Vis Comput (2010) 26: 445–455
DOI 10.1007/s00371-010-0436-z

ORIGINAL ARTICLE

Real-time coherent stylization for augmented reality

Shandong Wang · Kangying Cai · Jian Lu · Xuehui Liu · Enhua Wu

Published online: 8 April 2010

Abstract This paper presents a new stylized augmented reality (AR) framework which can generate line drawing and abstracted shading styles. In comparison with the state-of-art work, our framework can significantly improve both the visual immersion of a single frame and the temporal coherence of augmented video streams in real time. In our framework, we first render virtual objects over the input camera images and then uniformly process the combined contents with stylization techniques. For generating line drawing stylization, we first propose a specially designed shading method to render the virtual objects, and then use an adapted Flow-based anisotropic Difference-of-Gaussion (FDoG) filter to yield the high-quality line drawing effect. For generating the abstracted stylization, a focus-guided diffusion filter and a soft color quantization operator are sequentially applied to the augmented image, and then the processed result is combined with the detected edges to produce the final abstraction effect. The presented algorithms are all sympathetic to highly parallel processing, allowing a real-time performance on contemporary graphics hardware.

Keywords Stylization · Augmented reality · Line drawing · Abstraction · Coherent · Real-time

S. Wang (✉) · J. Lu · X. Liu · E. Wu
State Key Lab. of CS, Inst. of Software, Chinese Academy of Sciences, Beijing, China
e-mail: sdwang10@gmail.com

J. Lu
e-mail: zzzdevil@gmail.com

X. Liu
e-mail: lxh@ios.ac.cn

S. Wang
Graduate University of Chinese Academy of Sciences, Beijing, China

K. Cai
Thomson C.R., Beijing, China
e-mail: cai_kangying@hotmail.com

E. Wu
University of Macau, Macau, China
e-mail: EHWu@umac.mo

1 Introduction

Augmented reality is a field of computer research which deals with the combination of physical real-world environment and the virtual computer-generated objects. The ultimate goal of AR technique is to keep the virtual and physical contents visually indistinguishable. The majority of AR systems use conventional photorealistic rendering methods for displaying virtual models. However, due to the lack of knowledge about the actual lighting conditions in the real surroundings, there is often apparent distinction between real and virtual contents.

An efficient way to improve users' immersion of AR scenes is to use nonphotorealistic rending (NPR) technique to create a stylized blending of the virtual and physical worlds. Fischer et al. [10] first presented a stylized AR system which achieves "sketch-like" and "cartoon-like" styles. After that, more NPR styles have been introduced into AR applications, including brush stroke [11] and watercolor-like [4] styles. The stylized AR algorithm with high efficiency has also been developed [9]. Some of the current stylized AR algorithms, such as [11] and [10], use different stylization operators on virtual and real contents, and then simply overlap them. Thus the coherent appearance of the resultant augmented video streams cannot be guaranteed. Other stylized

Fig. 1 Stylization for augmented reality: (**a**) is a conventional AR frame containing a virtual model of sculpture, in which there exist the apparent distinctions between real and virtual contents; (**b**) is the effect of line drawing for AR, which significantly reduce such discrepancies; (**c**) shows the abstraction result and (**d**) is the focus-level of abstraction using the information of real–virtual contents

AR systems, such as [9] and [4], uniformly stylize the combined contents after the virtual objects are rendered on the camera images. Although the visual coherence of a single frame can be improved, these algorithms have not been optimized for improving the temporal coherence. Thus, there is still much room left to improve the coherence of the resultant video streams generated by the stylized AR system.

We present a new stylized AR framework which has been optimized for both the visual immersion of a single frame and the temporal coherence of augmented video streams. In our framework, the stylized AR video streams are generated by first rendering virtual objects over the input camera images and then uniformly processing the combined content. We can generate two stylized AR effects: line drawing and abstraction with focus-level control (as shown in Fig. 1). Both types of stylization can be achieved in real time.

For generating the line drawing style, unlike the previous stylized AR systems which render the virtual objects by standard procedures, we use a specifically designed shading method to guarantee that both the silhouette and the internal boundaries of the virtual objects can be detected in the following steps. Instead of the Canny edge detector [2] used by Fischer et al. [10], we use an adapted FDoG filter [17], which is famous for the smoothness of its edge detection results, as our edge detector. Another benefit of our algorithms is that the detected edges could be nonbinary. This will not only make the line drawing effect of the combined image much more coherent but also increase the temporal coherence in the AR video streams. Thus the quality of our line drawing stylization can be guaranteed.

For generating the abstracted stylization, a focus-guided diffusion filter is applied to the combined contents to blur small discontinuities while preserving the sharpen regions. Besides, the virtual–real information in AR scenes can control the focus of abstraction by using such a diffusion filter. We also propose a soft color quantization step to further deepen the viewers' impression and increase the temporal coherence. The processed result is combined with the detected edges to produce the final abstraction effect. To our knowledge, there is hardly any stylized AR system including the quantization step. Moreover, for the edge detection and image filtering procedure, as we use the same neighboring window, the two key steps can be finished in one rendering pass using multiple render targets (MRT) techniques. Thus the efficiency of our stylized AR algorithm is also guaranteed.

2 Related work

Stylized AR The psychophysical study on the effectiveness of stylized AR has shown that it is significantly more difficult to distinguish virtual objects from real objects [12]. Haller et al. [14] directly combine the computer generated images (by NPR methods) with the real images and thus the visual coherence of the resultant images is not satisfied. Fischer et al. [10] present a stylized AR system with the goal of improving immersion. Specifically, they achieve "sketch-like" style by using the Canny edge detector [2] for line extraction of the camera image and silhouette rendering for the 3D models. The "cartoon-like" style is achieved by using the bilateral filtering [28] of the camera images and nonlinear shading of 3D models. Because the virtual objects and real environments are stylized by different algorithms, some disharmonic parts still exist in these styles. Chen et al. [4] present "watercolor-like" NPR rendering for AR application. In [9], Fischer et al. use a novel post-processing filter, which is implemented on the modern graphics processing unit (GPU), for cartoon-like color segmentation and high-contrast silhouettes. Both these systems first render the virtual objects over the real camera images and then use the same stylization operator to process the combined images. Thus the visual coherence of the AR scenes generated by these systems can be improved. However, they both render the virtual objects by the standard rendering and take no consideration of the temporal coherence of the AR video streams. To our knowledge, the output video streams of most existing stylized AR systems tend to suffer from temporal noise.

Line drawing Line drawing is one of the most popular styles used in NPR applications because it offers effective and natural visual representation of scenes [5]. Countless methods have been studied on the line drawing of 3D models; accordingly, many types of lines come into being: occluding contours [15], ridges and valleys [23, 26], suggestive contours [6, 7], apparent ridges [16], demarcating curves drawn in black [20], highlight lines drawn in white [8, 21] and so on. As most of these line types are too expensive to compute, these methods are not fairly well qualified in the real-time AR applications. There are also many edge detectors in 2D image applications [2, 24, 27]. Gooch et al. [13] present a facial illustration system based on a DoG filter, which is the computationally simple approximation to Marr and Hildreth edge detector [22]. Winnemöller et al. [29] extend this technique using a slightly smoothed step function to increase the temporal coherence. Kang et al. [17] modify the line extraction filter, and propose the flow-based [1] anisotropic DoG (FDoG) filter, which dramatically enhances the spatial coherence of lines and also suppresses noise. However, FDoG edge detector is computationally expensive, and thus it cannot be directly used in real-time AR systems. In order to achieve real-time performance, we optimize the FDoG filter, which allows a GPU-based implementation.

Abstracted shading Abstraction of images and videos is another rendering style that focuses on facilitating visual communication and data reduction. Winnemöller et al. [29] present an automatic, real-time video and image abstraction framework that abstracts imageries by employing the soft DoG filter for line drawing and the separable bilateral filter [25] for region smoothing. Zhao et al. [30] propose a video abstraction system built upon [29], which chooses the edge-aware bilateral grid [3] operator to approximate the anisotropic diffusion filter for acceleration. Kang et al. [18] recently presented a flow-based image abstraction system based on shape or color filtering guided by a vector field that describes the flow of salient features in the image. Specifically, they adopted FDoG filter [17] for extracting lines and flow-based bilateral filter (FBL) for removing all "insignificant" details. Our abstracted shading style for AR system is based upon [18]. However, we implement the algorithms on modern GPUs with MRT technique for real-time performance.

3 Our approach

We propose a new stylized AR framework that automatically generates stylized images for augmented video streams, with the goal of improving the visual immersion and temporal coherence of virtual and real contents. Our framework uses identical nonphotorealistic rendering techniques for both camera images and virtual objects to achieve two different stylized AR effects: line drawing images consisting of pure silhouette lines, and abstracted shading images considering not only outlines of regions but also their interior filled with uniformly colored patches. Figure 1 shows the effects of these stylization types.

An overview of our stylized AR framework is shown in Fig. 2. For line drawing stylization, we apply a special designed shading method to render the virtual objects over the luminance image, which is derived from the CIE-Lab color space of the input camera image. Note that the designed shading result is also a luminance image. Then the combined image is processed using the optimized FDoG filter to generate the coherent line drawing result. For abstraction stylization, we first render the virtual objects over the input camera image using traditional photorealistic rendering methods. Then the combined image is processed by the diffusion and quantization techniques. At last the result is combined with the detected edges to generate the final abstracted shading results. It is worth mentioned that using the mask map achieved from the virtual–real information, our diffusion filter can be diversely applied on the different regions of the combined image.

3.1 Line drawing

Line drawing is a simple and effective tool for shape visualization and visual communication, since it enables quick recognition and appreciation of the subject with little distraction from relatively unimportant contents. The application of this style to AR systems can significantly reduce the discrepancies between virtual objects and real environment, resulting in improving the visual immersion.

In our stylized AR framework, we first render the virtual objects over the video background images in a special designed luminance shading manner. Then the combined images are processed with the line extraction method. We use the following formula for shading the virtual objects:

$$Color = D \cdot |N \cdot V| \cdot L_m \tag{1}$$

where

$$D = 1 - \log\left(\frac{z}{z_{\min}}\right) \Big/ \log\left(\frac{z_{\max}}{z_{\min}}\right).$$

We derive the depth value $D \in [0..1]$ from the computed depth z via two parameters $z_{\min}$ and $z_{\max}$, which are the nearest and farthest distances to the viewpoint respectively. N and V are the unit normal and view vector. L_m indicates the luminance of the surface material or texture of the virtual objects. The reason that we choose this shading method instead of other photorealistic renderings is that it cannot only highlight view-dependent silhouettes and strong feature lines of the objects in terms of geometry properties, but also reveal the boundaries between smooth regions with different materials or textures.

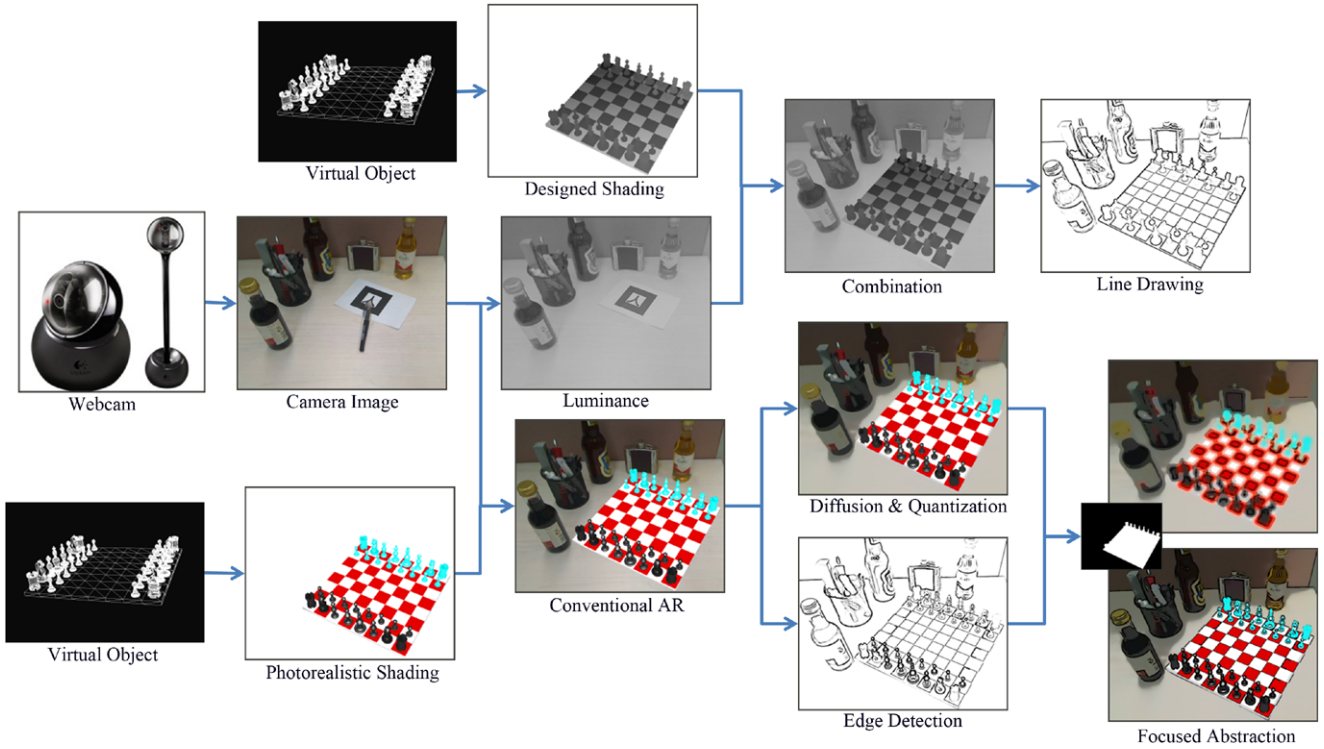

Fig. 2 Our stylized AR framework

Our solution to line extraction is based on the flow-based anisotropic filtering framework introduced by Kang et al. [17]: we first construct the smooth edge flow field from the input image by a kernel-based nonlinear vector-smoothing technique, and then use the anisotropic DoG filter to extract lines. Because Kang's method is too computationally expensive to be directly used in our framework, we design a modified approach that not only can run on modern GPUs for a real-time performance, but also can achieve smooth lines with much coherence.

Given an input image $I(x)$, $t(x)$ denotes the normalized edge direction of a pixel x, which is a vector perpendicular to its gradient $g(x) = \nabla I(x)$. The edge flow construction filter is defined as follows:

$$t^{\text{new}}(x) = \frac{1}{k} \iint_{\Omega(x)} \omega_m(x, y)\omega_d(x, y)t^{\text{cur}}(y)\, dy \tag{2}$$

where $\Omega(x)$ represents the neighboring pixels of x, and k is the vector-normalizing term. The magnitude weight function $\omega_m(\cdot)$ is defined as

$$\omega_m(x, y) = \frac{1}{2}\left(1 + \left[\hat{g}(y) - \hat{g}(x)\right]\right)$$

where $\hat{g}(z)$ denotes the normalized gradient magnitude at z. Note that this weight function monotonically increases with respect to the magnitude difference $\hat{g}(y) - \hat{g}(x)$, and thus can ensure the preservation of the dominant edge direction. The other parameter ω_d is called the direction weight function:

$$\omega_d(x, y) = t^{\text{cur}}(x) \cdot t^{\text{cur}}(y)$$

with $t^{\text{cur}}(x)$, the 'current' edge direction vector at pixel x. This weight function induces tighter alignment of edges with close direction while avoiding swirling flows [17].

Note that (2) is computationally expensive and cannot be directly used in our system. Inspired by the idea of separable bilateral filtering [25], we accelerate the edge flow construction by iteratively applying the following 1D filters:

$$t^{\text{new}'}(x) = \frac{1}{k_t} \sum_{y \in T(x)} t^{\text{cur}}(y)\omega_m(x, y)\omega_d(x, y), \tag{3}$$

$$t^{\text{new}}(x) = \frac{1}{k_g} \sum_{y \in G(x)} t^{\text{new}'}(y)\omega_m(x, y)\omega_d^{\text{new}'}(x, y). \tag{4}$$

Here $T(x)$ and $G(x)$ denote the neighboring pixels in the edge and gradient directions respectively; k_t and k_g are corresponding normalizing terms.

Instead of separately accelerating in x and y dimensions [18], we conduct two 1D nonlinear vector-smoothing operations, one along the edge direction and the other along the gradient direction, to reduce the time consumption without noticeable artifacts. The edge flow field is updated by iteratively applied (3) and (4). In this processing step, gradient $g(x)$ evolves accordingly but the normalized magnitude $\hat{g}(x)$ is unchanged. The initial normalized edge vector

$t_0^{\mathrm{cur}}(x)$ is obtained by taking perpendicular vector from the initial gradient vector, which is computed by the Sobel operator in our framework.

After the edge flow field is constructed, we detect edges by applying an anisotropic DoG filter whose kernel shape is defined by the edge flow. We first apply a 1D linear DoG filter $F(\cdot)$ only in the gradient direction of each pixel, and then accumulate the individual filter responses along the edge flow by the filter $H(\cdot)$, as a way of collecting enough edgeness information. Rather than using a black-and-white image, we define our edge operator $D(\cdot)$ using a slightly smoothed step function [29] to increase temporal coherence in video streams:

$$D(x,\varphi_e,\sigma_m,\sigma_c,\sigma_s,\tau) = \begin{cases} 1.0, & H(x) \geq 0, \\ 1.0 + \tanh(\varphi_e \cdot H(x)), & \text{otherwise} \end{cases} \tag{5}$$

where

$$H(x) = \frac{1}{k_h} \sum_{s \in T_s} G_{\sigma_m}\big(\|s-x\|\big) F(s), \tag{6}$$

$$F(s) = \frac{1}{k_{f1}} \sum_{t \in G_t} G_{\sigma_c}\big(\|t-s\|\big) I(t) - \tau \frac{1}{k_{f2}} \sum_{t \in G_t} G_{\sigma_s}\big(\|t-s\|\big) I(t). \tag{7}$$

In the above formulas, T_s and G_t denote the neighborhoods along the edge flow curve and the gradient direction respectively, as illustrated in Fig. 3. $G_\sigma(\|x-y\|) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\|x-y\|^2}{2\sigma^2}}$ is Gaussian function, with $\|x-y\|$ representing the 1D distance between pixels x and y. Here we set $\sigma_s = 1.6\sigma_c$ to let (7) be an approximation to Laplacian-of-Gaussian [22]. Another parameter σ_m controls the length of the elongated flow kernel and also the degree of line coherence to enforce. The parameter τ determines the sensitivity of the edge detector, and φ_e controls the sharpness of edge representations. Once σ_c and σ_m are given by the user, it automatically determines the size of G_t and T_s.

Intuitively, our algorithm for edge detection can be implemented on GPU through two rendering passes in the fragment shader. In the first pass, (7) is computed for each pixel, and next pass is performing (6). In fact, this process can also be iteratively implemented to improve the line coherence. Specifically, after each performance of (5), we re-initialize the filter input by superimposing the edge pixels ($D(\cdot) < 1.0$) upon the original image $I(\cdot)$ and then using this combined image as input to re-compute (5) (the edge flow field remains unchanged). Figure 4 shows the line drawing effect of an AR video frame with the virtual model of a monster.

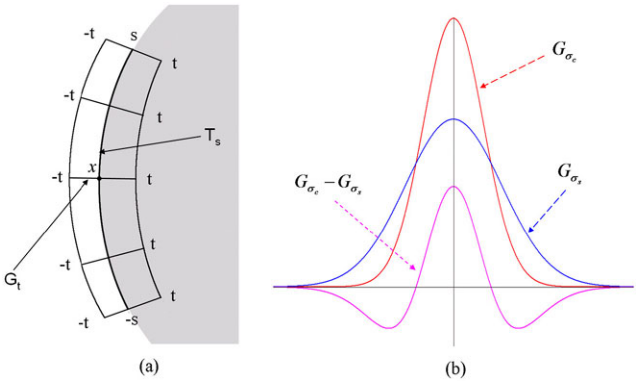

Fig. 3 (**a**) Kernel for constructing the edge flow field. (**b**) Difference of Gaussians (DoG) used for edge detection

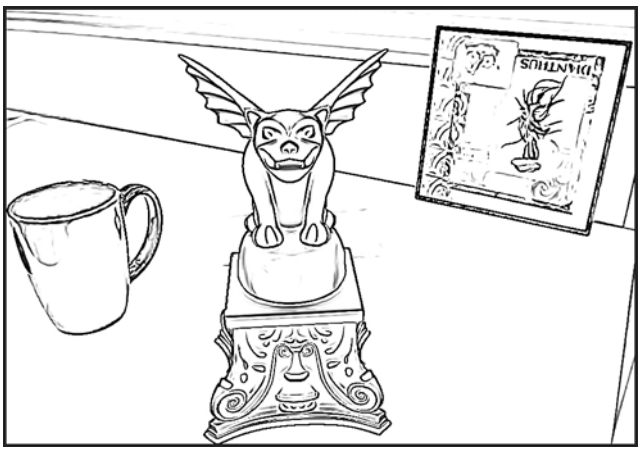

Fig. 4 Line drawing effect for an AR frame with a virtual monster object

3.2 Abstracted shading

Besides pure line drawing, the abstracted effect can also improve immersion in AR systems. The aim of abstraction is to convey important visual cues to the viewer by simplifying regions of low contrast while enhancing high contrast regions. Accordingly, image and video abstraction usually contains two steps: image diffusion and edge detection. In our stylized AR framework, we propose a focus-guided diffusion filter to blur small discontinuities while preserving the sharp regions. The edge detection step is implemented by the method introduced in Sect. 3.1.

For an input image $I(x)$ and an output image $O(x)$, which are already mapped into CIE-Lab color space, our focus-guided diffusion filter is constructed by the following formula:

$$O(\hat{x},\sigma_d,\sigma_r) = \frac{1}{k} \iint_{\Omega(\hat{x})} \big(C - G_{\sigma_d}\big(\|x-\hat{x}\|\big)\big) \omega(x,\hat{x},\sigma_r) I(x)\, dx \tag{8}$$

where

$$\omega\big(x,\hat{x},\sigma_r\big) = \big(1-m(\hat{x})\big)G_{\sigma_r}\big(\|I(x)-I(\hat{x})\|\big)+m(\hat{x}). \tag{9}$$

In the above two equations, $\hat{x}$ is a pixel in the output image $O(\cdot)$; $\Omega(\hat{x})$ denotes the set of spatially neighboring pixels of $\hat{x}$ and x is any pixel belonging to $\Omega(\hat{x})$; k is the normalization factor. Gaussian function $G_{\sigma_d}(\|x-\hat{x}\|) = e^{-\frac{\|x-\hat{x}\|^2}{2(\sigma_d)^2}}$ is the weighting factor of the spatial distance between pixel locations while $G_{\sigma_r}(\|I(x)-I(\hat{x})\|) = e^{-\frac{\|I(x)-I(\hat{x})\|^2}{2(\sigma_r)^2}}$ is used to weight the photometric similarity between nearby pixels. Here we define $m(\cdot)$ as the focus-control map, which determines how the input image is diffused by applying (8).

Our definition of (8) extends the well-known bilateral filter to become more customizable for focus-level control. If we set $m(\cdot)=1.0$, (8) becomes an inverted Gaussian blur filter. $C-G_{\sigma_d}(\|x-\hat{x}\|)$ is the spatial weight, where C is a user-defined constant. In contrast to the traditional Gaussian blur that assigns heavier weights to the nearer pixels, this inverted form can ensure that the further pixels are assigned heavier weights than the nearer ones. As a result, it is easy to eliminate the single spot noise and get more blurred result. For another case of $m(\cdot)=0.0$, (8) becomes a bilateral filter by a spatial weight in the form of inverted Gaussian [31]. Compared with the standard bilateral filter, this Inverted-Gaussian-Spatial Bilateral filter not only can preserve the distinct edges but also can remove small spots in large smooth areas more efficiently than the classical bilateral filter.

In our application, it is easy to create a focus-control map $m(\cdot)$ by means of the virtual–real content information. In an augmented image, we can either set $m(\cdot)=0.0$ for the regions where the virtual objects are and $m(\cdot)=1.0$ for the other regions of the image, or vice versa. Therefore it is essential to be able to focus on a given zone of the image, and efficiently grab visual attention.

In order to achieve a real-time performance, here we still assume that (8) could be approximated by 1D filters separately, without strict mathematical proving as in [18, 25]:

$$O_g(\hat{x},\sigma_{dg},\sigma_{rg}) = \frac{1}{k_g}\sum_{x\in G_t}\big(C-G_{\sigma_{dg}}\big(\|x-\hat{x}\|\big)\big)\omega(x,\hat{x},\sigma_{rg})I(x)\,dx, \tag{10}$$

$$O_t(\hat{x},\sigma_{dt},\sigma_{rt}) = \frac{1}{k_t}\sum_{x\in T_s}\big(C-G_{\sigma_{dt}}\big(\|x-\hat{x}\|\big)\big)\omega(x,\hat{x},\sigma_{rt})I(x)\,dx, \tag{11}$$

where $\frac{1}{k_g}$ and $\frac{1}{k_t}$ represent corresponding weight normalization terms. T_s and G_t denote neighborhoods along the edge flow direction and the perpendicular (gradient) direction (as shown in Fig. 3).

The goal of region smoothing can be achieved by alternating O_g and O_t in an iterative fashion. Fortunately, the results show that this separable version works well. Few visual artifacts can be noticed in most cases. More importantly, because of the same convolution kernel, with the help of MRT technique, we can perform edge operator (see (6), (7)) and diffusion filter (see (10), (11)) in the same rendering pass.

To further deepen the viewers' appreciation of the image and improve the temporal coherence of the generated video streams, we perform a soft color quantization step on the above diffused images. First, we define an emphasis function $E:[0,1]\to[0,1]$ by

$$E(x,\beta) = \frac{x}{e^{\beta}(1-x)+x}. \tag{12}$$

The emphasis function E is used to exaggerate or reduce local variations via a user-defined parameter β (as shown in Fig. 5). Then the quantization function can be defined:

$$Q(\hat{x}) = \big[q_n + E\big(q_r,(-1)^{q_n}\beta\big)\big]\Delta q, \tag{13}$$

where $Q(\hat{x})$ is the quantized result of the input diffused image $O\hat{x}$). In (13), Δq is the quantum width, $q_n = \lfloor\frac{O(\hat{x})}{\Delta q}\rfloor$ is the nearest floor boundary and $q_r = \mathrm{frac}(\frac{O(\hat{x})}{\Delta q})$ is the fractional portion. In comparison with the discontinuous function used by [29], (13) is formally a continuous one, which can make the transition sharpness in large smooth regions less noticeable. Thus much more coherent abstraction effect can be provided. Figure 6 shows the abstracted shading results.

4 Experimental results and discussion

We have developed our stylized AR framework based on the ARToolKit framework [19], which tracks the user viewpoint by means of calculating the real camera position and orientation relative to physical markers in real time. Our experimental environment includes a PC with a 2.66 GHz Intel Core 2 Q9400 CPU and an NVIDIA Geforce GTX 285 GPU and a Logitech webcam. The algorithms are implemented using OpenGL shading language and Cg shader programs to accelerate the rendering in the GPU. Most computations of the algorithms are performed in fragment shaders with the help of Frame Buffer Object (FBO) technique. The maximal video resolution is 960×720 and the corresponding frame rate is 15 fps. For the resolution of 640×480, the rate is more than 30 fps. This proves that our framework can achieve a real-time performance.

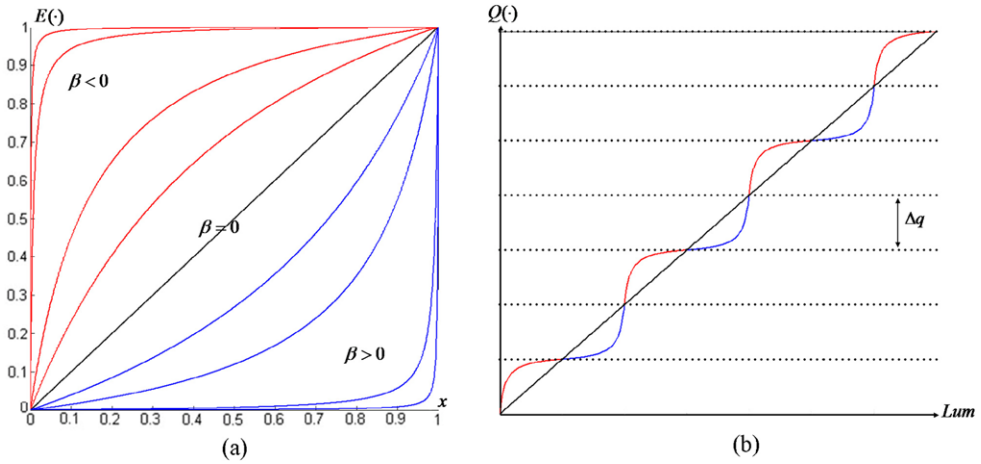

Fig. 5 (**a**) The emphasis function is to exaggerate ($\beta < 0$) or reduce ($\beta > 0$) local variations. (**b**) Quantization function based on the emphasis function

Fig. 6 Focus-guided abstraction results. (**a**) The default abstraction ($m(\cdot) = 0$ for everywhere) of the original AR image with the same level of abstraction everywhere. (**b**) (**c**) The abstraction results with different focus-levels, which are guided by the masks shown in the *bottom right corners*. The masks for generating (**b**) and (**c**) are calculated based on the real–virtual information. The *black parts* on the masks correspond to the clearer parts of the abstracted images

We have implemented our new stylized AR algorithms and the related algorithms in [10] and [4] for comparison. Figure 7 lists the stylization results generated by these algorithms. In Fig. 7, (b) and (c), the edges of the virtual model are much smoother and cleaner than those of the other regions. Thus the visual coherence is not satisfied. In comparison with the other methods, our algorithms can extract a set of cleaner, smoother and more coherent lines for both virtual and real contents, see Fig. 7(d). Moreover, our method can convey more important feature lines inside the rendered virtual objects, which are missing in Fig. 7, (b) and (c).

Figure 7(e) shows the "cartoon-like" stylization effect by [10]. And Fig. 7(f) shows the abstracted shading effect by our algorithm. Then it is clear that our method can produce much more coherent abstraction effect. Besides the better line drawing effect, the visual coherence of our abstracted shading effect is also guaranteed by the same diffusion and soft color quantization steps applied to the whole combined image. On the contrary, [10] applied the bilateral filter to the camera image parts and the traditional hard color quantization to the 3D objects rendering, respectively. Thus the coherence of its result is not satisfied, as shown in Fig. 7(e). Figure 7(g) provides the focus-controlled abstraction of our method, in which it focuses primarily on the regions of the virtual statue. Figure 7(h) is another abstraction style by our method, where the appearance of the image becomes much blurrier using the inverted Gaussian filter.

Figure 8 shows more stylized AR images generated by our algorithms. These results demonstrate that our stylized AR framework can greatly improve the visual immersion of a single AR frame. As we use the slightly smoothed edge representation and the soft color quantization operation, the temporal coherence of the stylized AR video streams can also be guaranteed (see the video attachment).

5 Conclusion

A new stylized AR framework is proposed in this paper for further improving the immersion of AR applications. We show how to use our framework to generate line drawing and abstracted shading styles for AR systems. An adapted FDoG edge detector and a focus-guided diffusion filter are included in our framework to improve both the visual immersion of a single frame and the temporal coherence of the augmented video streams. Instead of the standard photorealistic rendering methods used by most existing stylized

Fig. 7 Comparison with other stylized AR techniques. (**a**) Conventional AR scene, in which the statue of a woman is virtual. (**b**) "Sketch-like" stylization by [10]. (**c**) Edge detection by [4]. (**d**) Line drawing result by our method. (**e**) "Cartoon-like" stylization by [10]. (**f**) Default abstraction by our method. (**g**) Focus-guided abstraction by our method. (**h**) Nonfocused abstraction by our method

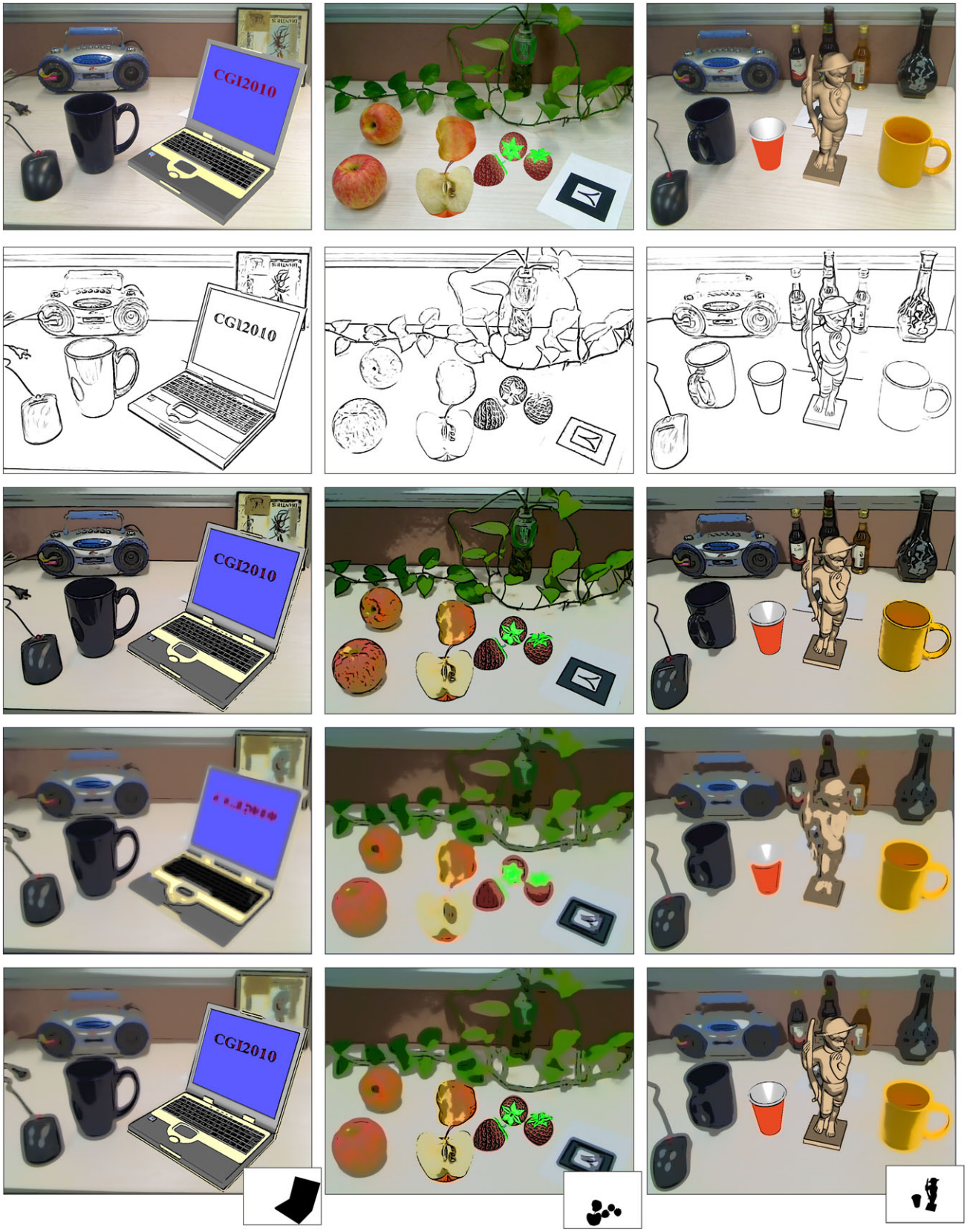

Fig. 8 More screenshots from our stylized AR framework. The *first row* displays three different conventional AR scenes. The *second row* shows the corresponding line drawing results. The *next row* shows the default abstraction results. The *fourth row* gives the abstraction with the inverted Gaussian blur filter and the *last row* shows the focus-guided abstraction by the mask maps in *lower right corners*

AR systems, our framework uses a special designed shading for displaying the virtual objects, which can convey some important feature lines using our edge detection algorithm. Furthermore, our focus-guided diffusion filter and the soft color quantization operation are able to achieve the meaningful focus-level of abstraction and grab visual attention efficiently. In addition, because of using the same convolution kernel, the two key steps in abstracted shading, edge detection and diffusion can be finished in one pass by using MRT technique. Thus our framework allows a real-time performance on modern graphics hardware. As AR applications are quite popular nowadays, especially on mobile devices, it is interesting to test our stylized framework on mobile platforms.

Acknowledgements We would like to thank Weiliang Meng and Kai Bao for their help in typesetting the paper draft. This research is supported by National Fundamental Research Grant of Science and Technology (973 Project: 2009CB320802), Science and Technology Development 863 project (2008AA01Z301) and research grant of University of Macau.

References

1. Cabral, B., Leedom, L.C.: Imaging vector fields using line integral convolution. In: SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, pp. 263–270. ACM, New York (1993)
2. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**(6), 679–698 (1986)
3. Chen, J., Paris, S., Durand, F.: Real-time edge-aware image processing with the bilateral grid. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers, p. 103. ACM, New York (2007)
4. Chen, J., Turk, G., MacIntyre, B.: Watercolor inspired non-photorealistic rendering for augmented reality. In: VRST '08: Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology, pp. 231–234. ACM, New York (2008)
5. Cole, F., Sanik, K., DeCarlo, D., Finkelstein, A., Funkhouser, T., Rusinkiewicz, S., Singh, M.: How well do line drawings depict shape?. In: SIGGRAPH '09: ACM SIGGRAPH 2009 papers, pp. 1–9. ACM, New York (2009)
6. DeCarlo, D., Finkelstein, A., Rusinkiewicz, S.: Interactive rendering of suggestive contours with temporal coherence. In: NPAR '04: Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering, pp. 15–145. ACM, New York (2004)
7. DeCarlo, D., Finkelstein, A., Rusinkiewicz, S., Santella, A.: Suggestive contours for conveying shape. In: SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, pp. 848–855. ACM, New York (2003)
8. DeCarlo, D., Rusinkiewicz, S.: Highlight lines for conveying shape. In: NPAR '07: Proceedings of the 5th International Symposium on Non-photorealistic Animation and Rendering, pp. 63–70. ACM, New York (2007)
9. Fischer, J., Bartz, D.: Real-time cartoon-like stylization of AR video streams on the GPU. Technical Report WSI-2005-18, Wilhelm Schickard Institute for Computer Science, University of Tübingen (2005)
10. Fischer, J., Bartz, D.: Stylized augmented reality for improved immersion. In: VR '05: Proceedings of the 2005 IEEE Conference on Virtual Reality, pp. 195–202, 325. IEEE Computer Society, Washington (2005)
11. Fischer, J., Bartz, D., Straßer, W.: Artistic reality: Fast brush stroke stylization for augmented reality. In: VRST '05: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, pp. 155–158. ACM, New York (2005)
12. Fischer, J., Cunningham, D., Bartz, D., Wallraven, C., Bülthoff, H., Strasser, W.: Measuring the discernability of virtual objects in conventional and stylized augmented reality. In: Eurographics Symposium on Virtual Environments (EGVE), pp. 53–61 (2006)
13. Gooch, B., Reinhard, E., Gooch, A.: Human facial illustrations: Creation and psychophysical evaluation. ACM Trans. Graph. **23**(1), 27–44 (2004)
14. Haller, M.: 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, pp. 189–196. ACM, New York (2004)
15. Hertzmann, A., Zorin, D.: Illustrating smooth surfaces. In: SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 517–526. ACM/Addison-Wesley, New York (2000)
16. Judd, T., Durand, F., Adelson, E.H.: Apparent ridges for line drawing. ACM Trans. Graph. **26**(3), 19 (2007)
17. Kang, H., Lee, S., Chui, C.K.: Coherent line drawing. In: NPAR '07: Proceedings of the 5th International Symposium on Non-Photorealistic Animation and Rendering, pp. 43–50. ACM, New York (2007)
18. Kang, H., Lee, S., Chui, C.K.: Flow-based image abstraction. IEEE Trans. Vis. Comput. Graph. **15**(1), 62–76 (2009)
19. Kato, H., Billinghurst, M.: Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: IWAR '99: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality, p. 85 (1999)
20. Kolomenkin, M., Shimshoni, I., Tal, A.: Demarcating curves for shape illustration. ACM Trans. Graph. **27**(5), 157 (2008)
21. Lee, Y., Markosian, L., Lee, S., Hughes, J.F.: Line drawings via abstracted shading. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers, p. 18. ACM, New York (2007)
22. Marr, D., Hildreth, E.: Theory of edge detection. Proc. R. Soc. Lond. B, Biol. Sci. **207**(1167), 187–217 (1980)
23. Ohtake, Y., Belyaev, A.G., Seidel, H.P.: Ridge-valley lines on meshes via implicit surface fitting. ACM Trans. Graph. **23**(3), 609–612 (2004)
24. Orzan, A., Bousseau, A., Barla, P., Thollot, J.: Structure-preserving manipulation of photographs. In: NPAR '07: Proceedings of the 5th International Symposium on Non-Photorealistic Animation and Rendering, pp. 103–110. ACM, New York (2007)
25. Pham, T.Q., van Vliet, L.J.: Separable bilateral filtering for fast video preprocessing. In: ICME, pp. 454–457 (2005)
26. Saito, T., Takahashi, T.: Comprehensible rendering of 3-d shapes. In: SIGGRAPH '90: Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques, pp. 197–206. ACM, New York (1990)
27. Son, M., Kang, H., Lee, Y., Lee, S.: Abstract line drawings from 2d images. In: PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications, pp. 333–342. IEEE Computer Society, Washington (2007)
28. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: ICCV '98: Proceedings of the Sixth International Conference on Computer Vision, pp. 839–846. IEEE Computer Society, Washington (1998)
29. Winnemöller, H., Olsen, S.C., Gooch, B.: Real-time video abstraction. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers, pp. 1221–1226. ACM, New York (2006)
30. Zhao, H., Jin, X., Shen, J., Mao, X., Feng, J.: Real-time feature-aware video abstraction. Vis. Comput. **24**(7–9), 727–734 (2008)

31. Zhu, L., Wang, C., Zhu, G., Han, B., Wang, H., Huang, P., Wu, E.: Image spatial diffusion on GPUs. In: IEEE Asia Pacific Conference on Circuits and Systems (APCCAS 2008), pp. 610–613 (2008)

Shandong Wang received his B.E. degree in Computer Science from Xidian University in 2007. He is currently a Ph.D. candidate in the State Key Laboratory of Computer Science at Institute of Software, Chinese Academy of Sciences. His research interests include stylized rendering, virtual and augmented reality.

Kangying Cai received her Ph.D. degree in 2006 from the Institute of Software, Chinese Academy of Sciences. After that, she has worked in Ritsumeikan University, Japan as a Postdoc Research Fellow. Since 2008, she is a Research Engineer in Thomson Cooperate Research Beijing. Her research interests include geometry processing, mixed reality and nonphotorealistic rendering.

Jian Lu is presently a Master candidate in the State Key Laboratory of Computer Science of Institute of Software, Chinese Academy of Sciences. His research interest field is mainly about nonphotorealistic rendering, especially rendering of line-drawing from 3D models.

Xuehui Liu received her B.Sc. and M.Sc. degrees from Xiang Tan University, Hunan, and received her Ph.D. degree in 1998 from the Institute of Software, Chinese Academy of Sciences. Since then she has been working at the Institute of Software, Chinese Academy of Sciences, and is now an Associate Professor. Her research interests include realistic image synthesis and animation.

Enhua Wu received his B.Sc. in 1970 from Tsinghua University, Beijing, and his Ph.D. in 1984 from University of Manchester, UK. Since 1985 he has been working at Institute of Software, Chinese Academy of Sciences, and from 1997 he has been also teaching in University of Macau. He is a member of IEEE & ACM. His research interests include realistic image synthesis, virtual reality, physically based modeling and scientific visualization.