# A real-time shadow approach for an Augmented Reality application using shadow volumes

Michael Haller
Upper Austria University of Applied Sciences
Media Technology and Design
Hagenberg, Austria
haller@fh-hagenberg.at

Stephan Drab
Upper Austria University of Applied Sciences
Media Technology and Design
Hagenberg, Austria
stephan.drab@fh-hagenberg.at

Werner Hartmann
Institute for Applied Knowledge Processing
JK University of Linz
Linz, Austria
wh@faw.uni-linz.ac.at

## ABSTRACT

Shadows add a level of realism to a rendered image. Moreover, they are used as visual clues to determine spacial relationships and real-time shadows will gain importance in current real-time computer graphics applications for this reason. Twenty-five years ago, Crow published the shadow volume approach for determining shadowed regions of a scene. In this paper we present a modified real-time shadow volume algorithm that can be used in an Augmented/Mixed Reality application. Finally, the proposed concepts provide a novel sense of visual output of an AR application.

## Keywords

Shadow volumes, Augmented Reality

## 1. INTRODUCTION

The seamless merging of the virtual world and the real world that we live in is a challenging topic in current Augmented (Mixed) and Virtual Reality research. There are a great deal of key issues that enhance the immersive feeling of the user. In this paper we want to focus more on the problem of shadows related to an Augmented Reality world. Needless to say that in an Augmented Reality system a number of problems need to be solved. One of the most important tasks is, of course, that superimposed objects have to be placed on the exact position as they really would exist in the real world. As mentioned by Naemura et. al. [15], the consistency of geometry, time (synchronized world to facilitate smooth interaction), and illumination is an important issue for Augmented Reality applications. In this paper we want to focus on the consistency of illumination and propose a technique for shadows in a superimposed Augmented Reality application. Shading and shadows in both worlds must match to achieve a natural merge [16]. Our approach supports rendering the shadows of real objects onto virtual objects and shadows of virtual objects onto real/virtual objects. Consistency between the real light and the virtual world would be very useful to enhance the realistic impression of virtual objects. A real-time estimation of the position and the direction of the real light source is required to calculate the right shading and shadows of the virtual objects (cf. inconsistent virtual shadow in image 16).

Shadows are essential for the improvement of visual perception. Moreover, they enhance the 3D impression in that the users get a better immersive 3D feeling - We believe that shadows are important in the same way as 3D stereo HMD, because users can sense more exactly the distance between two virtual objects. As a result, the interaction and manipulation of objects is enhanced [13].

Until now virtual worlds (computer games, VR applications) use shadows for enhancing the 3D impression. Although these virtual environments can animate and render the world with near photo-realism in real-time, we cannot say the same about the rendering of Augmented Reality applications. We investigated in methodologies for computing real-time shadows of real/virtual objects onto real/virtual objects. These methodologies also cope with virtual and real objects that change their position/orientation over time.

## 2. RELATED WORK

Raskar et al. [20] use projectors to graphically augment neutral physical objects of the real world. The neutral object is geometrically identical with a real object (with the exception of an occasional scale to decrease the size of the neutral object for indoor usage) but its surface does not contain any color or texture. Those features are projected onto the surface of the neutral object by projectors, so called "Shader lamps". These shader lamps take into account the orientation and distance between the neutral object and the projector and the geometry of the neutral object. The system also visualizes shadows within the neutral object.

Naemura et al. [15] and [16] propose concepts of virtual lights and virtual shadows with the aim of achieving a Mixed Reality environment focused on shadows. The virtual shadows are subdivided into four categories according to the involvement of real and virtual objects. These categories are "real to virtual shadows for rigid objects", "real to virtual shadows for non-rigid objects", "image-based virtual to virtual shadow" and "virtual to real shadow". Shadows of the first and third category can be generated by a virtual light consisting of a stick that is a substitute for a flashlight and a 3D sensor that measures orientation and position of the virtual flashlight. The rigid real objects are also equipped with a 3D sensor to measure their position and orientation. The tracking information of both the rigid object and the virtual flashlight permits the calculation of the shadows of virtual objects and rigid real objects. To display the shadows of non-rigid real objects, a virtual flashlight has a camera attached that records the image in front of the virtual flashlight. Shadows of non-rigid real objects are calculated by first capturing the silhouettes of the non-rigid real objects. The capturing takes advantage of the illumination conditions of CAVE-like VR/MR applications. It is assumed that the non-rigid real object is not directly lighted, and its background is the screen that shows the VR-scene. Thus, the non-rigid real object and its silhouette can easily be distinguished from the background. Using a tracked projector as light source creates Virtual onto real shadows. The projector projects the shadow of the virtual object at the real scene. Thus, without wearing HMDs, the user would only perceive the shadow of the virtual object, but not the object itself. The virtual object can only be seen through HMDs.

Debevec [2] presents a method that creates photo realistic augmentations of real images and videos. The method partitions the scene into a distant scene, a local scene and synthetic objects that augment the real scene. The distant scene contains objects relatively far away from the synthetic objects, whereas the local scene contains all objects that are near to the synthetic objects. It is assumed that the illumination changes of the distant scene, caused by the local scene or the synthetic objects, are very low and can be disregarded. This eases the computation of a global illumination model. Nevertheless, the method is not usable for real-time applications.

Loscos et al. [11], [12] present a system for interactively remodelling and relighting real scenes. The system uses two sets of photographs of real scenes to estimate a calculation of the geometry and the lightning model of the real scene. The first set of photographs consists of images of the scene, taken from different viewpoints. These photographs are used to reconstruct the geometry. The second set of photographs consists of images of the scene taken from one viewpoint but with a real light source at different positions. These images are used to calculate the initial lighting model of the scene. The system permits interactive modification of the scene geometry, including removing objects and adding, changing or removing light sources. The system needs a preparation phase to calculate the lighting model of the real scene, thus it is not suitable for real-time usage.

Everitt et al. [4] present a robust and artifact-free technique for hardware accelerated rendering of stencilled shadow volumes. The method addresses the problem of artifacts that arise from shadow volume polygons that do not completely intersect with objects and thus are clipped "at infinity". The method uses "depth clamping", a feature provided by new graphic accelerators that moves the far clipping plane to infinity.

MIND-WARPING is an augmented reality based multi-user game presented by Starner in [22]. The game uses an infrared camera and a set of infrared emitters which can independently be switched on and off by the system. The system analyzes the shadows that are cast by the object when it is illuminated by different infrared emitters and estimates the geometry of the object from the cast infrared shadows.

Nishino et al. [18] [19] present a method called Eigen-Texture method that synthesizes 3D models of real objects from sequences of range images and color images. The method uses the color images of the object, which are aligned and pasted to the surface of the 3D model of the object. The proposed method is capable to generate realistic images of objects under complicated illumination conditions.

## 3. PROPOSED METHOD

In Mixed Reality and Augmented Reality applications we can see both virtual and real objects simultaneously.

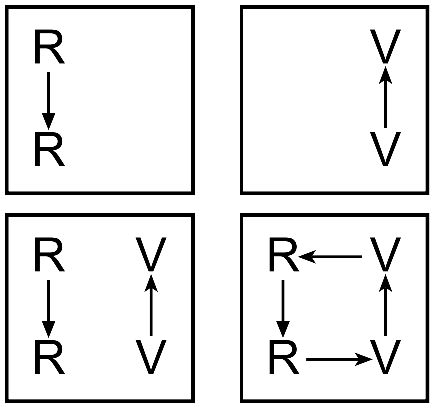


**Figure 1: We distinguish four possibilities of shadow casts. The last figure shows the shadow cast in an Augmented Reality application.**

Based on [16] figure 1 shows in the first two pictures the shadow cast in a real world and in a virtual world. The third picture depicts a complete but isolated shadow cast of real objects onto real objects (the same for virtual objects). Finally, the last image shows the shadow cast in an Augmented Reality scenario. Given one light source, a virtual object can cast a shadow onto a real and onto a virtual object. In contrast, a real object supports shadows onto virtual objects.

### 3.1 General Method of shadow volumes

In this section we give a short overview of the general shadow volume algorithm for virtual objects. The algorithm consists of two basic steps. First, the volumes of the shadows that are cast must be computed. Then, the shadows can be rendered. This is done by determining the intersection between the scene geometry and the shadow volumes. Shadows must be rendered on those parts of the geometry that intersect with shadow volumes.

#### *3.1.1 Generation of shadow volumes*

In 1977 Crow introduced the shadow volume approach in [1], where a shadow volume defines a region of space that is in the shadow of a particular occluder with a given light source (cf. figure 2) [4]. The front- or back-facing orientations of the rendered shadow volume polygons with respect to the viewer indicate the enters into and exits out of shadowed regions [4]. Heidmann proposes a time-saving solution by using the stencil buffer for counting the entries and exits [6]. The stencil buffer is incremented wherever a front-facing polygon of the shadow volume is drawn. In contrast, the stencil buffer is decremented wherever a back-facing polygon of the shadow volume is rendered. Finally, the whole scene is rendered again depending on the stencil bit. If the counter of the stencil buffer is greater than zero, then a pixel is in the shadow, and therefore it is not drawn - otherwise it is not in the shadow and it has to be rendered.

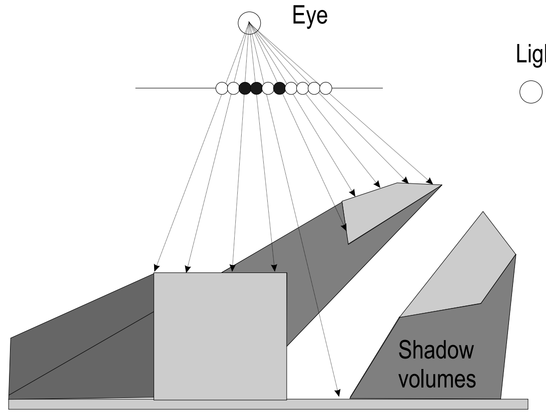


**Figure 2: Principle of the shadow volume algorithm.**

The first step of the algorithm is devoted to finding the silhouette of those objects that are potential candidates for a shadow volume with one given light source. With the given silhouette we extrude the shadow volume that is used for the algorithm. The silhouette of the objects can be calculated with the OpenGL Utility 1.2 library's boundary tessellation routine (`gluTesselator`).

After the calculation of the silhouettes we extrude the shadow volumes. Indeed, each object generates its own shadow volume. Therefore, an overlapping of shadow volumes is possible. Everitt and Kilgard presented a very robust solution in [4] for the problem that the side polygons of the shadow volume are not sufficient for the shadow volume algorithm. In our approach the shadow volume is limited to a given radius that can be modified in a configuration file. In the future, the radius should be calculated based on the distance between two objects. Usually, the radius does not have to be too big. Imagine a small circle far away from the object that is caught by the circle shadow. In general these shadows are no longer visible and therefore a limitation of the radius makes sense.

A second problem is depicted in figure 3, where the user's ray intersects the front face of a shadow volume, but it never exits the back face. In this case, the shadow volume algorithm sets the stencil buffer to one - which, in fact would be wrong. As a result, we have to guarantee even a top face and a base face of the shadow volume (it has to become a closed 'frustum'), to allow for a correct execution of the

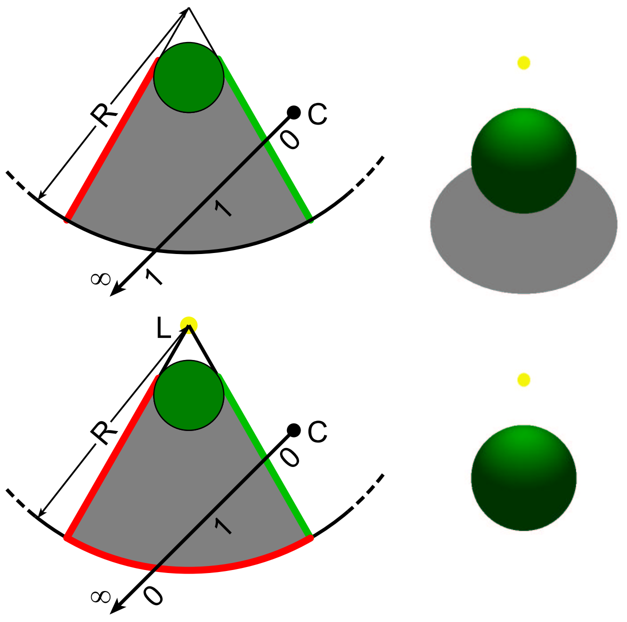
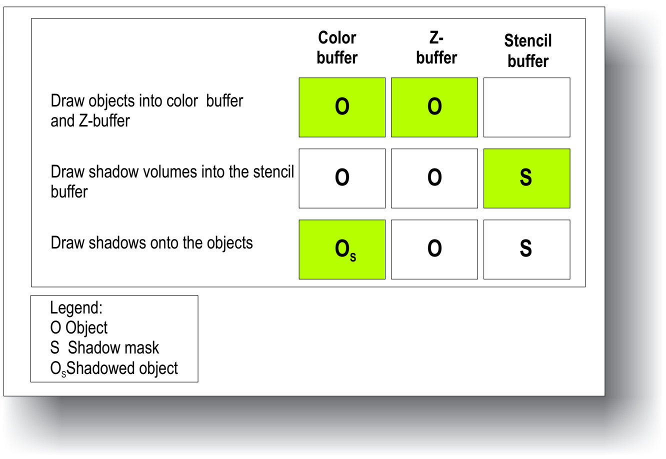


**Figure 3: The first figure shows a scenario in which the user's ray does not intersect the object. However, the stencil bit is set to 1 - even if we are outside of the shadow volume. Consequently, we see a shadow. In the second scenario, the stencil volume is limited in its size and it decrements the stencil bit, even when the ray leaves the stencil volume on the base face. In contrast to the first scenario, we have no shadow from the sphere.**

shadow volume algorithm (as depicted in figure 3).

### 3.1.2 *Rendering algorithm for shadow volumes*

Figure 2 gives a general (and short) overview of the rendering algorithm and describes the steps of the rendering process for the shadows. A closer description of the shadow volume algorithm can be found in [6] [14], and [21].



| | Color buffer | Z-buffer | Stencil buffer |
|---|---|---|---|
| Draw objects into color buffer and Z-buffer | O | O | |
| Draw shadow volumes into the stencil buffer | O | O | S |
| Draw shadows onto the objects | $O_s$ | O | S |

Legend:
O Object
S Shadow mask
$O_s$ Shadowed object

**Figure 4: Overview of the shadow volume algorithm including the snapshots of the different buffers.**

First of all, the color buffer, the stencil buffer, and the depth buffer are cleared. Afterwards, the ambient scene is drawn into the color buffer and its depth information is drawn into the Z-buffer. Next, both buffers, the color buffer and the Z-buffer are set to read only. Then the front-facing shadow volumes polygons are drawn (using the depth test). During this process, the values of the stencil buffers are incremented whenever a polygon is drawn. Then, the algorithm starts drawing the back-facing polygons of the shadow volumes. In this case, the values of the stencil buffers are decremented, whenever a polygon (in this case we are drawing the back face polygon) is drawn. Finally, the entire scene is rendered again, but only where the value of the stencil buffer is 0. In this case the location is illuminated by the light source. Moreover, when we render the scene, the material is switched to specular and to diffuse settings.

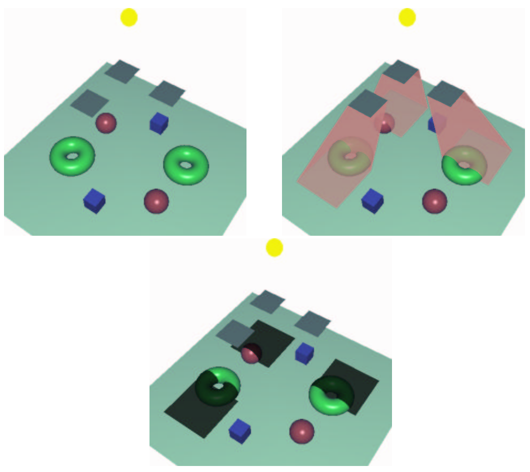


**Figure 5: The first snapshot shows the silhouettes of three objects. In this case the silhouettes match the objects. Their extrusion is depicted in the next image. Finally, the result, the casted shadow, is depicted in the last figure. The yellow point in the center of the objects represents the virtual light source.**

## 3.2 Modified Method in an AR environment

The presented shadow algorithm using shadow volumes works perfectly in a virtual environment. But if we want to use it in an augmented reality environment, we have to make some modifications in the sense that it works for both, virtual and real objects. Therefore, the shadow volume approach of section 3.1 has to be modified accordingly.

The correct order of the instructions is essential, otherwise the algorithm does not work correctly in an AR environment. In any case, our approach consists of two different rendering passes. In the first pass all real objects (including the shadows for the virtual objects) are rendered. Then, the second pass renders all virtual objects. In the following subsections we give a closer description of the modified shadow volume algorithm approach we used in our AR application.

### *3.2.1 Generation of shadow volumes*

First of all, we have to calculate the shadow volumes depending on the objects that potentially cast a shadow. Real objects are represented by virtual objects that are invisible for the user. We call these objects *phantom objects* (cf. figure 6). The phantom model becomes important for the shadow volume calculation. As described later in section 4, it is not necessary to have the exact phantom model for each real object in the world. However, the user usually does not recognize if the phantom object does not match the real object exactly. Next, we can start to calculate the silhouette of both kinds of objects (virtual and real) and extrude the shadow volume. The implementation of the silhouette calculation and the extrusion of the shadow volume can be easily implemented with the same method as used in the general shadow volume algorithm of section 3.1.1.

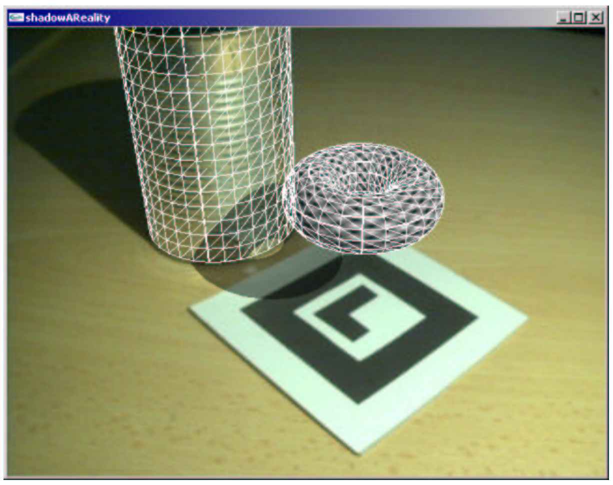


**Figure 6: Not only the torus, but also the real tin exists as a virtual object (phantom).**

Figure 6 shows a possible scenario of two objects that generate shadow volumes. The first object, the tin, is a real object and the second one, the torus is virtual.

### *3.2.2 Rendering algorithm for shadow volumes*

Figure 7 gives a short overview of the modified shadow volume algorithm from section 3.1.2.

As shown in figure 1, the algorithm has to work for all cases, for real/virtual objects that cast shadows onto real/virtual objects. Consequently, the four different cases are rendered in the following three steps:

- Step 1: Real objects cast their shadows onto real objects
- Step 2: Virtual objects cast their shadows onto real objects
- Step 3: Real and virtual objects cast their shadows onto virtual objects

Step 1 and step 2 are different from the shadow volume algorithm described above. The final step is comparable to the well known rendering algorithm of section 3.1.2.

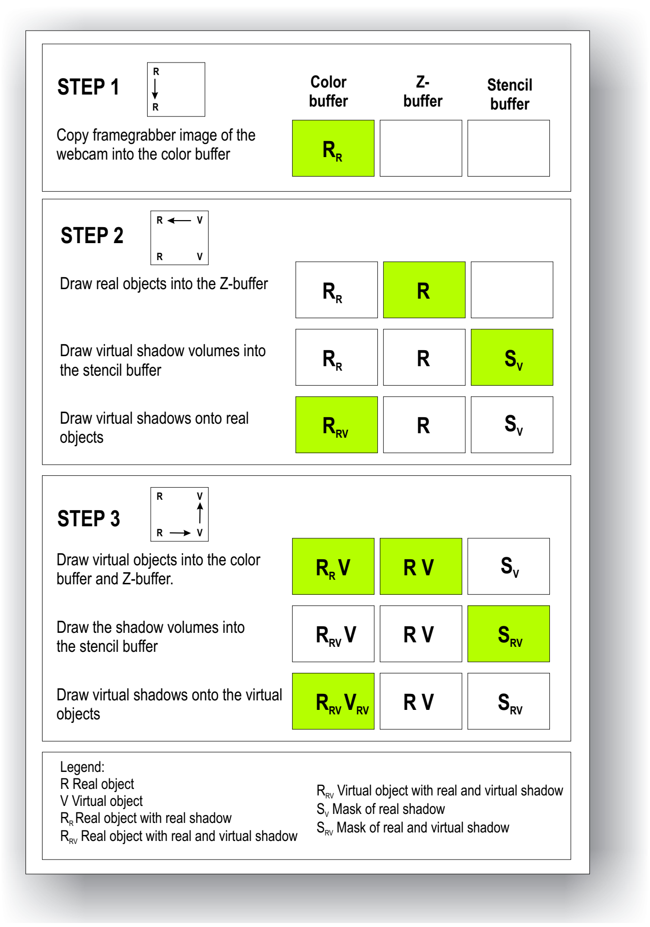

STEP 1
Color buffer
Z-buffer
Stencil buffer
Copy framegrabber image of the webcam into the color buffer
$R_R$
STEP 2
Draw real objects into the Z-buffer
$R_R$ R
Draw virtual shadow volumes into the stencil buffer
$R_R$ R $S_V$
Draw virtual shadows onto real objects
$R_{RV}$ R $S_V$
STEP 3
Draw virtual objects into the color buffer and Z-buffer.
$R_R$ V  R V  $S_V$
Draw the shadow volumes into the stencil buffer
$R_{RV}$ V  R V  $S_{RV}$
Draw virtual shadows onto the virtual objects
$R_{RV}$ $V_{RV}$  R V  $S_{RV}$
Legend:
R Real object
V Virtual object
$R_R$ Real object with real shadow
$R_{RV}$ Real object with real and virtual shadow
$R_{RV}$ Virtual object with real and virtual shadow
$S_V$ Mask of real shadow
$S_{RV}$ Mask of real and virtual shadow

**Figure 7: Overview of the modified shadow volume algorithm for an Augmented Reality application.**

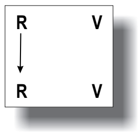


**Figure 8: First, the shadows that real objects cast on real objects is drawn.**

In the first step we have to realize the shadow cast onto real objects, see figure 8. There is nothing special in this case. Our AR system works with a normal webcam. Therefore, we can easily copy the picture of the real world that we grab from the webcam into the color buffer. Figure 7 shows the modified algorithm including the different buffers (color buffer, Z-buffer, and stencil buffer) and their actual content depicted on the right side. As a result of the first step, the grabbed webcam content is drawn into the color buffer. As a result of the first step, all real objects are in the color buffer (including their shadows on the other real objects; cf. figure 9).

In the following step we augment the shadow cast by vir-

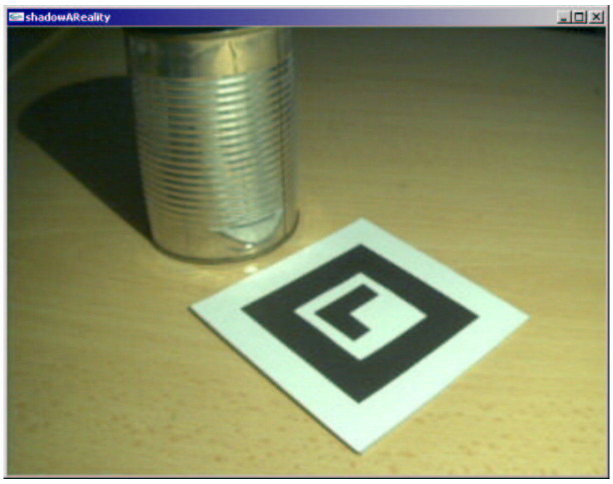


**Figure 9: The real content is shadowed by real objects.**

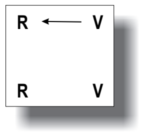


**Figure 10: The virtual shadow is casted on the real objects.**

tual objects onto real objects (cf. figure 10). This process consists of three sub-steps: First, the phantoms (virtual 3D models of the real objects) are drawn into the Z-buffer. There are two reasons for this step. Firstly, it permits the execution of a depth-test of the real objects and the virtual objects (a virtual object should appear in front of a real object in the case that the real object is in the background and behind the virtual object). Secondly, we need the depth-information for the shadow volume algorithm. Next, the shadow volumes of the virtual objects (cf. $S_V$ of step 2 in figure 7) are drawn into the stencil buffer. We do not have to do the same for the shadow volume of the phantom objects that represents the real objects. The reason for this is that all real objects already cast their shadows onto real objects (cf. step 1 of figure 2). Finally, we draw the virtual shadow onto the real objects. For this reason, we render the shadow that is in the stencil buffer. As a consequence of the missing material information from the real world, we use black and transparent shadow polygons that are blended over the parts of the the real scene (real image) that correspond to a virtual shadow. This concludes the procedure that generates shadows that are cast by virtual objects onto real objects.

Finally, we have to implement the shadow cast from real/virtual objects onto virtual objects (cf. figure 12). Similarly to the standard shadow volume algorithm, we distinguish three sub-steps. In this step we have to draw the entire scene (phantoms and virtual objects) into the color buffer and the Z-buffer. In contrast to the traditional shadow volume algorithm, for which we used only ambient and emission

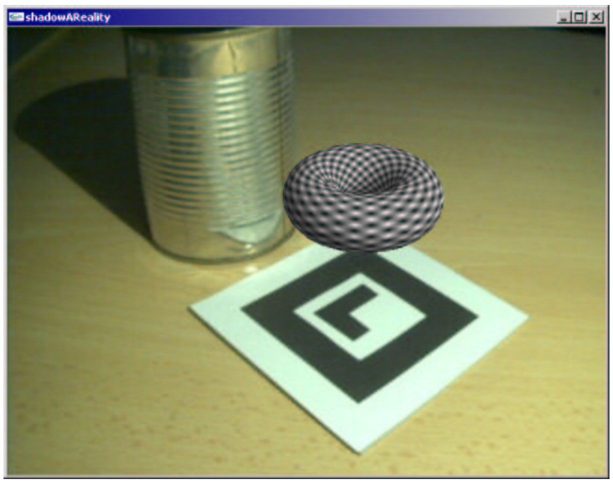


**Figure 11: In this step both, the real and virtual objects are rendered.**

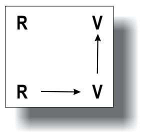


**Figure 12: Next, the real/virtual shadows are casted onto the virtual objects.**

components, we use diffuse and specular components. The reason for that is the first step of the algorithm, in which we already render the real scene. Next, we have to generate the shadow volumes for both virtual and real objects and draw the shadow volume polygons into the stencil buffer (cf. $S_{RV}$ of step 2 in figure 7 and figure 13). Indeed, the shadow volumes of step 2 (cf. $S_V$) that are in the stencil buffer have to be overwritten, because we also need the shadow volumes of the real objects. Again, the stencil value is increased wherever a front-facing polygon is drawn and it is decreased wherever a back-facing polygon is drawn. Finally, the whole scene is rendered wherever the stencil value is 0. In contrast to section 4, the whole scene is rendered with only ambient and emission components (cf. figure 14).

## 4. EXPERIMENTS

The results of *shadowAReality* were quite convincing and impressive. Figure 16 shows one of the four possibilities, in which a virtual torus casts its shadow onto a real house. Similarly, figure 15 depicts a real house casting its shadow onto a virtual torus.

In both scenarios, the real house exists as a phantom model. In fact, the real scene has to be implemented as a 3D model. However, it is sufficient if the 3D model (we used 3ds loader based on [3]) corresponds approximately to the real object. The position and orientation tracking of the real object is accomplished by using ARToolKit as marker detection system. The configuration of an AR scene (including the marker definition and the assignment of the corresponding markers) is

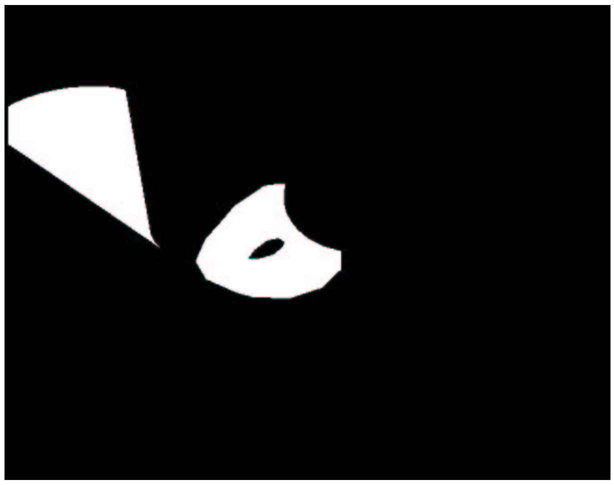


**Figure 13: The shadows of both the real and virtual objects are drawn into the stencil buffer.**

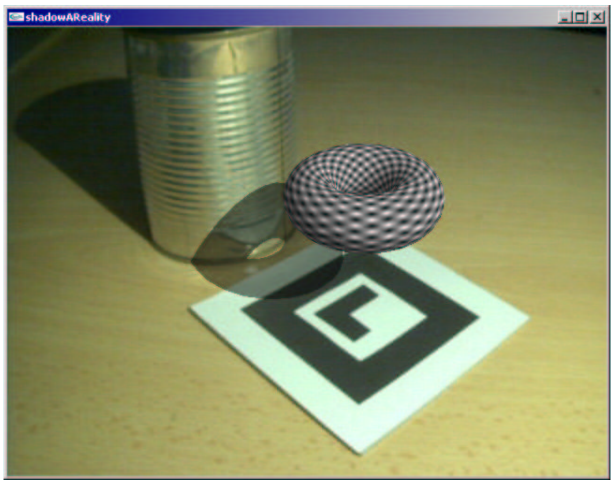


**Figure 14: Finally, the entire scene is drawn based on the content of the stencil buffer.**

defined in a simple configuration file. Both the phantoms and the virtual objects have been modelled in advance by using 3DStudioMax. Next, the models are placed into the scene by using the configuration file. Similarly, the light has to be placed into the scene. Therefore, the authoring of a new scene can be done quickly, but at the moment, this configuration has to be done manually. Currently, the virtual light position and its properties are independent of the real light conditions. Moreover, the current prototype is limited to one light source. But an extension to more light sources is planned for the next version.

As mentioned before, the real object and the corresponding phantom do not have to match exactly one to one. In addition, an approximation of the 3D model (a coarse model of the real object with less polygons) is enough for the shadow volume algorithm (see figure 18).

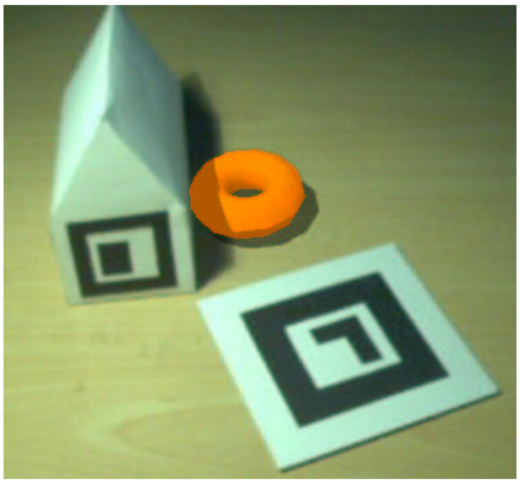


**Figure 15: The real house casts its shadow onto the virtual torus.**

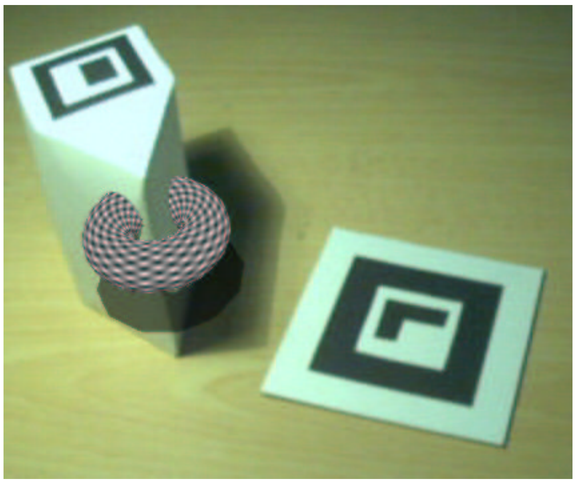


**Figure 16: The virtual torus casts its shadow onto the real house.**

We have not yet included any animated objects. Animated objects would be very simple as virtual objects, but could problematic as real objects (more precisely for their phantom objects). In the case of a real object, we have to observe the position/orientation of each frame of each motion. Consequently, we have to put a marker on each movable object to be able to recognize the movements (cf. figure 19). Finally, the corresponding phantoms have to be moved accordingly. Indeed, the shadow volume has to be recalculated whenever the objects change their silhouettes (in relation to the light source).

In our prototype, the shadow of the real/virtual objects casted onto the real/virtual objects was achieved by shading the scene with black-transparent color blending. In the beginning we worried about the problems with the real shadows onto the real objects. Much to our surprise we found that this did not become a problem. The 'virtual' shadows (casted by virtual/real objects) onto virtual/real objects did not interfere with the real environment. In most situations, users do not even recognize the differences between real and virtual generated shadows (cf. figure 17).

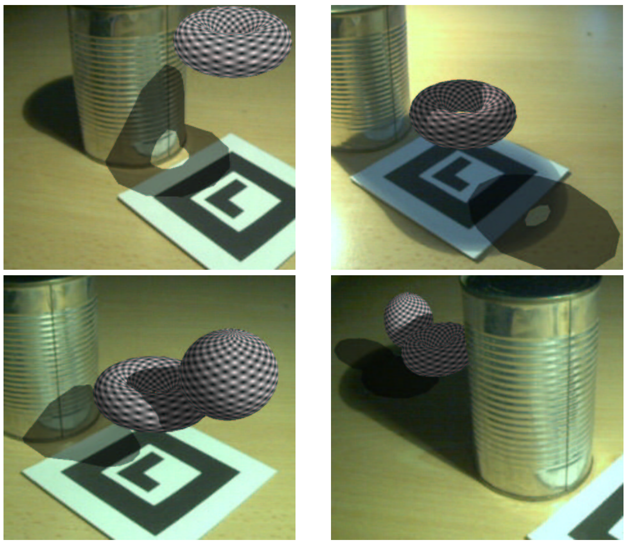


**Figure 17: The first figure shows a virtual shadow cast onto a real tin. The second figure depicts the shadow of the tin onto the torus. Next, we have two virtual objects (sphere and torus) casting a shadow onto the tin and finally, the tin casts a shadow onto the torus and onto the sphere depicted in the last snapshot.**

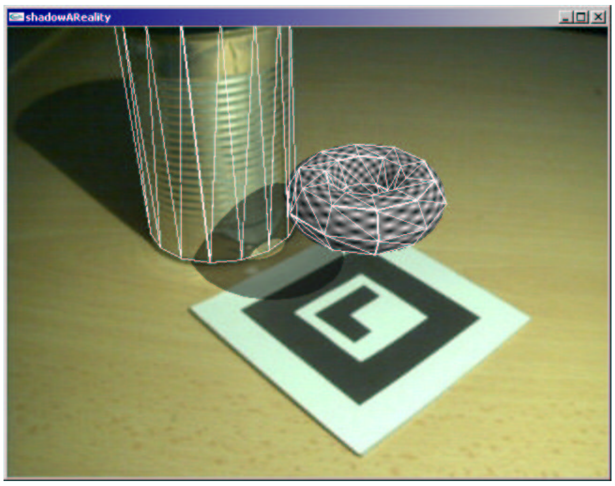


**Figure 18: Phantoms can be rendered with a lower resolution.**

Finally, we performed some tests with real objects for which no phantoms were available. For example, we tested moving the user's hand into the scene depicted in figure 20. The phenomenon was that even if we did not have any phantom, it happens very often, that the user often interferes with the shadow volumes of the scene. As a result, the shadow is casted onto the user's hand.

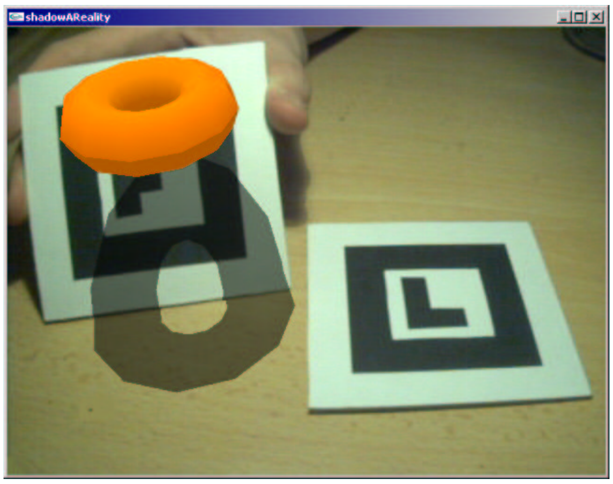


**Figure 19: The user can move the marker and the shadow of the virtual torus is casted correctly onto the real object (in this case the marker).**

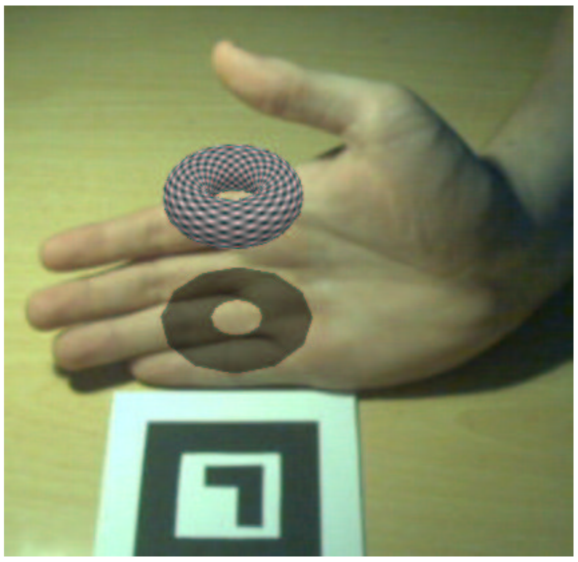


**Figure 20: Even if the model does not exist (i.e. the hand), the results are often impressive.**

## 5. CONCLUSIONS AND FUTURE WORK

In this paper we proposed the concepts for real-time shadows for Augmented Reality applications using shadow volumes. These concepts have been realized in the proof-of-concept prototype, *shadowAReality*, with very convincing results. The augmented shadows enhance the real scene a great deal and they offer the user a more intuitive and very photorealistic world. Similarly, the 3D interaction and manipulation possibilities of augmented scenes are improved by using shadows. Users can more easily measure the distance of two objects and the interaction/manipulation does not seem to be a big problem. In other words, the presented algorithm provides new opportunities for Augmented Reality applications and it allows new means of expression in an AR environment.

Our approach was implemented on the ARToolKit framework and should become available under LGPL. As described in [4] the shadow volume algorithm can be improved by using Portals, BSP, occlusion, and view frustum culling techniques that avoid the rendering of unnecessary shadow volumes. Moreover, we want to improve the shadow volume algorithm by using nVIDIA's shading language Cg [5] [10]. In addition, the proof-of-concept should be integrated into various AR applications to improve the immersive feeling of the users and finally more effort should be invested into the improvement of the shadows (e.g. soft shadows, etc.)

## 6. ACKNOWLEDGEMENTS

Special thanks to Jeremiah Diephuis for re-reading the last version of the paper and for his suggestions about possible improvements.

## 7. ADDITIONAL AUTHORS

Additional authors: Juergen Zauner, Upper Austria University of Applied Sciences, email: `jzauner@fh-hagenberg.at`).

## 8. REFERENCES

[1] F. Crow. Shadow Algorithms for Computer Graphics. In *Proceedings of SIGGRAPH 77*, pages 242–248, 1977.

[2] P. E. Debevec. Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography. pages 189–198, July 1998.

[3] DigiBen. GameTutorials, URL: http://www.gametutorials.com/TermsOfUse.htm, 2001.

[4] C. Everitt and M. J. Kilgard. Practical and Robust Stenciled Shadow Volumes Hardware-Accelerated Rendering. nVIDIA Corporation, March 2002.

[5] R. Fernando and M. J. Kilgard. *Cg Tutorial, The: The Definitive Guide to Programmable Real-Time Graphics*. Addison-Wesley, 2003.

[6] T. Heidmann. Real shadows, real time. *Iris Universe*, (18):23–31, 1991.

[7] M. Kanbara and N. Yokoya. Geometric and Photometric Registration for Real-time Augmented Reality. In *International Symposium on Mixed and Augmented Reality (ISMAR'02)*, 2002.

[8] A. Katayama, Y. Sakagawa, and H. Tamura. A method of shading and shadowing in image-based rendering. In *Int. Conference on Image Process*, volume 3, pages 26–30. IEEE, 1998.

[9] H. Kato, M. Billinghurst, B. Blanding, and R. May. ARToolKit. 1999.

[10] D. Kirk. *CG Toolkit, User's Manual*. nVIDIA Corporation, Santa Clara, CA, 2002.

[11] C. Loscos, G. Drettakis, and L. Robert. Interactive modification of real and virtual lights for augmented reality. In *Siggraph technical sketch: SIGGRAPH'98 (Orlando, FL)*. ACM SIGGRAPH, New York, Jul 1998.

[12] C. Loscos, M.-C. Frasson, G. Drettakis, B. Walter, X. Granier, and P. Poulin. Interactive virtual relighting and remodeling of real scenes. In D. Lischinski and G. Larson, editors, *Rendering techniques '99 (Proceedings of the 10th Eurographics Workshop on Rendering)*, volume 10, pages 235–246, New York, NY, Jun 1999. Springer-Verlag/Wien.

[13] M. D. McCool. Shadow volume reconstruction from depth maps. *ACM Transactions on Graphics*, 19(1):1–26, 2000.

[14] T. A. Moeller and E. Haines. *Real-Time Rendering*. A K Peters, Natick, Massachusetts, 2002.

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

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

[17] U. Neumann, S. You, Y. Cho, J. Lee, and J. Park. Augmented reality tracking in natural environments, 1999.

[18] K. Nishino, Y. Sato, and K. Ikeuchi. Appearance compression and synthesis based on 3d model for mixed reality. In *Proceedings of IEEE ICCV'99*, volume 1, pages 38 – 45, September 1999.

[19] K. Nishino, Y. Sato, and K. Ikeuchi. Eigen-texture method: appearance compression based on 3d model. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'99)*, volume 1, pages 618 – 624, June 1999.

[20] R. Raskar, G. Welch, K. Low, and D. Bandyopadhyay. Shader Lamps: Animating Real Objects with Image Based Illumination. In *Eurographics Workshop on Rendering*, June 2001.

[21] S. Roettger, A. Irion, and T. Ertl. Shadow Volumes Revisited. In *Proceedings of the 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'02)*, pages 373–379, 2002.

[22] T. Starner, B. Leibe, B. Singletary, and J. Pair. Mind-warping: towards creating a compelling collaborative augmented reality game. In *Proceedings of the 5th international conference on Intelligent user interfaces*, pages 256–259. ACM Press, 2000.

## APPENDIX

## A. IMPLEMENTATION OF THE RENDERING PROCEDURE

We developed our prototype, *shadowAReality*, by using OpenGL and ARToolKit as a marker detection system [9]. To make the algorithm of section 3.2.2 more concrete, the following section provides a pseudo-code like description of the major parts of the algorithm including relevant OpenGL commands.

Clear the color buffer and depth buffer.

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Render the real scene shadowed by the real objects only. Consequently, the content of the camera picture is copied to the color buffer. This is accomplished with two ARToolKit functions.

```
argDrawMode2D();
argDispImage(frameBuffer, 0, 0);
```

Render the real objects into the depth buffer. The objects should not be drawn into the color buffer. Therefore, the color mask sets the bit to `GL_FALSE`.

```
glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE);
renderRealObjects();
```

Next, render the shadow volumes of the virtual objects into the stencil buffer. The stencil value is increased whenever a front facing polygon is drawn. In the same way, the stencil value is decreased when a back facing polygon has been found.

```
glDepthMask(GL_FALSE);
glStencilOp(GL_KEEP, GL_KEEP, GL_INCR);
renderShadowVolumes(VIRTUAL_OBJECT_FLAG);
glCullFace(GL_FRONT);
glStencilOp(GL_KEEP, GL_KEEP, GL_DECR);
renderShadowVolumes(VIRTUAL_OBJECT_FLAG);
glCullFace(GL_BACK);
```

Finally, the last sub-step of step 2 draws all virtual shadows onto the real objects.

```
glStencilFunc(GL_NOTEQUAL, 0, ~0);
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);
renderPlane();
```

Next, we render the shadows of the real/virtual objects onto the virtual objects (cf. standard shadow volume algorithm). First, we reset the stencil buffer and we render the virtual objects (virtual and real objects (phantoms)) into the color buffer and Z-buffer.

```
glStencilFunc(GL_ALWAYS, 0, ~0);
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);
glDepthMask(GL_TRUE);
renderRealVirtualObjects();
```

Then, we render the shadow volumes of the real and virtual objects. Based on the front-facing and back-facing polygons, we increment and decrement the stencil buffer value accordingly.

```
glDepthFunc(GL_LEQUAL);
glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE);
glDepthMask(GL_FALSE);
glStencilFunc(GL_ALWAYS, 0, ~0);
glStencilOp(GL_KEEP, GL_KEEP, GL_INCR);
renderShadowVolumes(REAL_VIRTUAL_OBJECT_FLAG);
glCullFace(GL_FRONT);
glStencilOp(GL_KEEP, GL_KEEP, GL_DECR);
renderShadowVolumes(REAL_VIRTUAL_OBJECT_FLAG);
glCullFace(GL_BACK);
```

Finally, we render the whole scene based on the stencil buffer.

```
glStencilFunc(GL_NOTEQUAL, 0, ~0);
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);
glDepthFunc(GL_EQUAL);
renderRealVirtualObjects();
```