Visual Comput (2008)
DOI 10.1007/s00371-008-0254-8

ORIGINAL ARTICLE

# Real-time feature-aware video abstraction

**Hanli Zhao**
**Xiaogang Jin**
**Jianbing Shen**
**Xiaoyang Mao**
**Jieqing Feng**



H. Zhao · X. Jin (✉) · J. Feng
State Key Lab of CAD & CG,
Zhejiang University,
Hangzhou 310027, P.R. China
hanlizhao@gmail.com,
{jin, jqfeng}@cad.zju.edu.cn

J. Shen
Indiana University-Purdue University,
Indianapolis, USA
shenjian@iupui.edu

X. Mao
University of Yamanashi, Japan
mao@yamanashi.ac.jp

**Abstract** This paper presents a novel feature-aware rendering system that automatically abstracts videos and images with the goal of improving the effectiveness of imagery for visual communication tasks. We integrate the bilateral grid to simplify regions of low contrast, which is faster than the separable approximation to the bilateral filter, and use a feature flow-guided anisotropic edge detection filter to enhance regions of high contrast. The edges detected in this paper are smoother, more coherent and stylistic than those of the isotropic difference-of-Gaussian filter. The presented algorithms are highly parallel, allowing a real-time performance on modern GPUs. The implementation of our approach is straightforward. Several experimental examples are given at the end of the paper to demonstrate the effectiveness of our approach.

**Keywords** Non-photorealistic rendering · Visual communication · Real-time video processing · Image processing

## 1 Introduction

When artists design imagery to portray a scene, they do not just render visual cues veridically. Instead, they select which visual cues to portray and adapt the information each cue carries. In recent years, automatic abstraction for efficient visual communication is becoming more and more popular to make images and/or videos easier or faster to understand.

Depicting information about shapes by *line drawing* is clearly effective and natural, and has been used for tens of thousands of years. It even outperforms photorealistic imagery in terms of efficacy of visual information transfer and subject identification. Line drawing styles can be found in many contexts, such as cartoon, storytelling, technical illustration, architectural design and medical atlases.

Recently, Winnemöller et al. [25] presented an automatic, real-time video and image abstraction framework. The system reduced contrast in low-contrast regions using the bilateral filter and artificially increased contrast in higher contrast regions with difference-of-Gaussian (DoG) edges. On one hand, to attain real-time frame rates, they used a separable approximation to the bilateral filter with a small kernel size and iterated the approximation to obtain a sufficiently large spatial support. Chen et al. [7] introduced the new *bilateral grid* data structure with GPU implementation to achieve an even higher time efficiency. On the other hand, the DoG edge model has proven to be efficient in computation on GPUs and non-uniform thickness scales. However, it is not without drawbacks. Due to the nature of isotropic filter kernels, the aggregate of edge pixels may not clearly reveal the sense of "directedness" and thus it may look less like lines [13].

In this paper, we present a real-time video and image abstraction system by employing the bilateral grid to simplify regions of low contrast and using a feature flow-guided anisotropic edge detection filter to enhance regions of high contrast.

Our approach has three contributions:

- By optimizing the algorithm of [13], a fast, efficient, and highly anisotropic edge detection filter driven by the "flow" of salient image features, is implemented.
- By integrating both the bilateral grid and our edge detector, an improved automatic real-time feature-aware video and image abstraction system, as compared with [25], is developed.
- The algorithms presented in this paper are highly parallel, enabling a real-time implementation on modern GPUs.

The rest of the paper is organized as follows. Section 2 gives an overview of some previous work. Section 3 describes our new approach, whereas experimental results and related discussions are presented in Sect. 4. Finally, Sect. 5 concludes the paper.

## 2 Related work

### 2.1 Image-based stylization and abstraction

Several video and image abstraction systems have been extensively developed but most of them are computationally expensive. Collomosse et al. [8], and Wang et al. [23] extended the mean-shift-based stylization approach of Decarlo and Santella [9]. Bousseau et al. [4] created watercolor-like effects with temporal coherence by incorporating bidirectional texture advection.

Other real-time stylization techniques have been studied. Fischer et al. [11] applied stylization effects in augmented reality applications. Gooch et al. [12] automatically created monochromatic human facial illustrations using DoG filters. Winnemöller et al. [25] extended this technique to abstract imagery by modifying the contrast of visually important features. We use a similar pipeline as Winnemöller's, that is with real-time performance and temporal coherence but producing smoother, more coherent and stylistic edges.

### 2.2 Bilateral filter

The bilateral filter is a non-linear diffusion filter [18] which blurs small discontinuities and sharpens edges [1, 20, 22]. It is produced by combining range filtering with domain filtering.

Several researchers have proposed a variety of numerical schemes for approximating the bilateral filter to reduce processing time. Pham et al. [19] described a separable approximation and Winnemöller et al. [25] parallelized it on GPUs. Weiss [24] maintained local image histograms, which limited his approach to box spatial kernels instead of smooth Gaussian kernels. Paris and Durand [17] recast the computation as a higher-dimensional linear convolution followed by a trilinear interpolation and a division. Recently, Chen et al. [7] generalized it into the bilateral grid that enabled a variety of real-time edge-aware operations.

Other edge-preserving techniques are related to the bilateral filter [2, 10]. Our video abstraction system is built upon the method presented by Winnemöller et al. Chen et al.'s bilateral grid for acceleration is also integrated into our system, but it is irrelevant to the size of the vertex buffer during grid creation in our system with the utilization of the shader model version 4.0 features [3, 15].

### 2.3 Line drawing

In addition to the isotropic DoG edge detector which is similar to the Marr-Hildreth edge detector [14] as mentioned in Sect. 1, there are a variety of edge detectors in 2D image applications. Canny edges [6] are guaranteed to lie on any real edge in an image, and are used in several works [9, 11, 16]. Son et al. [21] extracted lines based on the likelihood of finding the genuine shape boundaries. Kang et al. [13] introduced the notion of flow-guided [5] anisotropic filtering for detecting highly coherent lines while suppressing noise. However, most of these edge detectors are computationally expensive, and thus they cannot be used in real-time applications. Due to the nature of isotropic filter kernels for the traditional DoG detector, the aggregate of edge pixels may not clearly reveal the sense of "directedness". Inspired by the idea of Kang et al., we propose a fast, efficient and highly anisotropic edge detection filter driven by the "flow" of salient image features. The edge detector in this paper is highly parallel, which allows a GPU-based and real-time implementation.

## 3 Our approach

### 3.1 Pipeline overview

Our goal is to develop an automatic and real-time abstraction system for videos and images by simplifying their visual content while preserving or even emphasizing most of the perceptually important image features. Intuitively, high contrast is linked to high visual salience, and low contrast to low salience.

Our video abstraction system is built upon the method presented by Winnemöller et al., and the framework of our approach is the same as theirs. The pipeline is described as follows. Firstly, the system collects input images from static image files, static video files, or even live video capturing devices (see Fig. 1a). Then, the input image is converted from *RGB* color-space to *CIE-Lab*-space [26], which is known to be a perceptually uniform color-space based on a large body of psychophysical data concerning color-matching experiments performed by human observers. In order to smooth small discontinuities and enhance edges in real-time, we choose the edge-aware bilateral grid operator to approximate the

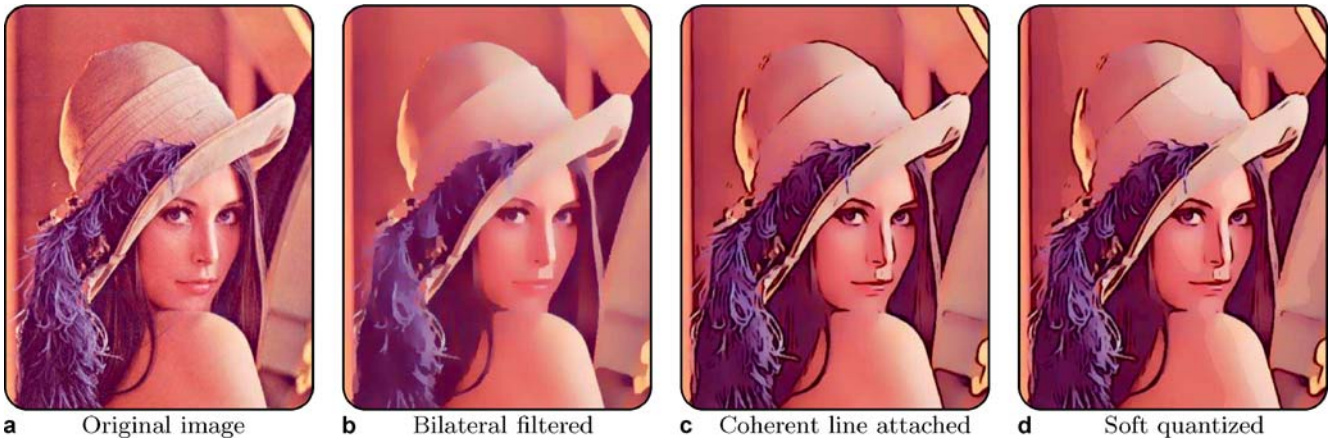



**Fig. 1a–d.** Our approach takes as input (**a**) an image and produces a variety abstraction effects: **b** bilateral filtered, **c** coherent line attached, **d** soft luminance quantized. Note how fiber and hair (**c**, **d**) are emphasized and stylized using our feature flow-based DoG filter

anisotropic diffusion filter (see Fig. 1b). Next, our feature flow-based DoG filter is applied to detect edges. The feature flow-based DoG filter controls the non-uniform thickness scales as the traditional DoG filter, however, the resulting lines are smoother, more coherent and more stylistic (see Fig. 1c). Optionally, techniques such as soft luminance quantization and image-based warping are used to further enhance the abstraction effect with good temporal coherence (see Fig. 1d). Lastly, the output image is generated by converting the result back to RGB space.

### 3.2 Bilateral grid

Given an input image $I$, which maps pixel locations into some feature space, the result of the bilateral filter is defined by:

$$F(I, \boldsymbol{x}, \sigma_s, \sigma_r) = \frac{1}{W(I, \boldsymbol{x}, \boldsymbol{y})} \sum_{\boldsymbol{y} \in \boldsymbol{N}(\boldsymbol{x})} w(I, \boldsymbol{x}, \boldsymbol{y}) I(\boldsymbol{y}) \quad (1)$$

$$W(I, \boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{y} \in \boldsymbol{N}(\boldsymbol{x})} w(I, \boldsymbol{x}, \boldsymbol{y}) \quad (2)$$

$$w(I, \boldsymbol{x}, \boldsymbol{y}) = G(\|\boldsymbol{y} - \boldsymbol{x}\|, \sigma_s) G(|I(\boldsymbol{y}) - I(\boldsymbol{x})|, \sigma_r) \quad (3)$$

$$G(d, \sigma) = e^{-\frac{1}{2}\left(\frac{d}{\sigma}\right)^2}. \quad (4)$$

In these formulae, $\boldsymbol{x}$ is a pixel location $(x, y)$ in the image, and $\boldsymbol{N}(\boldsymbol{x})$ are its neighboring pixels. The parameter $\sigma_s$ defines the size of the spatial neighborhood used to filter a pixel. Increasing $\sigma_s$ results in more blurring, but features may blur across significant boundaries if $\sigma_s$ is too large. The parameter $\sigma_r$ determines how much an adjacent pixel is downweighted because of the intensity difference.

The bilateral grid is a primary data structure that enables a variety of edge-aware operations in real-time [7]. Essentially, it is a 3D array (see Fig. 2), where the first two dimensions $(x, y)$ correspond to 2D pixel position and

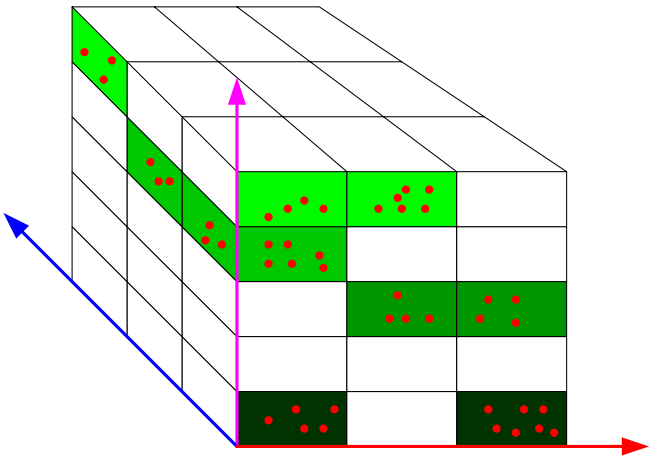


**Fig. 2.** Illustration of the bilateral grid. The *red points* denote input image pixels and they locate in corresponding grid voxels

form the spatial domain, whereas the third dimension $z$ corresponds to the intensity range. Both the bilateral filter and the edge detector in this paper are the major time-consuming components in our real-time application, thus we choose the bilateral grid method instead of the separable one used in [25].

The bilateral grid is implemented in three steps. First, the grid $\Gamma$ is constructed by accumulating the value of each input pixel into the appropriate grid voxel, which is inherently a scatter operation in the vertex shader in GPU implementation. The bilateral grids are stored in a 2D texture by tiling the $z$ levels across the texture plane.

$$\text{Initialization: } \Gamma(i, j, k) = (0, 0) \quad (5)$$

$$\text{Filling: } \Gamma([x/s_s], [y/s_s], [I(\boldsymbol{x})/s_r]) += (I(\boldsymbol{x}), 1), \quad (6)$$

where $s_s$ and $s_r$ are the sampling rates of the spatial axis and range axis, respectively. $[\cdot]$ is the closest-integer operator.

We use a vertex shader to rasterize the input pixel position and then determine the output grid voxel position. Chen et al. [7] rasterized a vertex array of single pixel points using a pre-computed vertex buffer, which may consume extra memory space and transfer time. Instead, we rasterize the input pixel position directly in the vertex shader. We utilize the value SV_VERTEXID which is automatically generated by the*input assembler* in the Direct3D 10 pipeline [3]. The input position is then computed according to the value SV_VERTEXID for each vertex and the image size (width and height). As a result, we render the pointlist with the image pixel number, however, no vertex buffer is required.

Next, the Gaussian convolution on the grid is applied to each dimension using a pixel shader. The user-defined bandwidths of the Gaussian $\sigma_s$ and $\sigma_r$ automatically determine the sampling rates $s_s$ and $s_r$.

Finally, we slice the grid to extract the final output image by accessing the grid at $(x/s_s, y/s_s, I(x, y)/s_r)$ using a trilinear interpolation. By taking advantage of the hardware bilinear interpolation, the values in the two nearest $z$ levels are sampled from the grid texture, and are then interpolated.

### 3.3 Feature flow construction on GPU

Inspired by the idea of Kang et al. [13], who proposed a kernel-based nonlinear vector-smoothing technique and a flow-based anisotropic DoG filter, our approach takes into account the "direction" of the local image structure in the DoG filtering and applies DoG filters only in a direction perpendicular to that of the local "feature flow" as well. However, our approach can be paralleled on graphics hardware, allowing a variety of real-time image processing applications related to the edge detection technique.

To gain smooth, coherent, and stylistic lines, firstly the approximated feature flow field $V$ from the input image $I$ is constructed. The feature flow field is a vector perpendicular to the image gradient $g(\boldsymbol{x}) = \nabla I(\boldsymbol{x})$. Kang et al. proposed three requirements that $V$ must satisfy: (1) The vector flow must describe the salient-edge tangent direction in the neighborhood; (2) the neighboring vectors must be smoothly aligned except at sharp corners; (3) important edges must retain their original directions. In order to achieve real-time performance, we modify the feature flow filter as follows:

$$V^{\text{new}'}(\boldsymbol{x}) = \frac{1}{k_h} \sum_{\boldsymbol{y} \in \Omega_h(\boldsymbol{x})} w_m(\boldsymbol{x}, \boldsymbol{y}) w_d(\boldsymbol{x}, \boldsymbol{y}) V^{\text{cur}}(\boldsymbol{y}) \tag{7}$$

$$V^{\text{new}}(\boldsymbol{x}) = \frac{1}{k_v} \sum_{\boldsymbol{y} \in \Omega_v(\boldsymbol{x})} w_m(\boldsymbol{x}, \boldsymbol{y}) w_d(\boldsymbol{x}, \boldsymbol{y}) V^{\text{new}'}(\boldsymbol{y}), \tag{8}$$

where $\Omega_h(\boldsymbol{x})$, $\Omega_v(\boldsymbol{x})$ denote the horizontal direction and vertical direction neighborhood of $\boldsymbol{x}$ respectively, and $k_h$, $k_v$ are corresponding normalizing terms.

The *magnitude weight function* $w_m$ is defined as:

$$w_m(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}(1 + \tanh(\eta(\hat{g}(\boldsymbol{y}) - \hat{g}(\boldsymbol{x})))), \tag{9}$$

where $\hat{g}(\boldsymbol{x})$ is the gradient magnitude at pixel position $\boldsymbol{x}$. When a neighboring pixel's gradient magnitude is higher, its function value is bigger, and vice versa. The value $\eta$ controls the fall-off rate, and is set to 1 throughout the paper.

The *direction weight function* $w_d(\boldsymbol{x}, \boldsymbol{y})$ is defined as:

$$w_d(\boldsymbol{x}, \boldsymbol{y}) = V(\boldsymbol{x}) \cdot V(\boldsymbol{y}), \tag{10}$$

where $V(\boldsymbol{x})$ denotes the previously calculated feature vector at $\boldsymbol{x}$. As we can see, the value of this direction weight function increases as the angle between the two vectors decreases. If the angle is bigger than 90°, the direction of $V(\boldsymbol{y})$ is reversed before smoothing, avoiding swirling flows.

The filter actually uses a separable approximation to construct the feature flow, and thus we can implement it on the GPU by taking advantage of its parallelism. The initial gradient field is calculated using the Sobel operator in this paper, and the initial feature flow field can be easily obtained. Then our modified filter is iteratively applied to smooth the feature flow. Note that the gradient field $g$ evolves accordingly whereas the magnitude $\hat{g}$ is unchanged. In our GPU implementation, 2 or 3 iterations are enough.

### 3.4 Coherent line extraction

After the feature flow field is filtered, feature lines are extracted by applying a flow-guided anisotropic DoG filter:

$$F(\boldsymbol{x}) = \sum_{\boldsymbol{s} \in c(\boldsymbol{x})} G(\|\boldsymbol{s} - \boldsymbol{x}\|, \sigma_m) W(\boldsymbol{s}) \tag{11}$$

$$W(\boldsymbol{s}) = \sum_{\boldsymbol{y} \in l(\boldsymbol{s})} (G(\|\boldsymbol{y} - \boldsymbol{x}\|, \sigma_c) - \rho \cdot G(\|\boldsymbol{y} - \boldsymbol{x}\|, \sigma_s)) I(\boldsymbol{y}), \tag{12}$$

where $c(\boldsymbol{x})$, $l(\boldsymbol{s})$ denote the neighborhood along the tangential direction at $\boldsymbol{x} \in I$ and perpendicular direction at $\boldsymbol{s} \in c(\boldsymbol{x})$ to the flow direction, respectively. Moving along both the positive and the negative directions of $c_{\boldsymbol{x}}$, a new position $\boldsymbol{s}$ is obtained. Thus $W(\boldsymbol{s})$ is convolved along the line $l_{\boldsymbol{s}}$ that is perpendicular to the direction vector $V(\boldsymbol{s})$ and intersects $\boldsymbol{s}$, as illustrated in Fig. 4. Note $l_{\boldsymbol{s}}$ is parallel to the gradient vector $g(\boldsymbol{s})$, and $l_{\boldsymbol{s}}(0) = \boldsymbol{s}$. Again we separate Eqs. 11 and 12 by drawing an image-sized quad with two passes in the pixel shader. $p \times q$ neighboring points from the kernel are sampled using hardware texture lookups. In pass 1, the value of $W(\boldsymbol{s})$ for each pixel can be computed by bidirectionally following the gradient vector $g(\boldsymbol{x})$ starting from $\boldsymbol{x}$. Initially, we set $\boldsymbol{z} \leftarrow \boldsymbol{x}$,

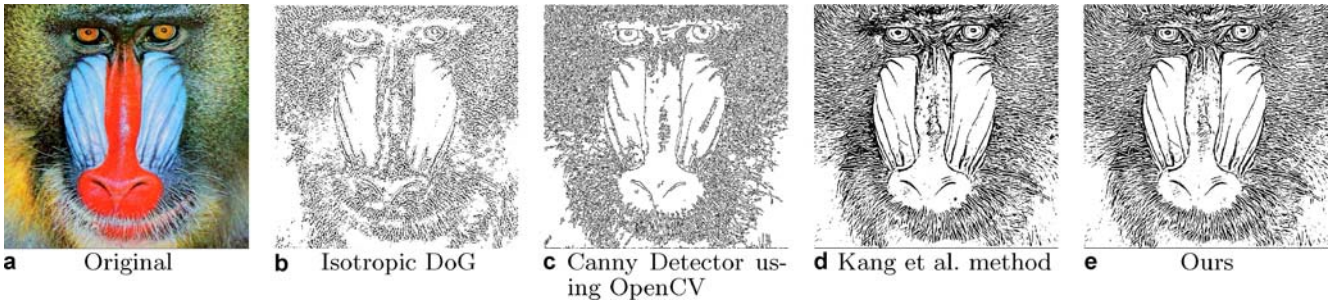



**Fig. 3a–e.** Comparison with other edge detection techniques. Note that our method can generate a similar result as proposed by Kang et al., and is better than the traditional difference-of-Gaussian and Canny detector

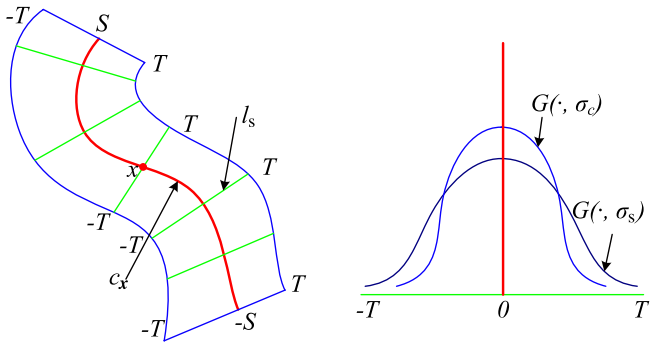



**Fig. 4.** Feature flow-based DoG filter: (*left*) kernel at $\boldsymbol{x}$, and (*right*) Gaussian component


**Fig. 5a–d.** Line control. Abstraction using **b** fine edges ($\varphi_e = 0.5$, $\sigma_c = 2$), **c** sharper edges ($\varphi_e = 2$, $\sigma_c = 2$), and **d** coarser edges ($\varphi_e = 2$, $\sigma_c = 5$). For all images we set soft quantization steps ($q = 8$, and sharpness values between [3, 14])

then iteratively obtain the next sample point by moving along $c_{\boldsymbol{x}}$ in one direction: $z \leftarrow z + g(z)$; similarly, we obtain the points on the other half of $c_{\boldsymbol{x}}$ by $-g(z)$. Then in pass 2, $F(\boldsymbol{x})$ is convolved along the line parallel to $V(z)$ using the same method. By taking advantage of graphics hardware bilinear interpolation, we only need two texture lookups to get the sampling position and then sample the image intensity in the pixel shader. Here we set $\sigma_s = 1.6\sigma_c$ to make the shape of $W(\boldsymbol{s})$ closely resemble that of the Laplacian-of-Gaussian [14]. Again the user-defined bandwidths of the Gaussian $\sigma_c$ and $\sigma_m$ automatically determine the convolution size $T$ and $S$, and thus the sampling size $p$ and $q$. The threshold level $\rho$ determines the sensitivity of the edge detector. We use $\rho = 0.99$ throughout.

Once we obtain $F$ for the input image from Eq. 11, the edges can be extracted. Rather than converting them to a black-and-white image by binary thresholding, as used in [13], we define our anisotropic DoG edges using a slightly smoothed step function, as suggested in [25]:

$$D(\boldsymbol{x}) = \begin{cases} 1, & \text{if } (F(\boldsymbol{x}) > 0); \\ 1 + \tanh(\varphi_e \cdot F(\boldsymbol{x})), & \text{otherwise.} \end{cases} \quad (13)$$

Here $\varphi_e$ is a parameter controlling the line sharpness, and $\sigma_c$ determines the line thickness scale as the isotropic DoG method (see Fig. 5). As the construction of the feature flow can be treated as a modified bilateral filter, small discontinuities are omitted and salient edges are strengthened. In addition, we can perform real bilateral filtering before extracting feature lines for a noisy image. Figure 3 shows the comparison of our method with other popular line extraction techniques. All images are converted to black-and-white by binary thresholding for clarification.

## 4 Experimental results and discussions

We have developed a new real-time video and image abstraction system based on Direct3D 10 APIs and Shader Model version 4.0, and tested it on some scenes in order to evaluate its efficiency. All tests were conducted on a PC with a 1.83 GHz Intel Core 2 Duo 6320 CPU, 2 GB main memory, an NVIDIA Geforce 8800 GTS GPU, 320 MB graphics memory, and Windows Vista 64 bit operating system.

**Table 1.** Statistics of frames per second of our system and processing time of edge detection

| Window resolution (Pixels) | 1.3 M | 1.0 M | 800 K | 500 K |
|---|---|---|---|---|
| Frames per second | 32 | 40 | 50 | 75 |
| Processing time (ms) for edge detection | 19 | 15 | 12 | 8 |

The efficiency statistics of our abstraction system are listed in Table 1. Using our GPU acceleration techniques, we are able to perform video abstraction at 32 Hz in a window resolution of 1.3 mega-pixels, satisfying the real-time requirement on most PC monitors. In the naive CPU implementation of Kang et al., the processing time for edge


**Fig. 6a–c.** Comparison with the case that the foreground objects have low contrast, whereas background regions have high contrast


**Fig. 7a–j.** Abstraction effects of various input images using our feature-aware method. Regions of low contrast are simplified and abstracted, whereas regions of high contrast are strengthened. It can be seen that the underlying features in the input images are also stylized

detection is about 6–10 s for a $512 \times 512$ image [13]. Our GPU implementation outperforms it by $2 \sim 3$ orders of magnitude speedup.

In Fig. 6b we show a failure case of Winnemöller et al.'s abstraction approach, where all high frequency features of background carpet were emphasized too much. Introducing user interactions can easily remove detail edges regardless of their contrast [16]. However, user manipulations are not feasible in automatic abstraction systems. In this paper, the feature flow filter is actually a modified bilateral filter, small discontinuities are smoothed while salient edges are strengthened. As a result, unnecessary high frequency features are smoothed out, whereas the major edges on the carpet are smoother and more coherent (see Fig. 6c). However, as both the bilateral filter and the edge detector operate on local kernels, a global scale subjective abstraction effect is still hard to generate.

Other abstraction effects automatically generated in our system are shown in Fig. 7. We can see how regions of low contrast are simplified and abstracted, and how regions of high contrast are strengthened. It can be seen that the underlying features in the input images are also stylized using our new method. Moreover, the accompanying video demonstrates that our video abstraction system can generate vivid visual perception. The video applications are decoded on the CPU in a separate thread that runs in parallel with the GPU. The frame sampling rate of the video capturer is 30 Hz.

Our approach performs our abstraction technique after transforming the original image to *CIE-Lab* color-space. It works well in most cases since luminance carries a lot of the feature information. However, when iso-luminance color regions "touch" each other in the input image, our method tends to fail to generate the correct effect. Moreover, the bilateral grid-based bilateral filter with a small spatial kernel requires fine sampling, which results in large memory and computation costs.

## 5 Conclusions and future work

In this paper, we present a non-photorealistic feature-aware rendering system that automatically abstracts videos and images by integrating the bilateral grid to simplify regions of low contrast and utilizing a feature flow-guided anisotropic edge detection filter to enhance regions of high contrast. The algorithms are highly parallel, allowing a real-time implementation on modern GPUs. Experimental results have shown that our proposed feature flow-based DoG filter is hundreds of times faster than Kang's method, while the lines are smooth, coherent, and stylistic as well. Additionally, the experimental results demonstrate both the feasibility and efficiency of our proposed algorithms.

Limitations mentioned above will be addressed in our future work. In addition, our future work includes exploring better methods for abstracting video by user control, adopting faster algorithms for very large high-definition video, extending to real-time HDR image and video abstraction, and investigating the extension to real-time information recognition.

**Acknowledgement** The authors would like to thank Olsen S.C. for providing his valuable source images. Many thanks also to Xiaoyan Luo for her help in completing the paper. This work is supported by the National Natural Science Foundation of China (Grant Nos. 60533080, 60573153), China 863 program (Grant No. 2006AA01Z314), the Key Technology R&D Program (Grant No. 2007BAH11B03), and the Science and Technology Plan of Zhejiang Province (Grant No. 2008C24008).

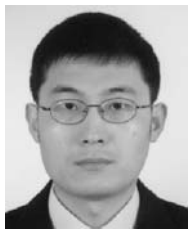
## References

1. Aurich, V., Weule, J.: Non-linear gaussian filters performing edge preserving diffusion. In: Proceedings of the DAGM Symposium, pp. 538–545. Springer, London (1995)
2. Barash, D.: A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(6), 844–847 (2002)
3. Blythe, D.: The Direct3D 10 system. In: Proceedings of the ACM SIGGRAPH '06, pp. 724–734. ACM, New York (2006)
4. Bousseau, A., Neyret, F., Thollot, J., Salesin, D.: Video watercolorization using bidirectional texture advection. In: Proceedings of the ACM SIGGRAPH '07, article 104. ACM, New York (2007)
5. Cabral, B., Leedom, L.C.: Imaging vector fields using line integral convolution. In: Proceedings of the ACM SIGGRAPH '93, pp. 263–270. ACM, New York (1993)
6. Canny, J.F.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**(6), 679–698 (1986)
7. Chen, J., Parism, S., Durand, F.: Real-time edge-aware image processing with the bilateral grid. In: Proceedings of the ACM SIGGRAPH '07, article 103. ACM, New York (2007)
8. Collomosse, J.P., Rowntree, D., Hall, P.M.: Stroke surfaces: temporally coherent artistic animations from video. IEEE Trans. Vis. Comput. Graph. **11**(5), 540–549 (2005)
9. Decarlo, D., Santella, A.: Stylization and abstraction of photographs. In: Proceedings of the ACM SIGGRAPH '02, pp. 769–776. ACM, New York (2002)
10. Durand, F., Dorsey, J.: Fast bilateral filtering for the display of high-dynamic-range images. In: Proceedings of the ACM SIGGRAPH '02, pp. 257–266. ACM, New York (2002)
11. Fischer, J., Bartz, D.: Stylized augmented reality for improved immersion. In: Proceedings of the IEEE Conference on Virtual Reality, pp. 195–202. IEEE Computer Society, Washington (2005)
12. Gooch, B., Reinhard, E., Gooch, A.: Human facial illustrations: creation and psychophysical evaluation. ACM Trans. Graph. **23**(1), 27–44 (2004)
13. Kang, H., Lee, S., Chui, C.K.: Coherent line drawing. In: Proceedings of the ACM International Symposium on Non-Photorealistic Animation and Rendering (NPAR07), pp. 43–50. ACM, New York (2007)

14. Marr, D., Hildreth, E.C.: Theory of edge detection. Proc. R. Soc. Lond. Ser. B, Biol. Sci. **207**(1167), 187–217 (1980)
15. Microsoft Corporation: DirectX 10 HLSL Compiler. DirectX Software Development Kit, Apr (2007)
16. Orzan, A., Bousseau, A., Barla, P., Thollot, J.: Structure-preserving manipulation of photographs. In: Proceedings of the ACM International Symposium on Non-Photorealistic Animation and Rendering, pp. 103–110. ACM, New York (2007)
17. Paris, S., Durand, F.: A fast approximation of the bilateral filter using a signal processing approach. In: Proceedings of the European Conference on Computer Vision, pp. 568–580. Springer, Berlin (2006)
18. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Trans. Pattern Anal. Mach. Intell. **12**(7), 629–639 (1991)
19. Pham, T.Q., van Vliet, L.J.: Separable bilateral filtering for fast video preprocessing. In: Proceedings of the IEEE Internationl Conference on Multimedia and Expo, pp. 454–457. IEEE Computer Society, Washington (2005)
20. Smith, S.M., Brady, J.M.: SUSAN – a new approach to low level image processing. Int. J. Comput. Vis. **23**(1), 45–78 (1997)
21. Son, M., Kang, H., Lee, Y., Lee, S.: Abstract line drawings from 2D images. In: Proceedings of the IEEE Pacific Conference on Computer Graphics and Applications (PG'07), pp. 333–342. IEEE Computer Society, Washington (2007)
22. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proceedings of the International Conference on Computer Vision (ICCV98), pp. 839–846. IEEE Computer Society, Washington (1998)
23. Wang, J., Xu, Y. Shum, H.-Y., Cohen M.F.: Video tooning. In: Proceedings of the ACM SIGGRAPH '04, pp. 574–583. ACM, New York (2004)
24. Weiss, B.: Fast median and bilateral filtering. In: Proceedings of the ACM SIGGRAPH '06, pp. 519–526. ACM, New York (2006)
25. Winnemöller, H., Olsen, S.C., Gooch, B.: Real-time video abstraction. In: Proceedings of the ACM SIGGRAPH '06, pp. 1221–1226. ACM, New York (2006)
26. Wyszecki, G., Stiles, W.S.: Color Science: Concepts and Methods, Quantitative Data and Formulae. Wiley, New York, NY (1982)







HANLI ZHAO is a Ph.D. candidate of the State Key Lab of CAD & CG, Zhejiang University, China. He received his B.Sc. degree in software engineering from Sichuan University in 2004. His research interests include non-photorealistic rendering and general purpose GPU computing. Contact him at hanlizhao@gmail.com.

XIAOGANG JIN is a professor of the State Key Lab of CAD & CG, Zhejiang University, China. He received his B.Sc. degree in computer science in 1989, M.Sc. and Ph.D. degrees in applied mathematics in 1992 and 1995, all from Zhejiang University. His current research interests include implicit surface computing, special effects simulation, mesh fusion, texture synthesis, crowd animation, cloth animation and non-photorealistic rendering. Contact him at jin@cad.zju.edu.cn.

JIANBING SHEN is currently a post-doctoral researcher in the Department of Computer and Information Science, Indiana University-Purdue University Indianapolis, USA. He received his Ph.D. degree in computer science from Zhejiang University in 2007. His research interests include texture synthesis, image completion, high dynamic range imaging and processing, and biomedical imaging and visualization. Contact him at shenjian@iupui.edu.

XIAOYANG MAO is a professor at the University of Yamanashi in Japan. Her research interests include flow visualization, texture synthesis, non-photorealistic rendering, and human–computer interactions. Mao has an M.S. and Ph.D. in computer science from Tokyo University. Contact her at mao@yamanashi.ac.jp.

JIEQING FENG is a professor in the State Key Lab of CAD & CG, Zhejiang University, China. He received his B.Sc. in applied mathematics from the National University of Defense Technology in 1992 and his Ph.D. in computer graphics from Zhejiang University in 1997. His research interests include geometric modeling, rendering and computer animation. Contact him at jqfeng@cad.zju.edu.cn.