Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, Kunming, 12-15 July 2008

GPU BASED VIDEO STYLIZATION

YANG ZHAO[1, 2], DENG-EN XIE[1], DAN XU[1]

[1]Department of Computer Science, Yunnan University, Kunming, 650091, China
[2]Department of Computer Science, Yunnan Normal University, Kunming, 650092, China
E-MAIL: bootcool@163.com, xde820@gmail.com.cn, danxu@ynu.edu.cn

Abstract:

In this paper, we present a GPU based video stylization framework that can artistically stylize video stream in real time. In this framework, firstly, we use a separable implementation of bilateral filter as an adaptive and iterative smoothing operation that selectively simplifies image color, leading to an abstracted look. Secondly, we perform a soft color quantization step on the abstracted video. A significant advantage of the soft color quantization implementation is preserving temporal coherence and reducing computation time as well. Successively, some optional approaches are designed to generate different artistic styles. We evaluate the effectiveness of our stylization framework with the experiment results.

Keywords:

GPU; Video stylization; NPR;

1. Introduction

Making the image and video to be more artistic and abstract, have become an indispensable component in contemporary films, special effects, computer generated animation, computer games and visual communication. Efficient and various methods for performing such works have been recently introduced to the computer graphics community.

To illustrate just how valuable image and video stylization and abstraction can be, consider the shot from the recent feature film A Scanner Darkly, where every frame of this film has been rendered as an oil-paint effect. It would have been extremely difficult, if not impossible, for an animator to achieve such detailed and precise scenes and motion by manually drawing from frame to frame.

DeCarlo [1] described a computational approach to stylize and abstract photographs that explicitly responds to specific design goal. Their system can stylized the meaningful elements of the image by using a model of human perception and a record of a user's eye movements in looking at the photo. But this system could only stylize a image or photograph and can not stylize and abstract a fragment of video, at the same time his methods should use some specific physical equipment , this make the system can not put in widely use in the market.

In a recent pioneering work, Jue Wang [2] described a system for transforming an input video into a highly abstracted, spatio-temporally coherent cartoon animation with a range of styles. While this method has produced some impressive results, it is highly computationally intensive, because it must use an anisotropic kernel mean shift technique which they have developed to segment the video data into contiguous volumes.

Jan Fischer [3] present a new method for the cartoon-like stylization of augmented reality images, which uses a novel post-processing filter for cartoon-like color segmentation and high-contrast silhouettes. In order to make a fast post processing of rendered images possible, the programmability of modern graphics hardware is exploited. Recently Holger Winnem¨oller [4] has presented a real-time video and image abstraction framework that abstracts imagery by modifying the contrast of visually important features. The main contributions of their work is that the framework they designed is highly parallel, allowing for real-time implementation.

Although both of them have proposed some methods to fast transform an input video into a highly abstracted, the style of the generated video is only limited at Cartoon-like, their system can not produced various and different artistic styles in a general and uniform framework, which is required in a feasible stylization system.

Inspired by their works, we also focus on video stylization, but explore a different rendering framework: GPU (Graphics Process Unit) based video artistic stylization. Rather than attempting to design a complex antistrophic image segmentation algorithm like [2], we present an automatic, real-time framework that simplifies imagery or video stream by using bilateral anisotropic filter. Our methods are very similar to [4]. Base on this method we have extend our technique to real-time generate various and different artistic styles by simply inputting an image or video streams into our system and changing some

978-1-4244-2096-4/08/$25.00 ©2008 IEEE

parameters of our system.

Specifically, our main contributions are: (i) we achieve real-time performance, which is essential for its applications. The complete stylized system is capable of generating an output video stream at real-time frame rates, so it can make an application in interactive scenarios possible; (ii)we borrow the idea of [3],[4] ,so by using the soft color quantization we can insure the temporal coherence of the output video; (iii)our system can automatic produce various and different artistic styles, such as cartoon-like, watercolor, pencil drawing, in a general framework which can satisfy different user's needs.

The remainder of this paper is structured as follows. In the next section we will derive the framework of our real-time stylization system and the details of our algorithm and how to implement our algorithm on the current GPU (Graphics Process Unit) hardware. Section 4 describes the extensions and applications of our work. We present some of the experiment results in section 5. Section 6 concludes the paper.

2. System overview

Figure 1 is the basic architecture of our real-time video stylization system. At the high level, the system processes in three steps. In the first step, we load the original video frame from the local frame buffer memory. A non-linear filter (bilateral filter) is then applied to the input video, which is executed by the graphics processing unit (GPU). In order to achieve sufficiently good color simplification and abstraction, several filtering iterations are consecutively applied to the frame. In the second step, we perform a soft color quantization on the abstracted video, which can result in abstracted effects. The second step is also executed by the GPU) to quicken the computational speed. In the final step, we design an optional way to generate different artistic styles in a common framework. By using other NPR filter methods in the framework we can create different artistic stylization with different parameters given by users.

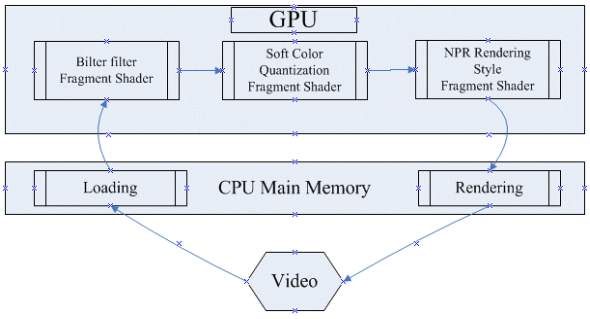


Figure 1. Basic architecture of our real-time video stylization system.

3. Generation of abstracted color image

To generate a stylized and abstracted color image we should do following steps. Firstly, the original frame is rendered into the local frame buffer of the graphics card. Secondly, the non-linear filter is then applied iteratively by using the output image of the last iteration as input texture for the next filtering step so that a smoothly scaled and abstracted version of the image is produced.

3.1. GPU based bilateral filter

Our non-linear filter is inspired by bilateral filtering, which became more important over the last years. It was developed by Tomasi as an alternative to anisotropic diffusion [5].The core of our GPU based implementation of bilateral filter is an adaptive and iterative smoothing operation that selectively simplifies image color, leading to a stylized and abstracted look. Iterated bilateral filtering is adaptive smoothers. It is similar to a mean-shift filter that moves pixels values towards local modes [5]. However, unlike mean-shift, adaptive smoothing requires only local information, and is thus better suitable for implementation on graphics hardware.

In practice, we use a separable implementation of the bilateral filter which was designed by Tuan Q. Pham [6]. The separable implementation offers equivalent adaptive filtering capability at a fraction of execution time compared to the traditional filter and it is easy to hardware implementing. A one-dimensional bilateral filter is applied to the first dimension and the intermediate result is filtered again in subsequent dimensions. In this way, the computational complexity of the separable implementation is just compared to for a full kernel implementation. To achieve real-time effects and obtain an easy implementing on the GPU, we create an adaptive smoothing effect by iteratively applying the separable approximation to a bilateral filter.

In our implementation we exploit the programmable features of the current graphics hardware such as fragment processing unit. These features make the graphics hardware very flexible and allow implementing even complex filters efficiently. We write a fragment by HLSL to implement separable bilateral in a straightforward approach. The implementation uses two registers for accumulation. One contains sum of the weights computed by multiplication of the geometric and photometric weight. The other sums the weight multiplied by corresponding neighboring pixel. The final output value is computed as a division between the two registers to normalize the accumulated color. We perform more iterations and passes to increase the

smoothing and stylized effect. Our implementation is easy to understand and gain highly efficiency .The filtering performs efficiently, because there is no conditional code at all.

Jan Fischer disregarded the geometric weight and modified the filter so that the photometric weight only depends on the actual color of each pixel [3]. In our experiments, we find that to get better stylized and abstracted fact, we should also take geometric weight into account, and this will only need some more computation complexity .In addition, in his algorithm, each pixel is converted into the YUV color space before the filter is applied. Paper [4] has done their non-liner filter in CIELab color space. However, we directly perform our filter in RGB color space .We found that it can also get the similar effect like doing in the YUV and the CIELab color space. Filtering directly in RGB color space, we should not do some conversion between different feather spaces, and thus improving the processing speed.

3.2. GPU based soft color quantization

Jue Wang [2] have described a system for transforming an input video into a highly abstracted, spatio-temporally coherent cartoon animation by using an anisotropic kernel mean shift technique they have developed to segment the 2D image or video data into contiguous volumes. But this method is highly computationally intensive and the anisotropic kernel mean shift algorithm is hard to implement on the current graphics hardware.

In Jan Fischer‘s paper, they have designed a cartoon-like non-linear shading method, which was also described by Lander [7], generates a limited and well-defined set of intensities. But their method only fit to 3D model (the system maintains normalized ni and L), it is hard to use in the 2D space for image quantization or segmentation.

Our soft color quantization concept is similar to the Holger Winnem¨oller‘s idea. The main contribution is that we find another quantization function named sigmoid which reduces the computation complexity. In their paper they used tanh function to control the sharpness of the transition from one bin to another. But in HLSL there is no tanh function ,in addition the tanh function should uses 4 times exp function to get the value, to get the similar effect the sigmoid function only uses 1 time exp function. Thus, it is simpler and easier to implement on the GPU. The soft image quantization procedure is defined as fallows:

$$Qi(x) = Bin_n + bin_w(\frac{1}{1+\exp(-f(x)+Bin_n)}) \qquad (1)$$

Where $Q_i(x)$ is the pseudo-quantized image; bin_w is the bin width; Bin_n is the bin boundary closest to f(x).

Because of human’s vision is most sensitive to the change of luminance of image, it is critical to quantize the luminance feature of the input image or video and that the color features of the input image or video should be reserved. In our algorithm, each pixel is converted into the YIQ color space before the soft color quantization is applied. The Y component represents the brightness of a pixel, while I and Q are the chrominance (color) components. For computing the pseudo-quantized image, our quantization procedure only takes the Y coordinates into account.

4. Simulate other artistic effects

In this section, we show how our algorithm can be configured to serve for a sequence of applications by using our NPR filtering methods. One attractive feature of our model is that it allows the incorporation of any kind of different artistic styles into our framework. The software Adobe Photoshop offers a lot of filter tools for such stylized image processing. In general, these tools can be categorized into those that make use of only different components of the source image including emboss, halftone pattern, Sharpen tool and so on. But it can not stylized the image or video in real-time. Comparing with Adobe Photoshop, our framework provides full flexibility for real time stylized image or video processing.

4.1. Cartoon effects simulating

It is very easy for our algorithm to generate cartoon-like effects. Only take performing the soft color quantization step on the abstracted images, which is generated by using adaptive separable bilateral filter, we could get a stylized result in cartoon or paint-like effects. In addition, in most of the cartoon images, the boundaries of the characters are highlighted by sketches, so we should detect the edges in the image or video. By adding visually distinct edges to regions of high contrast further increases the visual distinctiveness of these locations. In our implementation, edges are detected using the gradient of the original input image or video. The gradient is computed on the GPU using a fast, symmetric kernel as fallows [8]:

$$\Delta P(x,y) = \left|p_{x-1,y} - p_{x+1,y}\right| + \left|p_{x,y-1} - p_{x,y+1}\right| \qquad (2)$$

The gradient intensity is computed by averaging the gradient of each color channel. Any other gradient computation could be used depending on the compromise between efficiency and realism needed by the user. Our

method is clearly on the efficiency side. We also use the similar method in the process of simulate the pencil drawing effects.

The cartoon effects stylization algorithm is composed of two distinct image processing steps as proposed above. Each component could be implemented as a single fragment shader pass, both the adaptive separable bilateral filter and soft color quantization are implemented as a pair of fragment shaders.

4.2. Watercolor effects simulating

Watercolor offers a very rich medium for graphical expression. As such, it is used in a variety of applications including illustration, image processing and animation .In recent years; many researchers have developed some methods to simulate these effects.

Our work is directly inspired from Adrien Bousseau' idea [8], but we use the bilateral filter and soft color quantization proposed above, not the mean-shift filter, to abstracted the input image or video. Thus make us can easily ensure the temporal coherence. Our method include two steps: an abstraction step that recreates the uniform color regions of watercolor as mentioned in section 3, and an effect step that filter the abstracted image to obtain watercolor-like images.

The most significant and specific effects in watercolor is that the pigments are mixed with water but do not dissolve totally. The result is that, after drying, even on a totally smooth paper, the pigment density is not constant. This manifests itself at two different scales: Firstly, there are low-frequency density variations on the canvas: it is the turbulence flow; Secondly, there are high-frequency variations due to non-homogeneous repartition of pigments in water: it is the pigment dispersion. To simulate this phenomenon, in effect step, we firstly generate a white noise map which is blended with the abstracted image or video to achieve the effects of pigment dispersion, then we generate a perlin noise map which is blended with the intermediate image or video we got in previous step to achieve the effects of turbulence flow. Our method is very simple because we do not use any physical based model. It is very easy to implement on the current GPU (by using two textures) to satisfy real-time requirement. Another property of watercolor is that the underlying paper structure affects the water flowing process significantly, and thereby also the visual results. To emphasize this process, we additionally modify the color layers alpha channel according to an intensity texture T. The color blending function is given by:

$$c^{/} = c(1-(1-c)(d-1)); \quad d = 1+\beta(I-0.5) \qquad (3)$$

Where $c^{/}$ is the modified color, 1-c is the original light attenuation, d is a pigment density parameter. I is the intensity of the noise map. Our color blend model is similar to the Adrien Bousseau's intuition. The experiments show our method is effective.

4.3. Pencil drawing effects simulating

We have also implement the pencil filter based stylization algorithm we designed previously [9] on GPU. With the GPU's powerful parallel processing ability, we have achieved the real-time video synthesis for pencil drawing style. In this section, we will describe the GPU implementation of our algorithm in detail.

It is difficult to generate pencil filter we designed directly on GPU because the instructions and the registers are limited in GPU. Fortunately, we can generate the weight values of pencil filter in advance (like what shown in Figure 2). This idea makes a way to using our method on GPU. Firstly we load the weight values of pencil filter into GPU, which is generated in CPU in advance, and then convolution is executed for each pixel.

0	0	0	0.1001	0.0608
0	0	0.1038	0.3543	0.1001
0	0.1038	0.3543	0.1038	0
0.1001	0.3543	0.1038	0	0
0.0608	0.1001	0	0	0

Figure 2. Pencil filters using our method generated with the template parameters (D=1, Len=5, θ=45).

Many effective means can achieve the convolution on GPU. Generally, we can store the weight values of the convolution template as a constant or uniforms type, and then execute the convolution operation. This method seems simple and feasible, but the fact is that if we solidify these constants into GPU's shader program, we can not easily expand the program function later on. It's not suitable to our method because we need to use the weight values to simulate the different property of the pencil stroke, such as stroke length, stoke width and stroke orientation. To solve the problem, we load the template data as a texture image, and call the image as Filter Texture. When a video fragment is synthesized in real-time, the corresponding filter texture will be chosen according to the user's need, and then convolution is executed. The convolution process can be expressed by:

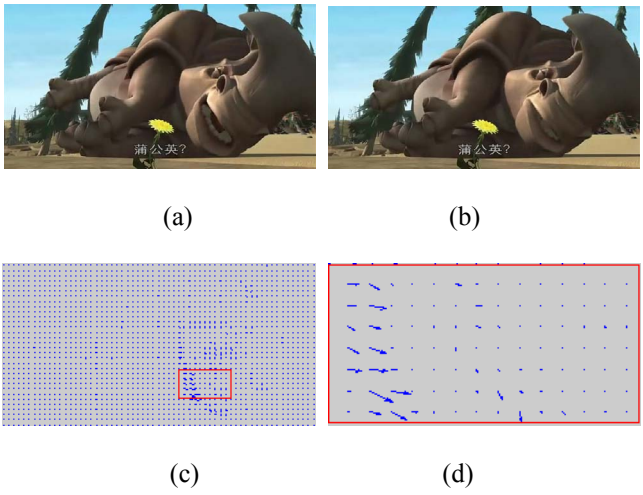
$$filtered\ color = \frac{\sum k(s-s_0, t-t_0)tex(s,t)}{\sum k(s-s_0, t-t_0)} \qquad (4)$$

Where the current pixel position is (s0, t0), tex is a function for searching the corresponding filter texture. To save the

memory, we store the total value of the template on the alpha channel.

To make sure the pencil texture has stochastic distribution, we generate the black noise image from the reference image. Our method for generating the black noise image is similar with the method for adding white noise in Mao [10].Unfortunately at present, there's no function supplied for generating the floating-point pseudo-random number in GPU. Therefore, we generate a noise texture image in advance; the pixel value of the noise image is making up of floating-point random number. GPU can thereby get the floating-point random number by sampling the noise texture image. Then, we can generate the black noise image on GPU according to the approach described in paper [9].

But our method in simulation pencil drawing effects on GPU usually brings a negative effect to the result video because it cannot preserve the interframe coherence by simply using the method we mentioned above. To solve the problem, we use Horn's method [11] to estimate the optical flow field of each two adjacent frames. Figure 3 shows the optical flow vector field. As synthesis a frame, we compare the magnitude of the current pixel with an appointed threshold. If the former is small, then we believe this pixel is almost still. So we need only to copy the previous frame's corresponding pixel to the current pixel. Otherwise, we recalculate the value of the current pixel following with the method described in paper [11].



(a) (b)

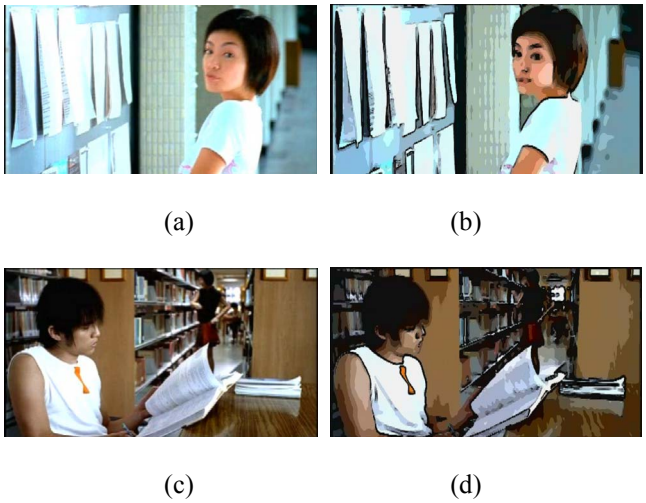
(c) (d)

Figure 3. Calculate the optical flow field ((a) (b) are two adjacent frames. (c) is the optical flow vector field. (d) is the enlarged image of the red region of (c)).

Generally, the scene's change is very small in a pair of adjacent frames. So the magnitudes of optical flow vectors are often to be zero or very small on most pixel positions, especially in the background. In this way, we can basically keep the interframe coherence of the video. Unfortunately the optical flow filed computation method we proposed have not yet implement on GPU.

5. Experiment result

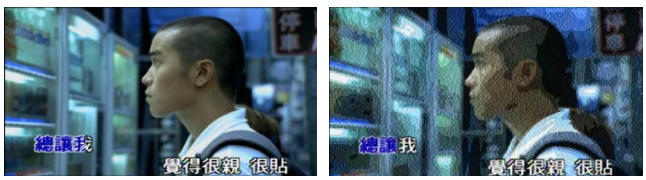
Figure 4－6 are some results generated by our system. All experiments run on a 1.7GHz Pentium PC with 512MB of RAM. We designed an implementation of the algorithm using the HLSL Shading Language. The graphics hardware we use is ATI Mobility Radeon 9700. Our rendering framework is based on the demo named VideoShader of the ATI Company. For all the input videos, the stylized procedure is fully or almost fully automatic. Our system can achieve real-time rendering speed (higher than 30 fps) by using the GPU.



(a) (b)

(c) (d)

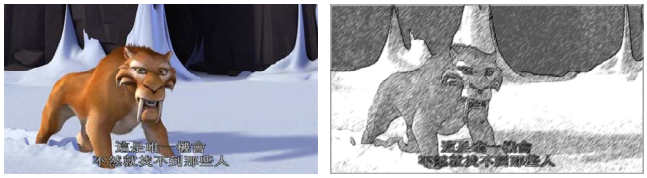
Figure 4. Stylized original video with cartoon-like style ((a), (c) is Original video get from mv of Jay Chou. (b), (d) is Stylized video frame).



(a) (b)

(c) (d)

Figure 5. Stylized original video with water color style ((a), (c) is Original video get from mv of S.H.E. (b), (d) is Stylized video frame).



(a) (b)

Figure 6. Stylized original video with pencil drawing style ((a) is Original video get from the movie Ice Age. (b) is Stylized video frame).

6. Conclusion

In this paper, we describe a system for GPU-based real-time video stylization. It has fallowed key features: (1) by using modern GPU hardware features, our system achieves real-time performance, which is essential for its applications. The complete stylized system is capable of generating an output video stream at real-time frame rates, so it can make an application in interactive scenarios possible; (2) we create an adaptive smoothing effect and simplifies the image color by iteratively applying the separable approximation to a bilateral filter; (3) by using the soft color quantization our system can insure the temporal coherence of the output video; (4) it can produce various and different artistic styles, such as watercolor, pencil drawing, eastern ink and so on in a general framework which can satisfy different user's needs in a general framework. Our system is not only fit to fast stylize of the image, but also to that of the video stream. Furthermore, according to users' selection from different parameters, it can provide interactive and real-time rendering of corresponding artistic style.

Acknowledgements

This paper is supported by NSFC (No. 60663010) and NSF (No. 2006F0017M) of Yunnan provinc., and the IEEE Systems, Man and Cybernetics Society. All video fragments are captured from MV.

References

[1] Doug DeCarlo, Anthony Santella, "Stylization and Abstraction of Photographs", In ACM SIGGRAPH 2002: 769-776(2002).

[2] Jue Wang, Yingqing Xu, Heung-Yeung Shum and Michael Cohen, "Video Tooning", In ACM SIGGRAPH 2004: Vol. 23, No. 3, 574-583(2004).

[3] Fischer, J., Bartz, D., Strasser, W, "Stylized Augmented Reality for Improved Immersion", In Proc. of IEEE VR 2005: 195–202(2005).

[4] Holger Winnemöller, Sven C. Olsen, Bruce Gooch, "Real-time video abstraction", In ACM SIGGRAPH 2006: Vol. 25, No. 3, 1221-1226(2006).

[5] Tomasi, C., Manduchi, R, "Bilateral filtering for gray and color images", In Proc. IEEE Int. Conf. on Computer Vision 1998, 836-846(1998).

[6] Tomasi, C., Manduchi, R, "Bilateral filtering for gray and color images", In Proc. IEEE Int. Conf. on Computer Vision 1998, 836-846(1998).

[7] J. Lander, "Shades of Disney: Opaquing a 3D World", Game Developer Magazine, 7(3):15–20(2000).

[8] Adrien Bousseau,Matt Kaplan,Joëlle Thollot,François X. Sillion, "Interactive watercolor rendering with temporal coherence and abstraction", In NPAR 2006:141 – 149(2006).

[9] Dang-en Xie, Yang Zhao, Dan Xu, Xiaochuan Yang, "Convolution Filter Based Pencil Drawing and Its Implementation on GPU", In APPT 2007: 723-732(2007).

[10] X. Mao, Y. Nagasaka and A. Imamiya, "Automatic Generation of Pencil Drawing from 2D Images Using Line Integral Convolution", In CAD/GRAPHICS2001: 240-248(2001).

[11] Horn B KP. Schunck B G: Determining optical flow. Artificial Intelligence, 1981, 17:185~203(1981).