendering pipeline.

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—Spherical Harmonics;

*e-mail:lgruber@icg.tugraz.at
†e-mail:trt@surface.at
‡e-mail:schmalstieg@icg.tugraz.at

IEEE International Symposium on Mixed and Augmented Reality 2012
Science and Technology Proceedings
5 - 8 November 2012, Atlanta, Georgia
978-1-4673-4661-0/12/$31.00 ©2012 IEEE

1 Introduction

Augmented Reality (AR) is frequently realized using a video see-through display, which renders virtual content registered in 3D on top of a camera feed, which represents the real world. To create a convincing AR experience, the virtual and the real world have to be lit with the same light model. Ideally, the spectator should not be able to tell the difference between virtual and real. This is what is also known as visual coherence.

Visual coherent rendering has been a topic of computer graphics and film production for a long time. Pioneering work in merging real video images with computer generated images has been published as early as 1993 [4]. The main distinguishing aspect of AR over film is that visually coherent renderings must be generated in real time and with a limited amount of preparation. This turns out to be a difficult problem, and thus coherent rendering is still not a standard feature of commercial AR applications.

1.1 Problem statement

In order to deliver visually coherent rendering in AR, two main problems must be addressed: registration and plausible illumination. The first problem is concerned with measuring the environment. Estimating the geometry through reconstruction allows correct occlusion rendering. Recent results in online reconstruction [13] have shown that geometry acquisition can be achieved without tedious preparations, which has important implications concerning the feasibility of high-quality AR.

A related topic addressed in this paper is the real-time estimation of the lighting environment. For AR, this means observing the result of real world lighting in every camera image and solving an inverse lighting problem based on these observations, all within a single frame's time.

Knowledge of the real-world lighting is a necessary input to the second problem: plausible illumination through lighting and shadowing in AR. Once the real-world lighting is known, it is possible to consistently light virtual and real objects, and allow consistent interaction of shadows between virtual and real objects. Recently there has been significant progress in differential rendering, which addresses the global illumination problem in AR scenes [10]. However, these approaches perform the real-time estimation of the lighting environment in an invasive manner, by placing auxiliary light probes such as a mirror ball or an omnidirectional camera in the scene. This limits the application of visually coherent rendering to situations where the auxiliary objects can be hidden or tolerated.

1.2 Contribution

In this paper, we present a novel interactive light estimation and rendering pipeline, which recovers the environment light directly from observations in the scene without any special and artificial light props. The solution of the light estimation is directly used in the AR rendering pipeline based on spherical harmonics (SH) and volume ray casting. Furthermore, we introduce a method to improve the robustness of the light estimation and systematically evaluate our system with synthetic and real-world data. For better comparison, we classify our contribution after the criteria proposed by Jacobs and Loscos [8]:

Amount of realism: Our rendering pipeline achieves *perceptively plausible lighting results*, using an SH representation for directional illumination and for radiance transfer on surfaces of scene objects. All computations run at interactive frame rates on the GPU, allowing fully dynamic situations with moving light sources, cameras and objects. Object occlusions are considered using real-time dense reconstruction as proposed in [13]. Differential rendering [2] is used to model global illumination effects of real to virtual and vice versa.

Input requirements: The minimal amount of raw input data required for the lighting estimation is an RGB camera image and a depth map. The depth map can conveniently be obtained by a depth sensor or a stereo camera system. In contrast to previous work, we *do not require auxiliary measures* to dynamically react to illumination changes: No mirror balls or omni-directional cameras are required. Moreover, the geometry used for the estimation is not restricted to planar objects as in [15]. Due to the ambiguity problem of surface texture color and light color, we restrict the light color to be white. The material reflectance is assumed to be Lambertian with arbitrary surface color and texture. The geometry can change dynamically. We do not require any pre-computation steps, which have to be done off-line.

Processing time: Our approach runs in interactive frame rates on consumer hardware.

Level of automation: The user does not have to provide any manual input during runtime.

Level of interaction: Our method provides a high level of interaction. The user is able to move camera and objects in the scene and change the lighting situation dynamically.

Overall, this work is the first solution of real-time photometric registration without artificial props or manual user input and global illumination techniques.

2 Related work

2.1 Inverse rendering

Inverse rendering is the counterpart to the more commonly known forward rendering. Instead of creating an image from known geometry and lighting, inverse rendering deals mainly with the estimation of lighting from an image. Ramamoorthi and Hanrahan [17] presented a mathematically consistent framework for inverse rendering. They describe the reflected light field as a convolution of lighting and bidirectional reflection distribution function (BRDF). Moreover, they represent the light field as a product of SH of the BRDF and the lighting. The same authors also presented an efficient way of rendering environment illumination using SH [16]. A main finding of this work is that a limited number of SH coefficients is sufficient for creating perceptually valid renderings. For a broader overview of inverse rendering problems, we refer to Patow et al. [14] and especially to the survey by Jacobs and Loscos [8] on illumination methods for mixed reality.

A major technical problems of inverse rendering is how to measure the visible radiance. We discuss this research area, which is also known as "photometric registration", in the following section.

2.2 Photometric registration

The principles of the previous work discussed in this section can be traced back to [3]. This seminal work showed how to recover high dynamic range irradiance maps from multiple photographs capturing a mirror ball. A subsequent work [2] demonstrated how to correctly light virtual objects with measured scene radiance and global illumination. However, this work was not designed for real-time applications. In the following, we discuss only real-time methods for photometric registration, organized by the methods how the environment light is observed.

Illumination estimation from light probes: Kanbara and Yokoya [9] presented a method how to estimate the light environment in real time for AR using a fiducial marker for geometric registration and a black mirror ball for capturing only high frequency light sources. They recover dominant light source directions assuming that all viewing vectors are parallel to the optical axis of the camera. In contrast, recent work by Aittala [1] captures diffuse lighting using a diffuse light probe, in this case a ping-pong ball or a rotated planar marker. The interesting aspects of these light probes are the diffuse reflectance characteristics. Aittala solves the relationship between a general light source model and the light intensity observations from the camera images using L1-regularized least squares minimization. This allows to robustly estimate the dominant light sources from the diffuse environment map.

Illumination estimation from shadows: Another method for estimating the dominant light sources is to observe the shadows in an image. The idea is based on the partial knowledge of the geometry of the shadow caster and the correct classification and measurement of the shadow in the image. For example, Haller et al. [7] utilize geometry with known characteristics to analyze shadows. Another work by Mei et al. [12] demonstrates how to efficiently find significant directional light sources and cast illumination recovery as L1-regularized least squares minimization. Wang et al. [19] presented a method for estimating multiple directional light sources from known geometry. Their approach has only been applied to static images and combines a shadow based method and a shading based method.

Illumination estimation from HDR cameras: Grosch et al. [5] and Knecht et al. [10] use special camera sensors for capturing the environment lighting. Their approach is based on a high dynamic range camera with a fish-eye lens. Compared to mirror balls, which can cover a field of view (FOV) of 300 degrees, the fish-eye lens provides only a FOV of about 60 degrees. Moreover, fish-eye HDR cameras are expensive and not commonly available.

Illumination estimation in outdoor AR: For outdoor AR, light estimation can be based on the usage of the geospatial position and the current time of the user. These parameters give a rough estimation of the current sun position and enable the implementation of the main light direction supporting cast shadows as presented in the work of Madsen et al. [11].

Illumination estimation from arbitrary geometry: Pilet et al. [15] propose a fully automated method for geometric and photometric calibration from an arbitrary textured planar pattern. In this work, the light probe is constrained to a planar surface. Another example is given by Van den Hengel et al. [18]. In this work the user has to specify the geometry through manual input. Then the appearance model of the geometry is estimated and transferred to other newly inserted virtual objects. However, the input of this work is an offline video sequence. In contrast, our work is interactive and does not require manual intervention.

In the following Section 3, we discuss the theory of our approach.

3 Lighting estimation

The primary scope of this work is to estimate the environment lighting. More specifically, we are interested in a distant light field F, which represents all incident light rays on a hemisphere. Assuming a distant light source implies that we can treat the light source and the lit scene independently from each other.

3.1 Environment lighting estimation using SH

The principal idea is to recover F from many different observations of arbitrary geometry. The basic relation between the incident light, the observations and the geometry is stated in the reflection equation (see Equation 1), as given by Ramamoorthi and Hanrahan [17]. The reflection $B(x,\vec{w_0})$ is an integrand of the three different terms - lighting $L(x,\vec{w_0})$, bidirectional reflectance distribution function (BRDF) $\rho(x,\vec{w_0})$ and texture $T(x)$.

$$B(x,\vec{w_0}) = \int_{\Omega_j} T(x)\rho(\vec{w_j},\vec{w_0})L(x,\vec{w_j})(\vec{w_j}\cdot\vec{n})d\vec{w_j} \tag{1}$$

The surface position is denoted with x and the outgoing direction is $\vec{w_0}$. The surface normal vector is defined with $\vec{n}$. T represents a simplified texture model with no explicit specular texture. We assume for the entire scene a diffuse Lambertian reflectance model for the surface BRDF $\rho(x,\vec{w_0})$. We do not solve the ambiguity of light color and texture color. Instead, *we assume that the light color is white*, and color contribution comes only from the surface texture itself. The light sources can be dynamic and are assumed to be distant, which implies a homogeneous lighting over the entire surface. We are interested in computing the lighting term L (intensity) in Equation 1 from all visible surface positions in one single view to estimate F. We express the incident light field F with SH.

In the following Equation (2), we express F as a reconstruction $\widetilde{F}$ of spherical harmonics basis functions Y weighted by the SH coefficients c. For simplicity, we choose a single index notion of the SH index terms usually called l and m, where $i = l(l+1)+m+1$. Moreover, the index k denotes the number of used bands. Obviously, the quality of the reconstruction of F depends on the number of basis functions.

$$\widetilde{F}(S) = \sum_{i=0}^{k^2} c_i Y_i(S) \tag{2}$$

Furthermore we implemented a **radiance transfer function** for diffuse shadowing, R_{DS} (see Equation 3), where $\vec{n}$ is the surface normal, ρ_x the surface albedo and $V(\vec{w_j})$ is a visibility term at surface point x:

$$R_{DS}(x) = \frac{\rho_x}{\pi}\int_S V(\vec{w_j})L_j(x,\vec{w_j})max(\vec{n}\bullet\vec{w_j},0)d\vec{w_j} \tag{3}$$

The final result of a rendering for a surface point x can be expressed as the dot product of the SH projected light source $\widetilde{F}$ and the SH radiance transfer function R_{DS}.

$$I(x) = \sum_{i=0}^{k^2} R_{DS}(x)_i \widetilde{F}_i \tag{4}$$

Our aim is to estimate the unknown $\widetilde{F}$ from $z \in \{1..Z\}$ reflection observations $I(x_z)$. For estimating $\widetilde{F}$, we stack the observations to a matrix system (see Equations (5, 6)) and obtain a linear equation system of the form $Ay = b$ of size $Z \times k^2$ with non-square A which we have to solve for y. Since the system is overdetermined, we minimize the error $|Ay-b|_2$.

$$A = \begin{pmatrix} R_{DS}(x_1)_1 & R_{DS}(x_1)_2 & \cdots & R_{DS}(x_1)_{k^2} \\ R_{DS}(x_2)_1 & R_{DS}(x_2)_2 & \cdots & R_{DS}(x_2)_{k^2} \\ \vdots & \vdots & \ddots & \vdots \\ R_{DS}(x_Z)_1 & R_{DS}(x_Z)_2 & \cdots & R_{DS}(x_Z)_{k^2} \end{pmatrix} \tag{5}$$

$$y = \begin{pmatrix} \widetilde{F}_1 \\ \widetilde{F}_2 \\ \vdots \\ \widetilde{F}_{k^2} \end{pmatrix},\ b = \begin{pmatrix} I(x_1) \\ I(x_2) \\ \vdots \\ I(x_Z) \end{pmatrix} \tag{6}$$

3.2 Making lighting estimation robust

Since the quality of the light reconstruction heavily depends on the quality of the input data, which is noisy under real world conditions, we developed two methods, which improve the robustness of our estimation. The main concern is the selection and interpretation of samples taken from the camera image. A sample $M(x_z)$ is represented by a position x in 3D space with a surface normal vector $\vec{n}$, the illumination observation from the camera frame $I(x_z)$ and the radiance transfer $R_{DS}(x_z)$ expressed by the SH coefficients (Figure 2).

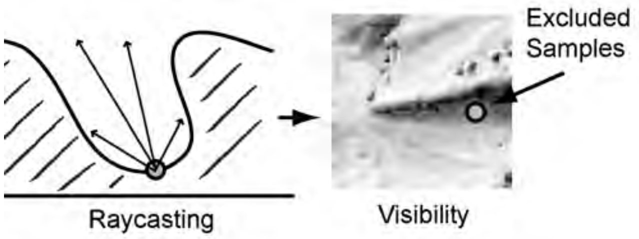

Figure 2: We evaluate the visibility of a sample by ray casting and exclude samples which potentially can receive only a very little amount of light.

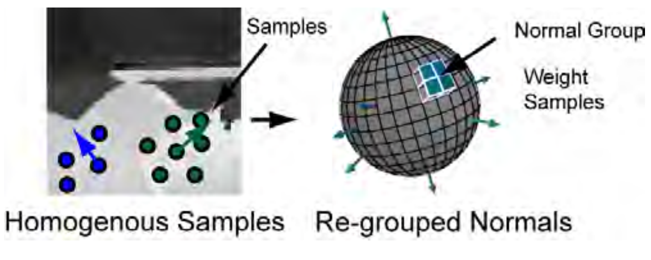

Figure 3: We group samples according to their surface normal vectors by a regular grid representing the surface of a sphere.

Sample selection by visibility: Surface points can be located in cavities or otherwise heavily occluded areas, where only a small amount of light transfer is potentially possible. From these areas, the measurements will not provide trustworthy data, since we do not model interreflections to simulate more complex light transfers. Therefore we exclude those areas from the measurements. While computing the radiance transfer function R_{DS}, we also determine the visibility term $V(\vec{w}_j)$. By doing so, we can measure the potential irradiance by casting a ray from each sample point into all possible directions $\vec{w}_j$ and determine if there is any occlusion from another surface in this direction. If a sample point has too much occlusion (e.g., 95 percent of the rays hit another surface), it is excluded from the lighting estimation (see Figure 2).

Weighting by surface normal vector distribution: The ideal surface for estimating the environment illumination would be a sphere, which provides observations from all possible directions. Unfortunately, such a uniform distribution of normal directions is not always naturally given in the real world. In practice, the scene often consists of a large planar surface (e.g., a table) with a single dominant normal vector. Naturally, samples taken from this surface will have a large impact on the final estimation. Other normal directions will not be sufficiently represented. For example, samples taken from smaller objects in the scene, which have a smaller pixel coverage in the video frame, will not yield a comparable number of samples. To improve our estimation, we therefore weight the samples according to a uniform distribution U of surface normal vectors on a sphere (see Figure 3). We bin the samples according to their surface normal vectors into a regular spherical grid C. For each bin c in the grid, we compute a weight m_c (see Equation 7), where D is the number of bins of the grid and Z is the total number of samples. The weight m_c is the number of samples Q_c in a bin divided by the uniform distribution $U = \frac{Z}{D}$.

$$m_c = \frac{Q_c}{U} \tag{7}$$

In the following we multiply the illumination value $I(x_z)$ and the radiance transfer $R_{DS}(x_z)$ with the corresponding weight m_c. The results are then inserted to the linear equation system in Equation 6. The weighting will have the following three effects. First, samples from overrepresented areas are normalized by the uniform distribution. Second, samples, which are represented uniformly over a sphere, are not changed. Third, the influence of samples, which are not strongly represented, will be diminished. The benefit of this method can be seen in the real world experiments in Section 5.2.

4 Implementation

Our system consists of two major parts, a lighting estimation part and an AR rendering part, which are explained in the subsequent sections.

4.1 Light Estimation

In Figure 4, a systematic overview of the light estimation pipeline is shown.

Color and depth image processing: We capture a color and a 16bit depth image from an RGBD camera (Microsoft Kinect). Since we are only interested in estimating white colored light sources, we convert the color image from RGB into CIE L*a*b color space to obtain the reflected radiance from the scene. To remove camera noise and high frequency texture parts, we run a TV-L1 de-noising algorithm [20] on the Luminance channel. The color conversion and the TV-L1 algorithm are both implemented on the GPU with CUDA. To improve the registration of the camera image and the depth image, we compensate the radial distortion on both sources, applying a third degree polynomial lens rectification model.

Geometry reconstruction and pose estimation: The geometry reconstruction and pose estimation is based on the KinectFusion algorithm [13]. A 3D voxel volume is continuously updated with the current depth map. The result is a surface reconstruction represented as a signed distance function (SDF) in the voxel volume together with a 6DOF pose estimation of the camera. To obtain a surface point x and a surface normal vector $\vec{n}$ for the current view, we have to perform ray casting on the reconstructed volume for every frame. Hence the real-world geometry is represented as an implicit surface, and no polygonal model needs to be computed at any time.

Radiance transfer computation and SH projection: For the light estimation, we have to model the possible real-world radiance transfer RT_R as closely as possible. We implemented the diffusely shadowed radiance transfer function R_{DS} presented in Equation 3, which takes visibility $V(\vec{w}_j)$ into account. To compute the visibility term, we cast rays in all possible directions of a sphere from every surface point x. Dynamically changing real-world geometry precludes pre-computed radiance transfer methods. Therefore we compute the radiance transfer every frame. This task has been implemented with CUDA. To improve real-time performance, it is also possible to compute the radiance transfer only every n^{th} pixel and use bilinear interpolation (see Figure 6 (a) to (c)). The solution of RT_R is then projected into the SH basis function. A further reason for computing RT_R is differential rendering, were real and virtual illumination effects are combined. A more detailed explanation is given in section 4.2.

Lighting estimation: The environment light is estimated with the solution proposed in Section 3. For each sample point, we have the SH coefficients and the irradiance observations from the camera. After applying the algorithms for robust estimation, proposed in Section 3.2, we obtain the SH coefficients representing the environment lighting by solving the equation system given in Equation 5. This is a rather lightweight computation and is done on the CPU. The light is estimated every frame and supports dynamically changing light sources.

4.2 Rendering pipeline

Our rendering pipeline is entirely based on SH lighting. In Figure 5 a systematic overview of the entire rendering pipeline is shown.

Asset preparation: Similar to the real-world geometry, we also create a voxel representation of the virtual geometry. This can be done in an off-line 3D mesh voxelizer such as binvox[1]. Choosing a unified data structure for the real world geometry and the virtual geometry enables a straight forward evaluation of the visibility term of the radiance transfer function by multi volume ray casting.

Differential rendering: To depict the influence between real-world models and virtual models, we implemented differential rendering as proposed by Debevec et al. [2]. We use the radiance transfer function R_{DS} stated in Equation 3 to model the local illumination.

Differential rendering is computed in two passes. In the first pass, a G-Buffer rendering of the virtual scene is created using standard OpenGL, using the camera pose that was determined through tracking with KinectFusion. This render pass produces a color buffer of the unlit virtual scene together with a geometry buffer (i.e., x/y/z coordinates of every fragment in camera coordinates) and a normal buffer. The color buffer is later used in the compositing step to compute the final shaded surface colors, while the geometry and normal buffers are used to initialize the raycasting employed in the second rendering pass.

In the second pass, two global illumination solutions used in the differential rendering are computed. The first solution is RT_R, which has been already computed for the light estimation. The second solution is RT_{RV}, which is the global illumination solution for real and virtual objects together. Like RT_R, RT_{RV} is computed using raycasting. Since the voxelized representation of the virtual objects creates noticeable sampling artifacts, we start the raycasting from the geometry buffer produced in the first pass. We also use the normal buffer from the first pass, as it has higher accuracy than normals that could be estimated from the voxel representation. See Figure 6(c) and (d) for an example of the different qualities of the representations.

[1]binvox: http://www.cs.princeton.edu/~min/binvox/

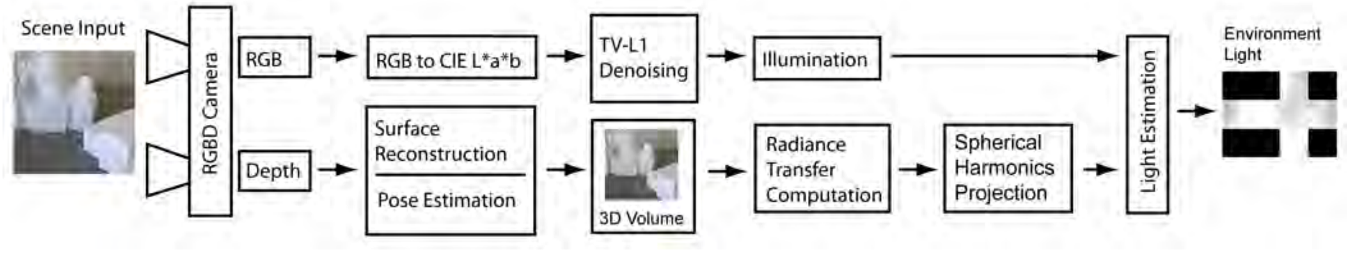

Figure 4: The lighting estimation pipeline computes SH coefficients of the physical environment in real time.

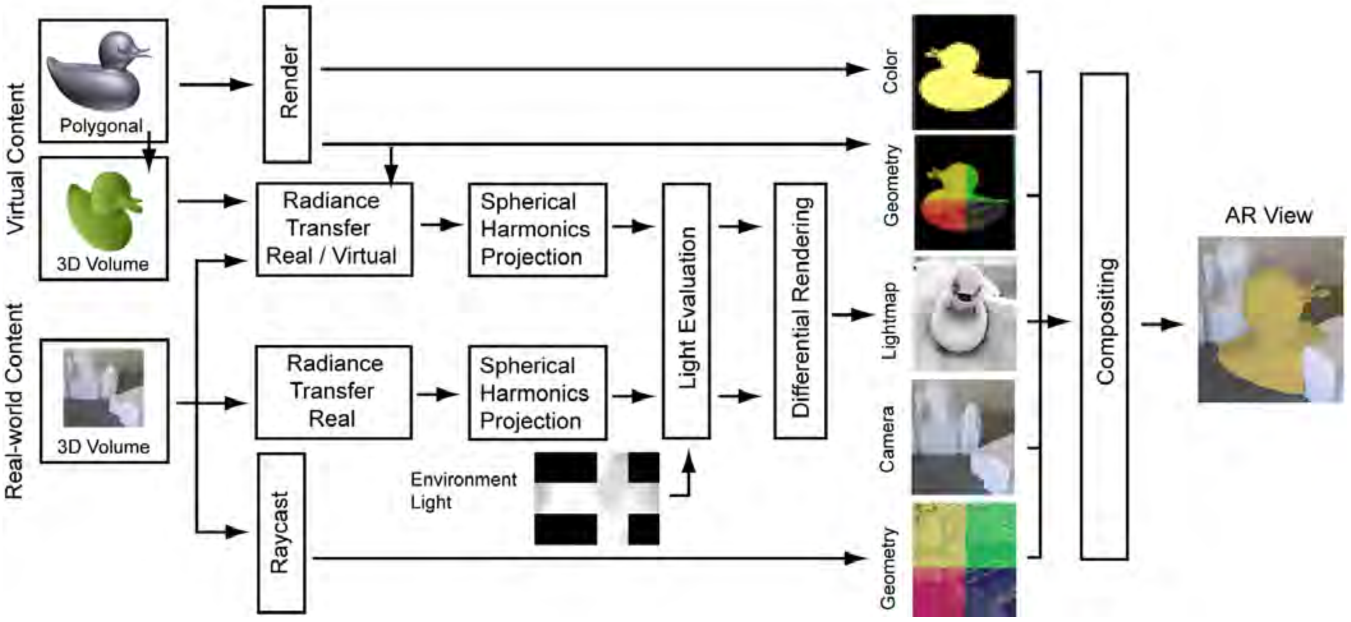

Figure 5: The rendering pipeline is a combination of differential rendering with a global illumination representation based on spherical harmonics.

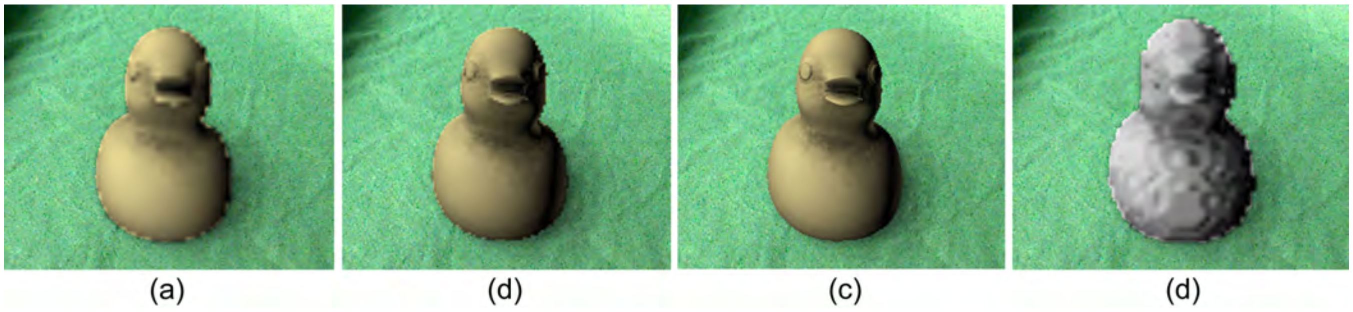

Figure 6: From left to right: Sample every 4th pixel, every 2nd pixel, every pixel, voxel representation.

We project both RT_R and RT_{RV} to SH coefficients. Finally the lighting is evaluated by computing the dot product of the SH coefficients of the environment lighting with the SH projected radiance transfer, and inserting the result into the differential rendering equation, which in our case yields a lightmap I_{final}:

$$I_{final} = \widetilde{F} \bullet RT_{RV} - \widetilde{F} \bullet RT_R \tag{8}$$

See Figure 10 for an example of the effects achieved with differential rendering. The raycasting is very computationally intensive and has the highest impact on the performance of the system. We considered screen-space ambient occlusion (SSOA) as a cheaper alternative, but preliminary tests with SSOA did not yield satisfactory results in terms of shadow quality. For performance tuning, radiance transfer computation can be restricted to only every n^{th} pixel.

Unlike other differential rendering solutions (e.g., Knecht [10]), our approach can render the results from RT_R and RT_{RV} in one pass and consequently avoids redundant geometry processing.

Compositing: The compositing step is necessary to create the final AR image. The main purpose is to handle occlusions of vir-

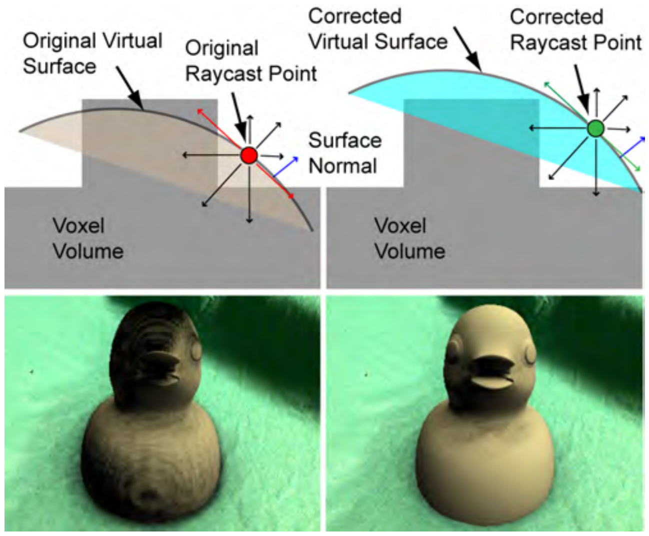

Figure 7: In the upper image the problem of self occlusion artifacts is depicted. The lower left image shows the virtual object (duck) without occlusion offset and the lower right image shows the duck with occlusion offset. As can be seen the surface of the duck in the lower right image has less stains.

tual and real objects correctly and to compute the final color. For handling occlusions, the depths values in the geometry buffer of the real-world surface and the geometry buffer of the virtual surface are compared. Depending on the depth test, the color of the camera frame or the color of the virtual object is taken and multiplied by the lightmap I_{final}.

Note that the real-world surface might contain discontinuities or holes, where KinectFusion was unable to provide a proper reconstruction. This can for example happen when the depth sensor is confused by highly specular surfaces. In these cases, a proper workaround is to assume that the virtual surface is closer than any real surface, and consequently use the shading information from the virtual shading. To work in linear color space, we removed the gamma correction in the video color frame at the beginning of the pipeline. As a post-processing step, we finally apply the gamma correction to the output color.

Self occlusion artifacts: The raycasting step can create unwanted self-occlusion artifacts, noticeable as stained surfaces. These artifacts can also be misinterpreted as lighting effects (shadows), since the overall color looks darker. Self-occlusion is typically caused when the resolution of the voxel grid does not match the resolution of the virtual geometry or when the reconstructed real surface is noisy. As can be seen in the left most image in Figure 7, this can produce aliasing artifacts. To overcome this problem, we offset the starting point for each ray along the surface normal by the length of one voxel cell.

5 Results

In the following, we present the evaluation of our algorithm and implementation. We will mainly focus on plausible and perceptively convincing results rather than on physically correct results. We evaluate the correctness and quality of our work in the following subsections. In our evaluation, we used props from the AR stage set "City of Sights" from Gruber et al. [6]. For SH computation, we used four bands, resulting into 16 coefficients for all the presented results.

5.1 Synthetic light estimation evaluation

As a first test, we load a known light source, represented as an HDR environment map, and use it to light the reconstructed scene without any virtual objects. The output is then directly used to estimate the synthetic light source. The estimated result should be comparable to the known input light. We compare the direct evaluation of the SH coefficients on a cube map, where each pixel represents a unique direction of the unit sphere. The results of this test are shown in Figure 8. In this evaluation we took HDR environment maps from Paul Debevec [2]. Note that in the interpretation of these results, the quality of the geometric reconstruction, for example the accuracy of surface normal vectors, must be considered. The major conclusion of this test is that the light estimation is working correctly with real-world data from the geometry reconstruction.

5.2 Real-world results

In Figure 12, we show the results of a systematical test series with three different scenes (A, B and C) and 4 different lighting situations. All three scenes have been exposed to the same 4 different lighting situations created using a bright directional light source, which changes direction through the series. Note that the light source is not optimally diffuse and creates hard shadows from the real objects. So far we did not model hard shadows in our system and estimate only low frequency lighting. Therefore the shadows from the virtual objects will appear more soft and blurred. In Figure 11, we show the effects of applying our robustness method presented in Section 3.2 on the more complex test scene A. The results show that the robust approach creates a better lighting estimation, which results in a perceptually better rendering. The tests shown in Figure 12 demonstrate that our system also works with colored objects, although we do not estimate any material properties. Scene B and C have identical geometrical properties and differ only in the surface color. While the objects in scene C have various colors, the objects in scene B are uniformly gray. Each result is accompanied by the visualization of the lighting estimation as in an unfolded cube map. For comparison, each series has been processed by the naive approach and the robust approach.

5.3 Influence of occlusion

In this section, we discuss the influence of the occlusion on the lighting estimation. The occlusion is computed through the visibility test $V(\vec{w}_j)$ in equation 3 and practically depends on the length of the ray of the ray casting step. A longer ray will increase the probability that occlusion from objects that are further away is incorporated into the visibility estimation. For the lighting estimation, we only consider the occlusion of the real world objects. We increase the ray length from 0 to $0.2m$ to evaluate the influence. As can be seen in figure 9, the solution of the lighting estimation improves with the ray length. The solution is more stable, which can also be noticed in the AR rendering. This is due to the fact that the real-world is modeled more accurately with occlusion than without. The optimal ray length depends on the entire volume of the scene. In our case, the ray length is given in meters, and the entire scene has a volume of $2 \times 2 \times 2$ meters. Note that compared to conventional SSAO, which only takes information from one view into account, the consistent reconstruction of the volume enables more realistic occlusions.

5.4 Differential Rendering

In Figure 10, the lighting effects between real and virtual created by differential rendering are shown. In this scenario, we directly rendered the voxel representation of the virtual object.

[2]http://ict.debevec.org/~debevec/Probes/

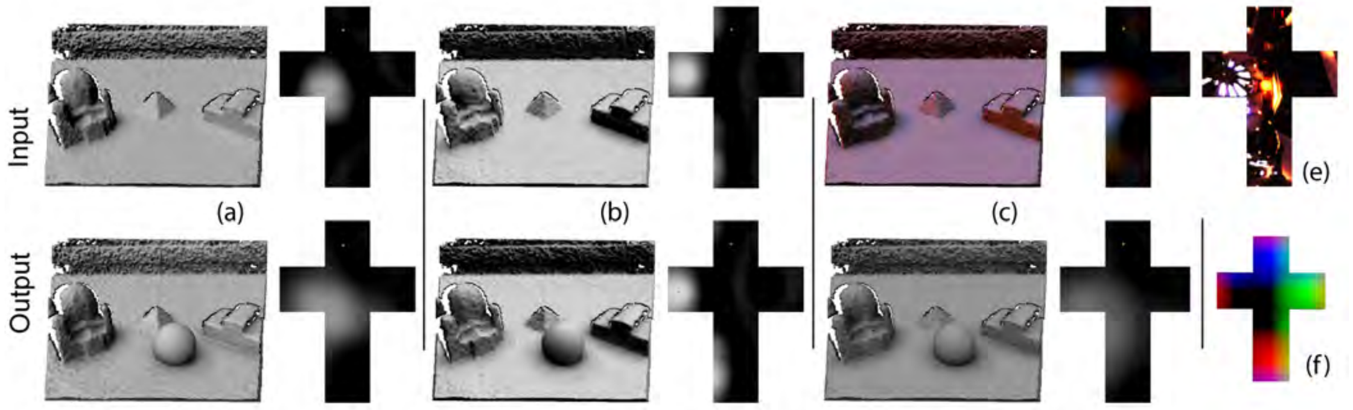

Figure 8: This image sequence shows results from the synthetic light reconstruction. The top row (Input) shows the entire real-world scene lit by the synthetic light. The rendered scene is then used as input for the light estimation algorithm. The light source is shown on the right of the rendering, represented as a cube map. The brightness of the cube map has been adjusted for better display in this document. The lower row (Output) shows the real-world scene plus a virtual sphere lit by the reconstructed light source. The solution of the light source reconstruction is shown on the right of the augmented rendering. In examples (a) and (b), we used a light source traveling from left to right. The effect is best observed looking at the pyramid. In example (c), we used the commonly available "'Grace Probe"' (e) HDR map from Paul Debevec. This light source is more complex. Note that we do not estimate the color of the light, therefore the result is also in gray. Image (f) shows the color encoded directions of the environment map.

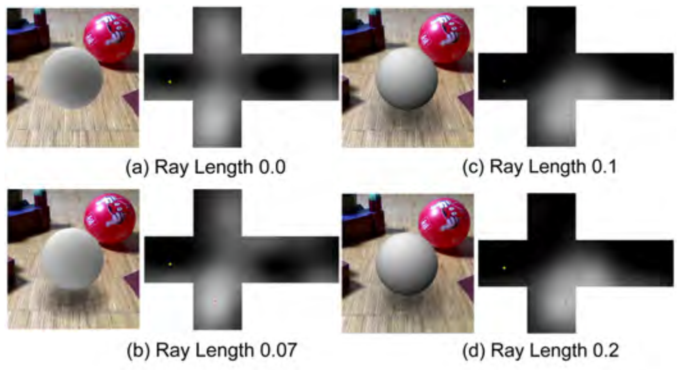

Figure 9: We increase the ray cast length (in meters) from (a) to (d). Comparing the solution of (a) and (d), a noticeable improvement of the estimation can be seen.

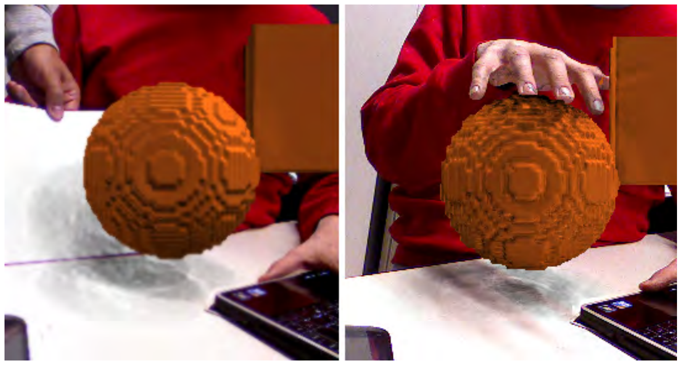

Figure 10: The left image shows shadows casting from the virtual object on the table and the paper. The right image shows a shadow from the real world geometry (hand) onto the virtual geometry.

5.5 Limitations

Our system is built upon several different techniques, each having its advantages and limitations, which we discuss here.

Geometry reconstruction: The dynamic reconstruction algorithm [13] has a great impact on the overall result. This algorithm improves the quality of the surface reconstruction (surface normal vectors, surface consistency etc.) over time. Since quality of the surface normal vectors is crucial for the light estimation, the quality of the light estimation depends on it. A further limitation of the surface reconstruction is that fast changes in the scene do not take effect immediately. Adding and removing objects is possible, but it will take some frames until the entire scene is updated correctly. Moreover, the reconstruction of the scene might be incomplete unless the user makes an effort to scan the scene properly, e.g., by walking around the scene. Nevertheless, we also demonstrated that our approach works with a static camera. A further restriction is given by the depth sensor and the resulting depth map quality. Using the projection-based Microsoft Kinect, we are not able to recover depth maps from specular surfaces, e.g., mirror balls.

Visual quality: The visual quality of the rendering depends on several factors. The number of samples (rays) used for the radiance transfer computation has a strong influence on the visual quality, on the light estimation and on the performance of the system. It controls how fine grained the visibility test is computed. We found that a number of samples ranging between 64 and 300 creates reasonable results. Using 64 samples provides already acceptable renderings. Second, our system computes the solution for every n^{th} pixel. Obviously the most expensive, but also visually most pleasant results are obtained if $n = 1$. Solutions with $n = 2$ and $n = 4$ have to be interpolated and create visual artifacts such as aliasing. Note that inconsistent reconstruction can lead to wrong or partially wrong occlusions.

Light probes: Naturally the light estimation also depends on the surface characteristics. There are surfaces which are problematic. For example, the reconstruction and tracking algorithm does not handle planar surfaces well, and the light estimation algorithm works better if the surface normal vectors cover a hemisphere as much as possible. Moreover, surfaces with strong reflective materials violate the assumption of diffuse surface materials. The quality of the light estimation also depends on the distribution of the surface material colors. For instance, if a scene is divided into a black and a white area, the algorithm would likely produce wrong results.

Light sources: We support only diffuse shadows and lighting in our estimation and rendering approach. No hard shadows are modeled. Furthermore, as already stated in Section 3.1, we do not estimate material properties yet. Therefore it is assumed that the light sources have a white color.

Frame coherence: We estimate the lighting at a per frame basis and do not take previous results into account. Although this allows the estimation of fast dynamically changing lights, it can also lead to instable results. We did not apply any method such as moving averages, and our system can suffer from flickering artifacts. This also depends on the quality of the RGB camera, which can suffer from noise at bad lighting conditions.

Performance: The main bottleneck in our system is the computation of the radiance transfer through ray casting including a visibility test. This task has to be performed twice because of the differential rendering. However, we achieve interactive frame rates of 5Hz performing the computation on every 4th pixel with a ray length of 10 cm, a total working volume of 2x2x2 m, 16 SH coefficients (4 bands) and 169 rays covering a sphere for visibility testing on commodity hardware (GeForce GTX 580).

5.6 Demonstration Application

In Figure 13, a more complex scenario with a colored environment is shown. The scenario models a fictive environment, which could be part of an AR game or story telling application. The inset shows how the estimation of lighting from an unprepared scene is used to add a virtual hat to the head of a person.

6 Conclusion and Future Work

In this work, we presented a system for estimating the environment lighting for real-time AR applications. These algorithms enable the

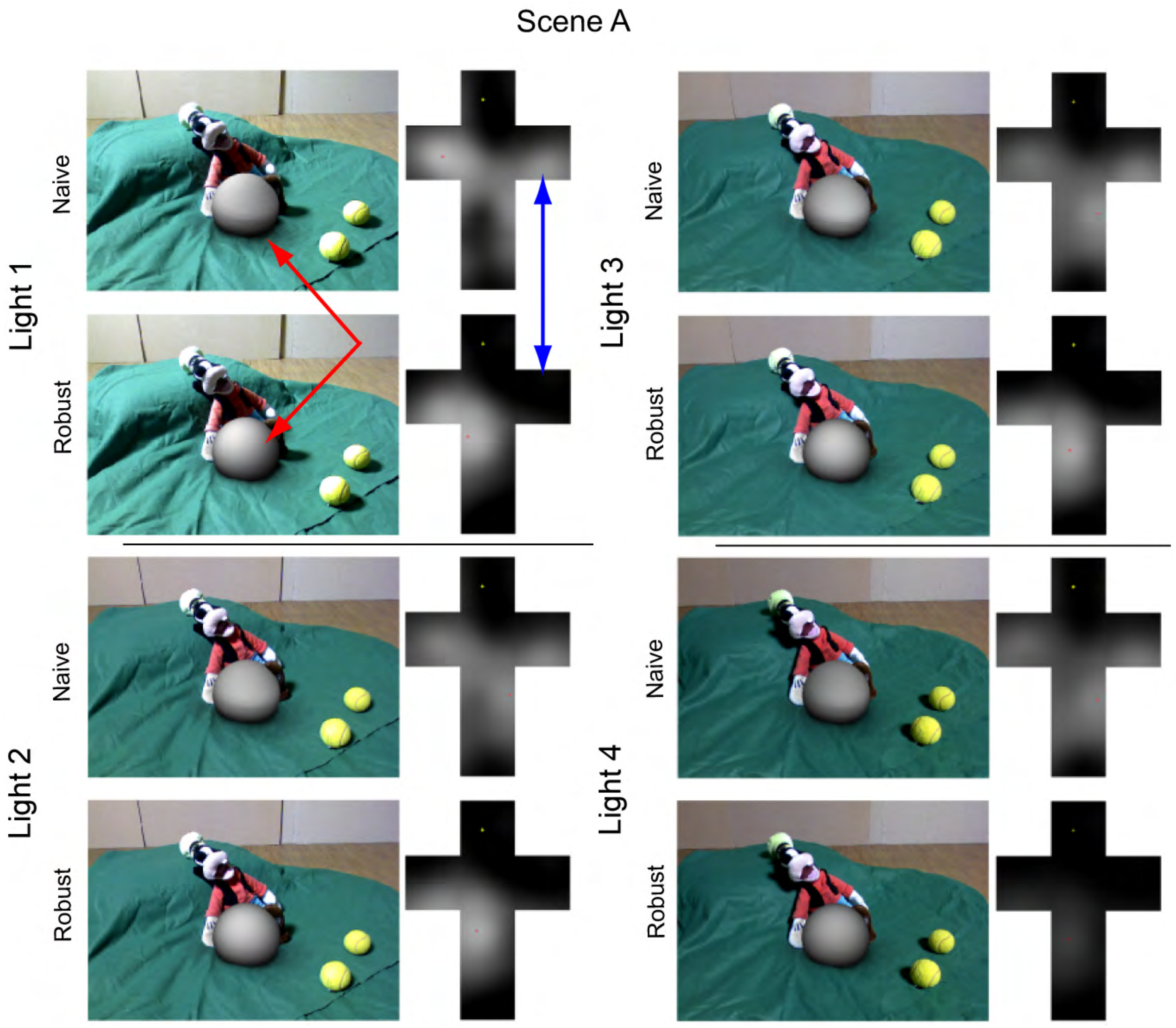

Figure 11: The red (scene space) and the blue (illumination space) arrows mark visible difference of the naive and robust approach. We can observe that the robust approach creates more visually pleasant results than the naive approach.

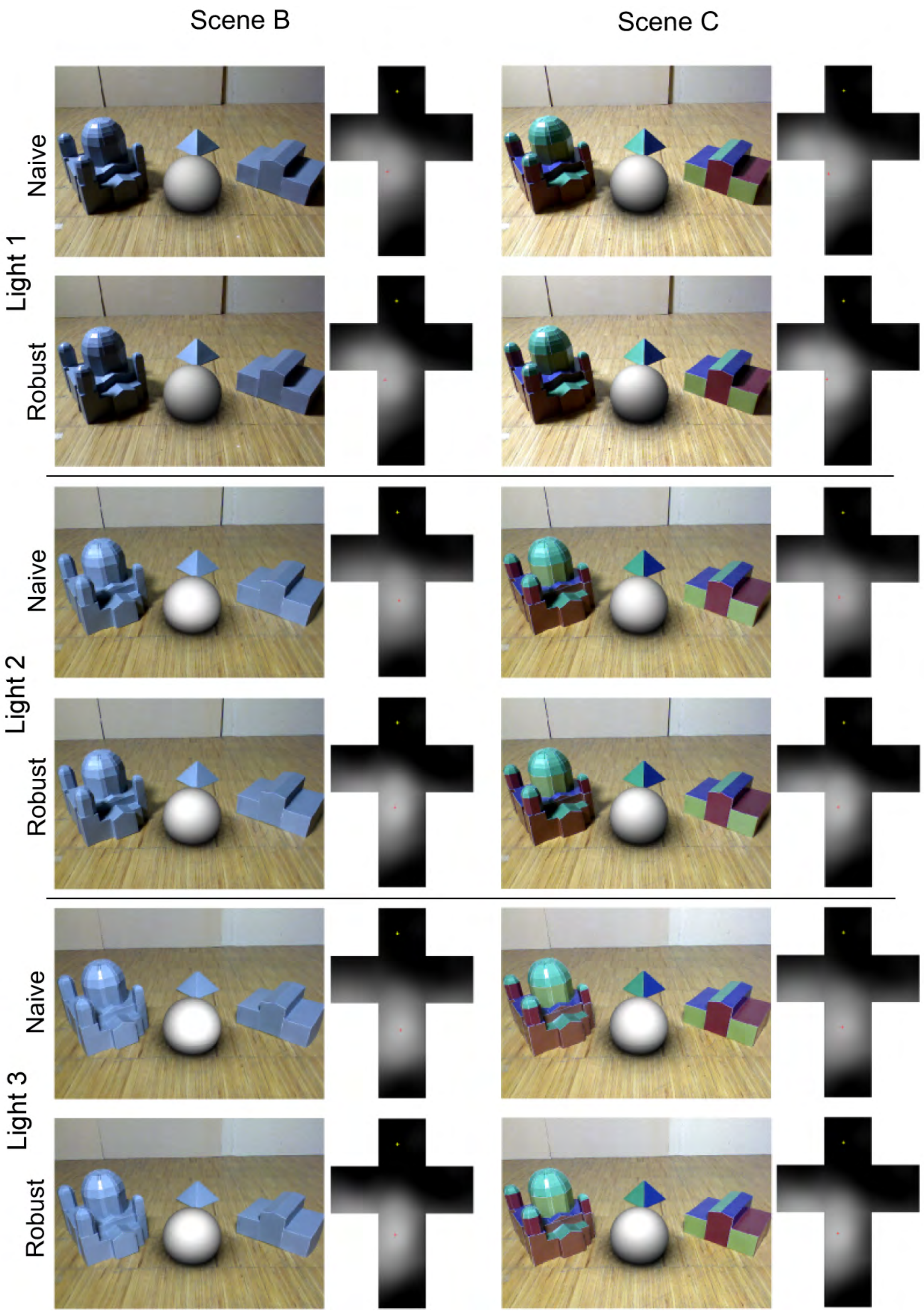

Figure 12: This Figure depicts the real-world results of the two scenes B and C, differing in surface colors. Note that there are very little differences in the lighting estimation.

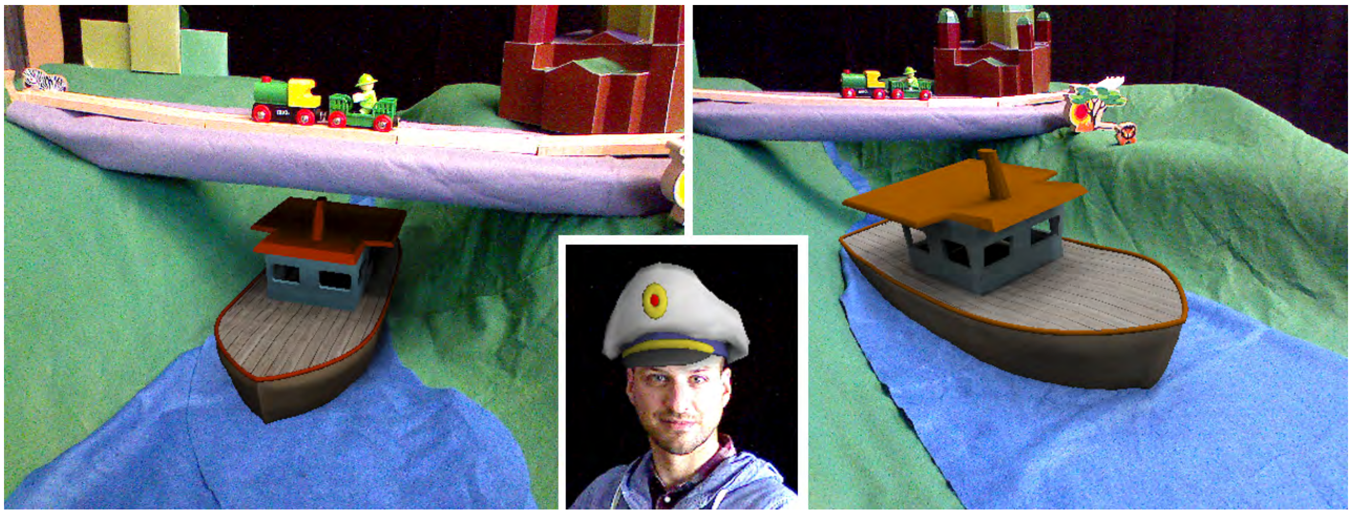

Figure 13: Left image: real-world bridge casts a shadow onto the virtual ship and the virtual ship casts shadows onto the real-world environment. Right image: the direction of the virtual shadows can be compared with the real-world shadows, for example of the little lion. Middle insert: we added a virtual hat onto the head of a real person to demonstrate the ability of the system to work in unprepared environments too.

lighting of virtual objects from a real-world lighting representation. Unlike previous work, we estimate the lighting directly from arbitrarily shaped objects. Thus, we can improve the visual coherence of AR without the need for impractical procedures or props. We demonstrate the quality of the results by comparing our algorithms with synthetic test data and with real-world data. In the future, we will extend our work by methods which will also allow to estimate higher frequency lighting.

7 Acknowledgement

This work has been funded by the Christian Doppler Laboratory for Handheld Augmented Reality. We thank Albert Walzer for providing virtual assets.

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. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 369–378. ACM Press/Addison-Wesley Publishing Co., 1997. ACM ID: 258884.

[4] A. Fournier, A. S. Gunawan, and C. Romanzin. Common illumination between real and computer generated scenes. In *Proc. Graphics Interface*, 1993.

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

[6] L. Gruber, S. Gauglitz, J. Ventura, S. Zollmann, M. Huber, M. Schlegel, G. Klinker, D. Schmalstieg, and T. Hollerer. The city of sights: Design, construction, and measurement of an augmented reality stage set. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*, pages 157–163, 2010.

[7] M. Haller, S. Drab, and W. Hartmann. A real-time shadow approach for an augmented reality application using shadow volumes. In *VRST*, pages 56–65, 2003.

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

[9] M. Kanbara and N. Yokoya. Real-time estimation of light source environment for photorealistic augmented reality. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02*, pages 911–914. IEEE Computer Society, 2004.

[10] 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.

[11] C. Madsen and M. Nielsen. Towards probe-less augmented reality. In *Proceedings: GRAPP 2008*, pages 255–261. Institute for Systems and Technologies of Information, Control and Communication, 2008.

[12] X. Mei, H. Ling, and D. W. Jacobs. Sparse representation of cast shadows via l1-regularized least squares. In *2009 IEEE 12th International Conference on Computer Vision*, pages 583–590. IEEE, Oct. 2009.

[13] 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.

[14] G. Patow and X. Pueyo. A survey of inverse rendering problems. *Computer Graphics Forum*, 22(4):663–687, Dec. 2003.

[15] J. Pilet, A. Geiger, P. Lagger, V. Lepetit, and P. Fua. An all-in-one solution to geometric and photometric calibration. In *IEEE/ACM International Symposium on Mixed and Augmented Reality, 2006. ISMAR 2006*, pages 69–78. IEEE, Oct. 2006.

[16] R. Ramamoorthi and P. Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 497–500. ACM, 2001. ACM ID: 383317.

[17] 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.

[18] A. Van Den Hengel, D. Sale, and A. R. Dick. SecondSkin: an interactive method for appearance transfer. *Computer Graphics Forum*, 28(7):1735–1744, Oct. 2009.

[19] Y. Wang and D. Samaras. Estimation of multiple directional light sources for synthesis of augmented reality images. *Graphical Models*, 65(4):185–205, July 2003.

[20] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In *In Ann. Symp. German Association Patt. Recogn*, pages 214–223, 2007.