Vis Comput (2009) 25: 973–984
DOI 10.1007/s00371-008-0308-y

ORIGINAL ARTICLE

Real-time saliency-aware video abstraction

Hanli Zhao · Xiaoyang Mao · Xiaogang Jin · Jianbing Shen · Feifei Wei · Jieqing Feng

Published online: 15 January 2009

Abstract Existing real-time automatic video abstraction systems rely on local contrast only for identifying perceptually important information and abstract imagery by reducing contrast in low-contrast regions while artificially increasing contrast in higher contrast regions. These methods, however, may fail to accentuate an object against its background for the images with objects of low contrast over background of high contrast. To solve this problem, we propose a progressive abstraction method based on a region-of-interest function derived from an elaborate perception model. Visual contents in perceptually salient regions are emphasized, whereas the background is abstracted appropriately. In addition, the edge-preserving smoothing and line drawing algorithms in this paper are guided by a vector field which describes the flow of salient features of the input image. The whole pipeline can be executed automatically in real time on the GPU, without requiring any user intervention. Several experimental examples are shown to demonstrate the effectiveness of our approach.

Keywords Non-photorealistic rendering · Image abstraction · Saliency map · Real-time video processing

H. Zhao · X. Jin (✉) · F. Wei · J. Feng
State Key Lab of CAD & CG, Zhejiang University, Hangzhou, 310027, China
e-mail: jin@cad.zju.edu.cn

H. Zhao
e-mail: hanlizhao@gmail.com

F. Wei
e-mail: weifeifei@cad.zju.edu.cn

J. Feng
e-mail: jqfeng@cad.zju.edu.cn

X. Mao
University of Yamanashi, Kofu, Japan
e-mail: mao@yamanashi.ac.jp

J. Shen
School of Computer Science & Technology, Beijing Institute of Technology, Beijing, 100081, China
e-mail: shenjianbing@bit.edu.cn

1 Introduction

Artists have learned that by making images look less photorealistic they enable audiences to feel more immersed in a story [18]. Simplified or even exaggerated features in an image can often improve perception and facilitate comprehension, which has led researchers to investigate new non-photorealistic image processing techniques. 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 [8, 13, 27].

In computational photography, an image is often decomposed into a piecewise smooth base layer and a detail layer [9]. The base layer contains the larger scale variations in intensity and is typically computed by applying an *edge-preserving smoothing* filter to the image. The detail layer captures the smaller scale details and is defined as the difference between the original image and the base layer. The image can be abstracted by discarding details at an appropriate scale. Very often, *bold lines* are employed to obtain a more stylized look. These lines are placed on locations of high contrast in the image that are likely to form the boundary of objects (or their parts) in a scene. To better convey visual information, the visual contents inside perceptually salient *regions of interest* (ROI) should be less attenuated than those in the background.

Existing real-time video abstraction methods assume that high contrast is linked to high visual salience, and low contrast to low salience [4, 27, 29]. However, in addition to the color and luminance, the contrast itself is also an important feature, and the contrast variation can be of perceptually importance. Moreover, as both the bilateral filter and the edge detector in these papers operate on isotropic kernels, we observe that there is room for improvement.

In this paper, we employ a more elaborate visual perception model and introduce an automatic real-time progressive abstraction framework. Our new system follows the flow-based idea of Kang et al. [13], employing the flow-based bilateral filter (FBL) for edge-preserving smoothing and the flow-based anisotropic difference-of-Gaussian filter (FDoG) for line extraction. To reduce both the spatial and temporal noise, we first apply a separable 3D bilateral filter to the video sequence by taking time as the third dimension. To guide the automatic progressive abstraction, we propose an ROI function which is derived from a saliency-based visual attention model proposed by Itti et al. [11]. Our framework design is highly parallel, enabling a real-time GPU-based implementation. In addition, the framework requires no user intervention, facilitating naive users to produce vivid cartoon-like effects and allowing online video processing.

This paper has the following contributions:

- An automatic real-time saliency-aware progressive abstraction framework.
- A real-time and temporally coherent flow-based abstraction component.
- A novel reformulation of saliency map computation in real time.

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 are presented in Sect. 4. Finally, the last section sumps up our conclusions.

2 Previous work

2.1 Saliency map

Low-level cues, i.e., color, luminance, and contrast, influence where in an image people will look and pay attention. Our approach is explicitly based on the perception-based model of Itti et al. [11], which combines information from center–surround mechanisms to compute a saliency map that assigns a saliency value to each image pixel.

The bottom-up saliency model has also been extended to 3D applications. Lee et al. [16] introduced the idea of mesh saliency as a measure of regional importance for graphics meshes. They demonstrated that the center-surround difference of curvature is more important than the curvature itself. Likewise we consider the contrast as an important feature, and the center-surround difference of contrast is more important than the contrast itself.

Lee et al. [17] presented a GPU-based algorithm for selective tracking in 3D scene. However, their method assumes that the 3D geometric information is known. Moreover, they did not take the contrast information as a perception feature.

2.2 Image and video abstraction

Several image and video abstraction systems have been developed for creating enhanced or abstracted renderings from arbitrary photographs or videos.

Many stylization methods can only process offline images or videos. Decarlo and Santella [8] progressively abstracted images using a perception model guided with the eye tracker. However, the eye tracker is expensive, and usually it is not a simple task to calibrate the eye tracker and track user's eye movements. Collomosse et al. [7] and Wang et al. [26] extended the stylization approach of Decarlo and Santella to video by treating a video sequence as a space-time 3D volume and then used the mean shift technique to segment the volume into contiguous volumes. However, those techniques require some least interactions from artists and are computationally expensive. Farbman et al. [9] abstracted a static image using an edge-preserving decomposition technique, which requires a time-consuming weighted least squares optimization over the whole image. Kang et al. [13] proposed a flow-based filtering framework for image abstraction. In this paper, we extend Kang et al.'s method to video abstraction and improve the processing time.

Recently, some real-time video abstraction systems have been proposed. Winnemöller et al. [27] proposed an automatic real-time image and video abstraction framework. They reduced contrast in low-contrast regions using the separable bilateral filter [20] and artificially increased contrast in higher contrast regions with isotropic DoG edges. Chen et al. [4] improved the frame rates by introducing the GPU-based bilateral grid. Zhao et al. [29] optimized the coherent line drawing algorithm [12] using graphics hardware to improve the abstraction effect. Kyprianidis and Döllner [15] abstracted images by the structure adaptive filtering. However, these approaches assume high contrast is linked to high visual salience and do not take the contrast variation into account.

There have been some non-photorealistic rendering (NPR) techniques related to image saliency. By introducing user interactions, Collomosse and Hall [6], Orzan et al. [19], and Chen et al. [4] performed progressive abstraction based on a control map. We are interested in a framework that is fully automatic since either eye-tracking or other user manipulations are usually difficult in case of videos compared with still images. Although Santella and DeCarlo [22] have shown through their evaluation experiments that the

eye-tracking approach produces better results than the automatic saliency map based method, they also showed that the saliency map based method is also effective in directing user's attention to those important positions in the image. Setlur et al. [24] succeeded in using the image saliency to produce vivid stylistic results from offline images or videos. Compared with prior algorithms, our approach is highly parallel, enabling real-time online video processing.

3 Saliency-aware abstraction

3.1 Workflow overview

Our goal is to abstract images by simplifying their visual contents while preserving or even exaggerating most of the perceptually important information. We aim to develop a framework that is fast to compute, temporally coherent, and fully automatic. By incorporating the perception-based progressive abstraction component and the flow-based filtering technique, our method improves the abstraction performance considerably as compared with Winnemöller et al. [27].

The basic workflow of our framework is shown in Fig. 1. The input frame is converted from RGB color-space to CIE-Lab color-space [28] before abstraction, and our approach operates on the luminance channel only. To reduce both the spatial and temporal noise, we first perform the bilateral filter over the frame volume with a small kernel size. Then, a smooth perception-based ROI function is derived from the saliency map [11], and a smooth, coherent, and feature-preserving edge tangent flow (ETF) field is constructed iteratively with a separable approximation. Next, low-contrast regions are smoothed using the FBL filter, and high-contrast regions are strengthened using the coherent and smooth FDoG lines. We then progressively interpolate between the denoised and the abstracted frames using the ROI function. As a result, detail in the background is deemphasized while the visually salient contents are abstracted appropriately. The abstracted image is flattened with a uniform-sized-bin soft quantization to improve the temporal coherence. Lastly, the output image is produced by converting the result back to RGB color-space.

3.2 Video denoising

Input frames often have noise and need to be smoothed first. Kang et al. [13] used the Gaussian blurred image to construct the feature flow and aimed to abstract static images. One limitation of the Gaussian filter is that salient edges are also smoothed. Winnemöller et al. [27] smoothed the spatial noise with the separable bilateral filter and depressed the temporal noise with a soft luminance quantization technique. However, the border between two consecutive quantization levels still exhibits temporal noise because of the input noise. Bilateral filtering is a term coined by Tomasi and Manduchi [25] to refer an edge-preserving filtering technique that takes both geometric closeness and photometric similarity of neighbor pixels into account. To reduce both the spatial and temporal noise, we view the video sequence as a spatio-temporal volume (a block of frames with time as the third dimension) and apply the bilateral filter to the volume.

Given an input frame $I(\mathbf{x})$, where $\mathbf{x} = (x, y)$ is a pixel position, we first perform a 1D bilateral filter to the temporal dimension:

$$F_t(\mathbf{x}) = \frac{1}{f_t} \sum_{n \in [-N,0]} G_{\sigma_t}(n) h_{r_t}\big(\mathbf{x}(n), \mathbf{x}\big) I\big(\mathbf{x}(n)\big) \tag{1}$$

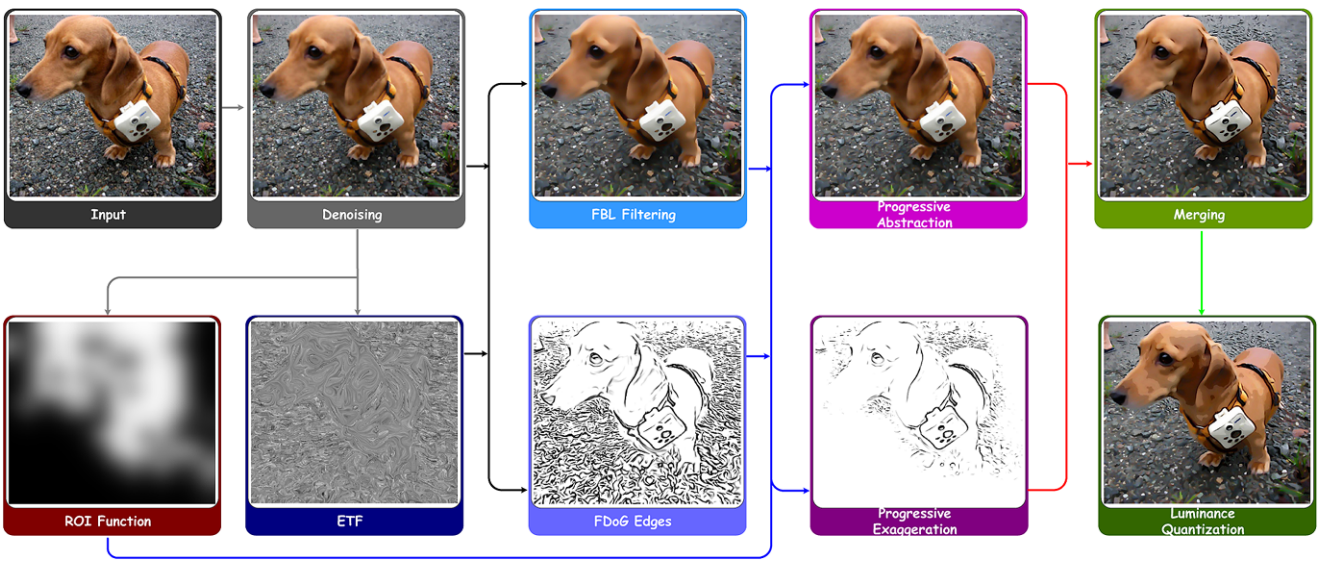


Fig. 1 Workflow overview

$$f_t = \sum_{n\in[-N,0]} G_{\sigma_t}(n) h_{r_t}\big(\mathbf{x}(n), \mathbf{x}\big) \tag{2}$$

where n denotes the nth neighbor frame, $G_\sigma(x)$ represents a univariate Gaussian function of variance σ^2, $h_\sigma(\mathbf{x}, \mathbf{y}) = G_\sigma(\|I(\mathbf{x}) - I(\mathbf{y})\|)$ is the range weight function, and f_t is the weight normalization term.

Then we smooth the spatial noise with a 2D bilateral filter. Chen et al. [4] introduced the bilateral grid to improve the time efficiency for large kernels. A limitation of the bilateral grid is that small spatial kernels require fine sampling, which results in large memory and computation costs. We employ the separable scheme with small kernel size in this paper.

3.3 Edge tangent flow construction

The nonlinear filtering of the ETF field is originally introduced by Kang et al. [12] and accelerated by Kang et al. [13] and Zhao et al. [29]. We construct the ETF $\mathbf{v}(\mathbf{x})$ iteratively using Zhao et al.'s GPU-based method. $\mathbf{v}^0(\mathbf{x})$ is initialized as a vector perpendicular to the image gradient $\mathbf{g}(\mathbf{x}) = \nabla I(\mathbf{x})$. After the ith iteration, $\mathbf{v}^{i+1}(\mathbf{x})$ is obtained using the separable filter defined as follows:

$$\mathbf{v}_h(\mathbf{x}) = \frac{1}{v_h} \sum_{\mathbf{y}\in\Omega_h(\mathbf{x})} w_m(\mathbf{x}, \mathbf{y}) w_d(\mathbf{x}, \mathbf{y}) \mathbf{v}(\mathbf{y}) \tag{3}$$

$$\mathbf{v}_v(\mathbf{x}) = \frac{1}{v_v} \sum_{\mathbf{y}\in\Omega_v(\mathbf{x})} w_m(\mathbf{x}, \mathbf{y}) w_d(\mathbf{x}, \mathbf{y}) \mathbf{v}(\mathbf{y}) \tag{4}$$

where $\Omega_h(\mathbf{x})$, $\Omega_v(\mathbf{x})$ denote the horizontal direction and vertical direction neighborhood of $\mathbf{x}$, respectively, and v_h, v_v are the corresponding vector normalization terms.

The magnitude weight function w_m is defined as

$$w_m(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\big(1 + \tanh\big(\hat{g}(\mathbf{y}) - \hat{g}(\mathbf{x})\big)\big) \tag{5}$$

where $\hat{g}(\mathbf{x})$ is the gradient magnitude at $\mathbf{x}$. We can see that w_m ranges in [0, 1]. When a neighbor pixel's gradient magnitude is higher, this weight function value is bigger, and vice versa. This ensures the preservation of the dominant edge directions.

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

$$w_d(\mathbf{x}, \mathbf{y}) = \mathbf{v}(\mathbf{x}) \cdot \mathbf{v}(\mathbf{y}) \tag{6}$$

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 $\mathbf{v}(\mathbf{y})$ is reversed before smoothing, avoiding swirling flows.

The initial gradient field $\mathbf{g}^0(\mathbf{x})$ is calculated using the Sobel operator, and $\mathbf{v}^0(\mathbf{x})$ can be easily obtained. Then the modified filter is iteratively applied to smooth the ETF. The filter actually uses a separable approximation to construct the ETF field, and thus we can parallelize it with two render passes on modern graphics hardware. It is known that the texture access operation is a bottleneck of current graphics hardware. To reduce the filtering time, $\hat{g}$ and $\mathbf{v}$ are rendered to the same render target in each render pass. As a result, we only need one texture lookup for each neighbor pixel. Note that the gradient field $\mathbf{g}$ evolves accordingly whereas the magnitude $\hat{g}$ is unchanged. In order to better illustrate the filtering effect, we visualize the ETF using line integral convolution [3]. Figure 2 shows ETF fields after each iteration.

3.4 Flow-based filtering

Instead of the isotropic kernel, this paper employs a flow-guided kernel to enhance the filtering quality, which is illustrated in Fig. 3. Let $c(s, \mathbf{x})$ denote the flow curve at $\mathbf{x}$, where s is an arc-length parameter that may take positive or negative values, and $c(0, \mathbf{x}) = \mathbf{x}$. Also, let $l(t, c(s, \mathbf{x}))$ denote a line segment that is perpendicular to $\mathbf{v}(c(s, \mathbf{x}))$ and intersecting $c(s, \mathbf{x})$, where t is an arc-length parameter, and $l(0, c(s, \mathbf{x})) = c(s, \mathbf{x})$. Note that $l(t, c(s, \mathbf{x}))$ is parallel to the gradient vector $\mathbf{g}(c(s, \mathbf{x}))$.

When performing the filtering, again the two-pass separation trick is employed to sample $2\alpha \times 2\beta$ points in the flow-based kernel. In the first pass, the point positions are computed by bidirectionally following the line segment l starting from $\mathbf{x}$. We begin with $\mathbf{z} = \mathbf{x}$ and iteratively obtain

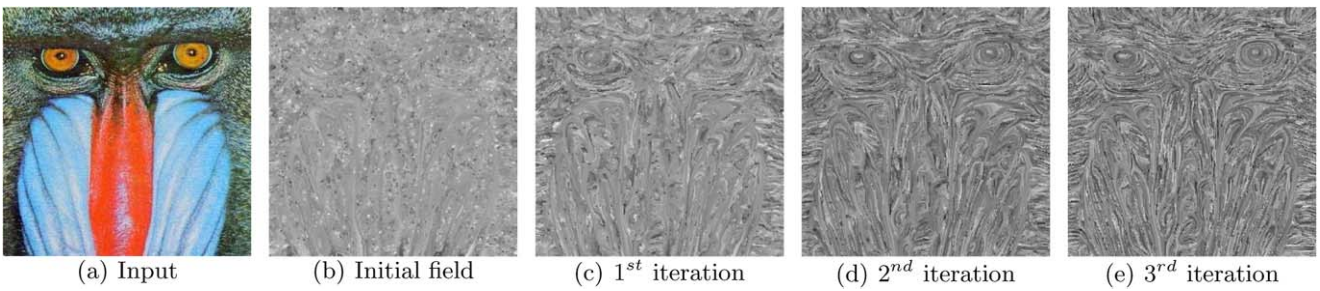
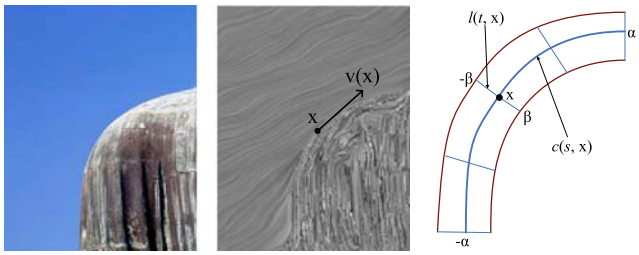


Fig. 2 Iterative ETF construction



(a) Input (b) ETF (c) Kernel at $\mathbf{x}$

Fig. 3 Convolution kernel

the next sample point by moving along $\mathbf{g}(\mathbf{x})$ in positive direction: $\mathbf{z} = \mathbf{z} + \mathbf{g}(\mathbf{z})$; similarly, we sample the points on the other half using $-\mathbf{g}(\mathbf{z})$. Then in the second pass, the point positions are sampled along the flow curve c using the same method. Only $2\alpha + 2\beta$ points are sampled for each pixel, while no quality degradation ensues from this conversion. To further improve the efficiency, we pack the luminance channel and the flow vector into one texture before convolution. By taking advantage of graphics hardware's bilinear interpolation, we only need one indirect texture lookup to sample a point.

Now the edge-preserving smoothing and line extraction filters of Winnemöller et al. are modified so that they are adapted to a curved kernel which follows the local edge flow.

As described in [13], the FBL filter is defined as follows:

$$F_g(\mathbf{x}) = \frac{1}{f_g} \sum_{t\in[-\beta,\beta]} G_{\sigma_g}(t) h_{r_g}\big(l(t,\mathbf{x}),\mathbf{x}\big) I\big(l(t,\mathbf{x})\big) \tag{7}$$

$$F_e(\mathbf{x}) = \frac{1}{f_e} \sum_{s\in[-\alpha,\alpha]} G_{\sigma_e}(s) h_{r_e}\big(c(s,\mathbf{x}),\mathbf{x}\big) I\big(c(s,\mathbf{x})\big) \tag{8}$$

$$f_g = \sum_{t\in[-\beta,\beta]} G_{\sigma_g}(t) h_{r_g}\big(l(t,\mathbf{x}),\mathbf{x}\big) \tag{9}$$

$$f_e = \sum_{s\in[-\alpha,\alpha]} G_{\sigma_e}(s) h_{r_e}\big(c(s,\mathbf{x}),\mathbf{x}\big) \tag{10}$$

Note that f_g and f_e are the weight normalization terms. Figure 4 shows the abstraction difference between the FBL filter and the standard bilateral filter. We can see that the FBL filter improves the performance of the standard bilateral filter in terms of enhancing shapes and feature directionality.

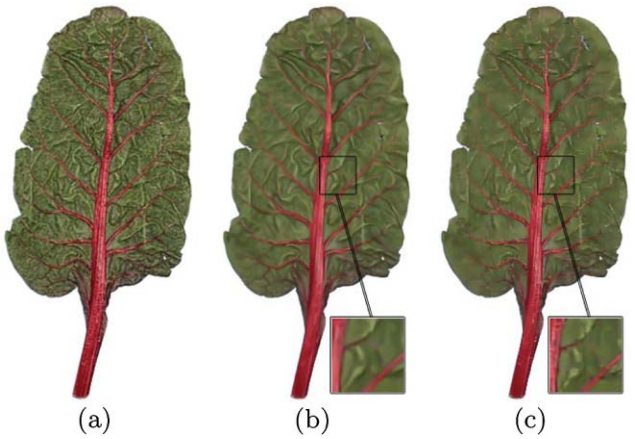
Similarly, the FDoG filter is formulated as

$$D_g(\mathbf{x}) = \frac{1}{d_g} \sum_{t\in[-\beta,\beta]} \big(G_{\sigma_c}(t) - \rho \cdot G_{\sigma_s}(t)\big) I\big(l(t,\mathbf{x})\big) \tag{11}$$

$$D_e(\mathbf{x}) = \frac{1}{d_e} \sum_{s\in[-\alpha,\alpha]} G_{\sigma_m}(s) I\big(c(s,\mathbf{x})\big) \tag{12}$$

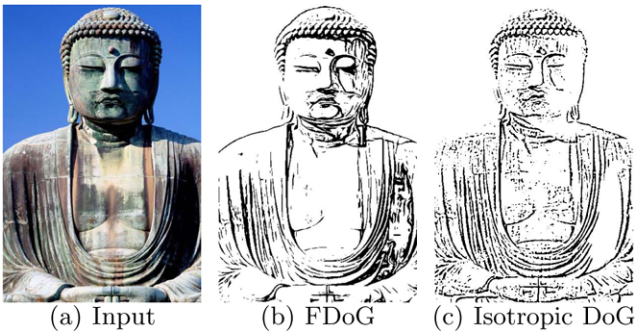
$$d_g = \sum_{t\in[-\beta,\beta]} \big(G_{\sigma_c}(t) - \rho G_{\sigma_s}(t)\big) \tag{13}$$

$$d_e = \sum_{s\in[-\alpha,\alpha]} G_{\sigma_m}(s) \tag{14}$$



(a) (b) (c)

Fig. 4 Comparison of (**b**) the FBL filter with (**c**) the standard bilateral filter



(a) Input (b) FDoG (c) Isotropic DoG

Fig. 5 Comparison of (**b**) the FDoG filter with (**c**) the isotropic DoG technique

where d_g and d_e are the corresponding weight normalization terms. The parameters σ_c and σ_s control the sizes of the center interval and the surrounding interval, respectively, and we set $\sigma_s = 1.6\sigma_c$. Thus σ_c determines the line thickness scale as the isotropic DoG method does. The threshold level ρ determines the sensitivity of the edge detector and typically ranges in $[0.97, 1.0)$. Note that the user-defined bandwidths of the Gaussian σ_e, σ_g, σ_c, and σ_m automatically determine the 1D convolution sizes and thus the sampling numbers α and β.

Once we obtain D_e for the input image from (12), the lines can be extracted. Rather than converting them to a black-and-white image by binary thresholding as used in [12] and [13], which may introduce temporal artifacts, we use a slightly smoothed step function as suggested in [27] and [29]:

$$L(\mathbf{x}) = \begin{cases} 1, & D_e(\mathbf{x}) > 0 \\ 1 + \tanh(\varphi_e \cdot D_e(\mathbf{x})), & \text{otherwise} \end{cases} \tag{15}$$

where φ_e is a parameter controlling the line sharpness. As a result, small discontinuities are omitted, and salient edges are strengthened. Figure 5 shows the comparison of

the FDoG method with the isotropic DoG filter. Unnecessary high-frequency features in the image are smoothed out, whereas the major edges are smoother and more coherent.

Kyprianidis and Döllner [15] estimated a similar ETF by performing an eigenvalue decomposition and linear Gaussian filtering. In contrast, the approach we used is actually a modified bilateral filter, taking both the magnitude difference and the direction difference into account. When performing separable flow-based filtering, we sample a point by 4-pixel bilinear interpolation as in [13], whereas Kyprianidis and Döllner interpolated each point with only two axis-aligned neighboring pixels, and their sampling points are not uniformly distributed along the flow curve. Most importantly, their approach assumes that high contrast is linked to high visual salience and may fail to accentuate an object against its background for the images with objects of low contrast over background of high contrast. In this work, we further employ a more elaborate visual perception model to improve the abstraction results.

3.5 ROI function computation

In this section, we present our novel algorithm to compute the perception-based ROI function. The architecture of the proposed model is illustrated in Fig. 6.

We begin with our acceleration techniques to compute the bottom-up image saliency. Three feature maps are employed for the luminance, color, and orientation information. The orientation information is obtained using four oriented Gabor filters [10] and represents the contrast variation. In addition to the color and luminance, the contrast itself is also an important feature, and the contrast variation can be of perceptually importance. In order to improve the time efficiency, the color and luminance are stored in one texture, and four orientations in another texture.

First, we need to construct two multiscale Gaussian pyramids (B_C, B_L) and one multiscale Gabor pyramid (B_O). We accelerate the pyramid construction by building a mipmap texture for the finest image as in [17], since the fast texture lookup operation for a coarser scale corresponds to the magnification of an image. We apply the Gaussian filter and Gabor filter to the finest mipmap level and approximate the coarser levels of the mipmap texture using the hardware bilinear mipmap generation. The Gabor filter can be implemented with a convolution mask. Repeated convolutions of Bartlett filters are equivalent to approximations of Gaussian filters if all nonzero matrix components of the resulting filters are considered [14]. To further improve the performance, we approximate the Gaussian filter with repeated Bartlett convolutions. The Bartlett convolution is defined as follows:

$$B_{3\times 3} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \otimes \frac{1}{4}\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{16}$$

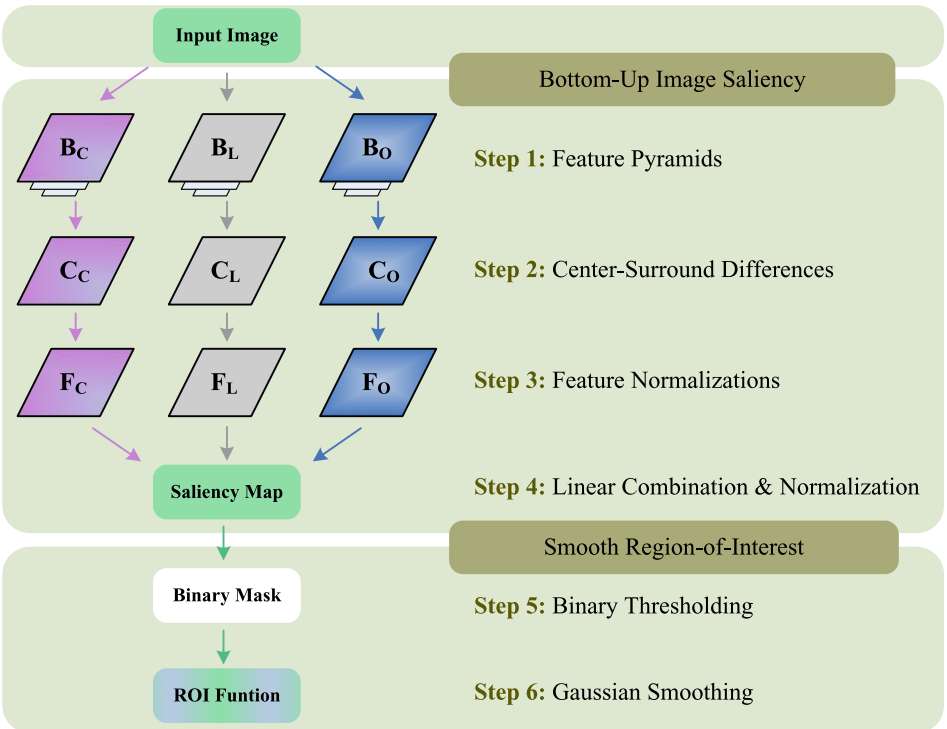


Fig. 6 General architecture of the model. The saliency map and ROI function are produced automatically

By taking advantage of the hardware bilinear interpolation, the convolution requires only two texture lookups with appropriate index biasing. An $n \times n$-sized Gaussian filter can be approximated with $(n-1)/2$ convolutions of the Bartlett filter and thus only $n-1$ bilinear texture lookups are required. For comparison, a separable Gaussian filter would require $2n$ nearest-neighbor texture reads.

In the second step, we take center-surround differences to get a contrast map (C_C, C_L, C_O) for each feature. We only need one bilinear texture lookup per mipmap level.

Next, each contrast map is normalized in the range $[0, 1]$ and scaled by $(1-\bar{m})^2$, where $\bar{m}$ is the global average of the map. The saliency map S is obtained by summing up the three feature maps (F_C, F_L, F_O). Again, we apply the normalization operation to S. The normalization operation is the major time-consuming component, which needs global statistic quantities (maximum, minimum, and average). There are three methods to evaluate these quantities:

1. One method is known as a reduction technique on the GPU. As suggested in [1, 30], we perform repeated downsamplings, averaging four neighboring values down to one in each pass. However, this method requires approximately $\log_2 n$ render passes, where n is the maximum of the width or height of the input image.
2. Another one is to evaluate these values on the CPU. Colbert et al. [5] chose this method to implement their real-time High-Dynamic-Range image painting. In their paper they pointed out that the expensive downsampling approach is due to the hardware design of the GPU, whereby the processor does not have the necessary registers. However, this CPU-based method needs reading back texture data from video memory.
3. Modern graphics hardware natively supports the alpha blend of floating-point data. The statistic quantities can be calculated by performing a scatter operation [4, 23] using vertex shader into a render target with one-pixel size. As the GPU supports different blend operations for the color channels and alpha channel, we can obtain multiple quantities with a single pass. The algorithm is particular efficient for render targets with large size. For the render target with one-pixel size in our case, frequent alpha blend operation is applied to a single pixel and limits the parallelism of the GPU.

We have implemented these three algorithms and find that the CPU-based method is the fastest on our PC. We believe that the GPU-based scattering method to calculate the global quantities will outperform the CPU-based method with the development of parallelism of GPUs in near future.

Lee et al. [17] created their saliency map using two image features (luminance and hue) and three 3D dynamic features (depth, object size, and object motion). In case of videos or images, 3D dynamic features are unknown. Although we may reconstruct 3D information using vision-based techniques, the robustness and the speed of the reconstruction process are usually not satisfactory. In addition to luminance and hue, the contrast itself is also an important feature. Thus we generate our saliency map using three image features. Furthermore, we employ the Bartlett filter to even improve the efficiency. Figure 7 shows our GPU-based result and Itti et al.'s CPU-based one. Three feature maps for color contrast, luminance contrast, and orientation contrast are fast extracted and combined to form a saliency map (S). It can be seen that both maps indicate the attended regions well.

Next, we obtain a binary mask from the saliency map using a user specified threshold T_m. Larger T_m indicates smaller salient regions, while smaller T_m generates larger salient ones. Empirically, $T_m \in [0.3, 0.7]$ produces good result. From this mask, we derive a smooth ROI function by blurring the mask with a large kernel which approximates the Gaussian filter (see Fig. 8b). By designing the ROI function in this way, we make sure that it can be used to control the progressive abstraction without introducing sudden gaps on the boundary. As suggested in [21], an image blur with a large Gaussian kernel can be implemented in a highly efficient manner using image pyramids. The large Gaussian blur

(a) F_C (b) F_L (c) F_O (d) S (e) S'

Fig. 7 Example of operation of the model with the input Fig. 8 a. Parallel feature extraction using the GPU yields three feature maps for color contrast, luminance contrast, and orientation contrast. These are combined to form the saliency map (S). For comparison, image (**e**) shows the CPU-based result S' of Itti et al.'s algorithm. As we can see from S and S', both images indicate the attended region well

Fig. 8 Our approach takes as input (**a**) an image, and derives (**b**) an ROI function. Images (**c**) and (**d**) are generated at different abstraction scales using Winnemöller et al.'s method, whereas the abstraction effect using our new saliency-aware algorithm is shown in image (**e**). Notice how detail is deemphasized in the background (guitarist's feet and ground), while the foreground visual contents (guitarist's face and body) are abstracted appropriately in image (**e**)

is implemented by generating a mipmap texture, and then upsampling from the coarser level with nearest-neighbor interpolation while applying a small (5×5) Gaussian blur at each level. Again we accelerate the Gaussian blur with two Bartlett convolutions.

3.6 Saliency-based progressive abstraction

As shown in Figs. 8c and 8d, the local contrast-based method proposed by Winnemöller et al. fails to emphasize the ROI (guitarist) when abstracting the background. We incorporate an automatic progressive abstraction component based on a more accurate visual perception model compared with Winnemöller et al.'s method.

First, the ROI function is used to linearly interpolate between the denoised frame and the bilareral filtered image. As the FBL filter operates on a local kernel, background regions with very high contrast cannot be abstracted appropriately. One solution is to increase the bandwidths of the flow-based filtering. However, it is still hard to obtain good results for images where background regions have very high contrast. Again, we utilize the ROI function to linearly interpolate between the image before and after coherent line extraction. As a result, the detail edges in background are simply removed regardless of their contrast (see Fig. 1). As our expectation, the visual contents are less simplified and more line features are emphasized in salient regions, whereas the pixels are more abstracted and less lines are emphasized in the background (see Fig. 8e).

We employ many techniques to preserve good temporal coherence, which is quite an important issue for video manipulations. We have reduced the input noise using a volume-based bilateral filter. In addition, the ETF and the ROI function are smoothly filtered before abstracting the frame. Rather than using a binary model of cell-activation, our FDoG edges are extracted using a slightly smoothed step function, which increases temporal coherence in animations. We also perform a soft quantization step on the abstracted images as suggested by Winnemöller et al. The luminance quantization technique can result in cartoon or paint-like effects. Another significant advantage is temporal coherence. As a result, sudden changes are further subdued in low-contrast regions, where they would be most objectionable. We expect a more sophisticated technique [2] to improve the temporal coherence but would be hard to achieve real-time performance.

4 Experimental results

We have developed a new automatic real-time saliency-aware 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 64bit operating system.

Figure 7 has shown the effectiveness comparisons of the saliency map computation between our GPU-based algorithm and the original CPU-based implementation from Fig. 8a. We approximate the pyramid construction using the hardware mipmap generation, which may blur some features. Itti et al. implemented the scaling operation using $(1 - \bar{m}')^2$, where $\bar{m}'$ denotes the average of all local maxima. We simplify this operation with $(1 - \bar{m})^2$. As a result, salient features in map S' seems clearer than in map S. We prefer an algorithm that is fast enough while can indicate visually salient regions to guide our progressive abstraction. Since the saliency map generated using our GPU-accelerated method indicates the attended region well, the approximate map meets our requirement.

The time comparisons are listed in Table 1. From this table we can see that our GPU-based implementation outperforms the naive CPU-based method by 2 orders of magnitude speedup. Our approximate algorithm can also be implemented on the CPU, but the efficiency will drop down without parallelization.

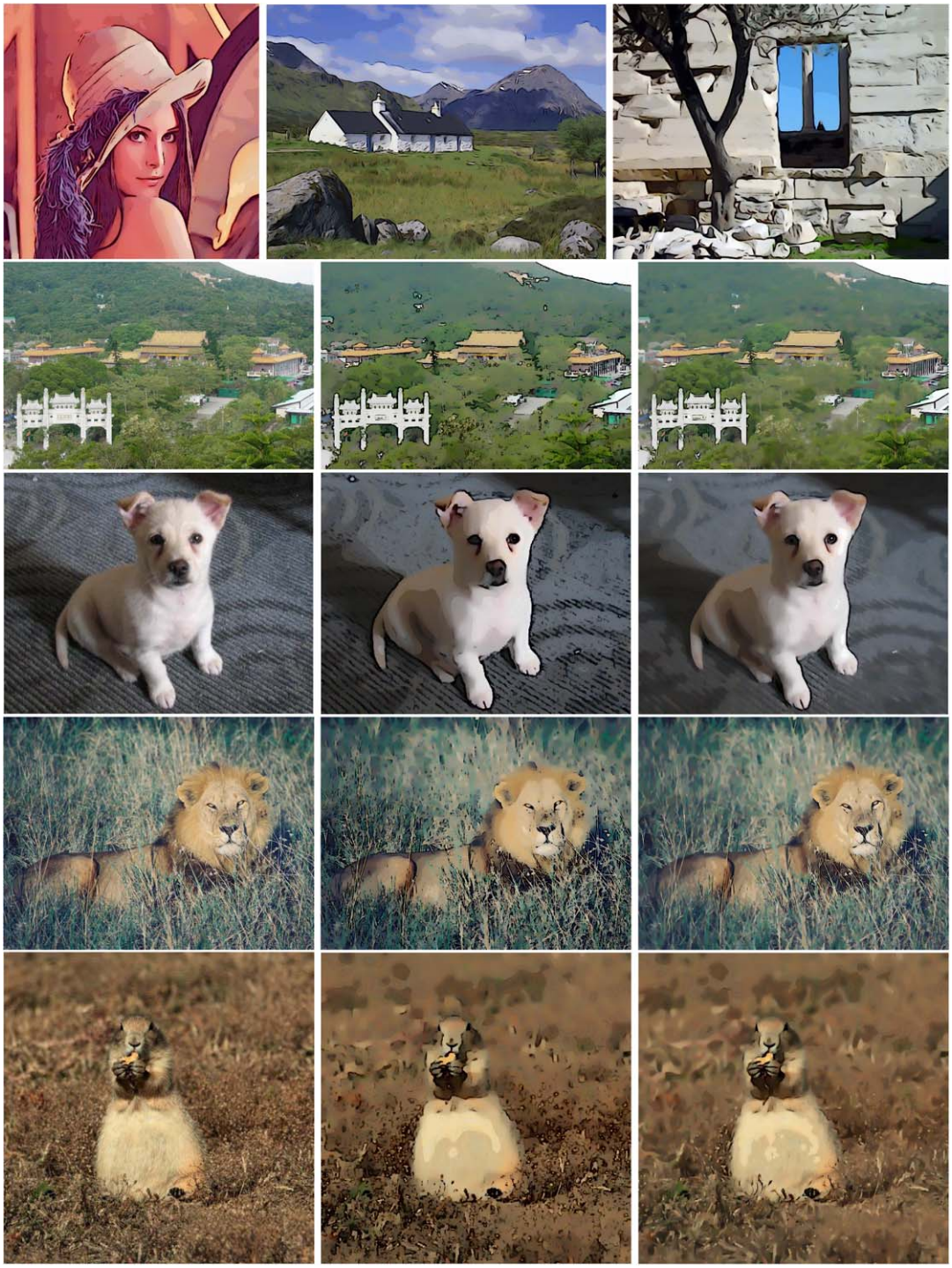


Fig. 9 The three images in the first row are abstracted using the flow-based filtering without the progressive approach. In the lower four rows, we compare the abstraction effects of various (*left column*) input images between (*right column*) our progressive saliency-aware approach and (*middle column*) Winnemöller et al.'s method

Figure 9 shows the comparison of various abstraction results between our saliency-aware approach and Winnemöller et al.'s method. A small region of low contrast (such as the houses and animals in the figures) surrounded by a large region of high contrast (such as the carpet and grass) can be visually salient. With Winnemöller et al.'s method, this small region of low contrast will become less salient after being low-pass filtered, while the large region will be emphasized with high-pass filtering. Our method, however, can extract such small region as the ROI based

Table 1 Running time (ms) comparisons of the saliency map computation between Itti et al's approach and our GPU-based method. The image size is in mega-pixels

Image size	1	2	3	4	5
Itti et al.'s	1,715	3,676	5,943	8,109	9,664
Ours	14	26	38	55	69

Table 2 Running times (ms) of our abstraction system. The other computations include CPU-GPU video stream transfer, the color-space conversion, interpolation with the ROI function, and the soft luminance quantization. The image size is in mega-pixels

Image size	1.5	1.0	0.8	0.5
Denoising	10	7	6	4
ROI function	21	15	12	8
ETF construction	15	10	8	5
FBL filtering	27	20	16	10
Line extraction	6	5	5	3
Other computations	6	4	2	2
Overall performance	85	61	49	32

on the saliency map, which is obtained by combining the center-surround difference of various features. Note that the underlying features in the images are also stylized because of the flow-based filtering. Moreover, the accompanying video demonstrates that our video abstraction system can generate vivid visual perception and exhibits good temporal coherence. 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.

The render time for each stage and the overall performance are listed in Table 2. We set $\alpha = \beta = 5$ and $\alpha = \beta = 3$ for FBL and FDoG filters, respectively. The kernel radius is 3 for the 3D separable bilateral filter and 5 for ETF construction. Both the ETF and FBL filters are iterated twice per frame, whereas the denoising is applied once. Note that both the CPU-GPU communication and the pure GPU processing are accounted for the reported overall performance. The ROI function generation stage and the FBL filtering are the major processing bottlenecks limiting the overall performance. Nevertheless, we are able to perform the progressive video abstraction at 11 Hz even for images with 1.5 mega-pixels, satisfying the real-time requirement on most PC monitors.

5 Conclusions and future work

In this paper, we present a novel real-time saliency-aware abstraction system that automatically abstracts videos and images based on a more accurate visual perception model compared with Winnemöller et al.'s method. We propose a progressive abstraction method based on an ROI function, which is derived from the notion of saliency map. Detail in background is removed, while image pixels in salient regions are abstracted appropriately. We demonstrate how the saliency map generation algorithm achieves real-time performance using modern graphics hardware. Moreover, we employ a feature flow field to guide our anisotropic abstraction. The whole pipeline in this paper is implemented automatically in real time without any user intervention. At last, the experimental results demonstrate both the feasibility and efficiency of our proposed algorithms.

Since saliency maps have been widely applied in many computer vision and graphics problems, we believe that most of those applications will benefit from our new GPU-based real-time saliency map implementation algorithm.

The abstraction technique is performed after transforming original images into *CIE-Lab* color-space. It works well in most cases since luminance carries lots of the feature information. However, when iso-luminance color regions "touch" each other in the input image, our method tends to fail in generating a correct rendition. Moreover, the filters used in this paper have linear time-complexity with respect to the kernel radius. However, the performance for large kernels is still not satisfactory. We expect that these filters may be accelerated by image downsampling or grid processing [4]. Limitations mentioned above will be addressed in our future work. In addition, our future work includes exploring better methods for very large High-Definition video, extending to real-time HDR image and video abstraction, and investigating the extension to other saliency based applications.

Current design of the alpha blend stage can only support limited and fixed function blending operations, and thus the calculation of global quantities in an image is even slower than the CPU implementation. We would like to see a more powerful and programmable blend stage extending current functionality, similar to simple pixel shader functionality.

Acknowledgements The authors would like to thank our anonymous reviewers for their dedicated help in improving the paper. Many thanks also to Xiaoyan Luo, Mingdong Zhou, and Shufang Lu for their help in presenting the manuscript. This work was supported by the National Key Basic Research Foundation of China (Grant No. 2009CB320801), the National Natural Science Foundation of China (Grant Nos. 60533080 and 60833007), and the Key Technology R&D Program (Grant No. 2007BAH11B03).

References

1. Bolz, J., Farmer, I., Grinspun, E., Schröder, P.: Sparse matrix solvers on the GPU: conjugate gradient and multigrid. ACM Trans. Graphics (SIGGRAPH '03) **22**, 3 (2003)
2. Bousseau, A., Neyret, F., Thollot, J., Salesin, D.: Video watercolorization using bidirectional texture advection. ACM Trans. Graph. (SIGGRAPH '07) **26**, 3 (2007)

3. Carbral, B., Leedom, L.: Imaging vector fields using line integral convolution. In: Proc. ACM SIGGRAPH '93 (1993), pp. 263–270
4. Chen, J., Paris, S., Durand, F.: Real-time edge-aware image processing with the bilateral grid. ACM Trans. Graph. (SIGGRAPH '07) **26**, 3 (2007)
5. Colbert, M., Reinhard, E., Hughes, C.E.: Painting in high dynamic range. J. Vis. Commun. Image Represent. **18**, 5 (2007)
6. Collomosse, J.P., Hall, P.M.: Cubist style rendering from photographs. IEEE Trans. Vis. Comput. Graph. **9**, 4 (2003)
7. Collomosse, J.P., Rowntree, D., Hall, P.M.: Stroke surfaces: temporally coherent artistic animations from video. IEEE Trans. Vis. Comput. Graph. **11**, 5 (2005)
8. DeCarlo, D., Santella, A.: Stylization and abstraction of photographs. ACM Trans. Graph. (SIGGRAPH '02) **21**, 3 (2002)
9. Farbman, Z., Fattal, R., Lischinski, D., Szeliski, R.: Edge-preserving decompositions for multi-scale tone and detail manipulation. ACM Trans. Graph. (SIGGRAPH '08) **27**, 3 (2008)
10. Greenspan, H., Belongie, S., Goodman, R., Perona, P., Rakshit, S., Anderson, C.H.: Overcomplete steerable pyramid filters and rotation invariance. In: Proc. IEEE Computer Vision and Pattern Recognition (1994), pp. 222–228
11. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. IEEE Trans. Pattern Anal. Mach. Intell. **20**, 11 (1998)
12. Kang, H., Lee, S., Chui, C.K.: Coherent line drawing. In: Proc. ACM Sym. Non-Photorealistic Animation and Rendering (NPAR '07). ACM (2007), pp. 43–50
13. Kang, H., Lee, S., Chui, C.K.: Flow-based image abstraction. IEEE Trans. Vis. Comput. Graph. **15**(1), 62–76 (2009)
14. Kraus, M., Strengert, M.: Pyramid filters based on bilinear interpolation. In: Proc. International Conf. of Computer Graphics Theory and Applications (GRAPP '07) (2007), pp. 21–28
15. Kyprianidis, J.E., Döllner, J.: Image abstraction by structure adaptive filtering. In: Proc. EG UK Theory and Practice of Computer Graphics (2008), pp. 51–58
16. Lee, C.H., Varshney, A., Jacobs, D.W.: Mesh saliency. ACM Trans. Graph. (SIGGRAPH '05) **24**, 3 (2005)
17. Lee, S., Kim, G.J., Choi, S.: Real-time tracking of visually attended objects in interactive virtual environments. In: Proc. ACM Sym. Virtual Reality Software and Technology (VRST '07) (2007), pp. 29–38
18. McCloud, S.: Understanding Comics. Harper Collins Publishers, New York (1993)
19. Orzan, A., Bousseau, A., Barla, P., Thollot, J.: Structure-preserving manipulation of photographs. In: Proc. ACM Sym. Non-Photorealistic Animation and Rendering (NPAR '07) (2007), pp. 103–110
20. Pham, T.Q., van Vliet, L.J.: Separable bilateral filtering for fast video preprocessing. In: Proc. IEEE International Conf. Multimedia and Expo (ICME '05) (2005), pp. 454–457
21. Rempel, A.G., Trentacoste, M., Seetzen, H., Young, H.D., Heidrich, W., Whitehead, L., Ward, G.: Ldr2hdr: on-the-fly reverse tone mapping of legacy video and photographs. ACM Trans. Graph. (SIGGRAPH '07) **26**, 3 (2007)
22. Santella, A., DeCarlo, D.: Visual interest and NPR: an evaluation and manifesto. In: Proc. ACM Sym. Non-Photorealistic Animation and Rendering (NPAR '04) (2004), pp. 71–78
23. Scheuermann, T., Hensley, J.: Efficient histogram generation using scattering on GPUs. In: Proc. ACM Sym. Interactive 3D Graphics and Games (I3D '07) (2007), pp. 33–37
24. Setlur, V., Lechner, T., Nienhaus, M., Gooch, B.: Retargeting images and video for preserving information saliency. IEEE Comput. Graph. Appl. **27**, 5 (2007)
25. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proc. IEEE International Conf. Computer Vision (ICCV '98) (1998), pp. 839–846
26. Wang, J., Xu, Y., Shum, H.Y., Cohen, M.F.: Video tooning. ACM Trans. Graph. (SIGGRAPH '04) **23**, 3 (2004)
27. Winnemöller, H., Olsen, S.C., Gooch, B.: Real-time video abstraction. ACM Trans. Graph. (SIGGRAPH '06) **25**, 3 (2006)
28. Wyszecki, G., Stiles, W.S.: Color science: concepts and methods, quantitative data and formulae. Wiley, New York (1982)
29. Zhao, H., Jin, X., Shen, J., Mao, X., Feng, J.: Real-time feature-aware video abstraction. Vis. Comput. (CGI '08) **24**, 7 (2008)
30. Ziegler, G., Tevs, A., Theobalt, C., Seidel, H.-P.: GPU point list generation through histogram pyramids. In: Proc. 11th Fall Workshop on Vision, Modeling, and Visualization (VMV '06) (2006), pp. 133–141

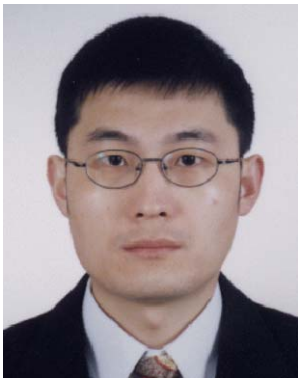
Hanli Zhao is a PhD candidate of the State Key Lab of CAD & CG, Zhejiang University, China. He received his BSc degree in software engineering from Sichuan University in 2004. His research interests include non-photorealistic rendering, image processing, and general-purpose GPU computing.

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 MS and PhD in computer science from Tokyo University.

Xiaogang Jin is a professor of the State Key Lab of CAD & CG, Zhejiang University, China. He received his BSc degree in computer science in 1989, MSc and PhD 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.

Jianbing Shen is currently an associate professor in the School of Computer Science and Technology, Beijing Institute of Technology, China. He received his PhD 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.

Feifei Wei is a PhD candidate of the State Key Lab of CAD & CG, Zhejiang University, China. He received his BSc degree in computer science from He'nan University in 2003 and his MSc degree in computer science from Northeast University in 2006. His research interests include real-time rendering and general-purpose GPU computing.



Jieqing Feng is a professor of 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.