A Pointillism Style for the Non-Photorealistic Display of Augmented Reality Scenes

Jan Fischer[†] and Dirk Bartz

Visual Computing for Medicine at WSI/GRIS, University of Tübingen, Germany

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.

We have recently proposed a novel approach for generating augmented reality images. 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. Special non-photorealistic image filters are applied to the camera video stream. The virtual scene elements are rendered using non-photorealistic rendering methods. Since both the camera image and the virtual objects are stylized in a corresponding way, they appear very similar. As a result, graphical objects can become indistinguishable from the real surroundings.

Here, we present a new method for the stylization of augmented reality images. This approach generates a painterly "brush stroke" rendering. The resulting stylized augmented reality video frames look similar to paintings created in the "pointillism" style. We describe the implementation of the camera image filter and the non-photorealistic renderer for virtual objects. These components have been newly designed or adapted for this purpose. They are fast enough for generating augmented reality images in real-time and are customizable. The results obtained using our approach are very promising and show that it improves immersion in augmented reality.

Categories and Subject Descriptors (according to ACM CCS): H.5.1 [Information Interfaces and Presentation]: Artificial, augmented, and virtual realities; I.3.7 [Computer Graphics]: Color, shading, shadowing, and texture

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 [Azu97].

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. The graphical primitives which constitute virtual objects in the AR scene are then rendered over the background image using standard computer graphics methods. Common real-time graphics libraries like OpenGL or high-level frameworks based on them are often utilized for this task.

In order to reach real-time rendering performance, 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 [FvFH97].

The resulting rendered images generally look artificial, es-
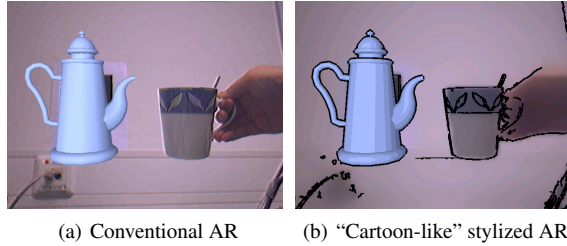

[†] fischer@gris.uni-tuebingen.de

(a) Conventional AR   (b) "Cartoon-like" stylized AR

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

pecially 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 [HS99], the problem of mismatched scene generation parameters would still persist. Since light sources and material properties are defined during definition of the AR scene, they generally do not correspond well to the lighting conditions in the actual surroundings.

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.

We have recently proposed a novel way of generating augmented reality images. This new method uses 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 a "cartoon-like" stylization method. In such an augmented view, it is significantly more difficult to distinguish between real and virtual objects, resulting in a dramatically improved immersion into the AR scene. In contrast to our earlier described "cartoon-like" stylization for augmented reality [FBS05], we here present a completely new method for stylized augmented reality, the "brush stroke artistic reality". In contrast to previous "brush stroke" stylized rendering approaches, our method renders both the video stream of the actual surroundings and the virtual objects with this stylization to maintain a coherent appearance. Furthermore, our approach achieves real-time performance, which is essential for augmented reality applications.

Our stylized augmented reality approach inherently removes details from the background camera image. Whereas this is inappropriate for some applications like medical di-

agnostics, many applications can benefit greatly from the improved immersion offered by our method. In particular in fields like art, 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 [KY02]. Their method requires a special marker and mirror ball to be visible in the camera image for computing the environment light map. A different approach to increase realistic appearance of virtual objects 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 [Hal04, HDHZ03]. As a drawback of this method, a static model of the geometry of the surfaces and objects in the real world is required.

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 (NPR) have been areas of very active research for several years. While many NPR approaches employ silhouette and hatching techniques, we focus in this paper only on brush-stroke techniques and NPR techniques for interactive scenes or video streams. An overview of many techniques is given by Reynolds [Rey03]. Strothotte and Schlechtweg have also published a good survey of methods used in the field [SS02].

The use of brush strokes for computer-generated images has been described before, for example by Meier [Mei96]. A more recent implementation and modification of Meier's algorithm has been presented by Haller and Sperl [HS04]. They also presented a system that integrates non-photorealistic rendering into augmented reality [Hal04, HS04]. However, their system applies artistic rendering techniques only to the virtual objects, whereas the background camera image is displayed in its original, unprocessed form.

Another application of NPR-rendering to virtual environments was presented by Klein et al. [KLK*00]. Again, NPR-rendering was only applied to virtual objects and no video information was included.

Litwinowicz presented an approach that generates an artistic rendering style [Lit97] very similar to our approach. He maintains temporal coherence of the brush-strokes with

motion estimation algorithms on the video images (with no additional virtual objects).

For pure video sequences and interactive scenarios (with no video content), Hertzmann and Perlin introduced an approach for an NPR-style [HP00] similar to Haeberli's "Paint By Numbers" approach [Hae90], which achieved interactive performance (>10fps). Another approach for interactive performance for NPR-rendering of virtual objects was presented by Cornish et al. [CRL01]. In their particle-based approach, they employ a view-dependent cluster algorithm to keep the number of particles in a manageable range.

An algorithm for semi-automatic conversion of a real video sequence into a stylized video has been presented by Wang et al. [WXSC04]. This method produces very good results, but it is an offline algorithm and computationally too expensive for real-time applications. Another offline technique for still images and video sequences was presented by Raskar et al. [RTF*04]. Xu et al. presented a stylized rendering technique for scanned outdoor scenes [XC04], which achieved real-time performance after preprocessing of the scanned data.

In the remainder of this article, the basic idea of stylized augmented reality is explained in Section 3. The "brush stroke" stylization method is described in Section 4. Section 5 discusses results obtained with the new algorithm. Finally, Section 6 summarizes our findings.

3. Stylized Augmented Reality

We propose a 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.
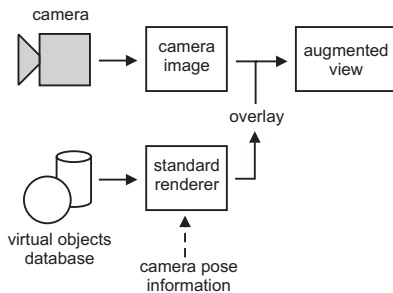

Figure 2: *The image mixing process in conventional AR.*

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 creates a stylized representation of the graphical objects. An overview of this process is given in Figure 3.
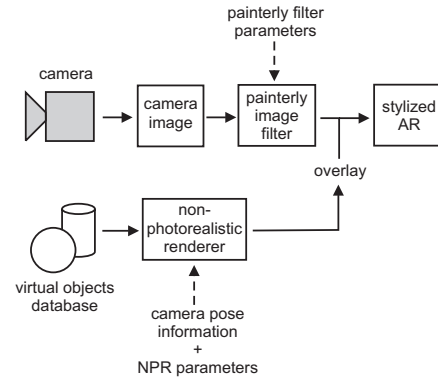

Figure 3: *The image mixing process used for stylized AR.*

An important precondition for an useful system of stylized augmented reality is the ability to generate images in real-time. We have thus devised a camera image filter and a non-photorealistic rendering technique, which are fast enough to ensure interactive overall frame rates. It is important to note that 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. In this paper, we present the new painterly "brush stroke" mode for the stylization of augmented reality images. Images generated with this algorithm are composed of a large number of small brush strokes.

4. Brush Stroke Stylization

In this section, we present a new approach to generating stylized augmented reality images. This novel method aims at reproducing a painting style found in real-life pictures. The generated images consist of a large number of small brush strokes, a method applied by painters who adhere to the *pointillism* style of painting [Art04].

As described in Section 3, a painterly filter is applied to the input camera image. This painterly filter randomly samples the camera image and paints brush strokes with the colors of the sampled camera pixels. Afterwards, the virtual model is rendered using the brush stroke style. In order to achieve an appearance similar to the processed camera image, the rendering of the virtual model also consists of colored brush strokes. The polygonal geometry of the virtual model is first converted to a three-dimensional particle model as a preparation for the painterly rendering. The renderer projects each of the particles into screen space and uses

these 2D positions as basis for the brush stroke representation of the model. Our method can thus be considered a simplified specialization of point based rendering (see for instance [PZvG00]).
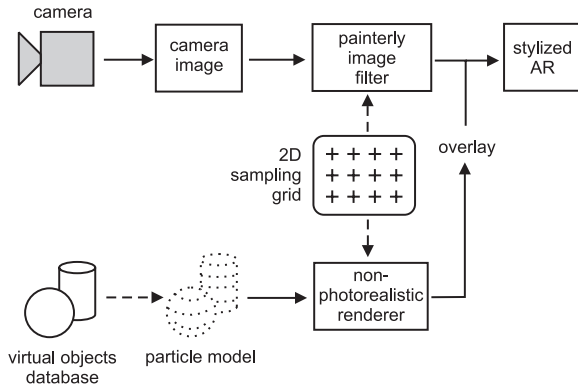

Figure 4: *Overview over the method for brush stroke stylization of augmented reality images.*

We have found that in order to make the virtual model and camera image look similar, the same brush stroke positions must be used. In early experiments employing a method which directly renders projected particles, the brush stroke rendering of the virtual model seemed to float above the background image. Our method thus assigns each projected particle position to the nearest point in a precomputed 2D brush stroke grid. This is the same grid that is used for sampling the camera image.

Figure 4 shows an overview of our system for brush stroke stylization.

4.1. Brush Stroke Filter for Camera Image

In this section, we describe the image filter for creating a brush stroke version of the camera image. Section 4.1.1 first explains the generation of the 2D sampling grid required by the filter, Section 4.1.2 describes the actual filtering operation.

4.1.1. Generation of 2D Grid

A two-dimensional sampling grid is generated in an one-time preprocessing step. The grid remains constant throughout the processing of consecutive input camera images. It is stored as an array of sampling point records. Each sampling point record contains the 2D position of the point and additional information about the brush stroke which is to be painted there.

Table 1 lists the attributes stored for each sampling point. $x_{sample(i,j)}$ and $y_{sample(i,j)}$ are the two-dimensional position of the point in the camera image, and $radius_{(i,j)}$ is the radius of the brush stroke to be drawn there. Moreover, a fixed RGB

Table 1: *Attributes stored for one sampling point*

Attribute	Data type
$x_{sample(i,j)}$	integer
$y_{sample(i,j)}$	integer
$radius_{(i,j)}$	integer
colorOffset$_{(i,j)}$	RGB color

color offset is stored for each sampling point. This color offset is later added to the color of any brush stroke which is to be drawn at that position.
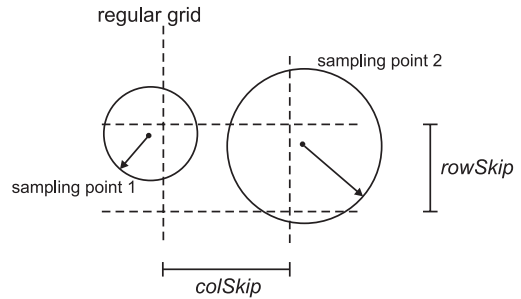

Figure 5: *Illustration of two sampling points randomly displaced from the regular grid with random brush stroke radii.*

Sampling points are generated over the entire area of the camera image. Each point is initially located on a regular grid with a horizontal step size of *colSkip* and a vertical step size of *rowSkip*. A random displacement vector is added to the point positions, as shown in Equation 1. The maximum random offset *range* is an user-definable parameter. Brush stroke radius $radius_{(i,j)}$ and RGB color offset **colorOffset**$_{(i,j)}$ are also generated randomly with random number ranges defined by the user. An illustration of two sampling points is shown in Figure 5.

$$x_{sample(i,j)} = i \cdot colSkip + rand(-range, range) \quad (1)$$
$$y_{sample(i,j)} = j \cdot rowSkip + rand(-range, range)$$

The random variations of point position, brush stroke radius and brush stroke color are introduced into the sampling point grid in order to create a more natural, irregular look for the generated image. Note that the total number of grid points depends on *colSkip* and *rowSkip*. The random number range for $radius_{(i,j)}$ has to be selected so that a good covering of the image area by brush strokes is achieved.

Another measure for generating a more natural look for the processed image is the application of a random drawing order for the brush strokes. An array containing the indices of all sampling points is generated. This array is then randomly shuffled. When the camera image filter is applied, this index array is traversed sequentially, resulting in a random brush stroke drawing order.

4.1.2. Camera Image Filter

The camera image filter samples the input image by reading pixel colors at the sampling point positions in the given, random order. Color offset **colorOffset**$_{(i,j)}$ is then added to each pixel color, and the resulting RGB components are clamped to the valid real number range $[0;1]$. Each brush stroke is drawn as a textured quadratic rectangle with side length $2 \cdot radius_{(i,j)} + 1$, centered at $(x_{sample(i,j)}, y_{sample(i,j)})$. The brush stroke texture is loaded from file beforehand. During brush stroke rendering, alpha blending is enabled to achieve partial transparency for overlapping brush strokes. An example of a brush stroke image generated by the camera image filter is shown in Figure 6.
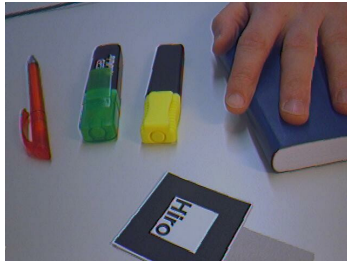
(a) Original camera image

(b) Result of brush stroke filter

Figure 6: *Example of an image generated by the brush stroke filter for the camera image.*

4.2. Brush Stroke Renderer for Virtual Objects

In order to be able to generate a brush stroke rendering of virtual objects in the augmented reality scene, they are first converted to particle models. This process is described in Section 4.2.1. For each frame, the projected particles are then assigned to the nearest 2D sampling point. The computation

of the lookup table required for this step is explained in Section 4.2.2. The actual rendering method is then described in Section 4.2.3.

4.2.1. Creation of Particle Model

For each of the polyons constituting the graphical object, a number of particles lying on the polygon is generated. Each three-dimensional particle position is computed as a weighted sum of the polygon vertices. In Equation 2, the $\mathbf{v}_i$ denote the vertices of the current polygon, and N is the number of vertices. The position of the currently regarded particle is called **particlePos**$_j$.

$$\mathbf{particlePos}_j = \sum_{i=1}^{N} w_i \cdot \mathbf{v}_i \qquad \sum_{i=1}^{N} w_i = 1 \qquad (2)$$

As expressed in Equation 2, the vertex weights w_i sum up to a value of one, so that **particlePos**$_j$ is located on the polygon. The weights are chosen randomly in order to generate a random particle position. Particle color **particleCol**$_j$ and normal vector **particleNorm**$_j$ are stored as additional attributes for each particle. The particle color is obtained from a texture lookup, for which texture coordinates are computed as a sum of the texture coordinates of the polygon vertices weighted with w_i. The user can load any bitmap as texture image for with the graphical model. The particle normal is calculated by correspondingly interpolating the normals of the polygon vertices and normalizing the result.

In order to achieve a homogeneous distribution of particles over the entire surface of the model, a specific number of particles is calculated for each polygon. The total number of particles to be generated is selected by the user as parameter *numParticles*. Before generating the particle model, the total surface area of the virtual model, *totalArea*, is determined. For each polygon, the number of particles is then computed as the ratio of its area to *totalArea* multiplied by *numParticles*.

An example of a particle model generated for a virtual object is shown in Figure 7.
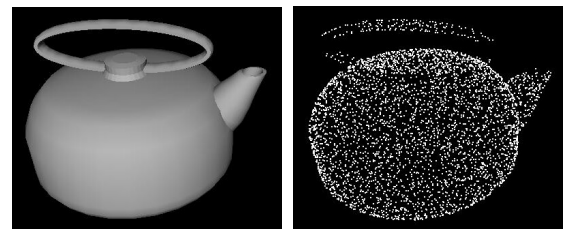
(a) Polygonal model   (b) Particle model

Figure 7: *Particle model generated for a teapot object.*

4.2.2. Sampling Point Lookup Table

During the brush stroke rendering process, the nearest 2D sampling point has to be determined for each projected particle. Due to the requirement for overall real-time performance, a lookup table containing the index of the nearest sampling point for each pixel in the camera image coordinate system is computed. The table can be considered a representation of the Voronoi diagram of the sampling point set [Wol04]. An illustration of the sampling point lookup map is shown in Figure 8. The lookup table is generated in an one-time preprocessing step directly after the creation of the sampling grid (see Section 4.1.1).
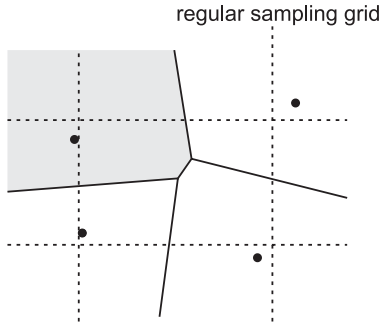

Figure 8: *Illustration of the Voronoi diagram of four sampling points. The black dots are the sampling point positions, randomly displaced from the regular grid. The gray area in the top left part represents the camera image pixels which are assigned to the top left sampling point. The three remaining segments belong to the respective other sampling points.*

The major challenge in computing the lookup map is the fact that the random displacement of sampling points can be large. Thus the nearest sampling point cannot be easily determined for a given camera image pixel. The naive approach of traversing the camera image and testing the pixel coordinates against the distance to each sampling point has proven to be be too slow even for a preprocessing step. Already for a relatively small number of points, this method can require a runtime of several minutes. We have therefore implemented the reverse approach to generating the lookup map. For each sampling point, the distances to camera pixels in a neighbourhood are compared with their distances to adjacent sampling points.

$$dist_{xy}(i,j) = \sqrt{(x_{sample(i,j)} - x)^2 + (y_{sample(i,j)} - y)^2}$$

for currently regarded (i,j):

$$\mathbf{map}(x,y) = \arg \min_{\substack{i'=i-1,...,i+1 \\ j'=j-1,...,j+1}} dist_{xy}(i',j') \qquad (3)$$

The points in the sampling grid are processed consecutively. Assuming that the grid position of the current sampling point is (i,j), the lookup map algorithm evaluates Equation 3 for each pixel position (x,y) in the neighbourhood. This means that lookup table entry $\mathbf{map}(x,y)$ will contain the grid indices of the nearest of the adjacent sampling points.

The size of the regarded pixel neighbourhood depends on the grid step sizes *rowSkip* and *colSkip* as well as the maximum random displacement *range* as shown in Equation 4. The neighbourhood of pixel (x,y) is defined as a rectangle which extends horizontally from $(x - xExtent)$ to $(x + xExtent)$ and vertically from $(y - yExtent)$ to $(y + yExtent)$.

$$xExtent = \left(\frac{colSkip}{2} + range\right)$$
$$yExtent = \left(\frac{rowSkip}{2} + range\right) \qquad (4)$$

We have found that this fast reverse approach to computing the sampling point lookup map results in a sufficiently short intialization stage of the renderer. We have measured computation times of the lookup map between two and four seconds for a 640 by 480 pixels camera image with typical sampling grid parameters on a standard PC.

4.2.3. Rendering Process

The actual rendering procedure for the particle models first determines the current color for each particle, projects the particles into screen space and sorts the particles according to their depth. The sorted particle list is then rendered from back to front, taking into account the positions and parameters of the 2D sampling points.

The renderer rotates the particle normals $\mathbf{particleNorm}_j$ according to the current transformation matrix. The active transformation and projection matrices as well as viewport parameters are retrieved from the current OpenGL state at the beginning of the rendering process. Based on the transformed normal vectors, particles on back-facing polygons are culled from the list of particles to be rendered. For visible particles, the current brightness is computed with diffuse reflection using the normal vector information. These brightness values are used for scaling the particle colors $\mathbf{particleCol}_j$ in order to achieve a shaded look for the model.

In the next step of the rendering process, the front-facing particles are projected into screen space on the basis of the stored matrices and viewport parameters. The resulting x and y coordinates and depth values are stored for each particle. Subsequently, the particle list is sorted according to the depth values.

Finally, brush strokes are rendered for the projected particles in the order of descending depth values. This drawing sequence constitutes an implementation of the

painter's algorithm for the correct mutual occlusion of brush strokes [FvFH97]. It is necessary, because Z-buffer tests are disabled for rendering the semi-transparent brush strokes with alpha blending.

The nearest 2D sampling point for each projected particle at (x, y) is looked up in $\mathbf{map}(x, y)$. The brush stroke is then drawn in the same way as applied by the camera image filter (see Section 4.1.2): The color offset stored in the sampling point record is added to the particle color, and then a semi-transparent textured rectangle with a side length depending on $radius_{(i,j)}$ is rendered at the sampling point location. Therefore, a brush stroke drawn for a model particle has the same visual properties as a brush stroke drawn by the camera image filter.

Figure 9 shows an example of a virtual model drawn by the brush stroke renderer.

Figure 9: *Brush stroke rendering of an uniformly colored plane model with diffuse lighting.*

5. Results

We have tested our brush stroke stylization methods for augmented reality with numerous different test scenes. Our augmented reality software contains an editor, which is capable of importing 3D models in the standard Wavefront OBJ file format. The user can freely place, scale and rotate the model. Bitmap files can be loaded and applied as textures to the virtual model. The augmented reality display can be switched interactively between the conventional AR and brush stroke modes at any time. Moreover, a graphical user interface for adjusting parameters of the painterly filter and the non-photorealistic rendering module is provided. For all of our test scenes, optical marker tracking based on the ARToolKit framework was used [KB99].

The brush stroke stylization algorithm is demonstrated in Figure 10. Here, conventional AR images are shown above the stylized versions. In the first column, an augmented reality scene containing a virtual teacup with a marble texture is depicted. The second scene contains a bridge model with a painting applied as texture. A wooden plane model is the virtual object in the third column of images.

The goal of our stylization methods 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. Our early experience, the accompanying video, and the images in Figure 10 show that this effect can be achieved by our methods. In particular for scenes in which the scale of the virtual object matches the physical world, the barrier between virtual and real is blurred. The teacup in Fig. 10(d) is a good example for a scene 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 the brush stroke method delivers a frame rate of approximately 13 fps for typical scene generation parameters. This benchmark was performed on a computer with a Pentium 4 processor running at 2.8 GHz using a graphics card with an ATI Radeon 9800 Pro chipset. The measurements show that our stylization algorithm is capable of delivering a video stream at interactive frame rates.

6. Conclusion

We have presented a new type of stylization for augmented reality images. The principle of using non-photorealism in both the virtual and real image layers in AR is a novel approach. The adapted levels of realism for camera image and graphical objects result in improved immersion. This way a more impressive augmented reality experience can be achieved. Applications in art, entertainment, training, and education can benefit from the blurred barrier between the virtual and real worlds.

The drawback of our brush stroke stylization algorithm is the fact that the design of the camera image filter results in the so-called "shower-door" effect. This is caused by the constant positions of the sampling points and the rendered brush strokes. In most non-photorealistic rendering systems, an effort is made to prevent this shower door effect. However, for our scenario of stylized augmented reality, we have found a constant position of the brush strokes to be necessary. As mentioned in Section 4, early experiments with brush strokes attached to the projected positions of model particles were not successful. In videos generated with such an algorithm, the virtual models were clearly distinguishable from the background image. The development of a brush stroke stylization algorithm for augmented reality without the shower door effect is one import trend for future research.

The presented stylization method and our previously published cartoon-like technique have different qualities. While the cartoon-like method works well for scenes containing mostly uniformly colored surfaces, the brush stroke stylization can handle textured objects well. The cartoon style can preserve finer details in the image, provided that they constitute sufficiently large intensity contrasts. One drawback of
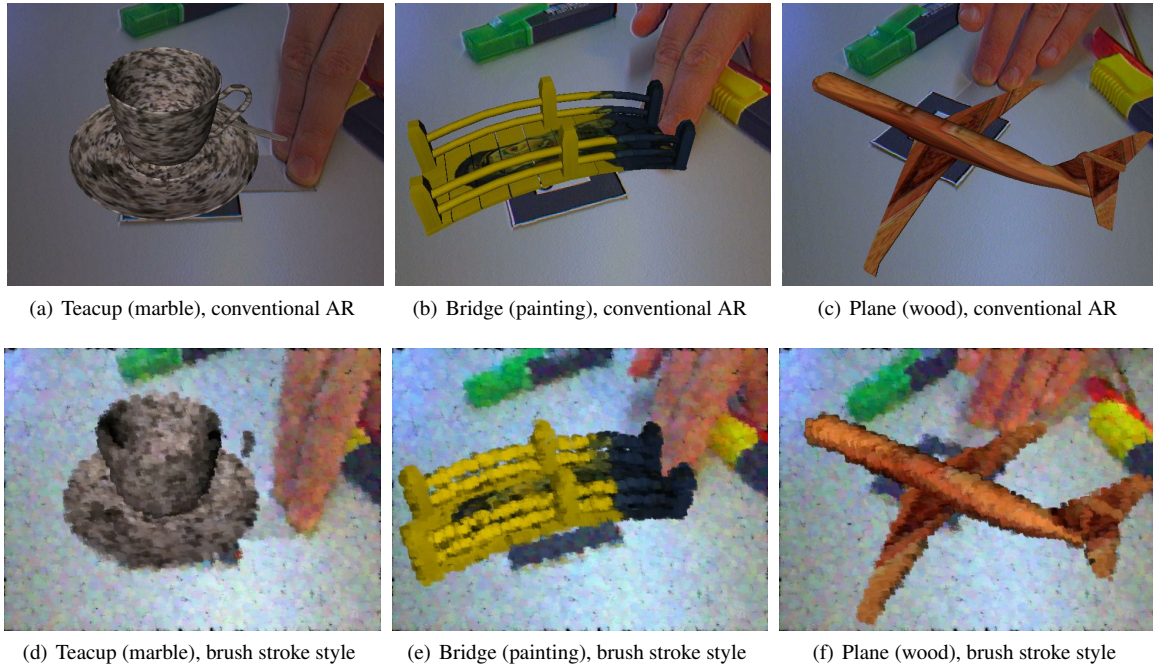
(a) Teacup (marble), conventional AR   (b) Bridge (painting), conventional AR   (c) Plane (wood), conventional AR

(d) Teacup (marble), brush stroke style   (e) Bridge (painting), brush stroke style   (f) Plane (wood), brush stroke style

Figure 10: *Three example scenes demonstrating the brush stroke stylization method presented in this paper. The first row contains conventional augmented reality images. In the second row, the brush stroke stylization of each image is shown. The description of the scenes contains the type of texture applied to the virtual model in brackets. All of the images were rendered in real-time.*

the current implementation of the cartoon-like method is the fact that silhouette lines generated from the camera image tend to flicker. This is caused by noise in the video stream delivered by the webcam. The brush stroke style can handle varying colors on objects better. It is thus more useful for textured virtual models. On the other hand, small details in the image are not preserved, due to the fact that the individual brush strokes can be rather large.

Acknowledgment

References

[3D 04]   3D CAFE: 3D CAFE'S FREE STUFF. http://www.3dcafe.com/, 2004.

[Art04]   ARTCYCLOPEDIA: Artists by Movement: Pointillism.   http://www.artcyclopedia.com/, 2004.

[Azu97]   AZUMA R.: A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments 6*, 4 (1997), 355–385.

[CRL01]   CORNISH D., ROWAN A., LUEBKE D.: View-Dependent Particles for Interactive Non-Photorealistic Rendering.   In *Proceedings of Graphics Interface 2001* (June 2001), pp. 151–158.

[FBS05]   FISCHER J., BARTZ D., STRASSER W.: Stylized Augmented Reality for Improved Immersion.   In *Proceedings of IEEE Virtual Reality* (March 2005).

[FvFH97]   FOLEY J., VAN DAM A., FEINER S., HUGHES J.: *Computer Graphics - Principles and Practice*, 2nd ed. Addison-Wesley Publishing company, 1997.

[Hae90]   HAEBERLI P.: Paint By Numbers: Abstract Image Representations.   In *Proceedings of ACM SIGGRAPH 90* (August 1990), pp. 207–214.

[Hal04]   HALLER M.:   Photorealism or/and Non-Photorealism in Augmented Reality.   In *ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry (VRCAI)* (June 2004), pp. 189–196.

[HDHZ03] HALLER M., DRAB S., HARTMANN W., ZA-UNER J.: A Real-time Shadow Approach for an Augmented Reality Application using Shadow Volumes. In *ACM Symposium on Virtual Reality Software and Technology (VRST)* (October 2003), pp. 56–65.

[HP00] HERTZMANN A., PERLIN K.: Painterly Rendering for Video and Interaction. In *Proceedings of Symposium on Non-Photorealistic Animation and Rendering 2000* (June 2000), pp. 7–12.

[HS99] HEIDRICH W., SEIDEL H.: Realistic, Hardware-accelerated Shading and Lighting. In *Proceedings of ACM SIGGRAPH 99* (August 1999), pp. 171–178.

[HS04] HALLER M., SPERL D.: Real-Time Painterly Rendering for MR Applications. In *International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia, Graphite* (June 2004), pp. 30–38.

[KB99] KATO H., BILLINGHURST M.: 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)* (October 1999), pp. 85–94.

[KLK*00] KLEIN A., LI W., KAZHDAN M., CORREA W., FINKELSTEIN A., FUNKHOUSER T.: Non-Photorealistic Virtual Environments. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), pp. 527–534.

[KY02] KANBARA M., YOKOYA N.: 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)* (September 2002), p. 279.

[Lit97] LITWINOWICZ P.: Processing Image and Video for An Impressionist Effect. In *Proceedings of ACM SIGGRAPH 97* (August 1997).

[Mei96] MEIER B.: Painterly Rendering for Animation. In *Proceedings of ACM SIGGRAPH* (1996), pp. 477–484.

[PZvG00] PFISTER H., ZWICKER M., VAN BAAR J., GROSS M.: Surfels: Surface Elements as Rendering Primitives. In *Proceedings of ACM SIGGRAPH* (2000).

[Rey03] REYNOLDS C.: Stylized Depiction in Computer Graphics - Non-Photorealistic, Painterly and 'Toon Rendering. http://www.red3d.com/cwr/npr, 2003.

[RTF*04] RASKAR R., TAN K., FERIS R., YU J., TURK M.: Non-photorealistic Camera: Depth Edge Detection and Stylized Rendering using Multi-Flash Imaging. In *Proceedings of ACM SIGGRAPH 2004* (Juli 2004), pp. 679–688.

[SS02] STROTHOTTE T., SCHLECHTWEG S.: *Non-Photorealistic Computer Graphics - Modelling, Rendering, and Animation*. Morgan Kaufmann Publishers, 2002.

[Wol04] WOLFRAM RESEARCH: MathWorld: Voronoi Diagram. http://mathworld.wolfram.com/, 2004.

[WXSC04] WANG J., XU Y., SHUM H., COHEN M.: Video Tooning. In *Proceedings of ACM SIGGRAPH* (August 2004), pp. 574–583.

[XC04] XU H., CHEN B.: Stylized Rendering of 3D Scanned Real World Environments. In *Proceedings of Symposium on Non-Photorealistic Animation and Rendering 2004* (June 2004), pp. 25–34.