International Journal of Virtual Reality, 2006, 5(3):1-10

Image Based Shadowing in Real-Time Augmented Reality

Peter Supan, Ines Stuppacher, Michael Haller
Digital Media, Upper Austria University of Applied Sciences
Hagenberg, Austria

***Abstract*—This work presents an approach to render appropriate shadows with Image Based Lighting in Augmented Reality applications. To approximate the result of environment lighting and shadowing, the system uses a dome of shadow casting light sources. The color of each shadow is determined by the area of the environment behind the casting light source. As a result it is possible that changes in the lighting conditions immidiately affect the shadow casting of virtual objects on real objects.**

***Keywords*— Augmented Reality, Image Based Lighting, Real-Time, Soft Shadows**

1. INTRODUCTION

Photorealistic rendering in Augmented Reality (AR) applications is still one of the most challenging research topics in AR. Bimber et al. postulate in [1] a consistent lighting situation between real and virtual objects to achieve a convincing Augmented Reality application. Sugano et al. and Madsen et al. underline the importance of consistent shadows in an AR scenario [2], [3]. Shading and shadows in both worlds must match to achieve a natural merge [4], [5]. Thus the shading and the shadows of the virtual lights have to be consistent with the real world. Agusanto et al. present in [6] the seamless integration of virtual objects in an AR environment using Image Based Lighting (IBL) techniques and environment illumination maps. They postulate a consistent and a coherent virtual world with respect to the real environment.

Image Based Lighting is an efficient technique to illuminate objects with textures of the real environment [7]. It can be seen as a combination of the techniques of Reflection Mapping [8] and Illumination Mapping [9]. While Image Based Lighting alone is well suited for real-time applications, problems arise when it comes to shadowing. Casting shadows requires light sources at distinct positions. Those are not available when lighting is done with textures.

This paper describes a technique to generate shadows for scenes lit with Image Based Lighting. The shadows are sensitive to any changes in the ubiquitous lighting. If the light source changes its size, the softness of the shadow adapts accordingly. Moreover, the color of the shadow depends on the color of the light source. Shadows add the level of realism to a rendered image. As described in [5], shadows are a very essential factor for the 3D impression of a scene. The seamless merging of the virtual world and the real world is a challenging topic in current Augmented (Mixed) and Virtual Reality research.

2. RELATED WORK

Fournier et al. were one of the first researchers, who combined real video images with computer generated content focusing on the problems of common viewing parameters, common visibility, and common illumination [10]. Global lighting models (e.g. progressive radiosity) are still not suitable for AR applications, because they require a complete scene description. Saito et al. present a method for measuring a radiance distribution of the real scene [11]. By using an omnidirectional stereo algorithm, they first create a geometric model of the scene. The radiance of the scene is computed from a sequence of omni-directional images taken with different shutter speeds and mapped onto the constructed geometric model. A similar algorithm is presented by Gibson et al. [12]. They use a large number of fixed pre-calculated light sources, and allow rendering of impressive quality at interactive frame rates. In contrast, Kakuta et al. [13] use a large amount of fixed light sources to render objects under changing lighting conditions. The objects are nevertheless pre-processed and cannot be moved during run-time.

A large amount of research in Image Based Lighting is done by Debevec [7], [14], [15]. He suggests a division of the environment map in areas of equal integrated brightness to find positions of virtual light sources for shadow creation in offline rendering [16]. The achieved rendering results are more than impressive. Nevertheless most of the results are not rendered in real-time. Agusanto et al. demonstrate in [6] a real-time IBL technique, where they mainly focus on improving lighting results without taking into account shadows.

Only few researchers are focusing on real-time shadows for AR-based scenerios [2], [5]. In our first prototype, we combined shadows of real objects with virtual objects and vice versa using shadow volumes [17]. However, shadow volumes seem to be too complicated to be used for achieving good results with high performance. Soft shadow maps in combination with Image Based Lighting techniques seem to be a good way to achieve better results. Karlsson and Selegard [18] integrated soft shadows in an AR scenario using IBL techniques. However, they only allow one fixed shadow. Unlike previous work, our system benefits from the following features:

International Journal of Virtual Reality, 2006, 5(3):1-10

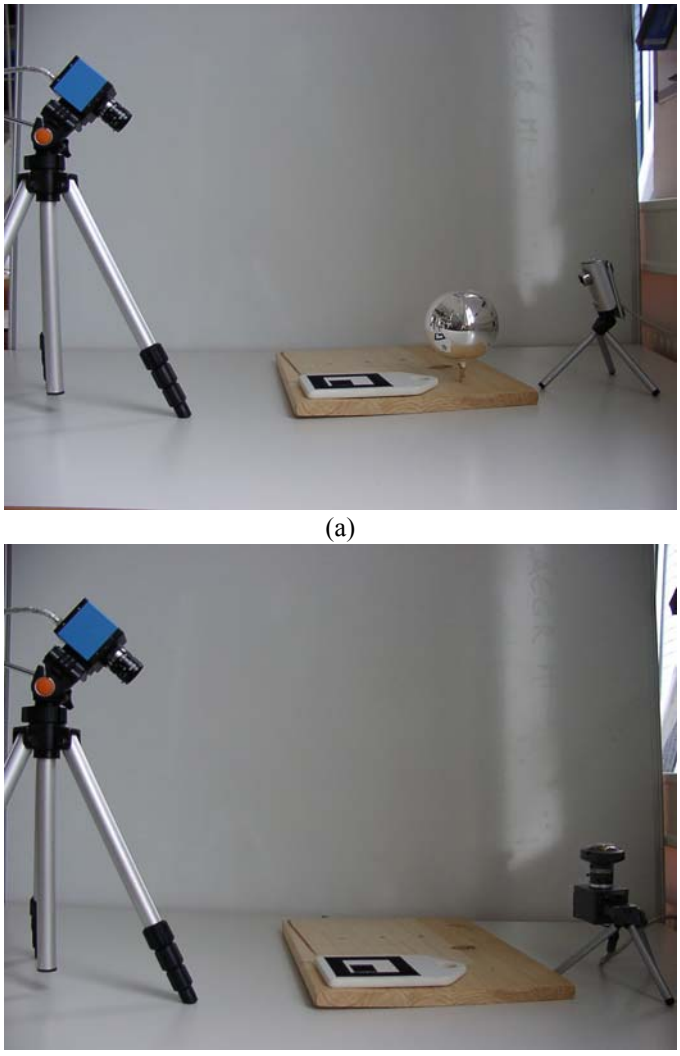
- *Seamless integration of a virtual scenario:* Virtual and real shadows always fall into the same direction with the same color and intensity.

- *Image Based Shadowing:* Shadows are controlled by an image of the environment. There are no limitations in the number and shape of real light sources in the environment.

- *Presentation of three setups:* We demonstrate three different AR scenarios using a single-camera setup with a gazing sphere, and two multi camera setups combined with a mirrored sphere and a fisheye lens for capturing the environment.

- *No pre-processed data:* Our approach does not rely on pre-processed data and thus allows changes of the light situation and changing of object positions during runtime. Nevertheless, this comes at the cost of higher computation.

3. SETUPS



(a)

(b)

Fig. 1. In picture (a) the setup with two cameras and a mirrored sphere is visible. Notice that in the single-camera setup, we just use the left camera for tracking both the environment and the mirrored sphere. Picture (b) shows the setup with two cameras and a fisheye lens, which is simpler and faster to assemble.

We implemented three different setups to test our approach (cf. fig. 1). In the first scenario, we had a single-camera setup in combination with a mirrored (gazing) sphere. The same camera captures both the real scene and the mirrored sphere in the scene. The sphere is in a fixed position relative to the origin of the tracking system (e.g. ARToolKit) and is cropped out of the video image.

The advantage of this setup is that we only have to use one camera. Conversly, the resolution of the environment map depends on the size of the mirrored sphere in the camera image. Since both the marker and the mirrored sphere have to be visible in the camera image at any time, the size of the sphere in the camera image is often very small. As a result the resolution of the sphere map is very low. Furthermore, it often happens that the mirrored sphere gets out of the camera image, which causes that the environment cannot be updated.

The second setup was a two-camera setup with a mirrored sphere. One camera captures the real scene and the other (at a fixed position) tracks the mirrored sphere. Since the mirrored sphere is captured by a separate camera, it is not possible that the sphere is outside of the camera image. Furthermore, the resolution of the environment map does not depend on the distance of the camera, since we always have the same distance from the camera to the mirrored sphere. Notice that both cameras have to be calibrated well, so that the environment map matches the image of the scene camera. Otherwise the cameras often have different white-balance and exposure setting which usually causes a mismatch of lighting of real and virtual objects.

Finally, in the third scenario, we used a two-camera setup in combination with a fisheye lens (180°). The mirrored sphere is replaced by the camera with the fisheye lens which captures the real environment. Thus the setup becomes simpler and easier to calibrate, since there is no mirrored sphere. However, a fisheye lens usually has a smaller field of view than the mirrored sphere. Consequently, a smaller part of the environment is visible in the map.

4. IMAGE BASED SHADOWING PIPELINE

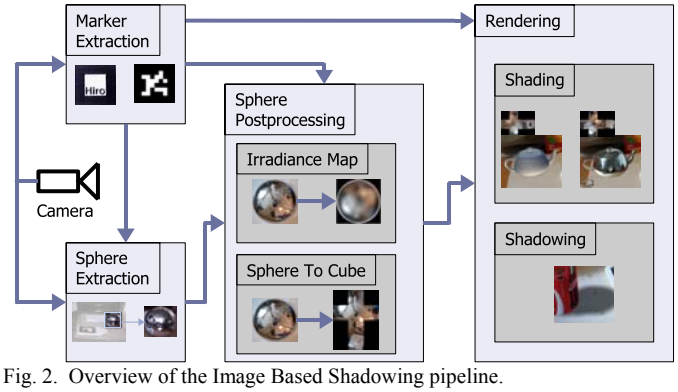


Fig. 2. Overview of the Image Based Shadowing pipeline.

An overview of our algorithm is depicted in fig. 2. Depending on the setup, the images of one or two cameras are used as input to the pipeline. From these images the position of the camera in world space is extracted. Moreover, the image of the mirrored sphere is cropped and copied into a texture. This

International Journal of Virtual Reality, 2006, 5(3):1-10

texture, the *Reflective Environment Map*, is then blurred to create a diffuse *Irradiance Map*. Since both the original Reflective Environment Map and the Irradiance Map are in *Sphere Map* format, they are converted to cube maps. In what follows the resulting cube maps are used for lighting while rendering the virtual objects.

The specular lighting is realized with reflection mapping [8]. The diffuse lighting is done as described in [9], where the surface normal of a point is used for a texture lookup into an environment map. Shadowing is performed by creating a reasonable amount of light sources around the scene which cast shadows with shadow maps. Each of these light sources determines its intensity and color from the area of the environment map behind the light source, as seen in fig. 3.

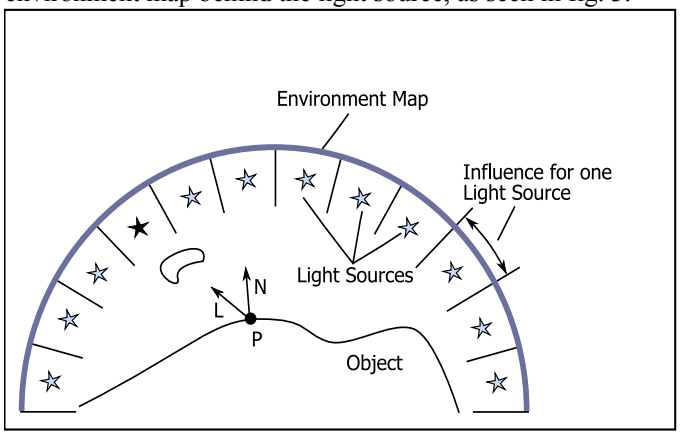


Fig. 3. The color of every light source depends on the average color of an area of the Environment Map behind the light source.

Similar to environment mapping, we assume that the environment is far away from the rendered scene. With this assumption, we can ignore the distance of light-emitting areas, which alleviates rendering. Our algorithm performs best when we simulate soft shadows cast from relatively large direct or indirect light sources far away in the environment. Our approach although fails to simulate a hard shadow cast by point lights or light sources located very near or inside the scene, due to the limited number of light sources.

5. ALGORITHM

Our algorithm for shadowing a scenario is based on three steps, which can be defined as follows:

1) *Creating the Shadow Maps:* We create the *Shadow Map* by rendering the scene from the view of every light source (in our setup we used up to 64 light sources simultaneously) and storing the depth buffer in a separate texture. Since the rendering of many shadow maps for each frame can become very expensive, we only update one shadow map (of only one light source) in each frame.
2) *Creating the Shadow Buffer:* Next, we create a texture called *Shadow Buffer* containing the accumulated shadows of all light sources. To create this texture we render the scene from the view of the camera. The shadows of all light sources are rendered and the results added (see section 5.2). The color of each shadow is determined by the environment map (cf. section 5.3).
3) *Rendering with the Shadow Buffer:* Finally, we render the scene from the point of view of the camera. The Shadow Buffer created before is projected on the scene. At every pixel the value stored in the Shadow Buffer is subtracted from the diffuse lighting value of the pixel to realize shadowing (see section 5.4).

5.1 Calculating the Shadow Maps

The scene is rendered from the view of the light source. The color writing flag is disabled so that only depth values are written into the buffer. Next, we copy the depth values to a depth texture, where one depth texture is stored along with every light source.

5.2 Calculating the Shadow Buffer

In step two of our algorithm, the shadow intensity of every light source is determined. Fig. 3 shows a cross section of the used scene. In theory each point P on the surface receives light from every visible light source. This means that the incident light on P equals the accumulated light from every light source not casting a shadow on P. If the surface is lit by Lambert's law, the influence I of every light is calculated as follows

$$I = C \cdot N \bullet L \cdot S \,, \tag{1}$$

where C is the color of the light source, N is the normal of the surface and L is the normalized vector from the rendered point P to the light source. The shadowing factor S equals 0.0 if the light casts a shadow on the rendered point and 1.0 if the point is in light. The illumination IL on every point is therefore calculated as the sum of all light intensities

$$IL = \sum_{i=0}^{n} C_i \cdot N \bullet L_i \cdot S \,, \tag{2}$$

where n is the number of all light sources. In our application there are objects which are already lit (e.g. the ground plane, which receives shadows from virtual objects but does not receive light from virtual light sources). To deal with these objects, we do not calculate the radiance a point receives from any light source but the radiance R a point does not receive because of shadowing. R is determinded by calculating the illumination as described in equation 2, but with inverted shadowing factor S:

$$R = \sum_{i=0}^{n} C_i \cdot N \bullet L_i \cdot (1 - S) \tag{3}$$

5.3 Acquiring the Light Color and Shadow Value

Notice that for equation 2 the color of the light source has to be calculated. This color depends on the color of the area of the environment map behind the light source. To achieve it, we simply "downsample" the cube environment map to a low resolution (4x4 or 8x8 texels per face as depicted in fig. 4(c)). As a result, every texel now "contains the average" of all surrounding texels in the original high resolution cube map (like fig. 4(b)). Now the positions of the light sources are set in

International Journal of Virtual Reality, 2006, 5(3):1-10

a way that every light source lies in the center of one cube map texel (see fig. 4(a)). The color of every light source is set to the color of the texel behind it.

The shadowing factor S is the result of the Shadow Map texture lookup and tells if a point is lit by the currently calculated light source or if it is in shadow. In fig. 3 and 4(a) S is shown by the fill color of the light source symbols.

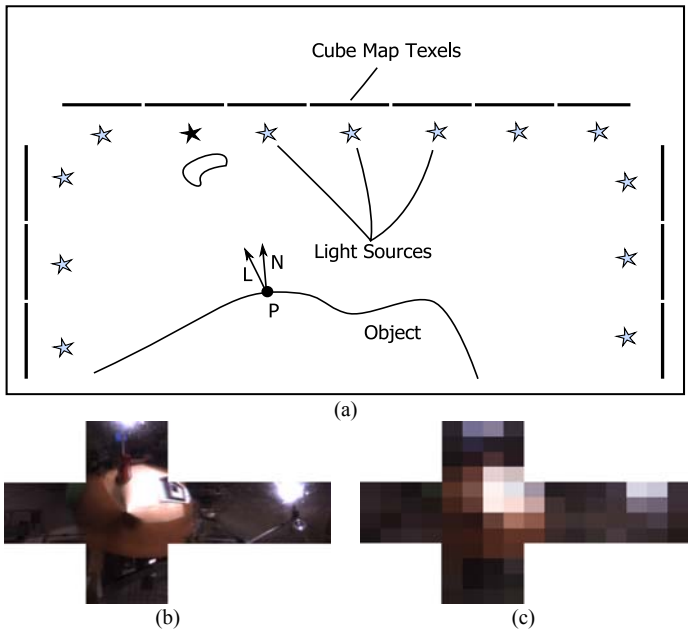


Fig. 4. The final color of the light source is determined by performing a lookup into a downsampled Cube Environment Map.

5.4 Combining Shadows and Lighting

Once the Shadow Buffer is ”finished”, the final lighting can be realized. Specular lighting is calculated by performing a lookup in the Reflective Environment Map at the direction of the reflected eye vector. In contrast, diffuse lighting is calculated by a lookup in the Irradiance Map at the direction of the surface normal of the rendered point. The received value is the amount of diffuse light the point receives if the whole environment is visible.

The amount of light which does not receive the rendered point, because it was blocked by another object, is stored in the Shadow Buffer. This amount has to be subtracted from the diffuse lighting value to achieve correct lighting, as illustrated in fig. 5. Thererfore, at every pixel, lighting is calculated as follows:

$$I = (d - s) + sp\,, \tag{4}$$

where I is the light intensity, d is the calculated diffuse lighting value, s is the value from the Shadow Buffer and sp is the calculated specular lighting value.

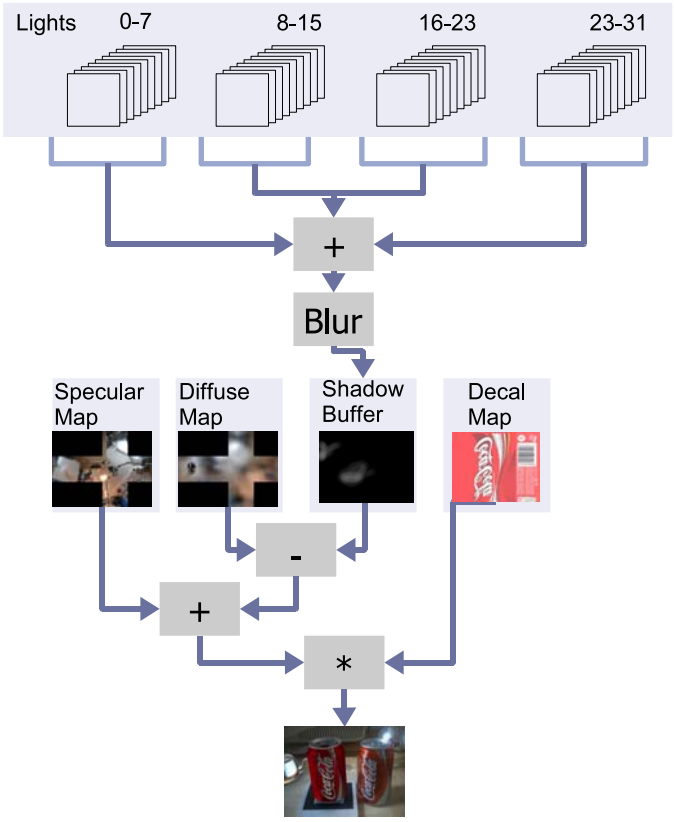


Fig. 5. First the Shadow Buffer is created in several passes with eight light sources each. Then the result is fed into the lighting pass and affects the diffse lighting value.

In our setup, real object should not receive lighting but shall receive shadows from virtual objects. We call these objecs Phantom Objects (cf. [19]). An example is the ground plane. For these objects, the lighting calulations are omitted (because they are already lit by real light). Finally, the value in the Shadow Buffer is subtracted from the original color value to get the impression of cast shadows (although correct results are only achieved for diffuse objects).

6. Implementation Details

The algorithm described above is implemented in OpenGL using the Cg shading language. It is split into several render passes.

1) Environment map processing steps:

Cropping the Sphere Map: We render a quad with the video image of the camera. The texture coordinates are chosen so that only the picture of the mirrored sphere is visible.

Creating the Irradiance Map: We render a quad with the Sphere Map and perform a seperable 2D-Gaussian blur on the map.

Converting to cube maps: We convert the sphere maps to cube maps by rendering six quads. Each quad corresponds to one face of the Cube Map and each pixel of the quad becomes a texel in the Cube Map. Notice that the texture coordinates of the quads are chosen in a way that the texture coordinates of every rendered pixel equal to the 3D-vector into space the Cube Map texel will represent. In the fragment shader, this 3D-vector is

International Journal of Virtual Reality, 2006, 5(3):1-10

converted to a 2D-vector for a simple texture lookup in the sphere map.

Although sphere maps of the environment would be instantly available after capturing the mirrored sphere, we use cube maps in the setups with two cameras. The reason is that with two cameras the camera for the real scene and camera for the environment map will look into different directions. Sphere maps are view dependent and must be taken from the same view direction as the camera which uses them for rendering. Cube maps are better suited for this task due to their view independency.

2) *Creating the Shadow Map:* We generate the Shadow Map by rendering the scene and copy the depth buffer into a GL_DEPTH_COMPONENT texture.

3) Creating the Shadow Buffer: The Shadow Buffer is created by rendering the scene from the point of view of the camera. To get the shadowing value for every pixel, a depth texture comparison has to be performed for every light source in the scene. The necessary texture coordinates are calculated in the vertex shader.

Since only eight texture coordinate sets can be transfered from vertex to fragment shaders in common shader profiles, not all light sources can be calculated in one pass. On the contrary the Shadow Buffer is created in several passes with eight light sources each. The results of these passes are accumulated with additive blending (see fig. 5).

To get the color of the light source a higher mip level of the environment cube map is sampled. As texture coordinates the positions of the light sources are used.

After all passes are finished, the Shadow Buffer is copied to a texture.

4) *Rendering with lighting and shadows:* We render the scene from the point of view of the camera. The texture with the Shadow Buffer is projected so that it fills the whole screen. This is done by mapping from screen coordinates S (in the range [−1.0,1.0]) into texture coordinates T (in the range [0.0,1.0]):

$$T = S \cdot 0.5 + 0.5. \quad (5)$$

7. RESULTS

All the images and performance measurements in the following section were generated on an Intel Pentium IV with 3 GHz and an nVidia GeForce 7800 GT graphics card with a screen resolution of 800×600.

7.1 Performance Results

Table 1 shows the frame rate of the same scene with different numbers of light sources.

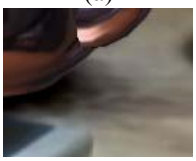
Number of Shadows	fps
0	35
8	32
48	21
64	15

Table 1. Frame rates with different number of shadows. Notice that the frame rate slows down using more than 8 shadows simultaneously.

As more light sources result in more rendering passes during creation of the Shadow Buffer, performance if heavily affected.

(a)

(b)



(c)

(d)

Fig. 6. Comparison of renderings with different number of shadows. Image (a) is rendered with 8 shadows, (b) with 48 and for image (c) 64 shadows are used.

On the other hand, the quality of the shadow increases with more light sources, because the single shadows become invisible. Fig. 6 depicts a comparison of a scene with 8, 48 and 64 shadows.

7.2 Rendering Results

The proposed algorithm offers shadows which change dynamically under varying lighting conditions. Fig. 7 shows a scene with different lighting. A strong light source is located at the upper right of the scene. Note the changing in the shadows of the virtual cola can and the teapot. The real lighting affects the shadow casting of the virtual objects. The resulting shadows are very soft and appropriate to simulate the shadowing caused by environments with a large amount of indirect light and large light sources.

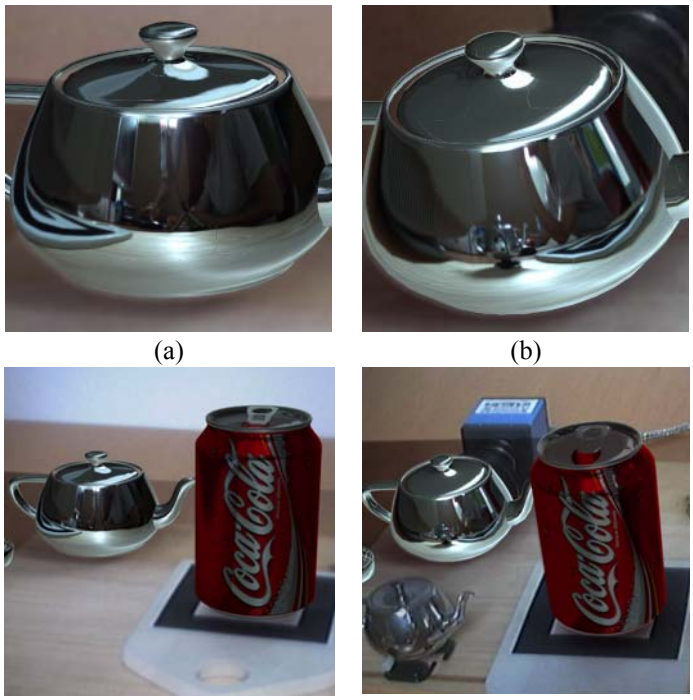
(a)

International Journal of Virtual Reality, 2006, 5(3):1-10

(b)

Fig. 7. The rendered scene under different lighting conditions. In (a) there is a strong light source at the upper right of the scene. (b) If the light source is relatively small, the virtual shadow ends up too soft.

If there is a small, bright light source in the environment, the shadows tend to be too soft (cf. fig. 7(b)). The reason for this is that the virtual shadow is actually a combination of many shadows from fixed light sources. If a real light source is placed between two virtual light sources, the shadow is simulated by two or more blended virtual shadows, which causes a softening effect.



(a) (b)

(c) (d)

Fig. 8. If the light source is relatively small, the virtual shadow ends up too soft. In the reflection of the teapot in figures (a) and (b) the marker is visible on the left hand side, although in reality it is at the right hand side. In images (c) and (d) camera and object are at the same position, the reflection is therefore correct.

Notice that in both setups (using the mirrored sphere or the fisheye lens), the reflection gets calculated wrong. Fig.s 8(a)-(d) depict a scenario, where the marker, on which we placed the can, gets rendered wrong on the teapot's surface. During our tests, we observed that people often do not recognize this effect. However, it will be recognized, once we go too close to the virtual object. Notice also that objects behind the camera are not seen by the fisheye lens.

8. Conclusions and Future Work

The algorithm proposed in this work presents an Augmented Reality setup, where virtual objects are rendered photo-realistically adapted to the real environment. Our system is based on both vertex and pixel shaders to achieve interactive frame rates. In addition, we focused on the implementation of soft shadows. We allow the movement of light sources during run-time which results in a perceptually correct shadowing.

In terms of future work, we want to investigatemore advanced rendering techniques for simulating objects with different surfaces. Our results would also be improved by the combination of BRDF (used for the environment map) with high dynamic range images and occlusion mapping. A website featuring videos of this work can be seen at http://www.officeoftomorrow.org.

References

[1] Bimber, O., Grundhoefer, A., Wetzstein, G., Knoedel, S.: *Consistent illumination within optical see-through augmented environments.* In: IEEE/ACMInternational Symposium on Mixed and Augmented Reality (ISMAR 2003), IEEE Computer Society (2003) 198–207

[2] Sugano, N., Kato, H., Tachibana, K.: *The effects of shadow representation of virtual objects in augmented reality.* In: IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR 2003), IEEE Computer Society (2003) 76–83

[3] Madsen, C.B., Sørensen, M.K.D., Vittrup, M.: *The importance of shadows in augmented reality.* In: Proceedings: 6th Annual International Workshop on Presence, Aalborg, Denmark. (2003) (4 pages) To appear.

[4] Naemura, T., Nitta, T., Mimura, A., Harashima, H.: *Virtual Shadows - Enhanced Interaction in Mixed Reality Environment.* In: IEEE Virtual Reality (VR'02). (2002)

[5] Naemura, T., Nitta, T., Mimura, A., Harashima, H.: *Virtual shadows in mixed reality environment using flashlight-like devices.* Trans. Virtual Reality Society of Japan **7**(2) (2002) 227–237

[6] Agusanto, K., Li, L., Chuangui, Z., Sing, N.W.: *Photorealistic rendering for augmented reality using environment illumination.* In: IEEE/ACMInternational Symposium on Mixed and Augmented Reality (ISMAR 2003), IEEE Computer Society (2003) 208–216

[7] Debevec, P.: *Image-based lighting.* IEEE Computer Graphics and Applications **22** (2002) 26–34

[8] Blinn, J., Newell, M.: *Texture and reflection in computer generated images.* In: Communications of the ACM 19(10). (1976) 542–547

[9] Miller, G.S., Hoffman, C.R.: *Illumination and reflection maps: Simulated objects in simulated and real environments.* In: Course notes for Advanced Computer Graphics Animation, SIGGRAPH 84. (1984)

[10] Fournier, A., Gunawan, A.S., Romanzin, C.: *Common illumination between real and computer generated scenes.* Technical report, Vancouver, BC, Canada, Canada (1992)

[11] Sato, I., Sato, Y., Ikeuchi, K.: *Acquiring a radiance distribution to superimpose virtual objects onto a real scene.* IEEE Transactions on Visualization and Computer Graphics **5**(1) (1999) 1–12

[12] Gibson, S., Howard, T., Hubbold, R.: *Rapid shadow generation in real-world lighting environments.* In: Proceedings of the Eurographics Symposium on Rendering. (2003)

[13] Kakuta, T., Oishi, T., Ikeuchi, K.: *Shading and shadowing of architecture in mixed reality.*In: Proceedings of ISMAR 2005. (2005)

[14] Reinhard, E., Ward, G., Pattanaik, S., Debevec, P.: *High Dynamic Range* Imaging. 1. edn. Morgan Kaufmann (2006)

[15] Debevec, P., Tchou, C., Gardner, A., Hawkins, T., Poullis, C., Stumpfel, J., Jones, A., Yun, N., Einarsson, P., Lundgren, T., Fajardo, M., Martinez,

P.: *Estimationg surface reflectance properties of a complex scene under captured natural illumination.* Technical Report ICTTR-06.2004, University of Southern California Institute for Creative Technologies Graphics Laboratory (2004)

[16] Debevec, P.*: A median cut algorithm for light probe sampling.* Technical Report 67, USC Institute for Creative Technologies (2005)

[17] Haller, M., Drab, S., Hartmann, W., Zauner, J.: *A real-time shadow approach for an augmented reality application using shadow volumes.* In: ACM Symposium on Virtual Reality Software and Technology, Tokyo, Japan (2003)

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

[19] Fuhrmann, A., Hesina, G., Faure, F., Gervautz, M.: *Occlusion in collaborative augmented environments.* Computers and graphics **23**(6) (1999) 809–819 http://www.cg.tuwien.ac.at/research/vr/occlusion.

Peter Supan is a researcher at the department Digital Media of the Upper Austria University of Applied Sciences, Austria. His research interests include augmented and virtual reality, real-time computer graphics and rendering, and multimodal human-computer interaction. Supan received a MSc. in engineering from the Upper Austria University of Applied Sciences, Austria. Contact him at peter.supan@fh-hagenberg.at.

Ines Stuppacher is a researcher at the department Digital Media of the Upper Austria University of Applied Sciences, Austria. Her research interests include real-time computer graphics and rendering, multimodal human-computer interaction, games, and augmented and virtual reality. Stuppacher received a MSc. in engineering from the Upper Austria University of Applied Sciences, Austria. Contact her at ines.stuppacher@fh-hagenberg.at.

Michael Haller is a researcher developing innovative computer interfaces that explore how virtual and real worlds can be merged to enhance and improve computer systems. Currently, he is working at the department of Digital Media of the Upper Austria University of Applied Sciences and responsible for computer graphics, multimedia programming, and augmented reality. In 2004, he received the Erwin Schroedinger fellowship award presented by the Austrian Science Fund for his stay at the HITLabNZ, University of Canterbury (New Zealand) and the IMSC, University of Southern California (USA). Contact him at haller@fh-hagenberg.at