Interactive Image Based Lighting in Augmented Reality

Peter Supan*
Ines Stuppacher†

Digital Media
Upper Austria University of Applied Sciences
Hagenberg / Austria

Abstract

Our work focuses on using Image Based Lighting in Augmented Reality applications. In our application the real environment has immidiate effects on the virtual objects. It is possible for us to render reflective objects and see our own reflection in the objects, updated 30 times per second. Moreover the virtual objects cast soft shadows on the real floor and on each other. The virtual shadows resemble the real ones because the real light sources are detected in the environment.

Keywords: image based lighting, augmented reality, photorealistic, realtime, interactive, soft shadows

1 Introduction

In this paper we present the use of realtime Image Based Lighting in an Augmented Reality (AR) setup. Augmented Reality is a way to extend the real world with a virtual overlay (also see [2]). This could be simple information about the real world or photorealistic renderings of objects like car prototypes or architectural models.

In many AR applications it is desirable for virtual objects to fit seamlessly into the real environment. This requires the virtual objects to be lit consistently with their neighbouring real objects. We try to enhance the lighting of the virtual objects by using the technique of Image Based Lighting (IBL), where an image or video of the environment is used to compute lighting. IBL is widely used in feature films and impressive filmlets. A detailed description of the technique is offered in [5] and [4]. In [8] a behind-the-scenes report of a filmlet is given. In these applications lighting is captured offline, requiring a huge amount of time. In contrast, our application captures the lighting in realtime. This permits the real environment to influence the lighting of the virtual objects immidiately. It is thus possible for a real person to see his/her own mirror image in one of the virtual objects. An application similar to ours is described in [12].

In this paper we first explain the principles of Image Based Lighting in section 2. Afterwards, we discuss the capturing of realtime environment maps in our setup in section 3. In section 4 the final rendering of the virtual objects is described. Here we also explain how to add shadows to the virtual scene. Finally we discuss the results of our approach and future work in section 5.

*peter.supan@fh-hagenberg.at
†ines.stuppacher@fh-hagenberg.at

2 Image Based Lighting

One of the most essential aspects to make virtual objects look like real ones is lighting. In traditional computer graphics a simple lighting model is calculated with point light sources, which have a distinct position and are infinitely small. Thus, an object only receives light from a few user-defined points. It is a problem to approximate non-point-shaped light sources (e.g. neon tubes or overcast sky), because a large number of point-lights would be necessary. In the real world an objects does not only receive photons emitted from light sources, but also photons which bounce off other objects. Image Based Lighting can simulate these effects by using an image of the environment to calculate lighting. Thereby it is also possible to simulate irregularly-shaped indirect light sources.

In Image Based Lighting it is usually assumed that the light-emitting objects are infinitely far away (see [9, chapter 19] for an exception). While this is not theoretically correct, it enables extremely efficient rendering.

2.1 Diffuse Lighting in IBL

A point on a Lambertian diffuse surface reflects light equally in all directions. But it also does not reflect light *from* only one direction. It is assumed that light is reflected from a hemisphere of approximately 180 °(see figure 1). The axis of this hemisphere is the surface normal $\vec{N}$ of the rendered point P. This is because the pixels along the surface normal have the strongest impact on the lighting. The impact gradually decreases in direction of the surface tangent. Therefore, instead of one or more distinct light-sources calculated with Lambert's law, all points in the environment map which face the surface are sampled. Afterwards they are weighted by the cosine between their direction and the surface normal, as if all of them were directional lights calculated with Lamberts law. The sum of these values is used to calculate lighting.

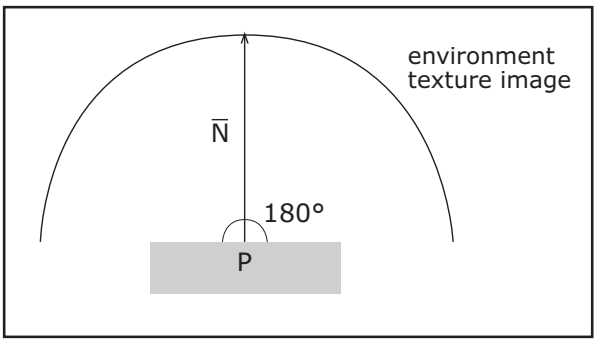


Figure 1: A point on a surface receives light from a 180°solid angle.

Unfortunately, that many texture lookups cannot be performed in realtime. Instead of performing all the lookups for every rendered pixel, the map is pre-convoluted. Every texel in the environment map represents a vector into the environment. For every texel all entries in the environment map can be sampled, weighted by the cosine of the angle between the vectors represented by the sample and the current texel and accumulated. This resembles the amount of light a surface pointing in the direction of the current sample receives. Accordingly one texture lookup at the direction of the normal vector retrieves the correct value of diffuse lighting. Another preconvolution method approximates the above results by blurring the environment map. This does not give exactly the same result but is close enough for lighting. The big advantage of this technique is that it can be implemented faster than the exact solution. By using a Gaussian blur filter, every point in the environment texture receives information from its surrounding pixels. The kernel of the filter has to include that area of the texture that represents half of the environment for diffuse lighting. The map resulting from the convolution is called an irradiance map (see [9, chapter 3] for further details). Table 1 (d) shows an example.

2.2 Specular Lighting in IBL

Highlights on objects are basically reflections of light sources. Thus it would theoretically be appropriate to use environment mapping for specular highlights. But this is only correct for perfectly reflective surfaces (like chrome). Most materials absorb and scatter light. Only bright objects like light sources are seen as reflections (see figure 2). Even these objects are not sharp, but blurred. Not only the reflection from one direction is visible, but from all directions in a certain solid angle (see figure 3). The size of the solid angle depends on the shininess of the object's material. The center of this solid angle is the reflection $\vec{R}$ of the eye vector $\vec{V}$ on the surface. Instead of one or more distinct light sources calculated with the formulas of Phong or Blinn, all points in the environment map at this solid angle are sampled. Afterwards they are weighted by the cosine between their direction and the reflection vector $\vec{R}$ raised to the power of shininess. Thus they are treated as if all of them were directional lights calculated with Phong's formula. The sum of these values is the amount of specular light coming from this direction of the environment.

Figure 2: On most sufaces (like the skin of an apple) the reflection is blurry and only bright objects are visible.

Again, so many texture lookups cannot be performed in realtime. Pre-convolution of the map is necessary (see section 2.1). Like with diffuse textures, all texels of the environment map can be sampled, this time weighted by the cosine of the angle between the vectors represented by the sample and the current texture taken by the power of the shininess of the material. The result is the amount of light that is seen if the reflection of the eye vector points in the direction of the current sample. Again another possibility is to approximate the result with a blur filter. The problem with specular maps is that the size of the filter kernel depends on the shininess, which varies between different materials. Theoretically a separate texture has to be generated for every material. An alternative is to generate Mip-Maps and use a lower Mip-Map-level for blurrier materials. Another possibility is using the technique of Summed Area Tables (see [11]).

To simulate the absorption of light in the material it may be necessary to darken the specular map, so that only bright lights are visible as highlights.

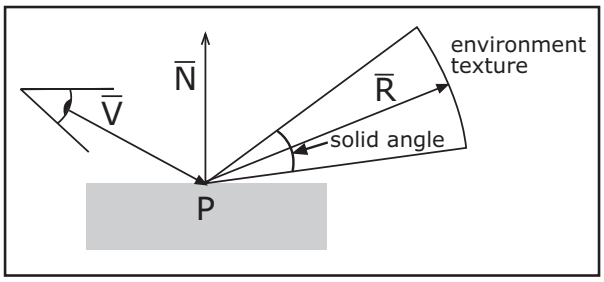


Figure 3: When simulating blurry reflections all texels in a certain solid angle have to be sampled.

2.3 Setup

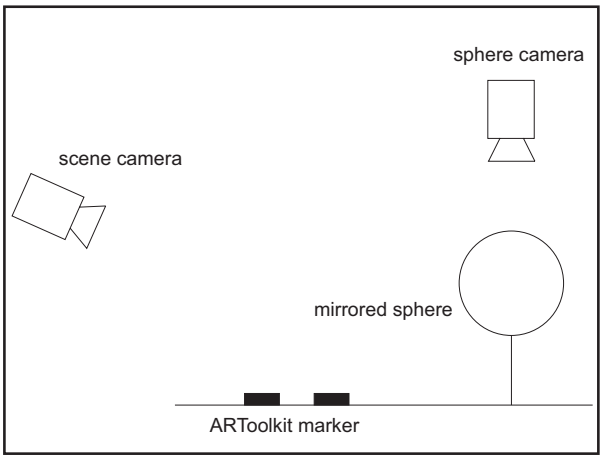


Figure 4: The setup using two cameras and the mirrored sphere to capture the environment.

The aim of our work is to use the techniques described above in an Augmented Reality environment, updating the environment map continuously. To capture the real environment a mirrored sphere (similar to a christmas sphere) is well suited. In its reflection nearly the entire environment can be seen. Images of this sphere are captured and used to create environment maps. We figured out two kinds of setup to achieve this.

1. The first setup uses two cameras as seen in figure 4. One camera captures the environment. This can be done either with a mirrored sphere, as mentioned above, or with a fisheye lens on the camera mount. The second camera captures a video of the scene which afterwards is overlaid with virtual objects.

2. The second setup uses only one camera which captures the scene to be overlaid (see figure 5 and 6). In this scene a mirrored sphere is used which reflects the environment. The sphere is in a fixed position relative to a marker of a visual tracking system (like ARToolkit [13] or ARTag [10]). The marker is tracked to calculate the position of the sphere in the video image. The image of the sphere is then cropped of the video image and used as an environment map.

In our application we use the second setup because it requires fewer hardware components.

3 Retrieving the Environment Maps

3.1 Finding the Mirrored Sphere

The first step is to find the image of the mirrored sphere in the camera image. The position of the sphere relative to the marker $\vec{S_m}$ is known. The transformation matrix M from the camera to the center of the marker is known too. To get the center of the sphere in camera space $\vec{S_c}$ we multiply $\vec{S_m}$ with M (see figure 7). Now we create a rectangle in viewspace which faces the camera. The side length of the rectangle is equal to the diameter D of the sphere. This rectangle is projected into screen space and now equals the bounding rectangle of the sphere image in the video image (see figure 8).

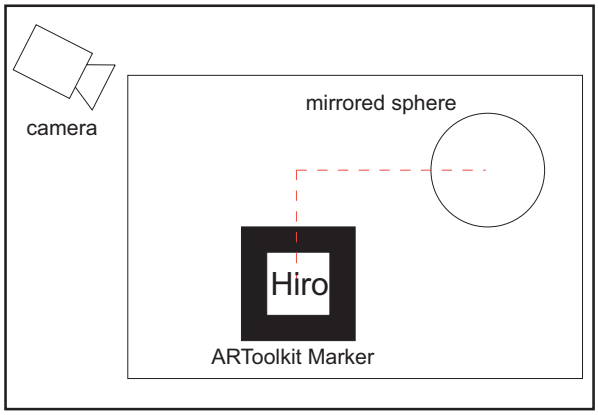


Figure 5: Our setup viewed from above.

To use the image of the sphere as a sphere map, it has to be extracted from the camera image. For this purpose the camera image is copied into a texture. A quad is rendered with this texture mapped onto it. The texture coordinates of this quad are set to equal the above mentioned bounding rectangle of the sphere. Thus in the rendered quad only the sphere is visible. This rendering is copied to another texture that can now be used as sphere map.

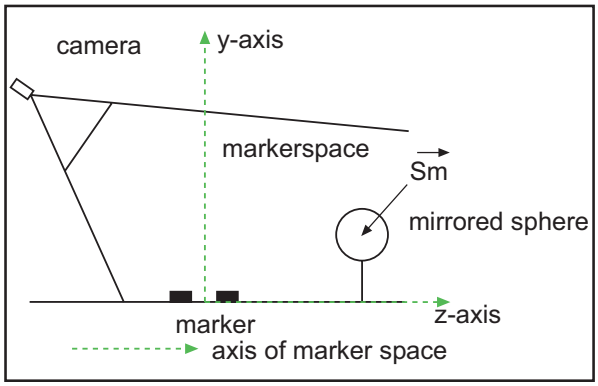


Figure 6: Our setup seen from the side. The sphere is in a certain position relative to the marker.

3.2 Creating Sphere Maps

3.2.1 Sphere Maps vs. Cube Maps

A big issue in the beginning of the project was the question if sphere maps or cube maps should be used for rendering.

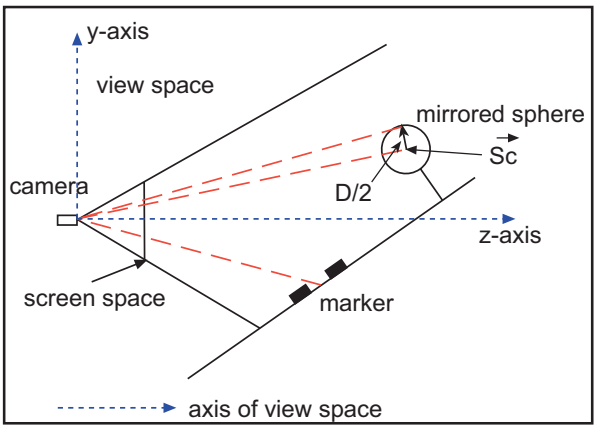


Figure 7: After transforming with the matrix of the marker, the position of the sphere is in camera coordinates.

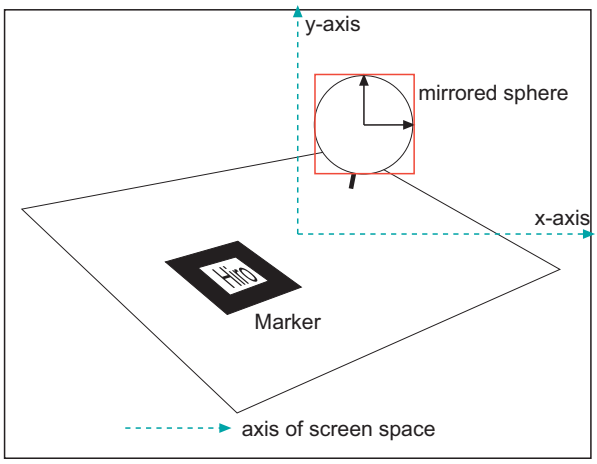


Figure 8: The scene as seen from the camera. The calculated bounding rectange of the sphere is marked red.

The image captured from the mirrored sphere is equal to a sphere map. But even if the sphere reflects (almost) the entire scene the resolution in the border regions is bad (see figure 9). Artifacts may also occur when retrieving samples across the border regions of the sphere map. When looking at reflective objects from behind, these drawbacks of sphere maps become visible through bad reflections. In our application the camera always views the mirrored sphere and the virtual objects from the same direction. Border samples from the sphere map are mapped to borders of the object, so the bad resolution is concealed. Using a separate camera to capture the mirrored sphere (see section 2.3) the disadvantages of sphere maps would most likely become visible. Thus it would be more attractive to use cube maps. But as mentioned above the captured image is already a sphere map. Using a cube map requires to convert this map into a cube map every time a new image is avaliable from the video.

We decided that sphere maps are the more efficient solution as long as the mirrored sphere is viewed from the same camera as the scene.

Figure 9: In the closeup the distortions and bad sample resolution at the border of the spheremap is visible.

3.2.2 The Specular Sphere Map

The image of the mirrored sphere can be used as environment map for perfectly reflective surfaces (like chrome). For glossy materials (like plastic) the map has to be preprocessed (see section 2 for details).

An ordinary 2D Gaussian blur filter is not sufficient because the sample resolution at the edge of the sphere map is different from the sample resolution in the middle. The solution is a radial blur in all 3 axis (see [14, page 100]). Table 1 illustrates this process. The radial blur is achieved by rendering a sphere with radius 1, seen from above, with the perfectly reflective sphere map mapped onto it. The texture coordinates are chosen so that the image of the sphere is seen exactly on the sphere object. The sphere is rotated step by step around the *x*-axis. At every step an image is taken and all the images are averaged, weighted by the amount of rotation of the sphere. The resulting blurred image (see table 1 (a)) is now mapped onto the sphere object. This time the sphere is rotated around the y-axis and the images are averaged. The resulting image (now blurred in x and y direction) is mapped onto the sphere object again and finally the sphere is rotated around the z-axis. Table 1 (d) shows the resulting map.

3.2.3 The Diffuse Sphere Map

Like the specular sphere map, the diffuse map has to be preprocessed too. The difference is that the diffuse sphere map has to be blurred over a solid angle of 180°. To achieve this with the technique described above, a lot of samples would be necessary to avoid banding artefacts. To reduce the number of samples the map is scaled down before blurring.

4 Rendering

4.1 Rendering IBL

Rendering and lighting with IBL is not much different from usual rendering. Instead of the results of the formulas of Lambert and Blinn the texture lookups into the diffuse

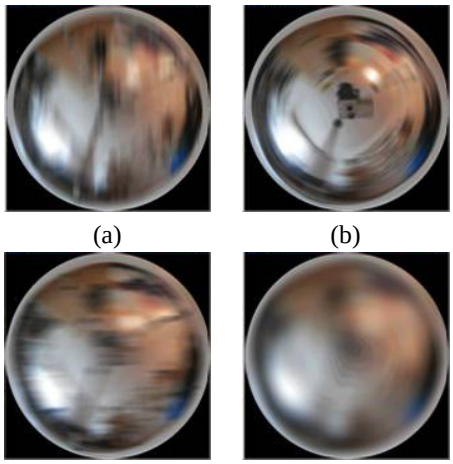


Table 1: The process of blurring the sphere map correctly. Picture (a) shows the effect of the radial blur filter in the x-axis, (b) shows the effect of the same filter in the z-axis, (c) the effect in the y-axis and (d) shows the resulting blurred image.

and specular environment maps are performed. Figures 11 and 10 show the vectors involved in the calculation of the texture coordinates. The diffuse environment texture lookup is done with the surface normal $\vec{N_{eye}}$. For the specular texture lookup the reflection of the vector ($\vec{V_{eye}}$) from the eye to the surface point at $\vec{N_{eye}}$ is used. This vector is called $\vec{R_{eye}}$. Since the sphere map is in view space, all

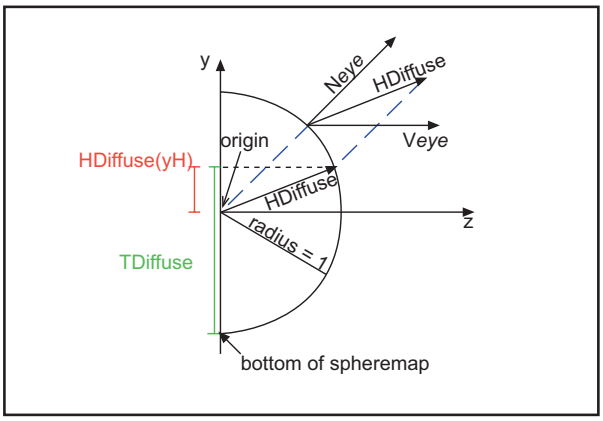


Figure 10: Calculation of sphere map texture coordinates for a diffuse lookup.

these vectors have to be in view space too. The calculation of the actual sphere map texture coordinates is performed as seen in [1, section 5.7].

4.2 Adding Shadows

Shadows are important to determine proportions and relative positions of objects. For a survey about the effect of shadows in Augmented Reality scenes see [7].

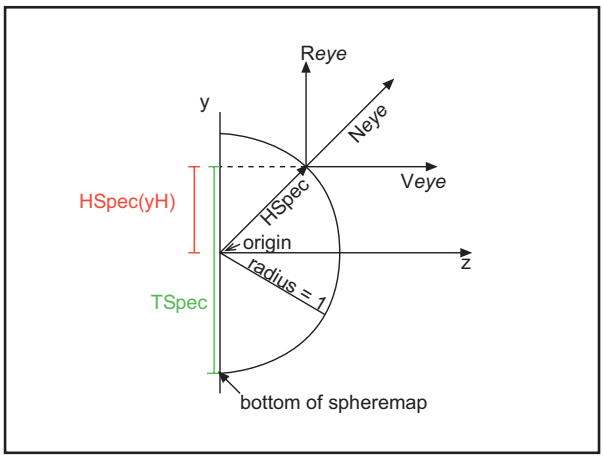


Figure 11: Calculation of sphere map texture coordinates for a reflective lookup.

Every real object in a real scene casts shadows. Virtual objects usually do not and so they are easily identifiable as virtual. In pure virtual environments approximated shadows may look visually acceptable. In Mixed and Augmented Reality environments virtual shadows can directly be compared to shadows of real objects. In this case they have to match the real shadows in shape as well as in color, softness and direction to look realistic.

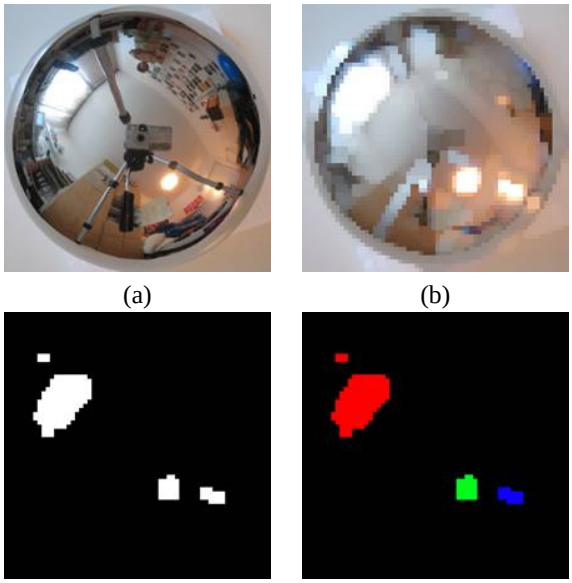
The geometry of the real environment is usually not known. Virtual objects only cast shadows on themselves and onto a ground plane determined by the markers. To overcome this virtual shadow receivers representing the real objects may be registered in the virtual scene like in [7]. So real objects may also receive virtual shadows.

4.2.1 Extracting the Light Sources

To calculate adequate shadows it is necessary to know the position and shape of all light sources in the real scene which are strong enough to cast shadows. Since all our lighting is done with environment maps, we do not have this information. We could either change the light parameters manually (which is a very inflexible solution) or extract them out of the environment map.

There are some very sophisticated methods for extracting light sources out of environment maps used in offline rendering (see [6]). For realtime we use only a very basic method. We retrieve the potential light sources by finding all pixels in the environment map which are above a certain threshold. If many points are found, those close together are clustered. Every pixel on the environment map represents a direction into the environment. So the center of a region of clustered potential light sources can be transformed into a directional vector, the area of the region can be used to determine the size of the light source.

This method may lead to problems in bright environments. Objects which do not emit enough light to cause



(a) (b) (c) (d)

Table 2: The process of finding useful light positions. In picture (a) the original sphere map is visible. This sphere map is scaled down with a maximum filter, as seen in (b). Picture (c) shows all potential light sources, and picture (d) shows them clustered to three main light sources.

shadows are also interpreted as light sources, because they appear as bright as the light sources themselves in the video image. In section 5.1 this problem is investigated in more detail.

4.2.2 Soft Shadows

Shadows in the real world almost always have a soft edge. Perfectly hard shadows are acceptable in games, but in AR they do not suffice in direct comparison with real shadows. There are several techniques to render soft shadows. Some of them are fairly realistic, e.g. penumbra wedges (see [15]) or smoothies (see[3]), others only blur the shadows as efficiently as possible [16, page 269ff].

We decided to use the shadow-map algorithm to render shadows in our scene because it is easier to simulate a kind of penumbra with this technique than with shadow volumes. For the moment we use one of the simplest solutions, which takes several samples from the shadow-map to create a blurred shadow. Although simple, this approach greatly enhanced the visual quality of the virtual objects. In the future, we want to use one of the more realistic algorithms mentioned above.

5 Conclusions and Future Work

5.1 High Dynamic Range

As mentioned in section 4.2.1 the extraction of the light sources from the sphere map image may cause difficulties. Due to the limited dynamic range of digital cameras it is very likely that a lot of pixels have the same value (pure white), although the real light emission of the objects differs greatly. Consider a scene where the sun is visible and illuminates a white wall. Most likely both the sun and the wall will appear white in the video image, although the sun is several hundred times brighter than the wall and thus causes a much more distinct shadow. A solution to this problem would be the use of images with higher dynamic range (HDR images, see [5]). These can be created by taking several images with different exposure settings. Either aperture stop, gain or exposure time can be adjusted to get a series of images with different brightness. Both dark and bright areas in the scene are exposed correctly in one of these images. The combination of the images can be used as one HDR texture to calculate lighting and extract the strongest light sources.

5.2 Other Setups

5.2.1 More cameras

A drawback of the setup with one camera (see section 2.3) is that a new environment map can only be created if the mirrored sphere is inside the camera image. At the moment we are experimenting with a two-camera setup, where one camera captures the scene and the other one the environment. This has the advantage that the resolution of the sphere map is not decreased when the scene camera moves away from the mirrored sphere. Another advantage is that instead of the mirrored sphere a camera with a wide fisheye lens can be used. This simplifies the hardware setup, because the camera does not need to be exactly aligned.

5.2.2 More Spheres

Video cameras have limited dynamic range. As discussed in section 5.1 this causes light sources of different intensity to have the same pixel value in the video image. To overcome this problem, different types of spheres for the environment map and light position extraction could be used. The mirrored sphere would create the environment map as usual. In the reflection of a black, glossy sphere, like the black sphere in a pool game, only bright lights are visible. The idea is to use a sphere like this for the extraction of the light source.

5.3 Visual Results

In the following images you see our setup with the mirrored sphere in the back. On the left side we placed a real

can for comparison. In figure 12 the virtual objects are lighted using only the irradiance map. Figure 13 shows them perfectly reflective. Finally figure 14 shows the objects lighted with their different material properties. The teapot is rendered perfectly reflective while the can adds a decal texture to its reflective material. The globe uses both diffuse and reflective lighting combined with a decal texture.

Figure 12: The virtual objects with diffuse lighting only.

Figure 13: The virtual objects with reflective lighting only.

6 Acknowledgement

The authors would like to thank Michael Haller for coaching and Jürgen Zauner for support with the JazzUp library.

References

[1] Tomas Akenine-Moeller and Eric Heines. *Real-Time Rendering*. A K Peters, 2. edition, 2002.

Figure 14: The virtual objects with combined diffuse and reflective lighting and decal textures.

[2] Ronald Azuma. Overview of augmented reality. In *GRAPH '04: Proceedings of the conference on SIGGRAPH 2004 course notes*, page 26, New York, NY, USA, 2004. ACM Press.

[3] Eric Chan and Frédo Durand. Rendering fake soft shadows with smoothies. In *Proceedings of the Eurographics Symposium on Rendering*, pages 208–218. Eurographics Association, 2003.

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

[5] Paul Debevec. Image-based lighting. *IEEE Computer Graphics and Applications*, 22:26–34, March–April 2002.

[6] Paul Debevec. A median cut algorithm for light probe sampling. Technical Report 67, USC Institute for Creative Technologies, August 2005.

[7] Stephan Drab. Echtzeitrendering von schattenwurf in augmented- und mixed-reality-anwendungen. Master's thesis, Fachhochschule Hagenberg, Medientechnik und -design, Hagenberg, Austria, Juli 2003.

[8] Debevec Paul et. al. Estimationg surface reflectance properties of a complex scene under captured natural illumination. Technical Report ICT-TR-06.2004, University of Southern California Institute for Creative Technologies Graphics Laboratory, June 2004.

[9] Randima Fernando. *GPU Gems*. Addison-Wesley, 2004.

[10] Mark Fiala. Artag revision 1, a fiducial marker system using digital techniques. In *NRC/ERB-1117*, November 2004.

[11] Justin Hensley, Thorsten Scheuermann, Greg Coombe, Anselmo Lastra, and Montek Singh. Fast summed-area table generation and its applications. Technical report, University of North Carolina at Chapel Hill and ATI Research, June 2005.

[12] Karlsson J. and Selegard M. Rendering realistic augmented objects using an image based lighting approach. Master's thesis, Linkpings Universitet, Department of Science and Technology, Sweden, June 2005.

[13] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *IWAR 99: Proceedings of the 2nd International Workshop on Augmented Reality*, October 1999.

[14] Bea Langsdorf. Gpu programming exposed: The naked truth behind nvidia's demos. Technical report, NVIDIA Corporation, August 2005.

[15] Tomas Akenine Möller and Ulf Assarsson. Approximate soft shadows on arbitrary surfaces using penumbra wedges. In *Proceedings on the 13th Eurographics Workshop on Rendering 2002*, pages 309 – 318, June 2002.

[16] Matt Pharr. *GPU Gems 2*. Addison-Wesley, 2005.