Please see the color plate on page 325.

Stylized Augmented Reality for Improved Immersion

Jan Fischer *
Visual Computing for Medicine,
WSI/GRIS
University of Tübingen

Dirk Bartz
Visual Computing for Medicine,
WSI/GRIS
University of Tübingen

Wolfgang Straßer
WSI/GRIS
University of Tübingen

(a) Conventional augmented reality

(b) Using "cartoon-like" stylized AR

Figure 1: Comparison of standard augmented reality and stylized AR using our image filter and non-photorealistic rendering. In both images, the teapot is a virtual graphical object, while cup and hand are real.

ABSTRACT

The ultimate goal of augmented reality is to provide the user with a view of the surroundings enriched by virtual objects. Practically all augmented reality systems rely on standard real-time rendering methods for generating the images of virtual scene elements. Although such conventional computer graphics algorithms are fast, they often fail to produce sufficiently realistic renderings. The use of simple lighting and shading methods, as well as the lack of knowledge about actual lighting conditions in the real surroundings, cause virtual objects to appear artificial.

In this paper, we propose an entirely novel approach for generating augmented reality images in video see-through systems. Our method is based on the idea of applying stylization techniques for reducing the visual realism of both the camera image and the virtual graphical objects. A special painterly image filter is applied to the camera video stream. The virtual scene elements are generated using a non-photorealistic rendering method. Since both the camera image and the virtual objects are stylized in a corresponding "cartoon-like" or "sketch-like" way, they appear very similar. As a result, the graphical objects seem to be an actual part of the real surroundings.

We describe both the new painterly filter for the camera image and the non-photorealistic rendering method for virtual scene elements, which has been adapted for this purpose. Both are fast enough for generating augmented reality images in real-time and are highly customizable. The results obtained using our method are very promising and show that it improves immersion in augmented reality.

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: augmented reality, immersion, painterly filtering, non-photorealistic rendering

1 INTRODUCTION

In augmented reality (AR), virtual graphical objects are overlaid over the actual surroundings of the user. One important aspect of useful reality augmentation is the correct spatial alignment of virtual scene elements. In order to achieve the correct three-dimensional orientation and positioning of graphical objects, the user's head or the video camera used in the system have to be tracked. Two different basic types of augmented reality can be distinguished. In video see-through AR a video stream is acquired continuously from a camera and used as basis for the image mixing process, whereas optical see-through AR is based on special translucent display devices [2].

Video see-through augmented reality systems acquire the digital input video stream and render the current video frame as background image for the augmented view. In this process, a 2-d transformation is usually applied to the image to account for the optical distortion caused by the camera lens. The graphical primitives which constitute virtual objects in the AR scene are then rendered over the background image using standard computer graphics

*e-mail: fischer@gris.uni-tuebingen.de

IEEE Virtual Reality 2005
March 12-16, Bonn, Germany
0-7803-8929-8/05/$20 ©2005 IEEE

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 IEEE

methods. Common real-time graphics libraries like OpenGL [19] or high-level frameworks based on them are often utilized for this task.

In real-time computer graphics, simple lighting and shading models are normally used. The common local illumination methods for the vertices of graphical primitives depend on manually placed virtual light sources and basic material parameters. The primitives are usually rendered using simple flat or Gouraud shading, which spreads the lighting parameters of the vertices over the whole graphical object in a simple manner [8]. The resulting rendered images generally look artificial, especially in contrast with the acquired real video background in an AR view. This is illustrated in Figure 1(a), in which the teapot on the left is a computer-generated rendering. Even if more sophisticated rendering methods with advanced illumination and shading were used, the problem of mismatched scene generation parameters would still persist. Since light sources and material properties are defined manually during definition of the AR scene, they do not correspond to the lighting conditions in the actual surroundings. This becomes particularly apparent when the camera is moved, which often results in varying image brightness and coloring, while the appearance of computer-generated virtual objects remains mostly uniform.

Due to the large discrepancies in visual appearance, even in a still image an observer can easily distinguish virtual objects from the real camera background. In the example in Figure 1(a), it is obvious that the teapot is a virtual object, since it almost seems to be "pasted over" the camera image.

In this paper, we propose a completely novel way of generating augmented reality images. We use stylization algorithms for reducing the visual realism of both the background camera image and the virtual graphical elements. The result of this approach is that both image layers in an AR scene have a much more similar visual appearance. Figure 1(b) shows the teapot scene rendered using our "cartoon-like" stylization method. In such an augmented view, it is significantly more difficult to distinguish between real and virtual objects, resulting in an improved immersion into the AR scene.

Our stylized augmented reality approach inherently removes details from the background camera image. Whereas this might be inappropriate for some applications like medical scenarios, many applications can benefit greatly from the improved immersion offered by our method. In particular in fields like entertainment, education and training, the blurred barrier between real and virtual can provide a much more impressive augmented reality experience.

2 Related Work

The opposite of our approach of reducing visual realism in augmented reality is to improve the realism of virtual objects in order to achieve a better visual correlation to the camera image. Research has been done into methods of analyzing the real illumination conditions in an AR setup. Kanbara and Yokoya describe an approach of analyzing the distribution of real light sources in real-time, which is used for adapting the representation of graphical objects accordingly [14]. Their method requires a special marker and mirror ball to be visible in the camera image for computing the environment light map. A similar technique for utilizing an acquired environment illumination map is proposed by Agusanto et al. [1]. In their system, a mirror ball and special camera are used in a specific procedure for determining the lighting conditions in the scene beforehand.

A different approach to make virtual objects appear more realistic adds shadowing to the AR image. This creates the impression that shadows are cast from virtual objects onto real surfaces. Haller et al. describe a method for computing such shadows in augmented reality [10, 11]. A similar technique is used for a user study on the effects of shadowing in AR by Sugano et al. [20]. As a drawback of these methods, a model of the geometry of the surfaces and objects in the real world is required. This model needs to be generated or measured beforehand and is expected to remain mostly static.

Our system for applying stylization to augmented reality is based on a painterly filter for the camera image and an artistic rendering scheme for the virtual objects. Non-photorealistic and painterly rendering and image filtering have been areas of very active research for several years. One overview of different techniques is given by Reynolds in [18]. DeCarlo and Santella have proposed an algorithm for processing photographs in order to achieve a simplified and stylized look [6]. Their technique uses color segmentation and edge detection, which partly inspired our approach. An algorithm for semi-automatic conversion of a real video sequence into a "cartoon-like" video has been presented by Wang et al. [23]. This method produces very good results, but it is an offline algorithm and computationally too expensive for real-time applications. Collomosse et al. have described a system for the stylized representation of motion in video sequences, which combines computer vision and graphics techniques [5].

Countless techniques exist for artistic rendering aimed at creating a cartoonish or sketch-like look. We have used two simple methods described by Lander for drawing the outline of a graphical model [17] and applying non-linear shading [16].

A system that integrates non-photorealistic rendering into augmented reality has been presented by Haller and Sperl [10, 12]. However, their system applies artistic rendering techniques only to the virtual objects, whereas the background camera image is displayed in its original, unprocessed form.

3 Stylized Augmented Reality

We propose a new paradigm for generating augmented reality images based on obtaining a similar, reduced level of realism for both the background camera image and virtual objects. The conventional method for overlaying graphical objects over the camera image is illustrated in Figure 2. Here, virtual objects are drawn over the background camera image using a standard renderer.

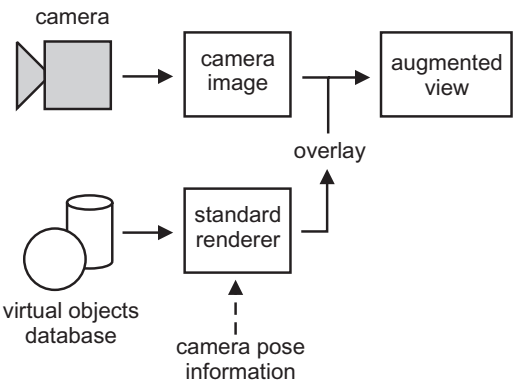


Figure 2: The image mixing process in conventional AR.

In stylized AR, the camera image is processed before it is used as background in the image mixing process. A painterly filter is applied to the input camera image. The aim of the painterly filtering step is to create a simplified, stylized version of the current camera view. After the camera image has been processed, the virtual objects are rendered over it. However, unlike in conventional AR, a non-photorealistic rendering (NPR) scheme is used instead of standard methods. The NPR renderer also creates a stylized representation of the graphical objects. An overview of this process is given in Figure 3.

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 IEEE

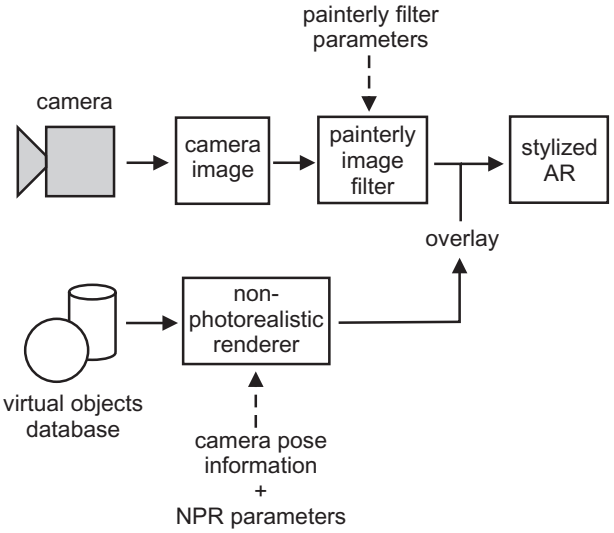
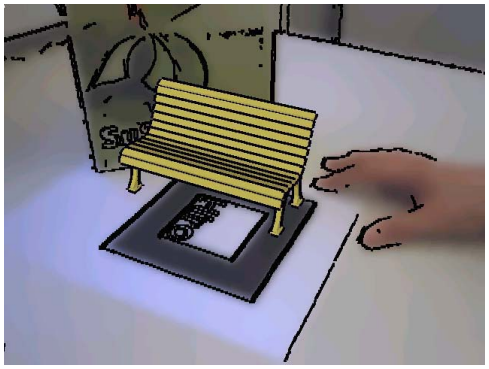
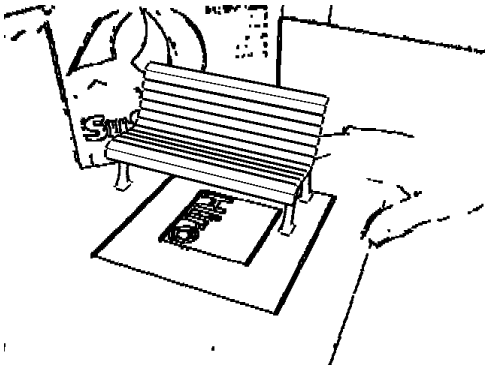


Figure 3: The image mixing process used for stylized AR.

An important precondition for a useful system of stylized augmented reality is the ability to generate images in real-time. We have thus devised a combination of painterly filter and non-photorealistic renderer, which is fast enough to ensure interactive overall frame rates. It is important to note that both the image filter and the rendering component can be customized using a set of parameters. In order to obtain a similar type of stylization for both AR image layers, the filter and rendering parameters must be tuned accordingly. Our system is currently capable of generating two different types of stylization. "Cartoon-like" AR images consist of flat, uniformly colored patches enclosed by silhouette lines. In a second, "sketch-like" mode, only black silhouettes are visible in front of a white background. Figure 4 shows a comparison of the two stylization modes.



(a) "Cartoon-like" stylization



(b) "Sketch-like" stylization

Figure 4: Comparison of "cartoon-like" and "sketch-like" stylization for a simple AR scene containing the virtual model of a bench.

4 Description of the Algorithm

We have implemented an augmented reality application providing the option to generate a stylized AR video stream. All of the AR test scenes shown in this paper were rendered using optical marker tracking based on the ARToolKit [15]. Nonetheless, our software framework is also capable of utilizing a special infrared tracking camera [7].

In the remainder of this section, the painterly filter (Section 4.1) and non-photorealistic rendering method (Section 4.2) used in our system are explained in detail.

4.1 Painterly Filter for Camera Image

The painterly filter used in our system is designed to simplify and stylize the input camera image. In order to achieve this effect, two separate steps are performed. One step aims at reducing detail in the camera image by generating large, uniformly colored regions based on the original image information. The second step detects high-contrast edges in the image. A post-processing step is applied to the edge image in order to generate thick lines which are adequate for the desired stylized look.

The images generated by the two steps of the painterly filter are then combined. The edge image is a binary map, in which detected edges have full intensity. Since the silhouette edges in the final output image are supposed to be black, the edge image is inverted and then combined with the color image using the binary AND operation.

If the "sketch-like" stylization mode is used, the color segmentation step is skipped. In this case, only edge detection is performed, and the inverted edge map is the output of the painterly filter. This way an image with a white background and black silhouette lines is created. An overview of the painterly filtering process is shown in Figure 5.

An essential design goal of the elements of the painterly filter is to achieve an interactive execution speed. Since it is embedded in a complex augmented reality and stylization pipeline, the painterly filter may only take a few milliseconds to compute. Due to these strict runtime requirements, most of the filtering process had to be designed from scratch, although more advanced, slower methods for painterly image processing have been described in the literature. In order to achieve a sufficiently short execution time for our algorithms, we use the speed-optimized OpenCV library [13] for basic image processing operations.

4.1.1 Color Segmentation

The first stage of the painterly filter aims at converting the input image into an image consisting of regions which are mostly uniformly colored. This way a simplified and stylized "coloring book" look is achieved. This task is closely related to color segmentation. However, due to the time constraints it is not possible to achieve a truly

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 IEEE

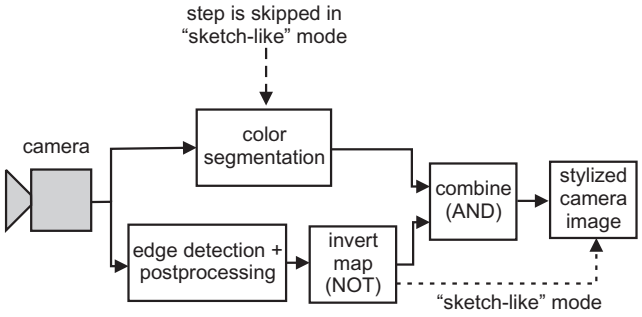


Figure 5: Overview of the painterly filtering process.

satisfying full segmentation in this filtering step. Our approach is based on bilateral image filtering as described by Tomasi and Manduchi [22]. The basic idea of bilateral filtering is to take spatial distance as well as signal difference into account when smoothing a function. Unlike a normal Gaussian filter, the bilateral filter thus leaves high-contrast edges mostly unaltered, while it has a strong smoothing effect on rather homogeneous regions.

Given the multi-channel (RGB) image function $\mathbf{f}$, the bilateral filter computes the smoothed image $\mathbf{h}$ using the following equation:

$$\mathbf{h}(\mathbf{x}) = k^{-1}(\mathbf{x}) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{f}(\xi) c(\xi, \mathbf{x}) s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))\, d\xi \qquad (1)$$

In Equation 1, $\mathbf{x}$ is the currently regarded point in the output image, and the integral is computed in two dimensions over neighboring image points ξ. In the discrete case, this is equivalent to a weighted sum of image pixels $\mathbf{f}(\xi)$ in the neighborhood of $\mathbf{x}$. The weight is a product of two factors. $c(\xi, \mathbf{x})$ is a function of the vector difference $\xi - \mathbf{x}$, i.e. the spatial distance. The second factor, $s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))$ depends on the similarity of values in the color channels $\mathbf{f}(\xi) - \mathbf{f}(\mathbf{x})$. In the implementation used by our algorithm, both c and s are Gaussian functions:

$$c(\xi, \mathbf{x}) = e^{-\frac{1}{2}\left(\frac{|\xi - \mathbf{x}|}{\sigma_d}\right)^2} \qquad (2)$$

$$s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x})) = e^{-\frac{1}{2}\left(\frac{|\mathbf{f}(\xi) - \mathbf{f}(\mathbf{x})|}{\sigma_r}\right)^2} \qquad (3)$$

Note that the value of c is a function of the Euclidean distance between ξ and $\mathbf{x}$ (Equation 2), while s depends on the absolute value of the color difference $\mathbf{f}(\xi) - \mathbf{f}(\mathbf{x})$ (Equation 3). The standard deviations of the Gaussian functions, σ_d and σ_r, determine the properties of the smoothing and can be chosen by the user as parameters for our painterly filter. In order to maintain the overall brightness of the image, the integral is divided by the normalization factor $k(\mathbf{x})$, which is computed as shown in Equation 4.

$$k(\mathbf{x}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, \mathbf{x}) s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))\, d\xi \qquad (4)$$

The effect of the bilateral filter is that smoothing only occurs in places, where nearby pixels have similar colors. In such places in the image, both $c(\xi, \mathbf{x})$ and $s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))$ are large. If near the currently regarded pixel colors are present which are relatively far away in color space, smoothing is suppressed. Thus strong edges in the image are preserved.

The disadvantage of bilateral filtering for our application is that it is too computationally expensive. Due to the necessity for computing the vector differences, the Gaussian functions, and the normalization factor, even an optimized implementation is too slow for our real-time requirements. We thus create a Gaussian pyramid of iteratively shrunk images and apply the bilateral filter to the smallest version only. The use of resolution pyramids for image processing is described for instance in [3].

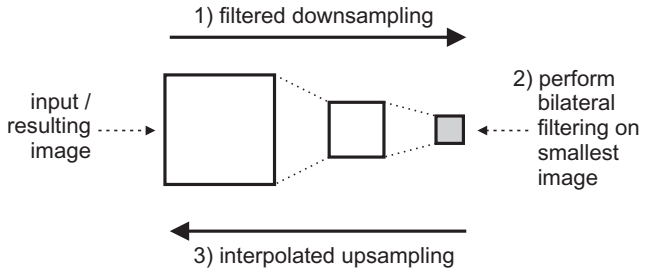


Figure 6: Construction of Gaussian pyramid for speeding up the bilateral filter.

Figure 6 shows the principle of using a Gaussian pyramid for speeding up the color segmentation step. For each level of the pyramid, the image resolution is reduced in both dimensions by a factor of two. This results in an image with only a quarter of the number of pixels of the previous level. The image is filtered with a standard Gaussian filter before the subsampling takes place, reducing aliasing effects. A bilateral filter with standard deviations σ_d and σ_r is then applied to the top level of the pyramid, the smallest version of the image. Afterwards, this image is iteratively scaled up, each time by a factor of two in both dimensions. In this process, a modified Gaussian filter is used for smooth interpolation of missing rows and columns of pixels. Finally, an image with the same resolution as the original camera image is created, which is the output of our color segmentation method. The number of levels of the Gaussian pyramid can be selected by the user as a parameter for the stylized AR system.

The required computation time for the bilateral filter decreases with an increasing number of pyramid levels. On the other hand, the resulting image is more blurred, if more pyramid levels are used. In Table 1 measured runtimes for the entire color segmentation process, including construction and decomposition of the Gaussian pyramid, are listed (column “filter time”). The third column contains average measured overall frame rates of the AR system, if only color segmentation is used for processing the camera image. The table clearly shows that a larger number of pyramid levels significantly increases overall system performance. Figure 7 shows two color segmented camera images of the same scene and illustrates how a greater number of pyramid levels leads to a more blurred result.

Table 1: Measured execution times of color segmentation.

#Pyramid levels	Filter time	Frames/sec.
1	0.217s	4.57 fps
2	0.092s	10.79 fps
3	0.063s	16.00 fps
4	0.055s	18.00 fps

4.1.2 Edge Detection

In the second stage of the painterly filtering process, edges are detected in the camera image. The detected edges are used as black silhouette outlines in the painterly filtered image, resulting in the “cartoon-like” or “sketch-like” look. Our system uses the Canny edge detector [4]. Note that the edge detection step is performed on the original camera image, not on the result of the color segmentation process.

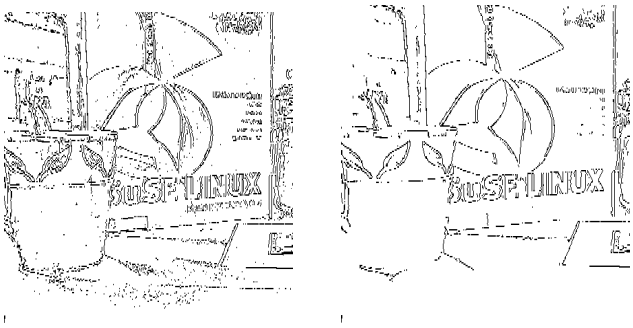
Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 **IEEE**

(a) One pyramid level (b) Three pyramid levels

Figure 7: Color segmentation results using a different number of Gaussian pyramid levels for the same scene.

The Canny edge detector is based on a standard gradient computation, normally using the Sobel operator, followed by methods for suppressing erreneous responses. The final step of the Canny detector is the so-called hysteresis, which eliminates detection responses based on two threshold values. The larger threshold is used for finding definite edge pixels, which can be thought of as starting points for continuous edge segments. Any pixel connected to an already identified edge pixel is also marked as an edge pixel, if its gradient is greater than the smaller threshold. This way continuous edges are generated, while detached and weak gradient operator responses caused by noise are suppressed.

In our system, these two thresholds for the Canny detector, t_1 and t_2, can be selected by the user. In order to achieve the goal of a stylized look with an emphasis on strong edges, we tend to use rather large threshold values. Smaller thresholds can result in visually nonrelevant edges being emphasized, or can even leave noise in the edge image. This is illustrated in Figure 8.



(a) t_1=80, t_2=60 (small thresholds result in image noise)

(b) t_1=150, t_2=105 (larger thresholds generate a better result)

Figure 8: Canny edge detection results for different thresholds.

It is one of the properties of the Canny edge detector, that the edge image it delivers consists of very thin edge segments (see Figure 8(b)). For the purpose of the painterly filter used in our system, the black lines derived from the edge image are supposed to be much thicker. In order to achieve this effect, a morphological operator is applied to the image. The morphological dilation processes a binary image by assigning a value of one to all pixels in the neighborhood of any pixel which has a value of one [9]. As shown in Figure 9, this morphological postprocessing makes the edges appear thicker and removes discontinuities.

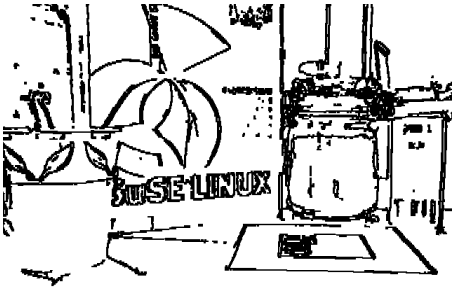


Figure 9: Application of morphological dilation to the edge image. The thresholds used for this image are the same as in Fig. 8(b).

4.2 Non-photorealistic Rendering Method

For rendering virtual objects in the stylized AR system, a non-photorealistic method consisting of two main components is used. A technique for drawing thick silhouette outlines is applied to the geometry of the objects. Moreover, a custom shading method is utilized, which reduces the number of colors in the generated image. These two methods make the rendered objects appear visually similar to the painterly filtered background image.

4.2.1 Silhouette Rendering

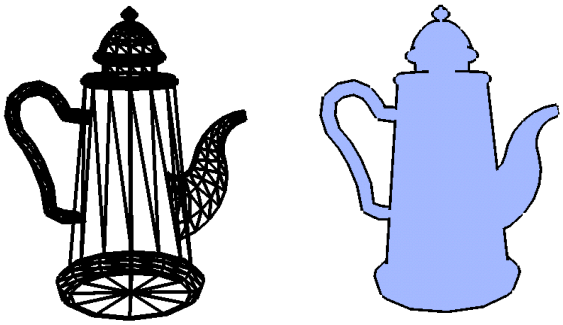
A simple method for rendering the silhouette outline of a graphical object by utilizing the capabilities of the OpenGL graphics library has been described by Lander [17]. This approach is based on drawing the geometry of each graphical object twice. In a first rendering pass, filled polygons are used for drawing the object. The Z-buffer test is applied in the normal way, so that pixels are only rendered, if they are closer to the viewer than primitives which have already been drawn. In this step, only front-facing polygons are rendered.

For the second rendering pass, the face culling performed by OpenGL is adjusted, so that only back-facing polygons are drawn. These polygons, which face away from the camera, are rendered in wireframe mode. This means that only the outlines are drawn for each polyon. The wireframe representation is made up of thick lines. In this rendering pass, the Z-buffer is set up to allow the display of all pixels which are closer to the viewer or at the same depth as previously rendered pixels. This second rendering pass affects such places in the image, where back-facing and front-facing polygon edges have the same distance to the viewer. For most graphical objects, these places correspond to silhouette edges. The result of this rendering pass is that thick lines are visible, where the silhouette outline of an object should be. The following code fragment explains how the two rendering passes can be performed using OpenGL (taken from [17]):

```
glPolygonMode(GL_FRONT, GL_FILL);
glDepthFunc(GL_LESS);
glCullFace(GL_BACK);
DrawModel(); // Draw primitives of virtual object
glPolygonMode(GL_BACK, GL_LINE);
glDepthFunc(GL_LEQUAL);
glCullFace(GL_FRONT);
DrawModel(); // Draw primitives of virtual object
```


Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 **IEEE**

Figure 10 illustrates the effect of the silhouette rendering method. In our implementation, the thickness of the silhouette lines can be selected by the user. Although the approach is based on two-pass rendering, its impact on the overall system frame rate is negligible except for extremely large virtual models. The advantage of this method is that it does not require advanced OpenGL functionality (e.g. shaders) and thus is highly portable.



(a) The second rendering pass: Only the outlines of back-facing polygons are drawn

(b) Final result of the silhouette rendering method

Figure 10: Silhouette rendered for the model of a teapot.

4.2.2 Non-linear Shading

The second method utilized by our system for giving a stylized look to virtual objects is non-linear shading. Normally, brightness and color parameters computed for the vertices of an object are spread over the visible surface using OpenGL shading. This means, that either flat shading is applied, which uses constant color and brightness, or Gouraud shading, which uses linear interpolation between the vertex parameters over the area of a polygon. While flat shading is too simplistic even for the purposes of a stylized AR renderer, the application of Gouraud shading generates too many different intensity levels. Although Gouraud shading can make surfaces appear round (as seen in Figure 1(a)), it also creates a too artificial look that does not correspond well to the painterly filtered background image.

The non-photorealistic non-linear shading method, which has also been described by Lander [16], only generates a very limited and well-defined set of intensities. This is achieved by modulating the color of the object with a special one-dimensional texture. The one-dimensional texture contains a function translating texture coordinates to a brightness value in discrete, quantized steps. An example of a one-dimensional shading texture is shown in Figure 11. In our stylized AR system, the shading texture is generated automatically. The number of quantization steps and the base intensity can be selected by the user. In order to be able to show the effects of full light intensity, the shading texture always contains full texel intensity in the last texels (127 for OpenGL signed byte intensity textures). Note that our non-photorealistic renderer always uses a texture with a size of 32 texels, unlike the 16-texel texture used for clarity in Figure 11.

For the actual rendering process, OpenGL lighting computations are disabled. Instead, light intensity is computed by our system for every vertex. This is done using simple diffuse reflection:

$$I_i = \mathbf{n_i} \cdot \mathbf{L} \qquad (5)$$

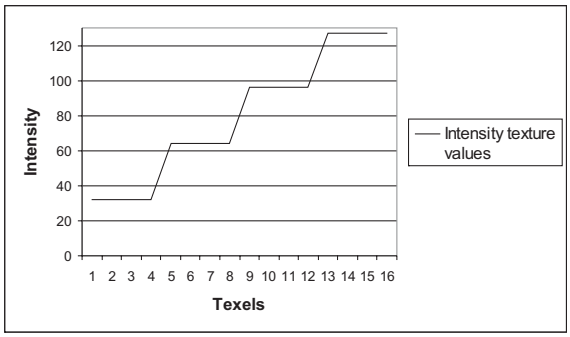
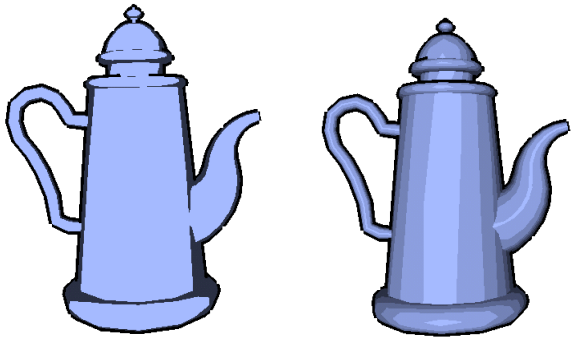


Figure 11: Example of one-dimensional intensity texture.

In Equation 5, the brightness I_i of vertex i is computed as the dot product of its normal vector $\mathbf{n_i}$ and the light direction $\mathbf{L}$. In order to obtain a correct lighting, each vertex normal is rotated by the rotational component of the current OpenGL transformation matrix beforehand. This ensures, that the current viewing parameters are taken into account. Our system maintains normalized $\mathbf{n_i}$ and $\mathbf{L}$, therefore, the result of the dot product is between -1 and 1. Negative values are clamped to zero. The resulting value between 0 and 1 is then used as index to the one-dimensional shading texture. This way the texture acts as a lookup table for the non-linear shading function. The effect of non-linear shading with a different number of quantization steps is illustrated in Figure 12. Our system provides functionality for manually specifying the light direction used by the shading algorithm.



(a) 2 quantization levels

(b) 6 quantization levels

Figure 12: Non-linear shading with a different number of quantization levels in the shading texture.

5 Results

We have tested the stylized augmented reality approach with numerous different test scenes. Our augmented reality software contains an editor, which is capable of importing 3-d models in the standard Wavefront OBJ file format. The user can freely place, scale and rotate the model. Moreover, a graphical user interface for adjusting all parameters of the painterly filter and the non-photorealistic renderer is provided. These include the number of Gaussian pyramid levels and standard deviations for the bilateral filter, as well as object color, lighting parameters and line thickness for the non-photorealistic rendering algorithm. Additionally the

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 **IEEE**

user can choose between the "sketch-like" and the "cartoon-like" display mode. All of these data, including object file name and transformation, can be stored in a single XML-file describing the scene. Thus experimental setups can easily be restored and compared.

For all of our test scenes, optical marker tracking based on the ARToolKit framework was used [15]. The user-defined transformation of the 3-d model is in relation to an ARToolKit marker, which determines the origin of the coordinate system.

Figure 15 shows four augmented reality test scenes. In each row of images, the leftmost column contains the augmented scene as generated by conventional AR image composition. In the second column, the image is rendered with our "cartoon-like" style. The "sketch-like" stylization mode for each test scene is illustrated in the third column. The first test scene contains a Moka Express coffee maker as the virtual object, which is located directly over the marker. In the second scene, a model representing a Volkswagen Beetle car is placed over the marker. A simple virtual teacup can be seen amid various real utensils in the third row of images. Finally, a model of the Statue of Liberty rendered on top of a stack of books is the virtual element in the fourth AR scene.

(a) Santa Claus model, conventional AR

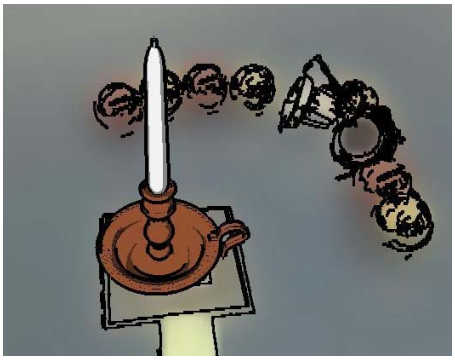
(b) Santa Claus model, "cartoon-like" style

Figure 13: Santa Claus example scene for the "cartoon-like" style.

Figures 13 and 14 show two more test scenes for the "cartoon-like" stylization method. A graphical Santa Claus model is placed over the marker in Fig. 13, and the model of a candle is the virtual object in Fig. 14.

The goal of our stylization approach for augmented reality is to achieve improved immersion. This means that it is less obvious to the user whether an object in the augmented image is real or virtual. The example images in Figures 13, 14 and 15 show that this effect is achieved by our method. Especially for scenes in which the

(a) Candle model, conventional AR



(b) Candle model, "cartoon-like" stylization

Figure 14: Candle example scene for the "cartoon-like" style.

scale of the virtual object matches the physical world, the barrier between virtual and real is blurred. The coffee maker (Fig. 15(b)) and teacup (Fig. 15(h)) are good examples for scenes in which the virtual model appears to be a natural part of the real environment thanks to the stylized display method.

Our augmented reality system uses a Firewire webcam delivering a resolution of 640 by 480 pixels. Benchmark measurements show that our stylized AR approach can generate augmented images in real-time, with frame rates of approximately 18 fps on the average for the "sketch-like" and 14 fps for the "cartoon-like" mode. Camera image acquisition and painterly filter take roughly 65 msecs for the "cartoon-like" style, which requires color segmentation, and roughly 45 msecs for the sketch style.

6 Conclusion

We have presented a novel approach to combining real and virtual images in augmented reality. The adapted levels of realism for camera image and graphical objects result in improved immersion. This way a more impressive AR experience can be achieved. Applications in entertaining, training and education can benefit from the blurred barrier between the virtual and real worlds.

Our experiments have shown that the stylization techniques used in our approach successfully make the two layers of an AR image - background image and rendered objects - look similar. Users of the stylized augmented reality system consistently assess it as more immersive than conventional AR. The improved similarity is also evident in still images generated with our technique. For some

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 **IEEE**

augmented reality scenes, users are genuinely unable to distinguish virtual elements from real objects, depending on the point of view.

The techniques and results presented in this paper constitute a good starting point for further research. This could include a formal user study on the effectiveness of stylized AR and the development of application scenarios based on the new approach.

ACKNOWLEDGEMENTS

We would like to thank Ángel del Río for his support during the experiments and for proofreading this paper.

Most of the graphical models used in our experiments were downloaded from the 3D Cafe website [21].

This work has been supported by project VIRTUE in the focus program on "Medical Robotics and Navigation" (SPP 1124) of the German Research Foundation (DFG).

REFERENCES

[1] K. Agusanto, L. Li, Z. Chuangui, and N.W. Sing. Photorealistic Rendering for Augmented Reality using Environment Illumination. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 208–216, October 2003.

[2] R. Azuma. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997.

[3] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice-Hall, 1982.

[4] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 8(6):679–698, November 1986.

[5] J.P. Collomosse, D. Rowntree, and P.M. Hall. Cartoon-Style Rendering of Motion from Video. In *Vision, Video and Graphics (VVG)*, pages 117–124, July 2003.

[6] D. DeCarlo and A. Santella. Stylization and Abstraction of Photographs. In *Proceedings of ACM SIGGRAPH*, pages 769–776, July 2002.

[7] J. Fischer, M. Neff, D. Freudenstein, and D. Bartz. Medical Augmented Reality based on Commercial Image Guided Surgery. In *Eurographics Symposium on Virtual Environments (EGVE)*, June 2004.

[8] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics - Principles and Practice*. Addison-Wesley Publishing company, 2nd edition, 1997.

[9] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1992.

[10] M. Haller. Photorealism or/and Non-Photorealism in Augmented Reality. In *ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry (VRCAI)*, pages 189–196, June 2004.

[11] M. Haller, S. Drab, W. Hartmann, and J. Zauner. A Real-time Shadow Approach for an Augmented Reality Application using Shadow Volumes. In *ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 56–65, October 2003.

[12] M. Haller and D. Sperl. Real-Time Painterly Rendering for MR Applications. In *International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia, Graphite*, pages 30–38, June 2004.

[13] Intel Corporation. *Open Source Computer Vision Library Reference Manual*, 2001.

[14] M. Kanbara and N. Yokoya. Geometric and Photometric Registration for Real-Time Augmented Reality (posters and demo session). In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, page 279, September 2002.

[15] H. Kato and M. Billinghurst. Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System. In *Proceedings of IEEE and ACM International Workshop on Augmented Reality (IWAR)*, pages 85–94, October 1999.

[16] J. Lander. Shades of Disney: Opaquing a 3D World. *Game Developer Magazine*, March 2000.

[17] J. Lander. Under the Shade of the Rendering Tree. *Game Developer Magazine*, February 2000.

[18] C. Reynolds. Stylized Depiction in Computer Graphics - Non-Photorealistic, Painterly and 'Toon Rendering. http://www.red3d.com/cwr/npr/, 2003.

[19] D. Shreiner, M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide*. Addison-Wesley Publishing Company, 4th edition, November 2003.

[20] N. Sugano, H. Kato, and K. Tachibana. The Effects of Shadow Representation of Virtual Objects in Augmented Reality. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 76–83, October 2003.

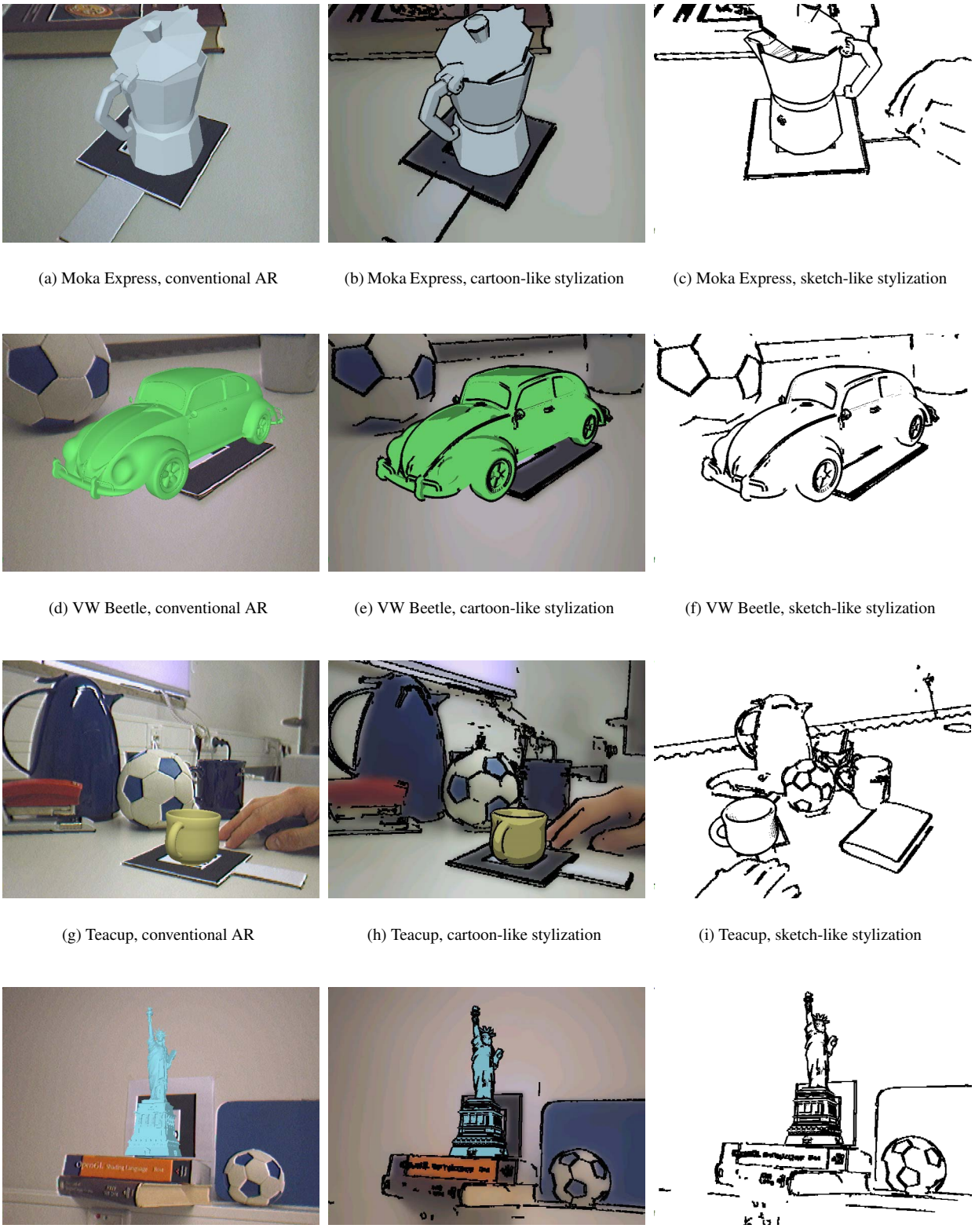
[21] The 3D Cafe website. 3D Cafe's Free Stuff. http://www.3dcafe.com/asp/freestuff.asp, 2005.

[22] C. Tomasi and R. Manduchi. Bilateral Filtering for Gray and Color Images. In *International Conference on Computer Vision (ICCV)*, pages 839–846, 1998.

[23] J. Wang, Y. Xu, H.-Y. Shum, and M. Cohen. Video Tooning. In *Proceedings of ACM SIGGRAPH*, August 2004.

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 **IEEE**

Color plate of paper by Fischer et al. on page 195



(a) Moka Express, conventional AR (b) Moka Express, cartoon-like stylization (c) Moka Express, sketch-like stylization

(d) VW Beetle, conventional AR (e) VW Beetle, cartoon-like stylization (f) VW Beetle, sketch-like stylization

(g) Teacup, conventional AR (h) Teacup, cartoon-like stylization (i) Teacup, sketch-like stylization

(j) Statue of Liberty, conventional AR (k) Statue of Liberty, cartoon-like style (l) Statue of Liberty, sketch-like style

Figure 15: Four example scenes illustrating the effect of the stylized augmented reality approach. In each row, the leftmost column shows conventional AR rendering. The second and third column contain the stylized versions. Note that Figure 15(c) and 15(i) show the respective scene from a different angle.

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 **IEEE**