# Flow-Based Image Abstraction

Henry Kang, *Member*, *IEEE*, Seungyong Lee, *Member*, *IEEE*, and Charles K. Chui, *Fellow*, *IEEE*

**Abstract**—We present a nonphotorealistic rendering technique that automatically delivers a stylized abstraction of a photograph. Our approach is based on shape/color filtering guided by a vector field that describes the flow of salient features in the image. This flow-based filtering significantly improves the abstraction performance in terms of feature enhancement and stylization. Our method is simple, fast, and easy to implement. Experimental results demonstrate the effectiveness of our method in producing stylistic and feature-enhancing illustrations from photographs.

**Index Terms**—Nonphotorealistic rendering, image abstraction, flow-based filtering, line drawing, bilateral filter.

## 1 Introduction

Nonphotorealistic rendering (NPR) in general involves abstraction and stylization of the target scene, which helps simplify the visual cues and convey certain aspects of the scene more effectively. For example, lines can be a simple yet effective tool for describing *shapes*, as demonstrated in many technical or artistic illustrations. *Line drawing* thus has drawn a lot of attention in recent NPR research, mainly focused on extracting lines from 3D models [1], [2], [3], [4], [5], [6], [7], [8]. However, attempts on making pure line drawings from photographs have been rare, in part due to the difficulty of identifying shapes that are implicitly embedded in a raw image, without depth information and often corrupted by noise.

While *color* may not be the essential ingredient in conveying shapes, NPR often paints object surfaces with a restricted set of colors to further assist the process of visual information transfer and subject identification. This is often witnessed in the cartoon renderings of 3D objects [9], [10], [11], where abstracted colors not only add to the stylistic look of the rendering, but also help convey the scene information in a clear and concise fashion. A raw photograph, however, can pose bigger challenges in achieving such cartoon-style color simplification, as it again involves nontrivial tasks of shape recognition and noise suppression.

In this paper, we present an automatic technique that generates a stylistic visual abstraction from a photograph. Our method is designed to convey both *shapes* and *colors* in an abstract but feature-preserving manner. First, it captures important shape boundaries in the scene and displays them with a set of smooth, coherent, and stylistic lines. Second, it abstracts the interior colors to remove unimportant details on the object surface while preserving and enhancing local shapes. What separates our approach from previous abstraction techniques is the use of a *flow-based filtering* framework. We employ existing filters for *line extraction* and *region smoothing* and adapt them to follow a highly anisotropic kernel that describes the "flow" of salient image features. We show that our approach improves the abstraction performance considerably in terms of feature enhancement and stylization, resulting in the production of a high-quality illustration from a photograph that effectively conveys important visual cues to the viewer. Such information reduction could facilitate quick data deciphering, as well as efficient data transmission over the network.

### 1.1 Problem Statement

Given an image that we view as a height field of pixel intensities, the task of image abstraction involves the following subproblems:

1. **Line extraction.** Capture and display "significant" height discontinuities.
2. **Region smoothing.** Remove all "insignificant" height discontinuities.

Solving the first problem results in a "line drawing" (see Fig. 1b), while the second results in a "smoothed" or "flattened" height field (see Fig. 1c). The combination of these two solutions often results in a cartoonlike image (see Fig. 1d). A line drawing is by itself an extreme case of image abstraction, since all the pixel colors except at edges are "flattened down" to the same level (white).

### 1.2 Related Work

Many of the existing image-based NPR techniques are intended to serve artistic purposes, that is, to elicit an aesthetic response from the viewer. These include painting [12], [13], [14], [15], pen-and-ink illustration [16], [17], pencil drawing [18], [19], stipple drawing [20], [21], mosaics [22], engraving [23], and cubist rendering [24], [25].

On the other hand, another paradigm exists for image-guided NPR, which we call *image abstraction*, that focuses more on facilitating visual communication and data reduction. Our present paper falls in this category. This line of work concerns capturing and conveying important image features while minimizing possible distractions from unimportant details. As *shape* and *color* are two of the most

- *H. Kang and C.K. Chui are with the Department of Mathematics and Computer Science, University of Missouri, St. Louis, One University Blvd., St. Louis, MO 63121. E-mail: kang@cs.umsl.edu, chui@arch.umsl.edu.*
- *S. Lee is with the Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, 790-784, South Korea. E-mail: leesy@postech.ac.kr.*

*Manuscript received 26 Oct. 2007; revised 29 Mar. 2008; accepted 29 Apr. 2008; published online 9 May 2008.*
*Recommended for acceptance by A. Hertzmann.*
*For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2007-10-0167.*
*Digital Object Identifier no. 10.1109/TVCG.2008.81.*

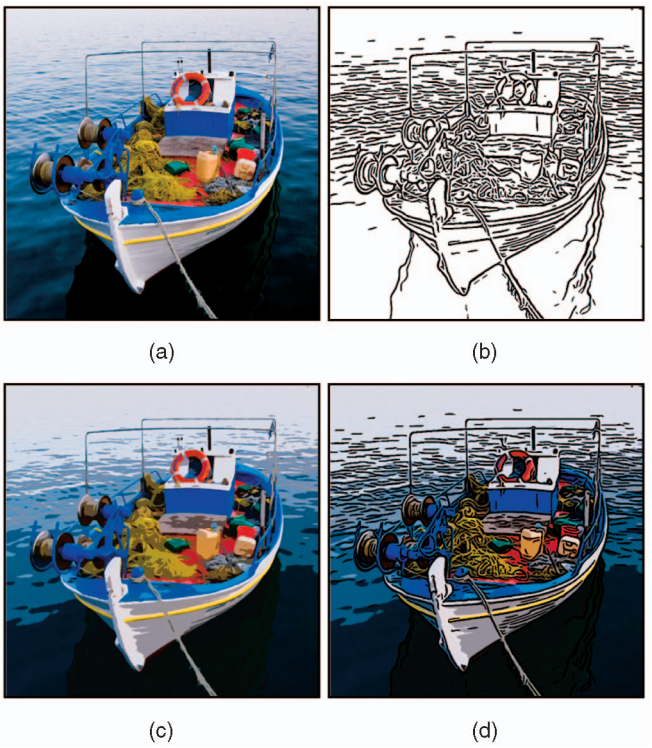



Fig. 1. Image abstraction by our method. (a) Input image. (b) Line extraction. (c) Region flattening. (d) Combined.

important features to convey, the existing approaches have focused on solving the corresponding two problems of *line drawing* and *region smoothing*, which we described in Section 1.1.

DeCarlo and Santella [26] employed the Canny edge detector [27] and the mean-shift filter [28] to obtain a cartoon-style image abstraction. They use the edge detector to produce line drawing, while the mean-shift filter performs region smoothing and segmentation. They also provide an eye-tracking-based user interface to allow for a user-guided specification of regional importance, which together with the hierarchical structuring of segmented regions, enables adaptive control of the level of abstraction. Wang et al. [29] developed an anisotropic mean-shift filter and applied it to create a sequence of image abstractions from a video. Collomosse et al. [30] similarly used the mean-shift filter to solve an offline video abstraction problem, focusing on achieving good spatiotemporal coherence. Wen et al. [31] presented a system that produces a rough sketch of the scene, again based on mean-shift filtering.

One limitation of the mean-shift segmentation is that it typically produces rough region boundaries as a result of the density estimation in a high-dimensional space. The resulting region boundaries thus require additional smoothing or postediting to obtain stylistic image abstraction [26], [31]. Region segmentation based on the mean-shift filtering is useful for flattening regions but less ideal for producing a sophisticated line drawing, because each segmented region inevitably forms a closed boundary (even for an open shape).

Fischer et al. [32] presented a system for producing a stylized augmented reality that incorporates 3D models into a video sequence in a nonphotorealistic fashion. They applied the Canny edge detector [27] and the bilateral filter [33] for solving the line extraction and the region smoothing problems, respectively. Orzan et al. [34] developed a multiscale image abstraction system based on the Canny edge detector and the gradient reconstruction method. Kang et al. [35] showed that it is also possible to obtain image abstraction via stroke-based rendering, constrained by the lines generated from a modified Canny edge detector.

While Canny's edge detector [27] has been often used for line drawing, there are other line extraction methods as well. Gooch et al. [36] presented a facial illustration system based on a difference-of-Gaussians (DoG) filter, originated from the Marr-Hildreth edge detector [37]. They used this filter in conjunction with binary luminance thresholding to produce a black-and-white facial illustration. Winnemöller et al. [38] recently extended this technique to abstract general color images and video, employing the DoG filter for line drawing and the bilateral filter for region smoothing.

This DoG edge model has proven to be more effective than Canny's method in terms of creating stylistic illustrations: It captures interesting structures better (as shown in [36]), and it automatically produces stylistic lines (in nonuniform thickness). Also, the bilateral filter [33] is a vastly popular and powerful tool for nonlinear image smoothing, and because of its simplicity and effectiveness, it has been quickly adopted as the standard solution for feature-preserving visual data processing in a variety of 2D or 3D graphics applications [39], [40], [41], [42], [38], [43], [44].

The advantages of the underlying filters make the abstraction scheme of Winnemöller et al. [38] a powerful one. From the perspective of feature enhancement and stylization, however, we observe that there is room for improvement. As for the DoG edge model, the aggregate of edge pixels may not clearly reveal the sense of "directedness" (and thus may look less like lines) due to the nature of the isotropic filter kernel. Also, the thresholded edge map may exhibit isolated edge components that clutter the output, especially in an area with image noise or weak contrast (see Fig. 14d). Although one may consider adjusting the threshold in order to improve the edge coherence, the result can be even poorer due to added noise. This problem is significantly diminished in our flow-based filtering framework (see Fig. 14e).

The inherent limitation of the isotropic kernel may similarly compromise the performance of the region smoothing technique such as the bilateral filtering. Since the original bilateral filter uses an isotropic (circular) kernel, some meaningful shape boundaries with low color contrast may be overly blurred. In addition, noise along the shape boundaries may not be properly removed. We show that the proposed flow-based filtering framework improves the performance of the region smoothing filter as well, in terms of feature enhancement and stylization.

### 1.3 Contributions and Overview

We present a flow-driven approach to solving the two main problems of image abstraction, that is, **line drawing** and **region smoothing**. The preliminary version of this work was presented in [45], where we focused on line drawing only. In this extension, we follow the abstraction framework of Winnemöller et al. [38], employing the DoG filter for line extraction and the bilateral filter for region smoothing. The

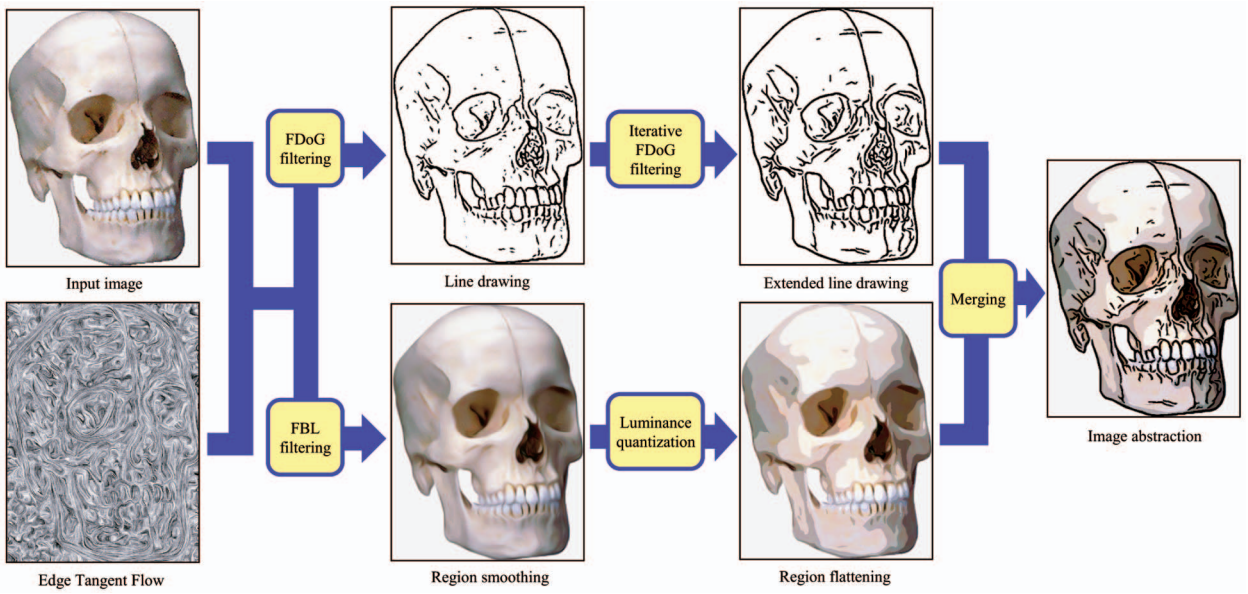



Fig. 2. Process overview.

main difference is that our approach takes into account the "direction" of the local image structure in shape/color filtering, rather than looking in all directions. That is, we modify these filters so that they are adapted to a curved kernel, which follows the local "edge flow." The resulting two filter responses are then combined to produce the final image abstraction (see Fig. 2 for the process overview).

We will show that this flow-based filter adaptation enhances the abstraction and stylization performance considerably. First, our modified line extraction filter, which we call the *flow-based DoG (FDoG) filter*, dramatically enhances the spatial coherence of lines and also suppresses noise. Second, our modified region smoothing filter, called the *flow-based bilateral (FBL) filter*, helps convey clear and enhanced shape boundaries.

In comparison to the existing approaches for image abstraction [26], [29], [30], [32], [31], [35], [34], [38], our scheme has the following advantages:

- **Feature enhancement.** Our line extraction filter (FDoG) differs from conventional edge detectors in that it uses a curve-shaped filter kernel in order to maximize the line coherence. Our region smoothing filter (FBL) similarly improves the performance of the standard bilateral filter in terms of enhancing shapes and feature directionality.
- **Cleanliness.** Flow-driven abstraction of shapes and colors results in smooth, clean, and clear lines and region boundaries.
- **Stylization.** Improved feature enhancing capability and cleanliness lead to the production of a high-quality illustration.
- **Simplicity.** Our method is straightforward and easy to implement. Also, both FDoG and FBL filters provide linear time complexity with respect to the kernel radius.

The remainder of this paper is organized as follows: In Section 2, we describe the construction of the filter-steering flow. Sections 3 and 4 discuss the FDoG filter and FBL filter, respectively. We then show various test results in Section 5, followed by the concluding remarks in Section 6.

## 2 Flow Construction

### 2.1 Edge Tangent Flow (ETF)

Given an input image $I(\mathbf{x})$, where $\mathbf{x} = (x, y)$ denotes an image pixel, we first construct a smooth, feature-preserving edge flow field. This flow field will be used as the guiding map of our filters. We define *edge tangent*, denoted $\mathbf{t}(\mathbf{x})$, as a vector perpendicular to the image gradient $\mathbf{g}(\mathbf{x}) = \nabla I(\mathbf{x})$. The term "tangent" is used in a sense that $\mathbf{t}(\mathbf{x})$ may be viewed as the tangent of the curve representing the local edge flow. We thus call this vector field an *ETF*.

Such a feature-preserving vector field is useful in many applications, and different approaches exist for constructing one. In painterly rendering, scattered orientation interpolation has been a popular method for creating a rough direction field [12], [15] with which to guide the placement of oriented strokes. A more sophisticated ETF may be constructed by taking into account the entire set of pixels. In the image processing community, it was shown that the diffusion process based on partial differential equation (PDE) can be used to regularize orientation fields [46], [47], such as optical flow. Paris et al. [48] presented an adaptation of bilateral filter for smoothing orientations in human hair images, taking advantage of the inherent strengths of the original bilateral filter, such as noniterative nature, simplicity, and controllability. These advantages led us to similarly employ a bilateral filter for constructing ETF. Our formulation is designed to deal with general input images, and we look to provide an efficient scheme suited for handling both still images and video.

### 2.2 Formulation

Our ETF construction scheme is essentially a bilateral filter [33] adapted to handle vector-valued data. In each pixel-centered kernel, we perform nonlinear smoothing of

vectors such that salient edge directions are preserved, while weak edges are redirected to follow the neighboring dominant ones. Also, to preserve sharp corners, we encourage smoothing among the edges with similar orientations.

The ETF construction filter is thus defined as follows:

$$\mathbf{t}'(\mathbf{x}) = \frac{1}{k}\iint_{\Omega_\mu} \phi(\mathbf{x},\mathbf{y})\mathbf{t}(\mathbf{y})w_s(\mathbf{x},\mathbf{y})w_m(\mathbf{x},\mathbf{y})w_d(\mathbf{x},\mathbf{y})d\mathbf{y}, \quad (1)$$

where $\Omega_\mu(\mathbf{x})$ denotes the kernel of radius $\mu$ at $\mathbf{x}$, and $k$ is the vector normalizing term. The tangent vector $t(\cdot)$ is assumed to be $2\pi$-periodic.

For the *spatial weight function* $w_s$, we use a box filter of radius $\mu$:

$$w_s(\mathbf{x},\mathbf{y}) = \begin{cases} 1 & \text{if } \|\mathbf{x}-\mathbf{y}\| < \mu, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The other two weight functions, $w_m$ and $w_d$, play the key role in feature preservation. We call $w_m$ the *magnitude weight function*, which is defined as

$$w_m(\mathbf{x},\mathbf{y}) = [\hat{g}(\mathbf{y}) - \hat{g}(\mathbf{x}) + 1]/2, \quad (3)$$

where $\hat{g}(\mathbf{z})$ denotes the normalized gradient magnitude at $\mathbf{z}$. Note that $w_m$ ranges in [0, 1], and this weight function monotonically increases with respect to the magnitude difference $\hat{g}(\mathbf{y}) - \hat{g}(\mathbf{x})$, indicating that bigger weights are given to the neighboring pixels $\mathbf{y}$ whose gradient magnitudes are higher than that of the center $\mathbf{x}$. This ensures the preservation of the dominant edge directions.

We then define $w_d$, the *direction weight function*, to promote smoothing among similar orientations:

$$w_d(\mathbf{x},\mathbf{y}) = |\mathbf{t}(\mathbf{x}) \cdot \mathbf{t}(\mathbf{y})|, \quad (4)$$

where $\mathbf{t}(\mathbf{z})$ denotes the normalized tangent vector at $\mathbf{z}$. This weight function increases as the two vectors align closely (that is, the angle $\theta$ between vectors approaches 0 or $\pi$) and decreases as they get orthogonal (that is, $\theta$ approaches $\pi/2$). For tight alignment of vectors, we temporarily reverse the direction of $\mathbf{t}(\mathbf{y})$ using the sign function $\phi(\mathbf{x},\mathbf{y}) \in \{1,-1\}$, in case $\theta$ is bigger than $\pi/2$:

$$\phi(\mathbf{x},\mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{t}(\mathbf{x}) \cdot \mathbf{t}(\mathbf{y}) > 0, \\ -1 & \text{otherwise.} \end{cases} \quad (5)$$

To further improve the robustness of orientation filtering, we may add another component to (1) such as the variance term suggested by Paris et al. [48], via collecting statistical measurements.

The initial ETF, denoted as $\mathbf{t}^0(\mathbf{x})$, is obtained by taking perpendicular vectors (in the counterclockwise sense) from the initial gradient map $\mathbf{g}^0(\mathbf{x})$ of the input image $I$. $\mathbf{t}^0(\mathbf{x})$ is then normalized before use. The initial gradient map $\mathbf{g}^0(\mathbf{x})$ is computed by employing a Sobel operator. The input image may be optionally Gaussian-blurred before gradient computation. Fig. 3 shows ETF fields obtained from sample images. The ETF preserves edge directions well around important features while keeping them smooth elsewhere. The ETF fields are visualized using line integral convolution [49].

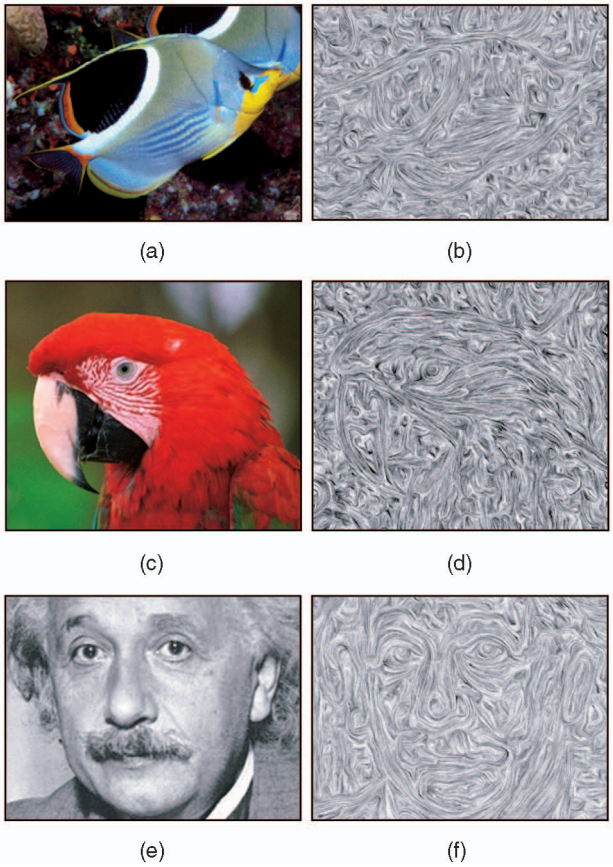



Fig. 3. ETF construction. (a) Tropical fish. (b) ETF (Tropical fish). (c) Parrot. (d) ETF (parrot). (e) Einstein. (f) ETF (Einstein).

### 2.3 Iterative Application

Our filter may be iteratively applied to update the ETF incrementally: $\mathbf{t}^i(\mathbf{x}) \rightarrow \mathbf{t}^{i+1}(\mathbf{x})$. In this case, $\mathbf{g}(\mathbf{x})$ evolves accordingly (but the gradient magnitude $\hat{g}(\mathbf{x})$ is unchanged). In practice, we typically iterate a few ($2 \sim 3$) times. Fig. 4 shows how the ETF gets smoother after each iteration.

### 2.4 Acceleration

Note that the original formulation of the ETF construction filter (1) is an $O(n \times \mu^2)$ algorithm, where $n$ is the number of image pixels and $\mu$ is the kernel radius. In practice, we accelerate the ETF construction by separately applying 1D versions of ETF filters in $x$ and $y$ dimensions. This idea is similar to the separable bilateral filtering, suggested by Pham and van Vliet [50].

The separable ETF construction reduces the time complexity down to $O(n \times \mu)$, without noticeable quality degradation of the vector field (see Fig. 5). In this figure, we represent orientations by RGB colors (with each component ranging in [0, 1]) to enable a clear comparison. For the input image in Fig. 5a, the average per-pixel color distance between the full-kernel ETF and the separable-kernel ETF is 0.00893.[1]

1. The separable ETF construction is more limited than the full-kernel version in capturing small-scale details or texture. In this case, a sufficiently small kernel must be used.

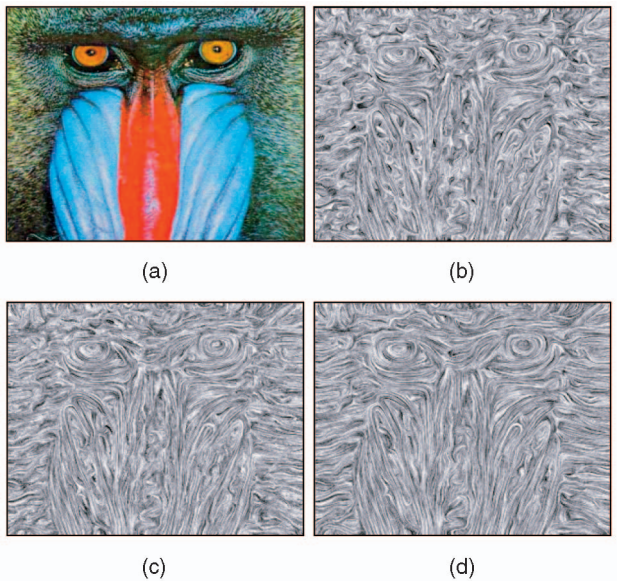



Fig. 4. Iterative ETF construction. (a) Input. (b) First iteration. (c) Second iteration. (d) Third iteration.

## 3 Line Extraction

For image-guided 2D line drawing, conventional edge (or line) detectors are often employed and adapted, such as Canny's [26], [32], [35], [34], mean-shift segmentation [29], [30], [31], DoG filtering [36], [38], and so on. We build on the DoG edge model suggested by Winnemöller et al. [38], mainly due to its simplicity and the stylistic nature that suits our purpose. We particularly focus on enhancing the quality of lines by steering the DoG filter along the ETF flow.

$\mathbf{t}(\mathbf{x})$ in ETF represents the local edge direction, meaning we will most likely have the highest contrast in its perpendicular direction, that is, the gradient direction $\mathbf{g}(\mathbf{x})$. The idea is to apply a linear DoG filter in this gradient direction as we move along the edge flow. We then accumulate the individual filter responses along the flow, as a way of collecting enough evidence before we draw the conclusion on the "edge-ness." This allows us to exaggerate the filter output along genuine edges, while we attenuate the output from spurious edges. Therefore, it not only enhances the spatial coherence of the edges but also has the effect of suppressing noise.

### 3.1 Flow-Based Difference-of-Gaussians Filter

Fig. 6 illustrates our filtering scheme. Let $c_{\mathbf{x}}(s)$ denote the flow curve at $\mathbf{x}$, where $s$ is an arc-length parameter that may take on positive or negative values. We assume $\mathbf{x}$ serves as the curve center, that is, $c_{\mathbf{x}}(0) = \mathbf{x}$. Also, let $l_{\mathbf{x},s}$ denote a line segment that is perpendicular to $\mathbf{t}(c_{\mathbf{x}}(s))$ and intersecting $c_{\mathbf{x}}(s)$. We parameterize $l_{\mathbf{x},s}$ with an arc-length parameter $t$, and hence, $l_{\mathbf{x},s}(t)$ denotes the point on the line $l_{\mathbf{x},s}$ at $t$. Again, we assume $l_{\mathbf{x},s}$ is centered at $c_{\mathbf{x}}(s)$, that is, $l_{\mathbf{x},s}(0) = c_{\mathbf{x}}(s)$. Note that $l_{\mathbf{x},s}$ is parallel to the gradient vector $\mathbf{g}(c_{\mathbf{x}}(s))$. We use the term *flow axis* for $c_{\mathbf{x}}$ and *gradient axis* for $l_{\mathbf{x},s}$.

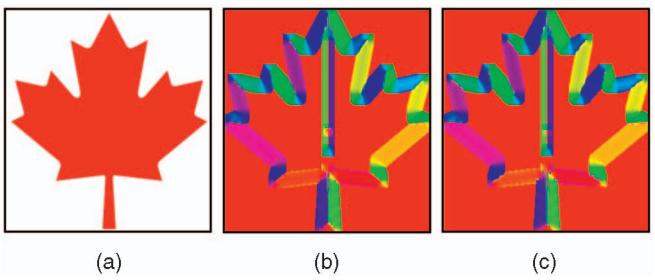



Fig. 5. Separable ETF construction. (a) Input. (b) Full kernel. (c) Separable kernel.

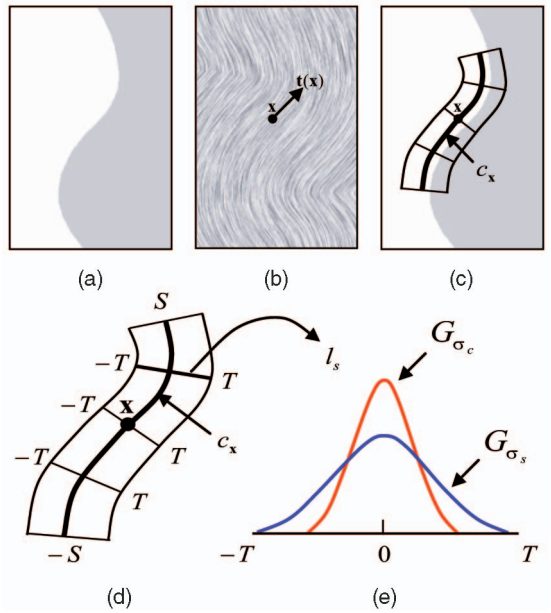



Fig. 6. FDoG filtering. (a) Input. (b) ETF. (c) Kernel at $\mathbf{x}$. (d) Kernel enlarged. (e) Gaussian components for DoG.

Our filtering scheme is then formulated as

$$\mathcal{H}(\mathbf{x}) = \int_{-S}^{S}\int_{-T}^{T} I\big(l_{\mathbf{x},s}(t)\big) f(t) G_{\sigma_m}(s)\,dt\,ds, \tag{6}$$

where $I(l_{\mathbf{x},s}(t))$ represents the value of the input image $I$ at $l_{\mathbf{x},s}(t)$. The above formulation can be interpreted as follows: As we move along $c_{\mathbf{x}}$, we apply a one-dimensional (1D) filter $f$ on the gradient line $l_{\mathbf{x},s}$. The individual filter responses are then accumulated along $c_{\mathbf{x}}$ using a weight function of $s$, denoted as $G_{\sigma_m}(s)$, where $G_\sigma$ represents a univariate Gaussian function of variance $\sigma^2$:

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}. \tag{7}$$

In (6), the user-provided parameter $\sigma_m$ automatically determines the size of $S$. Thus, $\sigma_m$ controls the length of the elongated flow kernel and also the degree of line coherence to enforce.

As for the underlying filter $f$, we employ the edge model based on DoG [38]:

$$f(t) = G_{\sigma_c}(t) - \rho \cdot G_{\sigma_s}(t), \tag{8}$$

where the two parameters, $\sigma_c$ and $\sigma_s$, control the sizes of the center interval and the surrounding interval, respectively. We set $\sigma_s = 1.6\sigma_c$ to make the shape of $f$ closely resemble that of Laplacian-of-Gaussian [37]. Therefore, once $\sigma_c$ is

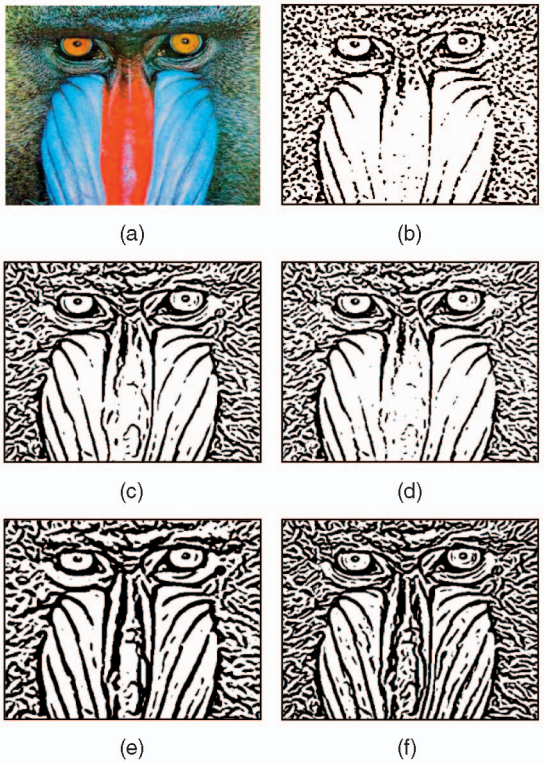



Fig. 7. FDoG filtering with parameter control. (a) Input. (b) Isotropic DoG. (c) FDoG: $\sigma_{\mathrm{m}} = 3.0$. (d) FDoG: $\sigma_{\mathrm{m}} = 1.0$. (e) FDoG: $\sigma_{\mathrm{c}} = 2.0$. (f) FDoG: $\rho = 0.997$.

given by the user, it automatically determines $\sigma_s$ and, thus, the size of $T$ in (6). It also directly affects the resulting line width. $\rho$ controls the level of noise detected and typically ranges in [0.97, 1.0].

Once we obtain $\mathcal{H}$ from (6), we convert it to a black-and-white image by binary thresholding, as suggested in [38]:

$$\tilde{\mathcal{H}}(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathcal{H}(\mathbf{x}) < 0 \text{ and } 1 + \tanh(\mathcal{H}(\mathbf{x})) < \tau, \\ 1 & \text{otherwise,} \end{cases} \quad (9)$$

where $\tau$ is a threshold in [0, 1], with the typical value of 0.5. This binary output $\tilde{\mathcal{H}}$ serves as our targeted line illustration.

Since our DoG filter is driven by the vector flow, we name it as the FDoG filter.

Fig. 7 shows results of our FDoG filtering with varying sets of parameter values. Each caption specifies the modified parameter value from the original setting: $\sigma_m = 3.0$, $\sigma_c = 1.0$, and $\rho = 0.99$. With FDoG filtering, the line coherence is improved compared to that of the isotropic DoG filter in Fig. 7b. Also, see how the kernel size parameters $\sigma_m$ and $\sigma_c$ affect the line coherence and the line width, respectively. Increasing $\rho$ results in the inclusion of more lines in the illustration.

Fig. 8 focuses on the capability of the FDoG filter in enhancing the spatial coherence of lines. Unlike conventional edge detectors, the FDoG filter enables constructing lines from a set of disconnected points by obtaining ETF from a Gaussian-smoothed input image and with a large ETF kernel size ($\mu$ in (1)). Compared to Fig. 8b (DoG), the result in Fig. 8c (FDoG) can be said of as "perceptually correct," considering the fact that people generally perceive

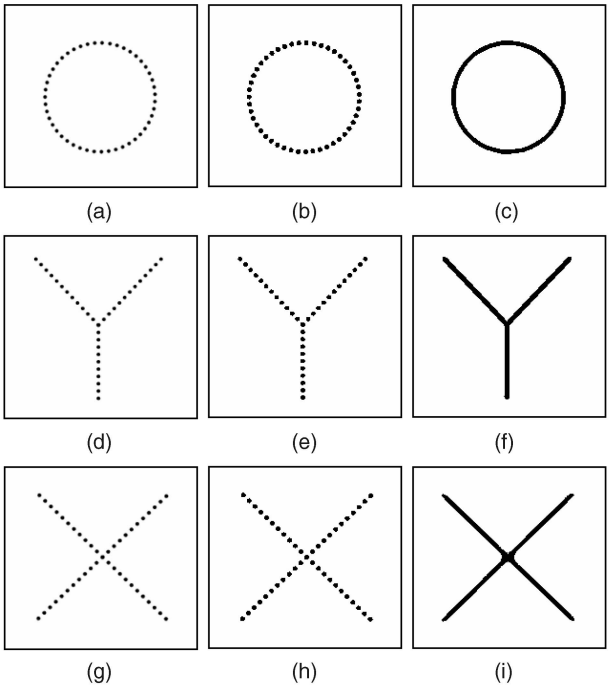



Fig. 8. FDoG: Extracting lines from isolated points. (a) Input. (b) DoG. (c) FDoG. (d) Input. (e) DoG. (f) FDoG. (g) Input. (h) DoG. (i) FDoG.

Fig. 8a as a picture of a circle rather than a collection of dots. The middle and the last rows illustrate that FDoG is capable of handling junctions and intersections as well.

Fig. 9 demonstrates the robustness of FDoG against noise. Fig. 9a is an image corrupted by Gaussian noise, and Fig. 9b is an output of the isotropic DoG filter followed by binarization with a low threshold ($\tau = 0.2$). Notice the weak line coherence due to noise. While a higher threshold ($\tau = 0.7$) as in Fig. 9c improves the coherence, the added noise clutters the output. On the other hand, Fig. 9e shows that FDoG filter constructs a clean and coherent line at a low threshold ($\tau = 0.2$) by taking advantage of the smooth ETF vectors around the target shape (Fig. 9d).

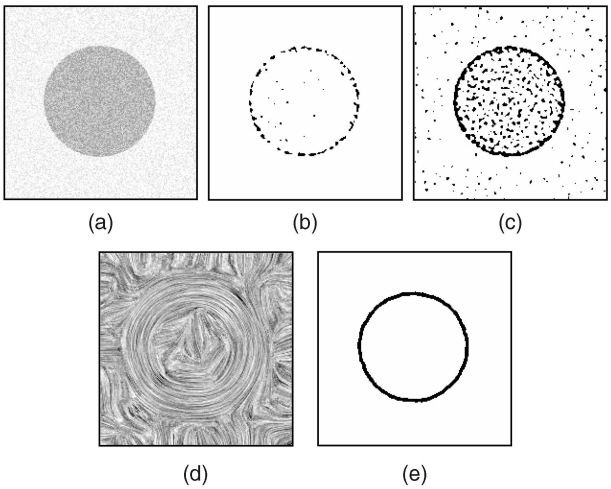



Fig. 9. FDoG: Noise suppression. (a) Input. (b) DoG. (c) DoG. (d) ETF. (e) FDoG.

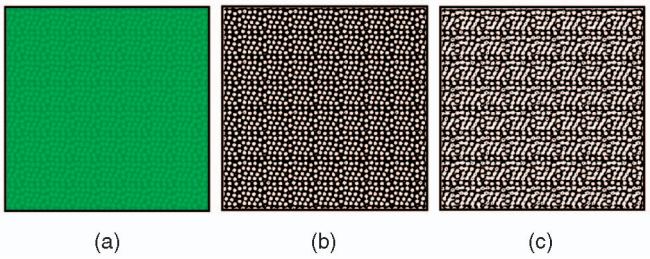



Fig. 10. FDoG: Handling of small-scale texture. (a) Input. (b) $\mu = 3$. (c) $\mu = 5$.

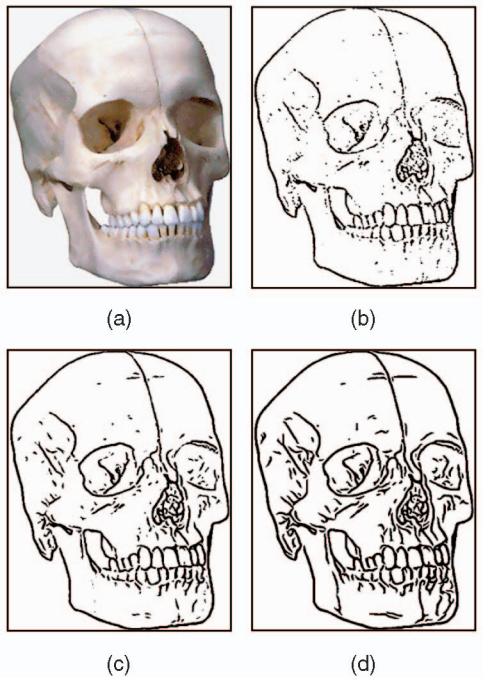



Fig. 11. Iterative FDoG: Skull. (a) Input image. (b) Isotropic DOG. (c) FDoG: First iteration. (d) FDoG: Third iteration.

While the FDoG filter is useful when it comes to protecting directional structures, it shows some limitations in handling small-scale nondirectional structures. Fig. 10 illustrates this case. Since the FDoG filter relies on ETF, it is important to construct an ETF that properly captures the shape, which, however, may be difficult for tiny-scale details or texture patterns. In such cases, one must set the ETF kernel size (represented by $\mu$ in (1)) to be sufficiently small, otherwise, the line detection may fail, as shown in Fig. 10c.

### 3.2 Iterative FDoG Filtering

For further enhancement of the line drawing, the FDoG filter may be applied iteratively. After each application of FDoG, we may reinitialize the filter input by superimposing the black edge pixels of the previous binary output $\tilde{H}$ (obtained by (9)) upon the original image $I$, then reapply the FDoG filter to this combined image (ETF remains unchanged). This process may be repeated until we reach a satisfactory level of line connectivity and illustration quality. For most of our test images, a few $(2 \sim 3)$ iterations were sufficient. Before each application of the FDoG filter, one may optionally Gaussian-blur the filter input to further smooth the line strokes.

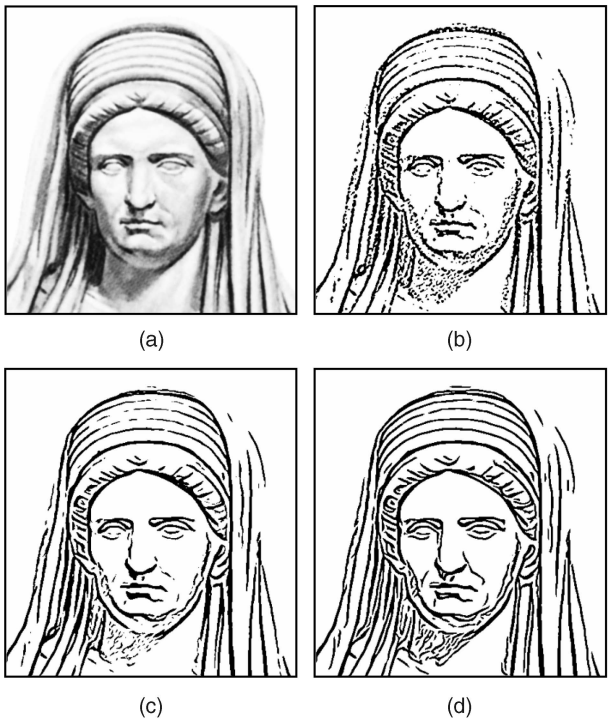



Fig. 12. Iterative FDoG: Venus. (a) Input image. (b) Isotropic DOG. (c) FDoG: First iteration. (d) FDoG: Third iteration.

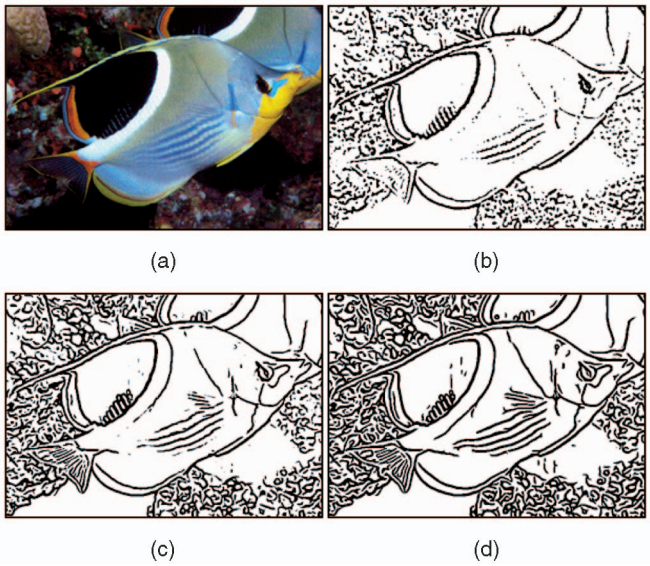
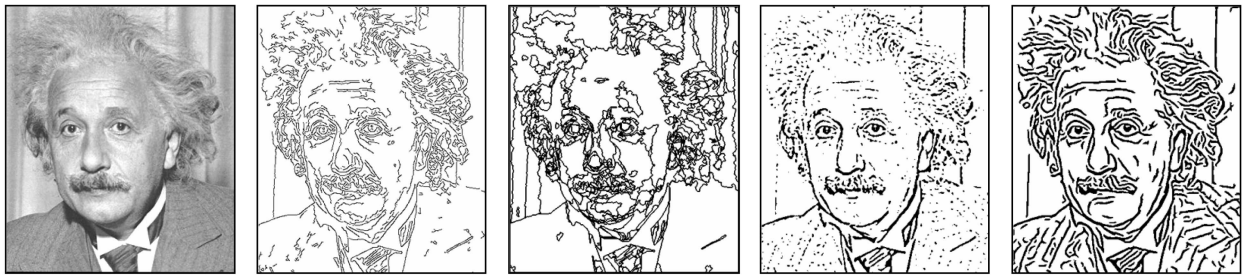



Fig. 13. Iterative FDoG: Tropical fish. (a) Input image. (b) Isotropic DOG. (c) FDoG: First iteration. (d) FDoG: Third iteration.

Black lines copied from the previous iteration form stark contrast with the background[2] and, thus, are recaptured in the current iteration due to the contrast-sensitive nature of the underlying DoG filter. In addition, FDoG filtering extends the detected lines along the ETF flow. Therefore, iterative FDoG filtering progressively improves line coherence. Figs. 11, 12, and 13 show that repeated applications of the FDOG filter successively improve the spatial coherence of the shape boundaries.

2. A possible exception is when both sides of the line are almost black, in which case, however, it is not likely that the line was detected by the linear DoG in the first place.



(a) (b) (c) (d) (e)

Fig. 14. FDoG: Comparison with other techniques. (a) Input. (b) Canny. (c) Mean shift. (d) Isotropic DoG. (e) FDoG.

In comparing DoG and FDoG, there are two important things to note: First, while the FDoG filter outperforms the DoG filter in terms of the line coherence and the cleanliness of illustration, DoG appears to do a better job of conveying "tone" information. For example, in Fig. 11b (DoG filtered), the black pixel aggregates naturally represent the tone in the area, whereas in Fig. 11d (FDoG filtered), the line thickness is somewhat equalized, and it becomes more so after each iteration of FDoG. Second, while the first application of the FDoG filter does suppress noise better than the DoG filter, the subsequent application of the FDoG filter may inherit noise pixels captured from the previous iteration and stabilize them due to our line superimposition principle. For example, see the neck of Venus (in Fig. 12c) that is filled with a set of short line segments. It is unclear whether these line segments are noise or not, but they are enhanced and stabilized in Fig. 12d. Careful parameter setting may help reduce noise in the first application of FDoG but at the risk of losing some details. One may also develop a more sophisticated strategy for FDoG iteration to reduce such artifacts.

### 3.3 Implementation of the FDOG Filter

For implementing the FDoG filter, we sample $2\alpha \times 2\beta$ points from the kernel and discretely approximate (6). We first sample $2\alpha$ points along the flow axis $c_{\mathbf{x}}$ by bidirectionally following the vector flow starting from $\mathbf{x}$ (thus, $\alpha$ sample points in each direction). Let $\mathbf{z}$ denote the sample points along $c_{\mathbf{x}}$. Initially, we set $\mathbf{z} \leftarrow \mathbf{x}$, then iteratively obtain the next sample point by moving along $c_{\mathbf{x}}$ in one direction using a fixed step size $\delta_m : \mathbf{z} \leftarrow \mathbf{z} + \delta_m \cdot \mathbf{t}(\mathbf{z})$. Similarly, we obtain the sample points on the other half of $c_{\mathbf{x}}$: $\mathbf{z} \leftarrow \mathbf{z} - \delta_m \cdot \mathbf{t}(\mathbf{z})$.

Now, at each $\mathbf{z}$, we sample $2\beta$ points along the gradient axis (the line perpendicular to $\mathbf{t}(\mathbf{z})$), similarly with the step size of $\delta_n$. We set $\delta_m = \delta_n = 1$ throughout. $\alpha$ and $\beta$ are automatically determined by $\sigma_m$ and $\sigma_c$, respectively. The time complexity of the FDoG filter is thus $O(n \times \alpha \times \beta)$, where $n$ is the number of image pixels.

We can accelerate the FDoG filtering by decomposing (6) as follows:

$$\mathcal{H}_g(\mathbf{x}) = \int_{-T}^{T} I(l_{\mathbf{x}}(t)) f(t) dt, \tag{10}$$

and

$$\mathcal{H}_e(\mathbf{x}) = \int_{-S}^{S} G_{\sigma_m}(s) \mathcal{H}_g(c_{\mathbf{x}}(s)) ds, \tag{11}$$

where $l_{\mathbf{x}}(t)$ is an abbreviated notation for $l_{\mathbf{x},0}(t)$. That is, $l_{\mathbf{x}}(t)$ denotes the gradient axis at $\mathbf{x}$. We first execute (10) and compute $\mathcal{H}_g(\mathbf{x})$, which is the linear DoG value in the gradient direction at each $\mathbf{x}$ in the image. This takes $O(n \times \beta)$ time, with $2\beta$ samples in the gradient direction. Given $\mathcal{H}_g(\mathbf{x})$, we then compute $\mathcal{H}_e(\mathbf{x})$ along the flow axis $c_{\mathbf{x}}$, as in (11). Computing $\mathcal{H}_e(\mathbf{x})$ has the effect of collecting the linear DoG values along the flow curve. This takes $O(n \times \alpha)$, with $2\alpha$ samples along the flow axis. Overall, this separation strategy reduces the original complexity of $O(n \times \alpha \times \beta)$ down to $O(n \times \alpha + n \times \beta)$. There is no quality degradation ensued from this conversion.

Fig. 14 shows the comparison of our method with other popular line extraction techniques, including Canny's, mean-shift segmentation, and isotropic DoG. From the line-drawing perspective, our method outperforms others in that it not only captures "perceptually meaningful" structures but also depicts them with smooth, coherent, and stylistic lines.

## 4 Region Smoothing

The goal of region smoothing is to remove unimportant details from the region interiors while preserving important shapes (that is, region boundaries). This in general calls for feature-preserving image smoothing, for which the bilateral filter is one of the most popular solutions out there. Due to its effectiveness and ease of use, the bilateral filter has drawn a lot of attention in recent years, and it is now being used for solving a variety of problems in computer graphics, including surface fairing [41], tone mapping [40], texture editing and relighting [39], image fusion [42], data upsampling [44], feature-aware filtering [43], and image/video abstraction [38]. The key to the bilateral filtering is to employ two weight functions, one in the spatial domain and the other in the color domain, then perform smoothing amongst similar colors only. This turns out a simple yet effective way of edge-preserving image smoothing.

From the perspective of image abstraction and stylization, the original bilateral filter does have some limitations, mainly due to the use of a circular (isotropic) spatial kernel. In smoothing the minor color differences in a circular neighborhood, it ignores the direction in which the color contrast is formed and thus may remove some subtle but meaningful shape boundaries. That is, it may lose feature directionality rather than protect it. This lack of direction-awareness may also cause the surviving edges to look rough, resulting in a poor stylization.

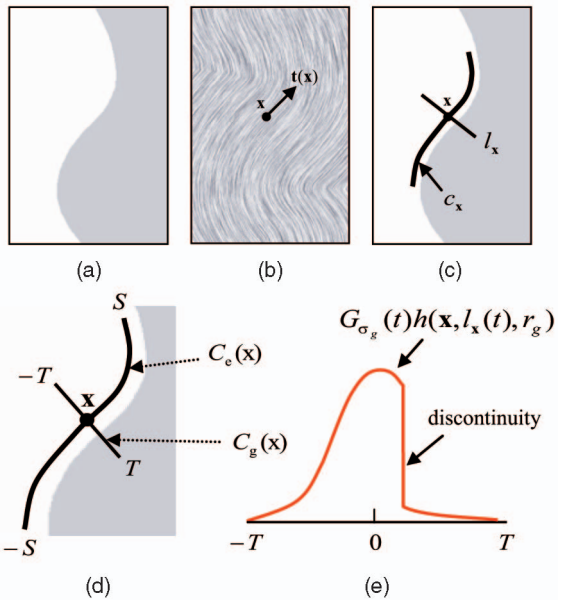



Fig. 15. FBL filtering. (a) Input. (b) ETF. (c) Two linear kernels at x. (d) Kernels enlarged. (e) Bilateral weight function for $\mathcal{C}_g$.

In this section, we show that the ETF-based adaptation of bilateral filter successfully overcomes these limitations. We conduct two separate linear bilateral smoothing operations, one along the edge directions and the other along the perpendicular (gradient) directions. Such reformulation of bilateral filter has direct connections to some of the existing structure-adaptive image denoising techniques, as will be discussed in Section 4.2.

### 4.1 Flow-Based Bilateral Filter

Equations (12) and (14) describe together the formulation of our FBL filtering scheme. It is also graphically illustrated in Fig. 15. The linear bilateral filter along the edge (or ETF) direction is defined as follows:

$$\mathcal{C}_e(\mathbf{x}) = \frac{1}{\nu_e}\int_{-S}^{S} I(c_{\mathbf{x}}(s))G_{\sigma_e}(s)h(\mathbf{x}, c_{\mathbf{x}}(s), r_e)ds, \qquad (12)$$

where $c_{\mathbf{x}}$ again denotes the flow curve of ETF, and $\nu_e = \int_{-S}^{S} G_{\sigma_e}(s)h(\mathbf{x}, c_{\mathbf{x}}(s), r_e)ds$ is the weight normalization term. As in the original bilateral filter, we provide two weight functions for the spatial and color domains, denoted as $G_{\sigma_e}(s)$ and $h(\mathbf{x}, c_{\mathbf{x}}(s), r_e)$, respectively. The *spatial weight function* $G_{\sigma_e}$ is a Gaussian function along the flow axis $c_{\mathbf{x}}$. $\sigma_e$ determines the kernel size $S$. The *similarity weight function* $h$ is defined similarly to the original bilateral filter, except that we compare the colors between the center point $\mathbf{x}$ and the points along the main axis $c_{\mathbf{x}}$:

$$h(\mathbf{x}, \mathbf{y}, \sigma) = G_\sigma(\|I(\mathbf{x}) - I(\mathbf{y})\|). \qquad (13)$$

This formulation can also be applied to color images by employing a distance metric in the color space.

Similarly, the following equation defines the other linear bilateral filter along the gradient direction:

$$\mathcal{C}_g(\mathbf{x}) = \frac{1}{\nu_g}\int_{-T}^{T} I(l_{\mathbf{x}}(t))G_{\sigma_g}(t)h\big(\mathbf{x}, l_{\mathbf{x}}(t), r_g\big)dt, \qquad (14)$$

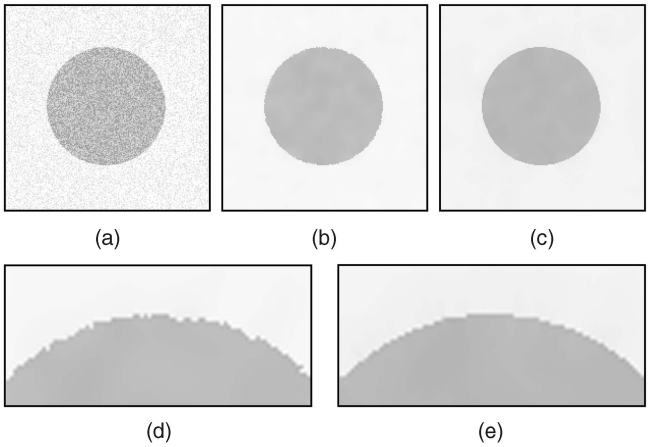



Fig. 16. FBL: Boundary restoration. (a) Input. (b) Bilateral filter. (c) FBL filter. (d) Bilateral filter (enlarged). (e) FBL filter (enlarged).

where $l_{\mathbf{x}}(t)$ is again an abbreviated notation for $l_{\mathbf{x},0}(t)$, which is the gradient axis at $\mathbf{x}$. Also, $\nu_g = \int_{-T}^{T} G_{\sigma_g}(t)h(\mathbf{x}, l_{\mathbf{x}}(t), r_g)dt$ represents the weight normalization term.

Figs. 15c and 15d exemplify how the two axes, $c_{\mathbf{x}}$ and $l_{\mathbf{x}}$, are formed for the corresponding linear bilateral filters $\mathcal{C}_e(\mathbf{x})$ and $\mathcal{C}_g(\mathbf{x})$, respectively. In particular, Fig. 15e shows a possible shape of the bilateral weight function for $\mathcal{C}_g$ at $\mathbf{x}$. As in the case of the FDoG filter, the discrete implementation of $\mathcal{C}_e$ and $\mathcal{C}_g$ is achieved by sampling $2\alpha$ points along $c_{\mathbf{x}}$ and $2\beta$ points along $l_{\mathbf{x}}$, respectively.

$\mathcal{C}_e$ and $\mathcal{C}_g$ play different roles. $\mathcal{C}_e$ mainly operates in the edge directions and thus protects and cleans up the shape boundaries. On the other hand, $\mathcal{C}_g$ suppresses the color differences within regions and thus smooths out the region interiors. Typically, we alternate $\mathcal{C}_e$ and $\mathcal{C}_g$ in an iterative fashion. We call the combination of $\mathcal{C}_e$ and $\mathcal{C}_g$ an *FBL* filter. Since $\mathcal{C}_e$ and $\mathcal{C}_g$ are both 1D operators, they are much faster than the full-kernel bilateral filter.

Fig. 16 shows the comparison between the original bilateral filter and our FBL filter. From a noisy input (Fig. 16a), Fig. 16b is obtained by iterating the bilateral filter five times, with domain and range parameters $\sigma_d = 2.0$, $\sigma_r = 10$ (see [33] for definitions). On the other hand, Fig. 16c is obtained by alternating $\mathcal{C}_e$ and $\mathcal{C}_g$ five times, with parameter values of $\sigma_e = 2.0$, $r_e = 50$, $\sigma_g = 2.0$, and $r_g = 10$. In this example, we set $r_e > r_g$ so that we can allow more aggressive smoothing in the edge directions than in the gradient directions. As compared in Figs. 16d and 16e, the FBL filter restores the shape boundary better than the full-kernel bilateral filter.

Fig. 17 further illustrates the performance of the FBL filter in terms of shape restoration. We picked a low-quality input image (Fig. 17a) that features some blockiness, possibly due to its compression scheme. While both the full-kernel bilateral filter and the FBL filter blur the image without destroying the original edges, the full-kernel bilateral filtering does not remove the blocking artifact (see Figs. 17e and 17g). On the other hand, the FBL filter restores clean and smooth shape boundaries (see Figs. 17f and 17h).

Fig. 18 shows how the FBL filter preserves subtle (but meaningful) shape boundaries. Unlike the full-kernel

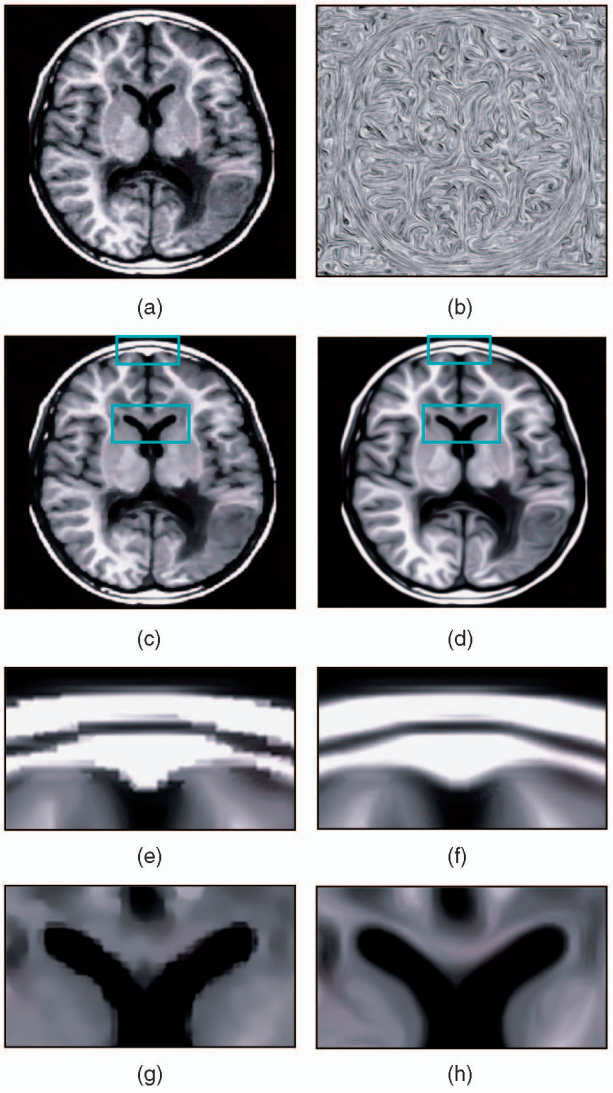



Fig. 17. FBL: Shape enhancement. (a) Input image. (b) ETF. (c) Bilateral filter ($\sigma_d = 2.0$, $\sigma_r = 10$, three iterations). (d) FBL ($\sigma_e = 2.0$, $r_e = 50$, $\sigma_g = 0.3$, $r_g = 10$, three iterations). (e) Bilateral filter (enlarged). (f) FBL (enlarged). (g) Bilateral filter (enlarged). (h) FBL (enlarged).

bilateral filter (see Fig. 18c), the FBL filter reinforces the shapes of the individual cat hairs by directing the smoothing along ETF (see Fig. 18d). The shape-enhancing nature of the FBL filter also makes a difference in stylization. We stylize the filtered image by *region flattening*, for which we perform a uniform-sized-bin luminance quantization [38]. Typical values for quantization levels are $4 \sim 10$. As shown in Fig. 18f, the FBL-filtered (and stylized) image "indicates" the sharp pattern of cat hairs better than the stylization through the full-kernel bilateral filter (Fig. 18e).

Due to its directional nature, the FBL filter is somewhat limited in stylizing small-scale nondirectional (isotropic) textures. See Fig. 19 for an example, where we similarly applied luminance quantization on each bilateral-filtered output. In this example, the full-kernel bilateral filter (Fig. 19b) results in shapes that are closest to the original. On the other hand, FBL filter (Fig. 19d) is capable of making the shape boundaries smoother, on account of the use of a

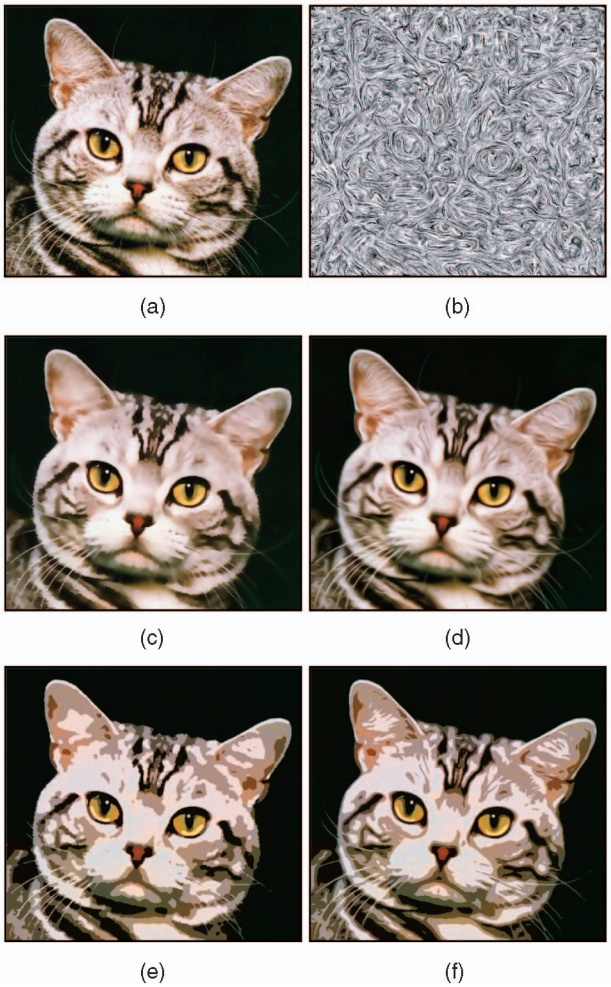



Fig. 18. FBL: Feature-enhancing stylization. (a) Input Image. (b) ETF. (c) Bilateral filter ($\sigma_d = 2.0$, $\sigma_r = 10$, three iterations). (d) FBL ($\sigma_e = 2.0$, $r_e = 10$, $\sigma_g = 0.3$, $r_g = 10$, five iterations). (e) Bilateral filter (flattened). (f) FBL (flattened).

smooth ETF. Also, FBL filter does not produce axis-aligned artifacts that may be present in the result of xy-separable bilateral filter [50] (see Fig. 19c).

Fig. 21 compares the stylization results through mean-shift segmentation, full-kernel bilateral filter, and FBL

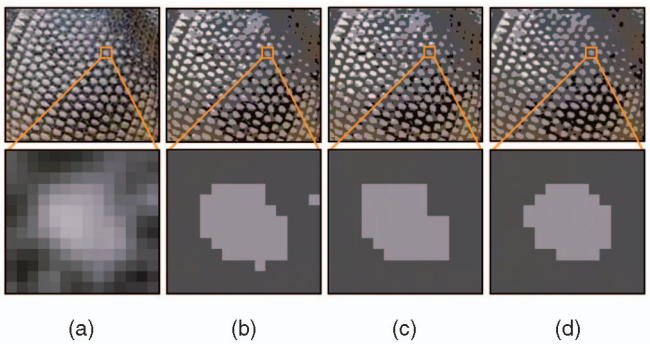



Fig. 19. FBL: Texture stylization. (a) Input image. (b) Bilateral ($\sigma_d = 2.0$, $\sigma_r = 10$, three iterations, four-level quantization). (c) Separable bilateral ($\sigma_d = 2.0$, $\sigma_r = 10$, three iterations, four-level quantization). (d) FBL ($\sigma_e = 2.0$, $r_e = 10$, $\sigma_g = 2.0$, $r_g = 10$, three iterations, four-level quantization).

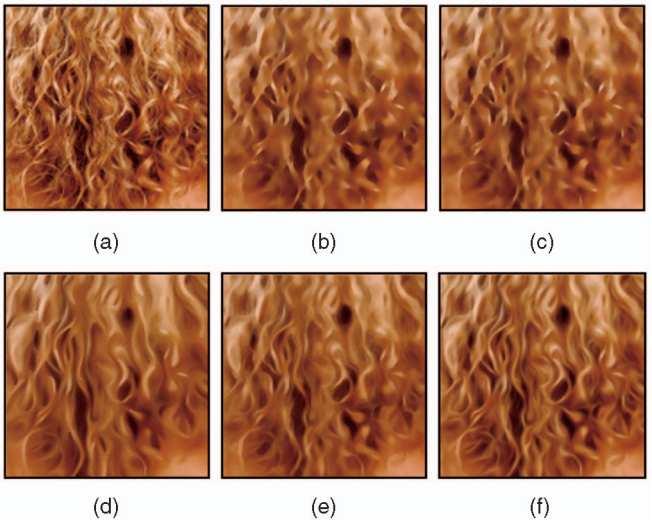



Fig. 20. FBL: Effect of kernel shapes. (a) Input. (b) Full-kernel bilateral (circle). (c) xy-separable bilateral (axis-aligned line). (d) Anisotropic bilateral (oriented line). (e) Structure-adaptive bilateral (oriented arc). (f) FBL (free-form curve).

filter. Figs. 21b and 21f are obtained by the mean-shift segmentation with parameter values of $h_s = 10$, $h_r = 5$, and $M = 20$ (see [28] for definitions). Note that the segmented region boundaries tend to be rough (due to the density estimation in a 5D space), and the loss of feature directionality is significant. Figs. 21c and 21g are obtained from the full-kernel bilateral filtering ($\sigma_d = 2.0$, $\sigma_r = 10$, three iterations), followed by luminance quantization. Compared with the mean-shift approach, the full-kernel bilateral filter does a better job at producing smooth region boundaries but still loses some directional features. Finally, the results of FBL filtering ($\sigma_e = 2.0$, $r_e = 50$, $\sigma_g = 0.3$, and $r_g = 10$, five iterations) plus luminance quantization shown in Figs. 21d and 21h, confirm that the FBL filter provides smooth region boundaries, as well as enhanced feature directionality.

### 4.2 Connections to Other Techniques

Pham and van Vliet [50] first introduced the idea of separating the bilateral filter into two directions ($x$-axis and $y$-axis), mainly for efficiency concerns. Pham [51] later presented another separable bilateral filter, named the structure-adaptive bilateral filter, where the separation axes are oriented along the feature directions recorded in the Gaussian-smoothed gradient structure tensor field. Our FBL filter is therefore a direct descendent of Pham's structure-adaptive bilateral filter [51] but with the following differences: 1) For steering the filter, we use ETF instead of Gaussian-smoothed gradient structure tensor. Note that an ETF is constructed by bilateral filtering, which preserves features better than Gaussian smoothing. 2) While the structure-adaptive bilateral filter employs a rigid arc-shaped kernel, we allow for a more flexible free-form kernel that faithfully follows an arbitrary shape of the edge flow. For example, the FBL filter captures an arbitrarily curved structure in Fig. 15c better than the structure-adaptive bilateral filter that uses a parabolic arc.

As for the kernel shape, our flow-based filtering scheme provides the highest flexibility as compared to the conventional isotropic/full kernel (circle), xy-separable kernel (line), anisotropic kernel (ellipse or line), and structure-adaptive kernel (arc). This could lead to better shape protection and enhancement, as demonstrated in Fig. 20.

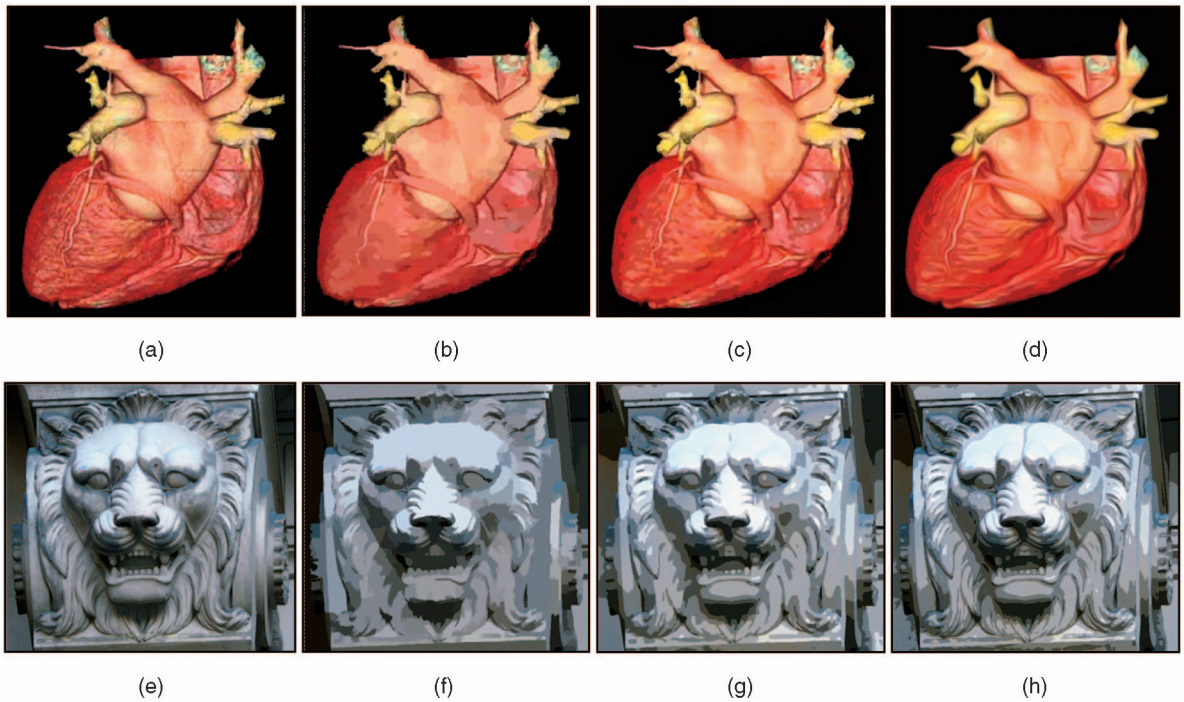



Fig. 21. FBL: Comparison of stylization results. (a) Input Image. (b) Mean-shift segmentation. (c) Bilateral filter + Quantization. (d) FBL + Quantization. (e) Input Image. (f) Mean-shift segmentation. (g) Bilateral filter + Quantization. (h) FBL + Quantization.

Fig. 22. Test images.

It should also be noted that such oriented bilateral filters (Figs. 20d, 20e, and 20f) are closely related to the PDE-based directional diffusion schemes [52], [53], [54], as pointed out by Pham [51]. Both the structure-adaptive bilateral filter and our FBL filter inherit the advantages of the original bilateral filter over the PDE-based approach, such as noniterativeness, simplicity, controllability, and so on.

## 5 Results

Fig. 24 shows various image abstraction results obtained from the test images in Fig. 22. Figs. 24a, 24c, 24e, 24f, and 24h are produced by combining results from FDoG and FBL filtering. The depiction of feature lines helps convey the shapes effectively, and the flattened colors convey the tonal information in a stylistic yet efficient (less-distracting) manner. Figs. 24d and 24i show that pure line drawing may sometimes work better in conveying the target shapes. On the other hand, pure color abstraction without lines (see Figs. 24b and 24g) can also be a good alternative for creating stylistic illustrations. Fig. 25 shows an abstraction result obtained from a large photograph. As shown in these results, our filters perform consistently well on a variety of images featuring humans, animals, plants, buildings, still objects, and outdoor scenes.

We have tested our filters on a 3-GHz dual-core PC with 2 Gbytes of memory. The performance mainly depends on the image size and the filter kernel size. Given an image of $n$ pixels, our accelerated ETF construction filter (see (1) and Section 2.4) is an $O(n \times \mu)$ algorithm, where $\mu$ is the kernel radius. The FDoG filter ((10) and (11)) and the FBL filter ((12) and (14)) both have the complexity of $O(n \times \alpha + n \times \beta)$, where $2\alpha$ and $2\beta$ are the number of sample points along the flow axis and the gradient axis, respectively. For a 512 $\times$ 512 color image and with the default parameter values $\mu = 5$, $\sigma_m = 3.0$, $\sigma_c = 1.0$, $\sigma_e = 2.0$, $r_e = 10$, $\sigma_g = 0.5$, and $r_g = 10$, an application of the ETF construction filter, the FDoG filter, and the FBL filter typically takes less than one second each.

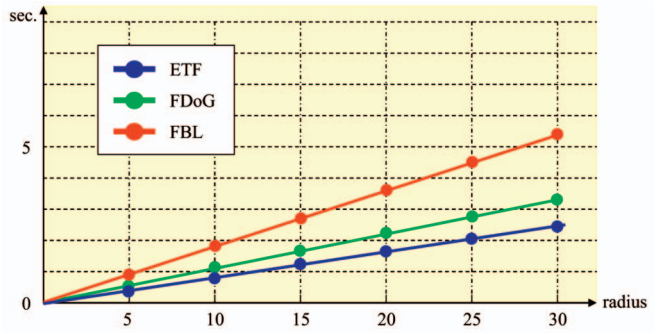



Fig. 23. Execution time for the filters.

Fig. 23 shows the timing data for the three filters obtained on a 512 $\times$ 512 RGB color image with varying kernel radius values. For the ETF filter, the term "kernel radius" refers to $\mu$, and for the FDoG and FBL filters, it refers to both $\alpha$ and $\beta$, where we set $\alpha = \beta$ for this experiment. All three of these filters have a linear time complexity with respect to the kernel radius. In case of the FBL filter, its speed is comparable to that of the separable bilateral filter [50], especially when $\alpha = \beta$.

It should be noted that, for bilateral filtering, there exist some sophisticated acceleration strategies based on histogram manipulation [55], image downsampling [40], [56], and grid processing [43]. We expect that our filters may be further accelerated when combined with such strategies, especially for large kernels. Also, the local nature of our filters should allow for a GPU-based parallel implementation, which would provide dramatic speedup and facilitate the processing of a large number of images.

## 6 Conclusions

We have presented an automatic technique for image abstraction based on a flow-based filtering framework. In particular, we have described the FDoG and FBL filters as new solutions to the two representative problems of image abstraction, line drawing, and region smoothing. Guided by the feature-preserving flow called ETF, these filters outperform their original counterparts in terms of feature enhancement and stylization, resulting in the production of high-quality image abstraction, as well as effective communication of shapes and colors.

Line drawing is generally considered the cornerstone of NPR. Since the FDoG filter constructs lines of style and quality, it may be used to enhance other image-based NPR effects such as pen-and-ink illustration, stippling, engraving, mosaics, pencil drawing, painting, and image puzzles. Its capability of coherence enhancement and noise suppression suggests its usefulness beyond NPR and thus deserves further investigation. As discussed in Section 4, the bilateral filter is employed in a variety of graphics applications, some of which may benefit from the feature-enhancing nature of the FBL filter. Our ETF-driven image filtering framework is general and independent of the underlying filter, and thus, it is possible to similarly apply other filters or algorithms to obtain feature-enhanced results or provide local guidance in a host of image-related tasks such as

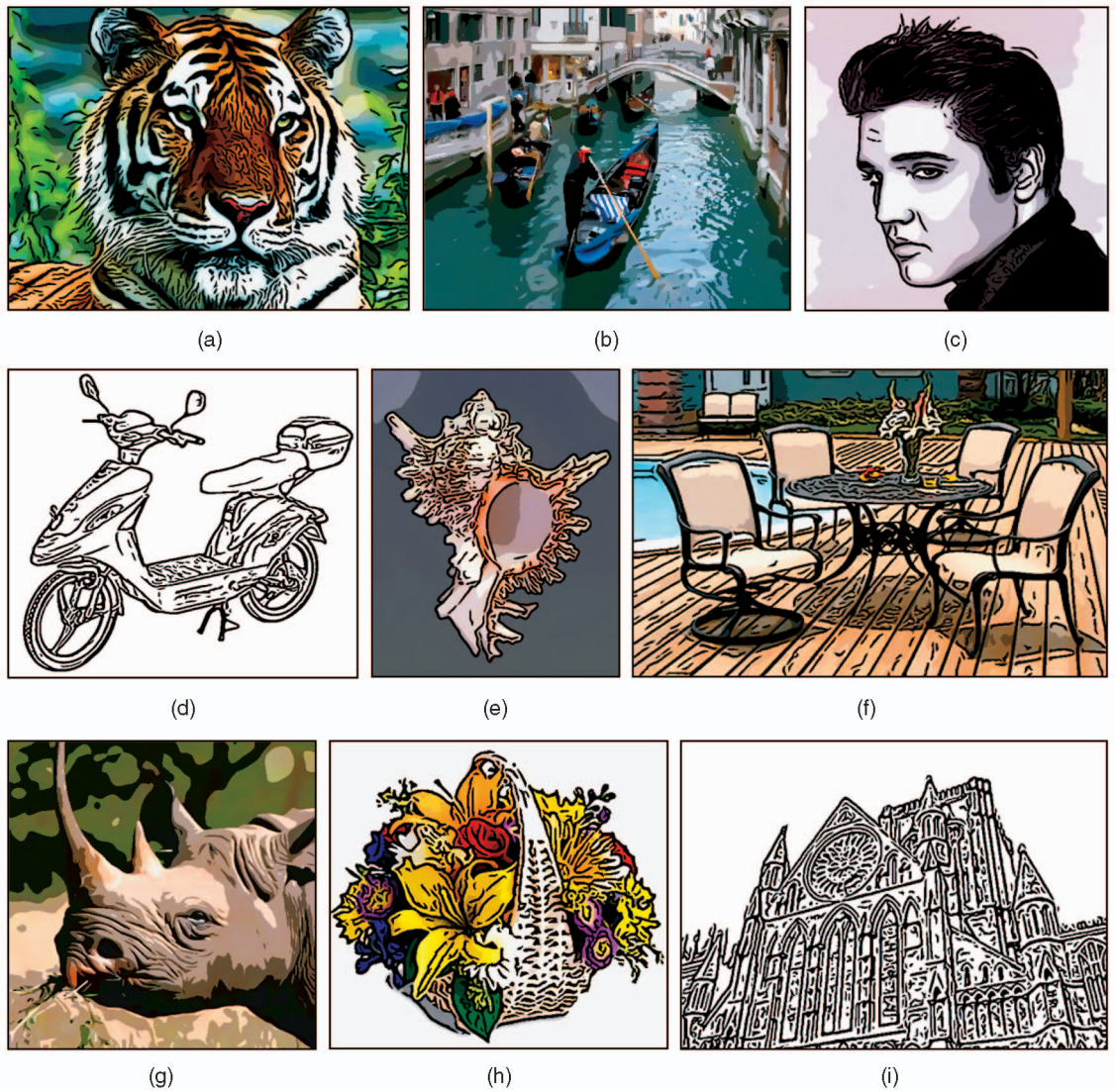



Fig. 24. Results. (a) Tiger. (b) Venice. (c) Elvis. (d) Bicycle. (e) Shell. (f) Terrace. (g) Rhino. (h) Basket. (i) Cathedral.

stroke-based rendering, texture synthesis, and image magnification to name a few.

Since our filters follow the precomputed vector directions, the quality of the underlying vector field (ETF in our case) is crucial in the success of filtering. That is, if ETF is somewhat inaccurate or does not properly represent the local features to preserve, the output of the FDoG and FBL filters could also suffer. This particularly limits the ability of our filters in capturing tiny-scale details and textures, which calls for the use of a proportionally small ETF kernel. One possible solution could be the use of an adaptive kernel based on the analysis of scene complexity. Another interesting future research involves the development of 3D tangent vector fields for surfaces [57], possibly through ETF construction on a 2D parameter space or its extension to a grid of 3D voxels. As for the abstraction scheme as a whole, a logical next step might be to incorporate more intelligence or prior knowledge to provide a significantly artistic or hand-made look in the final illustration.

Although video is not the primary target of this work, we also tested our scheme on some videos (some of which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2008.81) using a frame-by-frame application of our filters. While preliminary, the test result on video was promising in that a similar feature-enhancing effect was obtained without significant temporal artifacts. This suggests that the flow-based filtering principle may be useful in handling a variety of video-related tasks as well, particularly the ones that involve feature detection and enhancement.

## Acknowledgments

This research was supported in part by ARO Grant W911NF-07-1-0525 and DARPA/NGA Grant HM-1582-05-2-2003. It was also supported by the IT R&D program of MKE/IITA (2008-F-031-01, Development of Computational Photography Technologies for Image and Video Contents). Photos courtesy of the US fish and wildlife digital library (www.fws.gov/dls), the Kodak true color image suite (r0k.us/graphics/kodak), the USC-SIPI image database (sipi.usc.edu/database), the MedPix image database (rad. usuhs.mil/medpix), and Flickr (www.flickr.com).

Fig. 25. Flow-based abstraction of a large image.

## REFERENCES


[1] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive Contours for Conveying Shape," *Proc. ACM SIGGRAPH '03,* pp. 848-855, 2003.

[2] R.D. Kalnins, P.L. Davidson, L. Markosian, and A. Finkelstein, "Coherent Stylized Silhouettes," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 856-861, July 2003.

[3] M. Sousa and P. Prusinkiewicz, "A Few Good Lines: Suggestive Drawing of 3D Models," *Computer Graphics Forum,* vol. 22, no. 3, 2003.

[4] T. Isenberg, B. Freudenberg, N. Halper, S. Schlechtweg, and T. Strothotte, "A Developer's Guide to Silhouette Algorithms for Polygonal Models," *IEEE Computer Graphics and Applications,* vol. 23, no. 4, pp. 28-37, July/Aug. 2003.

[5] Y. Lee, L. Markosian, S. Lee, and J.F. Hughes, "Line Drawings via Abstracted Shading," *Proc. ACM SIGGRAPH,* 2007.

[6] T. Judd, F. Durand, and E. Adelson, "Apparent Ridges for Line Drawing," *Proc. ACM SIGGRAPH,* 2007.

[7] T. Goddwin, I. Vollick, and A. Hertzmann, "Isophote Distance: A Shading Approach to Artistic Stroke Thickness," *Proc. Non-Photorealistic Animation and Rendering (NPAR '07),* pp. 53-62, 2007.

[8] D. DeCarlo and S. Rusinkiewicz, "Highlight Lines for Conveying Shape," *Proc. Non-Photorealistic Animation and Rendering (NPAR '07),* pp. 63-70, 2007.

[9] A. Lake, C. Marshall, M. Harris, and M. Blackstein, "Stylized Rendering Techniques for Scalable Real-Time 3D Animation," *Proc. Non-Photorealistic Animation and Rendering (NPAR '07),* pp. 13-20, 2000.

[10] K. Anjyo and K. Hiramitsu, "Stylized Highlights for Cartoon Rendering and Animation," *IEEE Computer Graphics and Applications,* vol. 23, no. 4, pp. 54-61, July/Aug. 2003.

[11] P. Barla, J. Thollot, and L. Markosian, "X-Toon: An Extended Toon Shader," *Proc. Non-Photorealistic Animation and Rendering (NPAR '07),* pp. 127-132, 2006.

[12] P. Litwinowicz, "Processing Images and Video for an Impressionist Effect," *Proc. ACM SIGGRAPH '97,* pp. 407-414, 1997.

[13] A. Hertzmann, "Painterly Rendering with Curved Brush Strokes of Multiple Sizes," *Proc. ACM SIGGRAPH '98,* pp. 453-460, 1998.

[14] B. Gooch, G. Coombe, and P. Shirley, "Artistic Vision: Painterly Rendering Using Computer Vision Techniques," *Proc. Non-Photorealistic Animation and Rendering (NPAR '07),* pp. 83-90, 2002.

[15] J. Hays and I. Essa, "Image and Video-Based Painterly Animation," *Proc. Non-Photorealistic Animation and Rendering (NPAR '07),* pp. 113-120, 2004.

[16] M.P. Salisbury, S.E. Anderson, R. Barzel, and D.H. Salesin, "Interactive Pen-and-Ink Illustration," *Proc. ACM SIGGRAPH '94,* pp. 101-108, 1994.

[17] M. Salisbury, M. Wong, J. Hughes, and D. Salesin, "Orientable Textures for Image-Based Pen-and-Ink Illustration," *Proc. ACM SIGGRAPH '97,* pp. 401-406, 1997.

[18] M. Sousa and J. Buchanan, "Observational Models for Graphite Pencil Materials," *Computer Graphics Forum,* vol. 19, no. 1, pp. 27-49, 2000.

[19] F. Durand, V. Ostromoukhov, M. Miller, F. Duranleau, and J. Dorsey, "Decoupling Strokes and High-Level Attributes for Interactive Traditional Drawing," *Proc. Eurographics Workshop Rendering (EGRW '01),* pp. 71-82, 2001.

[20] O. Deussen, S. Hiller, K. Van Overveld, and T. Strothotte, "Floating Points: A Method for Computing Stipple Drawings," *Computer Graphics Forum,* vol. 19, no. 3, pp. 40-51, 2000.

[21] A. Secord, "Weighted Voronoi Stippling," *Proc. Non-Photorealistic Animation and Rendering (NPAR '02),* pp. 37-43, 2002.

[22] A. Hausner, "Simulating Decorative Mosaic," *Proc. ACM SIGGRAPH '01,* pp. 573-578, 2001.

[23] V. Ostromoukhov, "Digital Facial Engraving," *Proc. ACM SIGGRAPH '99,* pp. 417-424, 1999.

[24] A.W. Klein, P.-P. Sloan, A. Finkelstein, and M.F. Cohen, "Stylized Video Cubes," *Proc. Symp. Computer Animation (SCA '02),* pp. 15-22, 2002.

[25] J.P. Collomosse and P.M. Hall, "Cubist Style Rendering from Photographs," *IEEE Trans. Visualization and Computer Graphics,* vol. 9, no. 4, pp. 443-453, Oct.-Dec. 2003.

[26] D. DeCarlo and A. Santella, "Stylization and Abstraction of Photographs," *Proc. ACM SIGGRAPH '02,* pp. 769-776, 2002.

[27] J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, no. 6, pp. 679-698, 1986.
[28] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach toward Feature Space Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, pp. 603-619, 2002.
[29] J. Wang, Y. Xu, H.-Y. Shum, and M.F. Cohen, "Video Tooning," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 574-583, 2004.
[30] J.P. Collomosse, D. Rowntree, and P.M. Hall, "Stroke Surfaces: Temporally Coherent Non-Photorealistic Animations from Video," *IEEE Trans. Visualization and Computer Graphics,* vol. 11, no. 5, pp. 540-549, Sept./Oct. 2005.
[31] F. Wen, Q. Luan, L. Liang, Y.-Q. Xu, and H.-Y. Shum, "Color Sketch Generation," *Proc. Non-Photorealistic Animation and Rendering (NPAR '06),* pp. 47-54, 2006.
[32] J. Fischer, D. Bartz, and W. Strasser, "Stylized Augmented Reality for Improved Immersion," *Proc. IEEE Virtual Reality (VR '05),* pp. 195-202, 2005.
[33] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images," *Proc. IEEE Int'l Conf. Computer Vision (ICCV '98),* pp. 839-846, 1998.
[34] A. Orzan, A. Bousseau, P. Barla, and J. Thollot, "Structure-Preserving Manipulation of Photographs," *Proc. Non-Photorealistic Animation and Rendering (NPAR '07),* pp. 103-110, 2007.
[35] H. Kang, C. Chui, and U. Chakraborty, "A Unified Scheme for Adaptive Stroke-Based Rendering," *The Visual Computer,* vol. 22, no. 9, pp. 814-824, 2006.
[36] B. Gooch, E. Reinhard, and A. Gooch, "Human Facial Illustrations," *ACM Trans. Graphics,* vol. 23, no. 1, pp. 27-44, 2004.
[37] D. Marr and E.C. Hildreth, "Theory of Edge Detection," *Proc. Royal Soc. London,* pp. 187-217, 1980.
[38] H. Winnemöller, S. Olsen, and B. Gooch, "Real-Time Video Abstraction," *Proc. ACM SIGGRAPH '06,* pp. 1221-1226, 2006.
[39] B.M. Oh, M. Chen, J. Dorsey, and F. Durand, "Image-Based Modeling and Photo Editing," *Proc. ACM SIGGRAPH '01,* pp. 433-442, 2001.
[40] F. Durand and J. Dorsey, "Fast Bilateral Filtering for the Display of High-Dynamic-Range Images," *Proc. ACM SIGGRAPH '02,* pp. 257-266, 2002.
[41] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral Mesh Denoising," *Proc. ACM SIGGRAPH '03,* pp. 950-953, 2003.
[42] G. Petschnigg, M. Agrawala, H. Hoppe, R. Szeliski, M. Cohen, and K. Toyama, "Digital Photography with Flash and No-Flash Image Pairs," *Proc. ACM SIGGRAPH '04,* pp. 664-672, 2004.
[43] J. Chen, S. Paris, and F. Durand, "Real-Time Edge-Aware Image Processing with the Bilateral Grid," *Proc. ACM SIGGRAPH,* 2007.
[44] J. Kopf, M.F. Cohen, D. Lischinski, and M. Uyttendaele, "Joint Bilateral Upsampling," *Proc. ACM SIGGRAPH,* 2007.
[45] H. Kang, S. Lee, and C.K. Chui, "Coherent Line Drawing," *Proc. Non-Photorealistic Animation and Rendering (NPAR '07),* pp. 43-50, Aug. 2007.
[46] P. Perona, "Orientation Diffusions," *IEEE Trans. Image Processing,* vol. 7, no. 3, pp. 457-467, 1998.
[47] D. Tschumperlé and R. Deriche, "Orthonormal Vector Sets Regularization with PDE's and Applications," *Int'l J. Computer Vision,* vol. 50, no. 3, pp. 237-252, 2002.
[48] S. Paris, H. Briceño, and F. Sillion, "Capture of Hair Geometry from Multiple Images," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 712-719, 2004.
[49] B. Cabral and L. Leedom, "Imaging Vector Fields Using Line Integral Convolution," *Proc. ACM SIGGRAPH '93,* pp. 263-270, 1993.
[50] T.Q. Pham and L. van Vliet, "Separable Bilateral Filtering for Fast Video Preprocessing," *Proc. IEEE Conf. Multimedia and Expo (ICME),* 2005.
[51] T.Q. Pham, "Spatiotonal Adaptivity in Super-Resolution of Undersampled Image Sequences," PhD dissertation, Delft Univ. of Technology, Jan. 2006.
[52] L. Alvarez, P. Lions, and J. Morel, "Image Selective Smoothing and Edge Detection by Nonlinear Diffusion II," *SIAM J. Numerical Analysis,* vol. 29, no. 3, pp. 845-866, 1992.
[53] J. Weickert, "Anisotropic Diffusion in Image Processing," PhD dissertation, Dept. of Math., Univ. of Kaiserslautern, Jan. 1996.
[54] D. Tschumperlé, "Curvature-Preserving Regularization of Multi-Valued Images Using PDE's," *Proc. European Conf. Computer Vision (ECCV '06),* pp. 295-307, 2006.
[55] B. Weiss, "Fast Median and Bilateral Filtering," *Proc. ACM SIGGRAPH '06,* pp. 519-526, 2006.
[56] S. Paris and F. Durand, "A Fast Approximation of the Bilateral Filter Using a Signal Processing Approach," *Proc. European Conf. Computer Vision (ECCV '06),* pp. 568-580, 2006.
[57] M. Fisher, P. Schroder, M. Desbrun, and H. Hoppe, "Design of Tangent Vector Fields," *Proc. ACM SIGGRAPH,* 2007.

**Henry Kang** received the BS degree in computer science from Yonsei University, Korea, in 1994 and the MS and PhD degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST) in 1996 and 2002, respectively. He is an assistant professor of computer science at the University of Missouri, St. Louis. His research interests include nonphotorealistic rendering and animation, illustrative visualization, image and video processing, image-based modeling and rendering, and facial expression animation. He is a member of the IEEE.

**Seungyong Lee** received the BS degree in computer science and statistics from Seoul National University in 1988 and the MS and PhD degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST) in 1990 and 1995, respectively. He is an associate professor of computer science and engineering at the Pohang University of Science and Technology (POSTECH), Korea. From 1995 to 1996, he worked at the City College of New York as a postdoctoral research associate. Since 1996, he has been a faculty member and leading the Computer Graphics Group, POSTECH. From 2003 to 2004, he spent a sabbatical year at MPI Informatik, Germany, as a visiting senior researcher. His current research interests include 3D mesh processing, nonphotorealistic rendering, image and video processing, 3D surface reconstruction, and mobile graphics systems. He is a member of the IEEE.

**Charles K. Chui** received the BS, MS, and PhD degrees from the University of Wisconsin, Madison. He is currently a Curators' professor at the University of Missouri, St. Louis, and a consulting professor of statistics at Stanford University. His research interests include approximation theory, computational harmonic analysis, surface subdivisions, and mathematics of imaging. He is a coeditor in chief of *Applied and Computational Harmonic Analysis* and serves on the editorial board of seven other journals. He is a fellow of the IEEE.