# ILLUMINATION RECONSTRUCTION FROM REAL-TIME VIDEO FOR INTERACTIVE AUGMENTED REALITY

*Sebastian Heymann[1,2], Aljoscha Smolic[1], Karsten Müller[1], and Bernd Froehlich[2]*

[1]Fraunhofer Institute for Telecommunications - Heinrich-Hertz-Institut
Image Processing Department,
Germany
[2]Bauhaus University Weimar,
Germany

## ABSTRACT

This paper presents an interactive real-time augmented reality system with realistic illumination of integrated virtual objects. A marker and a mirror sphere are placed in a scene that is captured by a video camera. Dynamic illumination (as the marker is interactively moved) is estimated from the video signal and used for realistic real-time integration of virtual objects into the video. To achieve interactive real-time performance, we have developed a hardware-accelerated approach, where self-occlusion and form factors are precalculated and stored in a low-resolution cube map for each vertex. This data is multiplied with the real-time illumination information estimated from the video to obtain diffuse per vertex illumination.
Our approach works well for small virtual objects moving around in real environments and is almost completely implemented on the GPU per performance purposes. The real environment operates as a large area light source, which casts soft shadows on our diffusely lit virtual objects.

## 1. INTRODUCTION

An important challenge for augmented reality systems is the issue of consistent lighting of virtual objects with respect to the real-world environment. This is in particular difficult to achieve for virtual objects moved around in completely unknown and dynamic environments, where the lighting might continuously change. One way to obtain knowledge of the real scene lighting is the observation of a mirror sphere using a video camera. This simple approach has been mostly used to create environment maps, which work well for highly specular objects.

In our work we also use a mirror sphere to capture the incident light from the real-world environment, but we compute the diffuse illumination of a virtual model in real time. The environment map is treated as a textured area light source, which encloses our virtual model. The direct light contribution of the area light source is computed by integrating over the visible hemisphere for each vertex of our model on a frame-by-frame basis. This approach allows us to move our virtual model freely around in dynamically changing environments.

In order to achieve real-time results, we had to introduce two main limitations into our approach: We are working with a single static virtual model and we do not consider indirect illumination. Since we use a single mirror sphere as our representative sample for the incident light, we also assume that the incident illumination does not change over the surface of the displayed virtual object apart from self-occlusion, which is handled correctly. These limitations allow us to compute form factor cube maps for each vertex in a pre-processing stage. These cube maps account for self-occlusion of the virtual model and the light transfer between the environment map and each vertex. During the use of our system, these form factor cube maps are multiplied with an environment cube map generated from the image on the mirror sphere. The results are summed for each vertex and multiplied with the reflection coefficients of our displayed virtual model.
Our approach is implemented to be executed almost completely in the graphics processing unit (GPU), which results in real-time frame rates for virtual models with ten thousands of polygons. Our tests show that the approach works well for small virtual objects moved around in real-world environments. Color bleeding from the real environment onto virtual objects as well as soft shadows result in a consistent appearance of purely diffusely lit virtual objects in the real environment. Our main contribution is the development of an approach, which allows the real-time computation of diffuse lighting from a real-time captured surround image.

## 2. RELATED WORK

Research and development in Computer Graphics have been focusing on creating synthetic environments with photo-realistic quality. By navigating in such environments, users become immersed with the virtual reality. Beside the creation of purely virtual scenes, augmented reality (AR) applications combine synthetic objects with

real world scenes. The different aspects of AR require research from different areas of computer graphics and computer vision.

For the creation of a high-quality AR experience three main aspects have to be considered: First, the object has to be positioned in the real scene by estimating the correct geometric projection. Second, illumination, originating from real-world light sources, has to be adapted for the synthetic objects to create the illusion of real objects being part of the original scene. The third aspect concerns real-time behavior of the rendering process for applications that capture real-world environments during runtime. One widely applied system in this domain is AR-Toolkit by Billinghurst et al.[3][4][5][6], which is also used in the proposed system due to its ease of use, high performance and stability. The system provides a simple detection and identification of multiple scene markers and calculates the necessary geometric transformation matrices to place geometric objects into arbitrary scenes.

In the area of illumination reconstruction, Fournier et al. [2] consider direct and indirect illumination using a radiosity algorithm for the diffuse lighting. Drettakis et al. [7] proposed a modified version of Fournier et al.'s work. Although the integration of computer vision techniques simplifies the reconstruction process, this technique is still restricted by the required input images, which have to be captured with a wide viewing angle of the scene to yield enough information about the real-world lighting environment. Another approach for capturing environmental lighting was presented and refined in [8][9] where reflective spheres are placed into the real scene. One basic limitation of this approach however is the lack of information about the positions of light sources.

A new method of reconstructing a lighting environment for later integration of virtual objects is proposed by Sato et al [10]. The reconstruction is done automatically using a pair of omni-directional images provided by cameras with fish-eye lenses but is restricted to a fixed view static lighting environment. Finally, Sloan et al. [1] presented another approach for lighting virtual objects using panoramic images. The technique is based on precomputed radiance transfer using spherical harmonics.

## 3. TECHNICAL OVERVIEW

The proposed algorithm consists of two main stages that are described in detail: The preprocessing step for self-occlusion calculation and the real-time rendering process.

### 3.1. Preprocessing Stage

In the preprocessing stage, an occlusion cube map for each vertex of the virtual object is calculated. This information is needed to support correct self-shadowing effects. For each vertex an image is computed in all six main axis directions. The virtual object is rendered completely in black and the background is left white to represent the occlusion factor for the form factor computation. In a second step the resulting occlusion map is weighted by the cosine of the angle $\theta_{ij}$ between the normal vector and each pixel $(I,j)$ on the cube to actually compute the so called delta form factors for each pixel on the occlusion map [11]. The delta form factors $\Delta F_{ij}$ for a pixel $(i,j)$ on the occlusion map are therefore

$$\Delta F_{ij} = \cos\theta_{ij} \frac{A}{\pi},$$

where $A$ denotes the area of the pixel. The factor $A$ divided by $\pi$ is constant and applied later in the process. The cosine calculations are performed by rendering a cosine-textured and normal-vector-aligned sphere, which is centered around the vertex. Thus, the occlusion maps turn into form factor maps. Fig. 4 shows an example precalculation.

The six faces of the occlusion cubes for all the vertices are down sampled and arranged in a single large image. By rendering high-resolution occlusion maps and down sampling them we get accurate estimates for the form factors. Afterwards, these down sampled form factor maps are multiplied by the incoming light from the captured environment map. The down sampling is necessary for efficiency reasons. Since we are dealing with diffuse reflection only, a smaller number of samples of the hemisphere is often quite sufficient, but might lead to blurry shadow boundaries. For a model of 30k vertices the preprocessing step takes about 5 minutes on our test setup.

### 3.2. Rendering Process

For capturing the light information of the scene in which the virtual object should be integrated we use AR-Toolkit [6]. It provides functionalities to capture video frames, create a *Glut/OpenGL* window with the video frames in the background and, most important, detects markers and computes a valid modelview matrix that enables the user to simply render virtual objects into the video stream as if they were present in the real scene.

The given modelview matrix can furthermore be used to find objects in a video stream whose relative position with respect to the marker is fixed. This advantage has been exploited for finding the mirrored sphere, which is to be used for obtaining of the lighting environment.

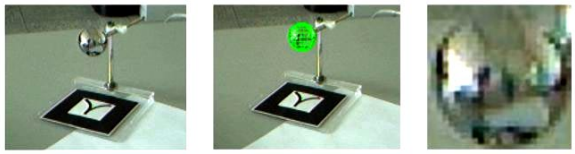

**Fig. 1: AR-marker and light probe construction: Real image (left), calibration mode (center) and captured sphere (right).**

As seen in Fig. 1 a wire-framed sphere is rendered onto the real sphere in the video frame. The position of the spheres is fixed with respect to the marker in both worlds, so both spheres behave similar when the marker is moved. With the given point of origin and the size for the sphere, the exact position and size of a rectangular part of the video frame can be obtained.
Once the sphere has been extracted from the video, it has to be unfolded to a geodesic panorama. Then this mapped sphere is used as a sky dome around the object containing information about the reflected environment from the mirrored sphere. A correct unfolding of the sphere is very time consuming to be done in software per frame. However, this can be avoided since the distortion of the sphere remains constant and can thus be precalculated and used at runtime.Since a mapped sphere which is textured like a globe is needed for further steps of the lighting calculations the unfolding is simply done by creating a sphere with the proper texture coordinates.

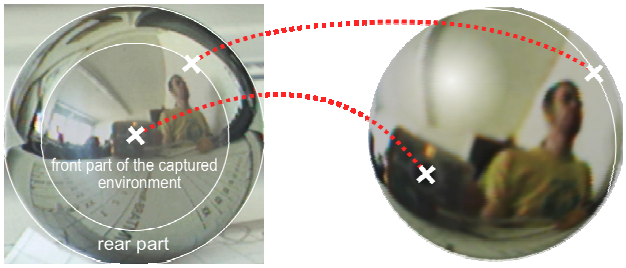


**Fig. 2: The unfolding of the captured sphere image (left) onto a geometric sphere (right) using texture coordinates.**

These are stored in an OpenGL display list and can be used afterwards for every frame using the last captured video sphere as a texture. As shown in Fig. 2 the center of the mirrored sphere reflects the real camera. So the top of the mapped sphere also shows the real camera.
In the next step the restored video panorama must be transferred into another form that fits into the preprocessed occlusion map. This specifically means that the textured panorama sphere must be rotated until it directly faces the camera. The process of transforming an object in a way that it faces the camera is called billboarding.
With the virtual object and a billboarded sphere enclosing it, we have actually placed the object into the captured environment. In the next step of the light calculation process, an environmental cube map is constructed, as shown in Fig. 3.


**Fig. 3: Sample reconstruction of the lighting environment**

In the next step, the environmental cube map and precomputed form-factor map merged and added up to get the final lighting data. Here the use of axis-aligned full-cubes is beneficial because the only step to be done is to create a multi-textured rectangle with the cube-map and the occlusion-map mapped onto.

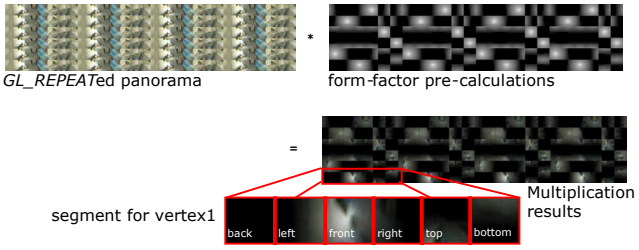


**Fig. 4: The repeated video cube map is multiplied with the preprocessed form-factors**

Fig. 4 shows the result of the multiply-operation between both maps. To accelerate the process, the environmental cube map (Fig. 4 left) is repeated on the GPU to fit the occlusion map (right) and texture multiplication is carried out using the multi-texture pipeline.
In the next step the resulting images need to be accumulated bit by bit to obtain the lighting value of the corresponding vertex. After each pixel block has been summed up and divided by the number of pixels in that block, we get a new image containing one pixel for each vertex of the target object. This image can be used as a lookup table for the final rendering of the object.
In the final step the resulting light values are mapped to the rendered object. Since the lighting is now available as a texture on the graphics card a texture lookup is carried out for each vertex of the object.

## 4. RESULTS

With the proposed system we achieve interactive frame rates for models with 30k+ vertices.

**Fig. 5: Real-time lighting calculation for synthetic object**

We used a 3Ghz Xeon equipped with a QuadroFX 3400 as a test system. The first example in Fig. 5 shows a model within the real-time captured environment showing the lighting effect. Here, a red-color object is placed to the left of the mirror sphere and its lighting change causes color bleeding and thus correct illumination adaptation to the virtual object.
The second example in Fig. 6 shows the results for different lighting intensity and direction for two virtual objects.

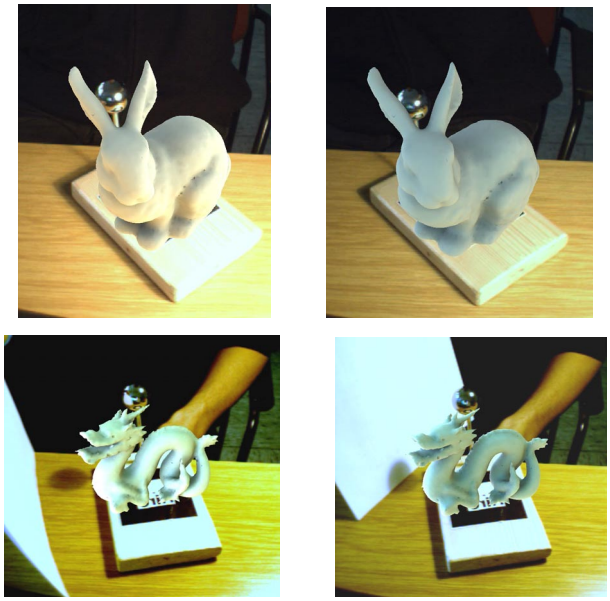

**Fig. 6: Illumination result for different lighting intensity and direction**

Especially for the dragon, the images show, how the virtual object is aligned with the marker on the ground plane and therefore the object is moved in the real scene by moving the marker. Also the precomputed and real-time computed lighting effects can be observed from both examples, namely the self-occlusion and external diffuse lighting respectively.
The current implementation is limited to the texture and rendering buffer sizes of the graphics hardware, with the textures limited to 4096x4096 pixels, which results in a limitation of the object sizes that can be used. Es an example, a form factor resolution of 8x8x6 pixels limits the algorithm to objects of a resolution of no more than 43.520 vertices or 174.080 vertices with a resolution of 4x4x6 pixels. These limits are implementation specific and can by removed by extending the algorithm to use multiple textures for the calculations

## 5. CONCLUSIONS

We presented a technique for capturing lighting information of dynamic environments that we use to consistently light diffuse objects placed in real environments. Our system performs at interactive rates, allowing the user to interact with the virtual objects in real-time. This performance was achieved by splitting the modeling process into precomputation and rendering part, allowing self-occlusion and form factors to be precomputed and used as lookup values in the rendering process. Furthermore, performance optimization was achieved by implementing most of the algorithm on the GPU and thus exploiting graphics hardware features, e.g. multi-texture stage processing and weighting. The approach is not limited to augmented reality scenarios but could also be applied for lighting objects in complex virtual scenes with preprocessed lighting.

## 6. ACKNOWLEDGEMENTS

We would like to thank the Stanford University Computer Graphics Laboratory for providing the happy Buddha, the bunny and the dragon model.

## 7. REFERENCES

[1] P.P. Sloan, J. Kautz, J. Snyder, “Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments”, *SIGGRAPH 2002*, 2002.
[2] A. Fournier, A. Gunawan, C. Romanzini, “Common Illumination between Real and Computer Generated Scenes”, *Proc. Graphics Interface ’93*, pp.254-262, 1993.
[3] M. Billinghurst, H. Kato, “Collaborative Mixed Reality”, *Proc. ISMR 1999, Mixed Reality-Merging Real and Virtual Worlds*, pp. 261-284, 1999.
[4] M. Billinghurst, H. Kato, S. Weghorst, T. A. Furness, “A Mixed Reality 3D Conferencing Application”, *Technical Report R-99-1*, Seattle, Human Interface Technology Laboratory, University of Washington, 1999.
[5] M. Billinghurst, H. Kato, “Real World Teleconferencing”, *Proc. CHI 1999*, Conferencing Companion, 1999.
[6] M. Billinghurst, H. Kato, “Collaborative Mixed Reality: Research Results”, *Technical Report R-99-12*, Presented at the Virtual Worlds Consortium, 1999.
[7] G. Drettakis, L. Robert, S. Bougnoux, “Interactive Common Illumination for Computer Augmented Reality”, *Proc. 8th Eurographics Workshop on Rendering*, pp. 45-57, 1997.
[8] P. E. Debevec, J. Malik, “Recovering High Dynamic Range Radiance Maps from Photographs”, *Proc. SIGGRAPH 1997*, pp. 369-378, August 1997.
[9] P. E. Debevec, “Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography”, *Proc. SIGGRAPH 1998*, pp. 189-198, July 1997.
[10] J. Kajiya, “The Rendering Equation”, *SIGGRAPH 1986*, pp. 143-150, 1986.
[11] M. F. Cohen, D. P. Greenberg, “The Hemi-Cube, A Radiosity Solution for Complex Environments”, *Computer Graphics*, vol. 19, no.3, pp.31-40, July 1985