Human Interface Technology Laboratory New Zealand

University of Canterbury Private Bag 4800 Christchurch Telephone (64 3) 364 2349 Fax (64 3) 364 2095 Website www.hitlabnz.org Email info@hitlabnz.org

HITLabNZ

Technical Report **TR-2008-01** April 2008

A Mixed-Reality Rendering Framework for Photorealistic and Non-Photorealistic Rendering

Jung Shin, Raphaël Grasset, Hartmut Seichter, Mark Billinghurst

ABSTRACT

This paper presents various techniques for the registration of real and virtual content and proposes an abstract representation for Augmented Reality (AR) through a pipeline of rendering of components Until now, few solutions wereavailable to create different AR rendering techniques without a high level of effort. In this paper, we propose a unifying framework which is flexible and easy to develop various rendering techniques in AR. Our framework supports combination of Photorealistic or Non-Photorealistic Rendering techniques in AR and Mixed-Reality (MR) applications. Based on a GPU shader implementation, our framework is a simple and extensible solution for rendering in AR. Finally, sample demonstrations of the various possibilities of the framework are described

Keywords: Augmented Reality, Rendering, Non-Photorealistic Rendering.

A Mixed-Reality Rendering Framework for Photorealistic and Non-Photorealistic Rendering

Jung Shin * Raphaël Grasset† Hartmut Seichter ‡ Mark Billinghurst§

HIT lab NZ
University of Canterbury
Private Bag 4800, Christchurch, NZ

ABSTRACT

This paper presents various techniques for the registration of real and virtual content and proposes an abstract representation for Augmented Reality (AR) through a pipeline of rendering of components. Until now, few solutions wereavailable to create different AR rendering techniques without a high level of effort. In this paper, we propose a unifying framework which is flexible and easy to develop various rendering techniques in AR. Our framework supports combination of Photorealistic or Non-Photorealistic Rendering techniques in AR and Mixed-Reality (MR) applications. Based on a GPU shader implementation, our framework is a simple and extensible solution for rendering in AR. Finally, sample demonstrations of the various possibilities of the framework are described.

Index Terms: K.6.1 [Management of Computing and Information Systems]: Project and People Management—Life Cycle; K.7.m [The Computing Profession]: Miscellaneous—Ethics

1 INTRODUCTION

In [22], Haller discusses the motivation for Photorealistic rendering versus Non-Photorealistic rendering in Augmented Reality applications. Photorealism aims to create a visually sound representation of the world, reducing the pictorially difference between real and virtual images. In Augmented Reality, photorealism provides a photometric registration and visual coherence between the real and the virtual content. The Non-Photorealistic rendering approach tends to deliver an abstract representation of the content which emphasizes the information implicitly rather than making it visually plausible. In the context of AR, Non-Photorealistic approaches can be an efficient method to focus the attention of the use on the content; for example displaying a virtual building with a charcoal style on a real table for a design review for architects - emphasizing the early stage of the design. Figure 1 shows examples of Photorealistic and Non-Photorealistic rendering in AR.

However, computer graphics in AR may not be limited to either photorealistic or non-photorealistic rendering. Mediated Reality provides a way to modifythe perception of the real world and combines elements of both photorealistic and non-photorealistic graphics. Recently, Haller et al.[25] and Fischer et al.[12] demonstrated new rendering techniques where various styles are applied to both the 3D virtual content and the background video image in an AR application to reduce the visual discrepancy between them. Their work can be considered as exploring the more general notion of *Mixed-reality rendering techniques* rather than specifically AR.

*e-mail:Jung.Shin@hitlabnz.org
†e-mail:raphael.grasset@hitlabnz.org
‡e-mail:hartmut.seichter@hitlabnz.org
§e-mail:mark.billinghurst@hitlabnz.org

(a) PR rendering: rendering of soft shadow [18] (b) NPR rendering: the loose and sketchy algorithm [24]

Figure 1: Examples of Photorealistic and Non-Photorealistic in AR

This diversity of rendering techniques remains a crucial issue in AR applications. For this purpose, we propose a generic framework which supports broader Mixed-reality rendering techniques. Our work supports not only basic photometric and non-photometric rendering techniques but also the combination and variation between them. Our framework is flexible and highly extensible due to its general structure. It allows users to create and combine various rendering techniques in a simple manner. Therefore users can sample, combine and change through various combinations of techniques for Mixed-Reality environments.

In the rest of the paper, we first present earlier related work, followed by detailed architecture descriptions of our system. Finally, we present applications results and describe the limitations of our approach.

2 RELATED WORK

Recently, various rendering techniques for AR applications have been investigated. In this section, we look at previous research related to our work in terms of Photorealism and Non-Photorealism in AR.

Jacobs et al. [32] give a good overview of illumination techniques for Photorealistic rendering. Their work provides some of the background foundation that our work is based on.

There are many researchers who have focused on real-time techniques for producing Photorealistic rendering, such as shadow rendering [27] [18] [33], realistic appearance of glossy material [6] [42] and real-time light acquisition [6], [42], [10].

A large body of work exists related to Photorealistic rendering including relighting of the virtual scene with a global illumination approach [38] or with a per-pixel lighting solution [10], [31], [30]. Various demonstrations not only have yield solutions for occlusion between static real and virtual objects for but also for dynamic objects. Different solutions have been proposed to reduce the visual conflict between using phantom objects [43] [8] [17] [15] [11] and image-based solutions [7] [37] [35].

Recently, the use of Non-Photorealistic rendering techniques have emerged in the AR domain. Haller et al. [26] introduced a real-time painterly rendering technique for AR application. In [24], a new Non-Photorealistic rendering technique is proposed which applies the rendering style to the composed image of the video background and 3D virtual contents. Fischer et al. [12] introduces a similar method, and coins this concept as *stylized AR*. In [14], they demonstrated the viability of their approach based on a psychophysical study.

More recently, Fischer et al.[13] describe another technique for reducing the visual discrepancy of video-based AR. The technique uses a GPU approach to modify the rendering of the virtual content to be visually coherent with the video image rather than correcting the problems of video acquisition such as noise and motion-blur. The modifications include adding noise to the virtual content.

Many researchers provide rendering engines or frameworks which support various rendering styles. For example, Grabli et al. [20] propose a library based on a script interface for Non-Photorealistic line rendering. The OpenNPAR project [28] presents an effective graphical tool that can be used to design and investigate different rendering styles of virtual contents. More recently, 3D rendering engines such as OGRE[3] offer a script language to support multi-pass rendering and combining different visual effects. However, most of them are not suitable to use in AR context directly. This is because they do not support basic AR functions including easy acquisition of a video stream, tracking of real objects, spatial registration of virtual objects, and none of the them have been designed to support MR rendering techniques implicitly.

Recently, Mendez et al.[39] explores the design of a context-driven framework where selective rendering can be applied to the presence of a local MagicLens. This local control of the representation is similar to our work, although our approach goes beyond this by supporting different rendering styles.

As can be seen from this short review, there is a large body of previous research in photorealistic and non-photorealistic rendering, but there are few frameworks that support a large range of rendering styles for AR applications. In this paper we address this deficiency and provide a unique solution that allows developers to quickly deploy a range of different rendering styles for AR applications. In the next section we describe our flexible and extensible architecture.

3 Architecture Overview

In this section we present the design of our software architecture. The properties of a library or framework can be summarize by the *functional* and *non-functional* services (or requirements). There are extensive lists of non-functional requirements [36] in the literature and different methods to describe them. Some of the most used are performance, extensibility, flexibility, modularity and robustness.

In our context, the functional requirements are defined by the visual rendering capabilities of our system for AR and MR applications. We can describe rendering techniques in terms of the Photorealistic and Non-Photorealistic rendering capabilities. We can also explain methods to combine them and blur further this distinction. In the following sections, we describe the design principles which describes the non-functional services of our library, various rendering techniques and how to combine them using our system.

3.1 Design Principles

3.1.1 AR Scene

Our framework is built on a scene-graph rendering library. We define nodes dedicated to support basic AR functionalities directly inside the scene-graph similar to that previously proposed [23], [9]. In video see-through AR, we consider that a typical scene consisting of a background and a foreground. The background is generally a two dimensional live video stream and the foreground the virtual content of the application. A simple scene-graph can therefore be built as shown in figure 2.

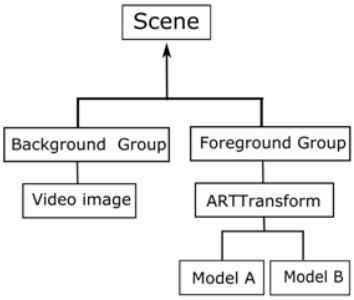


Figure 2: Basic AR Scene-Graph structure

3.1.2 The Layer System

Our rendering pipeline consists of *layers* of images (OpenGL textures in our implementation). A *layer* is a special type of a node which contains a rendered image of the target nodes. Each *layer* defines an *effect* and they can be combined by a simple alpha blending techniques or by using a more complex operator. This operator can be define procedurally by using a fragment shader. The final result is combined the different layers and other nodes. However, not all the effects have to be rendered into layers. It is only necessary when the rendered effect is used as an input into other process in later phases.

This generic approach is flexible and modular, and a user can add, remove or change the effects. For example, shadows in a scene can be rendered using a 'Shadow volume' technique onto a layer. The user can easily change the shadow technique later with a 'Shadow mapping' technique since the shadow effect is just a layer and can be replaced (see figure 3). The concept of layers also helpes to unify Photorealistic and Non-Photorealistic rendering in a more a flexible manner, and to support their integration.

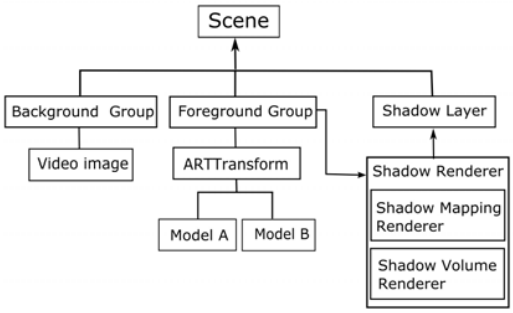


Figure 3: Flexible scene-graph structure

The following is the pseudo code of figure 3. This scene-graph approach with layers allows an easier development for users.

```
ARScene arScene;
arScene.InitialiseVideoBackground();
arScene.InitialiseForeground();

arScene.AddModel( Model A );
arScene.AddModel( Model B );
```

```
ShadowLayer shadowLayer;
ShadowVolumeRenderer shadowVolueRenderer;

shadowVolumeRenderer.AddReferenceGeometry(
            arScene.GetForegroundGroup());

shadowLayer.Connect( shadowVolumeRenderer );

arScene.AddModel( shadowLayer );
```

The system does not limit the number of layers that can be chained and stacked together. However, in a realistic AR application, real-time performance is a constraint and users need to limit the number of effects depending on their hardware capabilities.

Each effect can be classified as a geometry- or image-based algorithm. Even though a layer can be overlaid on rendering or 3D content, it is necessary to render the geometry to a layer if it is been used for other image-based techniques. Therefore geometry-based techniques need to be performed before image-based techniques. In other words, geometry-based techniques cannot be performed after they have been rendered into a layer. A similar approach can be found in OpenNPAR [28].

We optimized our system to support GPU shaders for various effects.Each layer can be associated with a shader defined externally tothe scene-graph and each of these effects can be simply configuredand the parameters associated with the effects can be easilymanipulated through shaders. However, our system still permits thedeveloper to implement CPU-based effects, limited by real-timeperformance considerations.

Users can modify the shader at run-time and the result will beautomatically reflected in the rendering. This functionality is intuitive during the prototyping and the development phase.Users can test the different parameters of the shader easily through trial-and-error.Therefore it supports a short development cycle due to the capability to runfunctions without losing time for recompiling and re-runningthe application. More detail is explained in section4.

The framework is extensible since the layer structure is highly generic. Users can create an effect layer easily using shaders, by combining different effects or applying any custom rendering using CPU based techniques. This is possible because of our layer-based approach.

3.1.3 Illustrated Examples

Figure 5 shows examples of scene-graphs built with our architecture. Figure 4 shows the corresponding result images.

In this example, a Gaussian blur and Sobel filter are used to enhance an AR scene. Even though two scene-graphs have the same components such as a 3D truck model, a video background, Sobel and Gaussian filters, the rendering result can be varied depending on how the user structures the scene with nodes. In ‘Scene A’, a Gaussian blur is applied to the video image before they are merged for the edge detection. In ‘Scene B’, the Gaussian Blur is applied after the background and the foreground are merged. As the result, the truck in ‘Scene A’ has more detailed silhouette highlights while the truck in ‘Scene B’ has only very coarse silhouette highlights [1].

3.1.4 Global and Local Control

The layer system provides a flexible structure to add, remove orchange different effects. However, it is also crucial that usershave full control of the rendering effects regarding where and howthe effects are applied. For example, a user wants to apply ablack and white effect to a specific region on the screen with therest of image

[1]The Gaussian blur is applied because applying a Sobel filter directly on a image can result some unstable, flaky silhouette especially on live video images due to noise from the camera sensor.

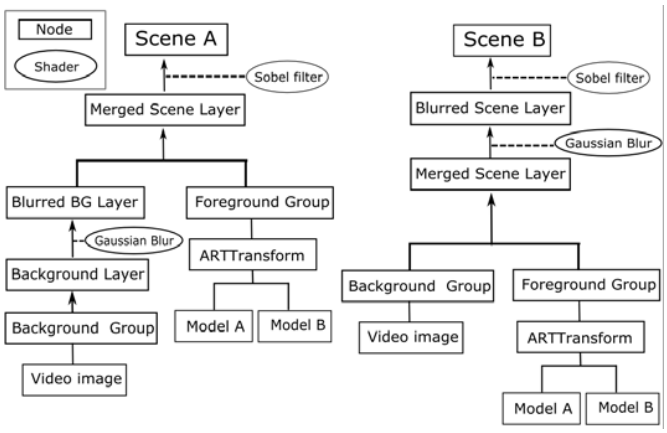


Figure 5: Scene-graph structure of Figure 4

rendered with charcoal style. Global and localcontrol of effects is of essence in this case.

In our framework, control can be applied in geometry-space and/or in image-space. Image space control only applies to regions on the result of the layer image whereas geometry space control is related to visual geometry in the image. For geometry-based effects, these effects can be applied individually to each virtual object. Several virtual objects can be grouped in the scene-graph to apply a certain effect. Therefore the users have full control over individual content in the geometry-space.

Image-based effects behave differently with the need for masks which specify the region on the screen to apply the image-based effects. This mask is also a layer which is used for combining other layers. This mask layer can be generated by rendering the virtual contents into a layer or users can supply user defined images and have full control of effects in the image-space as well. Therefore our system provides global and local control of effects in both geometry-based and image-based techniques. Furthermore, control is dynamic as masks can be generated from underlying effects.

In the next section, we describe the supported effects in our system and the functional services of our toolkit.

3.2 Photorealistic Rendering

Figure 6: Shadows of virtual and real objects: a) shadow from the phantom objects, b) real shadow of the can, c) shadow of the truck

(a) Scene A (b) Scene B

Figure 4: Combination of two image-based techniques: Gaussian blur and Sobel filter

Our system supports shadow casting and correct occlusions between virtual and real objects as these are essential elements of an AR scene making the rendering result more convincing [34, 22].

In our framework, there are two types of 3D contents; virtual and phantom objects. Phantom objects are used to provide correct occlusion of virtual content by real world objects. The virtual object is created and treated as a normal 3D object in the AR scene. For the phantom objects, however, they are separately stored in a 'phantom group'. Our system renders the depth of phantom objects at the beginning of the rendering state before any other virtual objects are rendered. This process allows culling of virtual objects which are occluded by phantom objects, creating the illusion that the virtual objects are occluded by the real objects. Phantom objects can also be used for rendering shadows.

Figure 7: Simple glass refraction in an AR Scene

In [40], Naemura et al. categorise the different types of virtual shadows in AR depending on what object casts shadow and what object receives the shadow (from Real/Virtual to Real/Virtual). Our framework can support these different combinations by using the phantom object.

Figure 6 shows shadows off virtual (a truck) and real (a can) objects using a shadow mapping technique. The can casts its virtual shadow on the desk and on the truck. Note that the truck model is lifted slightly to show the shadow of the truck clearly and the virtual light position is set differently from the real light position on purpose, to show both the real and generated shadows. In the figure, there are one virtual object (truck) and two phantom objects for the can and the desk.

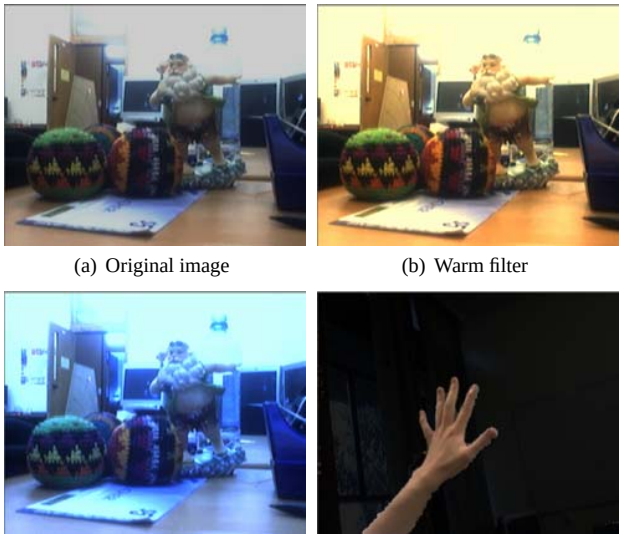
In Photorealistic AR rendering, lighting is another crucial element. Even though we did not implement any image-based lighting techniques, they can be easily integrated into our framework because of the flexible rendering pipeline and the scene-graph based approach that we use.

Another active area of research in Photorealistic rendering is rendering special material such as translucent, partially-reflective or transmissive materials [29, 16, 44]. Figure 7 shows an example of simple glass refraction in our framework.

3.3 Non-Photorealistic Rendering

Recently, Non-Photorealistic rendering in AR environments has become an active research area [22, 25, 12]. Due to the characteristic of video see-through AR, there are both 2D and 3D elements in an AR scene. The video background is one of the most important 2D elements in the scene and the AR experience can be greatly enhanced by controlling the video background.

For example, figure 8(a), 8(b) and 8(c) show a colour temperature technique to change or exaggerates the colour of the light in the scene. These effects can change the perception of the users and increase the immersiveness of the AR application. Image-based techniques are not only useful to enhance the AR scene visually but also useful for other functions. Figure 8(d), shows a simple skin colour detection technique on the video image. The segmented skin colour mask can be used for both visual enhancements and interaction functions. While 2D components can be enhanced by image-based techniques, 3D content can be improved by using geometry-base techniques. Figure 9 shows examples of geometry-based techniques to change the appearance of the input geometry such as lighting, colour and textures. NPR algorithms using image- and geometry-based techniques can be integrated independently to the video background and the virtual objects. Figure 10 shows a simple example of applying both image-based and geometry based techniques independently.



(a) Original image (b) Warm filter

(c) Cold filter (d) Skin color detection

Figure 8: Examples of Image-based techniques

(a) Cartoon shading (b) 3D hatching

Figure 9: Geometry-based techniques

3.4 Combination

As stated in section 3.1.2, all the effects are rendered into layers of images. Therefore it is not only easy to add new effects but also intuitive to combine all the available effects to create new results. Figure 11 shows a simple combination of PR and NPR techniques. First, the shadow layer is generated using the truck model and a phantom geometry made for the desk. Then the shadow layer is used as a mask layer to a sketch image layer generating a sketchy style shadow layer. Then the truck is rendered with Gooch shading [19] on the video background. Finally, the sketchy style shadow layer is overlaid on the rest of the layers.

4 Implementation

Our framework is based on the OSGART [5] library which is built on OpenSceneGraph rendering engine [4]. OpenSceneGraph inherently supports GLSL shaders at the scene-graph level, which supports easy integration. The system is developed in C++.

In our system, users load shaders with our shader manager which loads GLSL shaders from files and once they are loaded, they are stored by the shader manager. The shader manager checks whether the shader file is modified. If the shader file is modified, the system reloads the shader automatically. This allows developers to modify shaders and see the result immediately while the application is running. For example, users can change the light colour or change the threshold values for an edge detection algorithm from the shader directly and see the result in real time. It allows users to debug and tune the shaders easily and saves the time of having to recompile and run the application again.

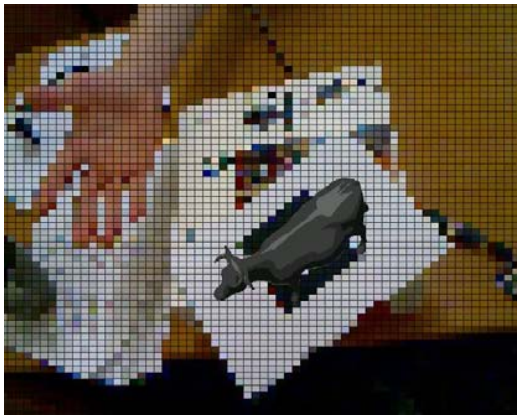


Figure 10: Geometry-based technique: cartoon shading, Image-based technique: tile Effect

Figure 11: Combination of gooch shading and shadow

5 Results

Our system supports a large range of effects and figure 12 shows some of them implemented effects using GLSL shaders.

Figure 12(a), 12(b) and 12(c) are examples of geometry-based techniques to modify the lighting, colour and the texture of the virtual models. Figure 12(a) shows gooch shading which is a famous Non-Photorealistic lighting model technical illustration method [19]. Figure 12(b) is a simple example of generating a brick-like texture using the vertex and fragment shaders. Figure 12(c) is real-time 3D hatching using multitexturing [41].

There are many image-based techniques implemented such as figure 12(d), 12(e) which are simple colour modifications of the image. Figure 12(g), 12(h) shows simple radial blur effects using multitexturing with a simple blurring on the background image. Figure 12(f) shows a more advanced example combining Gaussian blurring with a Sobel filter. Figure 12(j),12(k) and 12(l) shows colour

temperature effects with white balancing. Animations between different colour temperatures is implemented for smooth transitions between them. Figure 12(m), 12(n) shows a simple example of a high dynamic rendering effect which is a popular image-based technique for enhancing the scene.

5.1 Applications

Our framework is used for the development of various research prototypes at our lab. The system has been recently used in the context of the Mixed-Reality book project [21] where different image-based and model-based effects have been implemented. We have used several filtering effect including the colour temperature effect. These effects have been used to improve the immersion of the user inside the book (see figure 13). The Mixed-Reality book application was presented for the Interactivity Session at the CHI2007 conference.

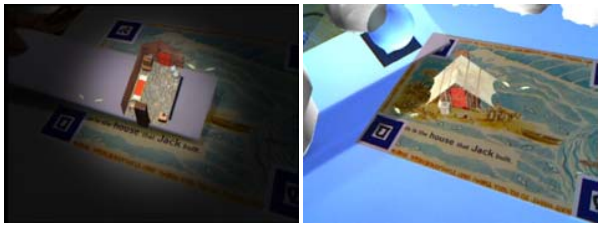


Figure 13: Effects of the Framework for the Mixed-Reality Book project

The framework has also been used for the prototyping of an artistic application - Semaphore. Semaphore is an augmented reality installation which visualises the information transport concept in the optical Telegraph system used during the 19th century.

The artist used Augmented Reality to provide a visual representation of the information transported between the semaphore tower. In this context, the artist was interested to explore how he could modify the appearance of the video background to create meaningful and aesthetics effects such as a grey level, luminance variation, and a combination of chrominance modifications (see figure 14(a)). The possibility to change the effect at run-time and see the result instantly helped the artist in this project. This installation was exhibited in COCA - Centre of Contemporary Art [1] (see figure 14(b)).

The Semaphore installation will be exhibited at the 17th International Conference of Artificial Reality and Telexistence (ICAT 2007) that is being run in conjunction with ArtAbilitation Conference in Denmark [2].

(a) In prototyping stage (b) In COCA exhibition

Figure 14: Artistic application

6 Conclusion

In this paper, we have introduced a Mixed-Reality Rendering Framework which is flexible and highly extendable because of the needs of modular rendering in an Mixed-Reality environment. We have identified the components in a typical AR scenario and demonstrated our rendering system with various examples in different configurations. Several rendering styles have been discussed including components of Photorealistic and Non-Photorealisctic rendering and a combination of both. Finally, we showed results of some of our implemented effects.

One of the limitations of the Layered rendering approach is that geometry-based algorithms have to be performed before image-based algorithms. Therefore, when a geometry-based algorithm requires an image-based algorithm output as its input, the same geometry has to be rendered more than once. However, recent graphics cards are powerful enough to render the same 3D models multiple times.

Another limitation of our system is that we utilise GPU shaders heavily which requires a high-end graphics card with vertex and fragment shading units. However, most of the recent graphics cards have both vertex and fragment shading functionality.

Currently, we are extending our framework with new functionalities. We are planning to include various techniques such as real light acquisition (e.g. using light probe analysis) and more advanced phantom techniques (e.g. an occlusion refinement technique presented in [35]).

Even though our system is powerful and flexible, it is a framework designed for developers. Therefore, it is not easy for designers without programming knowledge to create an AR application. Currently, we are developing an intuitive authoring tool with a visual programming interface that solves this issue. This tool will not only allow users to add virtual 3D content into an AR scene easily but also configure and create various rendering effects.

References

[1] Coca. http://www.coca.org.nz/.

[2] Icat2007. http://www.icat2007.org/.

[3] Ogre rendering engine. www.ogre3d.org.

[4] Openscenegraph. www.openscenegraph.org.

[5] Osgart. http://www.artoolworks.com/community/osgart.

[6] K. Agusanto, L. Li, Z. Chuangui, and N. W. Sing. Photorealistic rendering for augmented reality using environment illumination. In *ISMAR '03: Proceedings of the The 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality*, page 208, Washington, DC, USA, 2003. IEEE Computer Society.

[7] M. Berger. Resolving occlusions in augmented reality: a contour-based approach without 3d reconstruction. In *CVPR (Ieee Conference on Computer Vision and Pattern Recognition (CVPR 1997)*, 1997.

[8] D. Breen, R. Whitaker, E. Rose, and M. Tuceryan. Interactive occlusion and automatic object placement for augmented reality. In *Eurographics '96*, 1996.

[9] E. M. Coelho, B. MacIntyre, and S. Julier. Osgar: A scene graph with uncertain transformations. In *International Symposium on Mixed and Augmented Reality (ISMAR04)*, 2004.

[10] S. DiVerdi and T. Hllerer. Combining dynamic physical and virtual illumination in augmented reality. Technical report, University of California at Santa Barbara, 2004.

[11] J. Fischer, D. Bartz, and W. Straßer. Occlusion handling for medical augmented reality using a volumetric phantom model. In *VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 174–177, New York, NY, USA, 2004. ACM Press.

[12] J. Fischer, D. Bartz, and W. Straer. Artistic reality: fast brush stroke stylization for augmented reality. In *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 155–158, New York, NY, USA, 2005. ACM Press.

[13] J. Fischer, D. Bartz, and W. Straer. Enhanced visual realism by incorporating camera image effects. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2006.

[14] J. Fischer, D. Cunningham, D. Bartz, C. Wallraven, H. Blthoff, and W. Straer. Measuring the discernability of virtual objects in conventional and stylized augmented reality. In *Eurographics Symposium on Virtual Environments (EGVE)*, 2006.

[15] J. Fischer, H. Regenbrecht, and G. Baratoff. Detecting dynamic occlusion in front of static backgrounds for ar scenes. In *EGVE '03: Proceedings of the workshop on Virtual environments 2003*, pages 153–161, New York, NY, USA, 2003. ACM Press.

[16] R. W. Fleming, H. W. Jensen, and H. H. Bülthoff. Perceiving translucent materials. In *APGV '04: Proceedings of the 1st Symposium on Applied perception in graphics and visualization*, pages 127–134, New York, NY, USA, 2004. ACM Press.

[17] A. Fuhrmann, G. Hesina, F. Faure, and M. Gervautz. Occlusion in collaborative augmented environments. *Computers and Graphics*, 23(6):809–819, 1999.

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

[19] A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 447–452, New York, NY, USA, 1998. ACM Press.

[20] S. Grabli, E. Turquin, F. Durand, and F. Sillion. Programmable style for npr line drawing. In *Rendering Techniques 2004 (Eurographics Symposium on Rendering)*. ACM Press, june 2004.

[21] R. Grasset, A. Dnser, H. Seichter, and M. Billinghurst. The mixed reality book: A new multimedia reading experience. In *CHI'2007 Interactivity*, 2007.

[22] M. Haller. Photorealism or/and non-photorealism in augmented reality. In *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 189–196, New York, NY, USA, 2004. ACM Press.

[23] M. Haller, W. Hartmann, T. Luckeneder, and J. Zauner. Combining artoolkit with scene graph libraries. In *IEEE International Augmented Reality Toolkit Workshop*, 2002.

[24] M. Haller, F. Landerl, and M. Billinghurst. A loose and sketchy approach in a mediated reality environment. In *Graphite 2005*, 2005.

[25] M. Haller, F. Landerl, and M. Billinghurst. More sketchy, more ar, more fun! In *SIGGRAPH '05: ACM SIGGRAPH 2005 Posters*, page 34, New York, NY, USA, 2005. ACM Press.

[26] M. Haller and D. Sperl. Real-time painterly rendering for mr applications. In *Graphite 2004*, 2004.

[27] H. W. Z. J. Haller M., Drab S. A real-time shadow approach for an augmented reality application using shadow volumes. In *ACM Symposium on Virtual Reality Software and Technology (VRST 2003)*, pages 56–65, 2003.

[28] N. Halper, T. Isenberg, F. Ritter, B. Freudenberg, O. Meruvia, S. Schlechtweg, and T. Strothotte. Openpar: A system for developing, programming, and designing non-photorealistic animation and rendering. In *Pacific Graphics 2003*, 2003.

[29] X. Hao and A. Varshney. Real-time rendering of translucent meshes. *ACM Trans. Graph.*, 23(2):120–142, 2004.

[30] S. Heymann, A. Smolic, K. Mller, and B. Frhlich. Illumination reconstruction from real-time video for interactive augmented reality. In *WIAMIS 2005*, 2005.

[31] C. E. Hughes, E. Reinhard, and S. N. Pattanaik. Achieving interactive-time realistic illumination in mixed reality. In *ASC2004, Army Science Conference*, 2004.

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

[33] K. Jacobs, J.-D. Nahmias, C. Angus, A. Reche, C. Loscos, and A. Steed. Automatic generation of consistent shadows for augmented reality. In *GI '05: Proceedings of the 2005 conference on Graphics interface*, pages 113–120, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[34] Y. Jung, T. Franke, P. Dähne, and J. Behr. Enhancing x3d for advanced mr appliances. In *Web3D '07: Proceedings of the twelfth international conference on 3D web technology*, pages 27–36, New York, NY, USA, 2007. ACM Press.

[35] G. Klein and T. Drummond. Sensor fusion and occlusion refinement for tablet-based ar. In *ISMAR 2004*, 2004.

[36] D. Kulak and E. Guiney. *Use Cases: Requirements in Context*. Addison Wesley, 2003.

[37] V. Lepetit and M.-O. Berger. Handling occlusions in augmented reality systems: A semi-automatic method. In *IEEE and ACM International Symposium on Augmented Reality*, 2000.

[38] C. Loscos. *Ré-éclairage et remodélisation interactifs des scènes réelles pour la Réalite Augmentée*. PhD thesis, Université Joseph Fourier (Grenoble), Dec 1999.

[39] E. Mendez, D. K. Denis, and D. Schmalstieg. Interactive context-driven visualisation tools for augmented reality. In *International Symposium on Mixed and Augmented Reality 2006*, Reality 2006.

[40] 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), 2002.

[41] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, page 581, New York, NY, USA, 2001. ACM Press.

[42] P. Supan, I. Stuppacher, and M. Haller. Image based shadowing in real-time augmented reality. *International Journal of Virtual Realit*, 5(3):1–7, 2006.

[43] M. M. Wloka and B. G. Anderson. Resolving occlusion in augmented reality. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 5–12, New York, NY, USA, 1995. ACM Press.

[44] D. E. Zongker, D. M. Werner, B. Curless, and D. H. Salesin. Environment matting and compositing. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 205–214, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

(a) Gooch Shading

(b) Brick Shading (Generated texture)

(c) 3D Hatching

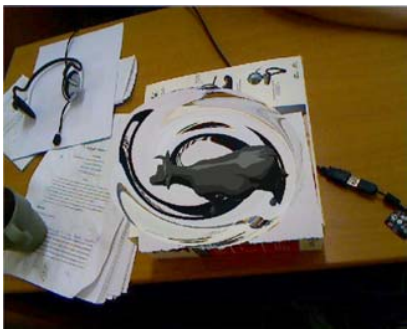
(d) Black and White Threshold

(e) Sepia

(f) Gaussian Blur with Soebl Filter

(g) Radial Blur 1

(h) Radial Blur 2



(i) Swirl Effect

(j) Original

(k) Warm Filter

(l) Cold Filter

(m) High Dynamic Range: Short Exposure

(n) High Dynamic Range: Long Exposure

(o) Skin color detection

Figure 12: Examples