Coherent Line Drawing

Henry Kang*
University of Missouri, St. Louis

Seungyong Lee†
POSTECH

Charles K. Chui‡
University of Missouri, St. Louis

(a) Input image (b) Isotropic DoG filtering (c) Our method

Figure 1: *Line drawing example*

Abstract

This paper presents a non-photorealistic rendering technique that automatically generates a line drawing from a photograph. We aim at extracting a set of coherent, smooth, and stylistic lines that effectively capture and convey important shapes in the image. We first develop a novel method for constructing a smooth direction field that preserves the flow of the salient image features. We then introduce the notion of flow-guided anisotropic filtering for detecting highly coherent lines while suppressing noise. Our method is simple and easy to implement. A variety of experimental results are presented to show the effectiveness of our method in producing self-contained, high-quality line illustrations.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation; I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms;

Keywords: non-photorealistic rendering, line drawing, edge detection, flow-based filtering

1 Introduction

Line drawing is arguably the simplest and oldest means of visual communication which dates back to prehistoric ages. As a crude form of human interpretation of a real scene, line drawing uses the minimal amount of data (that is, lines) and yet effectively conveys object shapes to the viewer. It even outperforms photorealistic imagery in terms of the efficacy of visual information transfer and subject identification. In this paper, we focus on 'pure' black-and-white line drawing, where lines are used to convey the shape of an object but not the tonal information on the object surface.

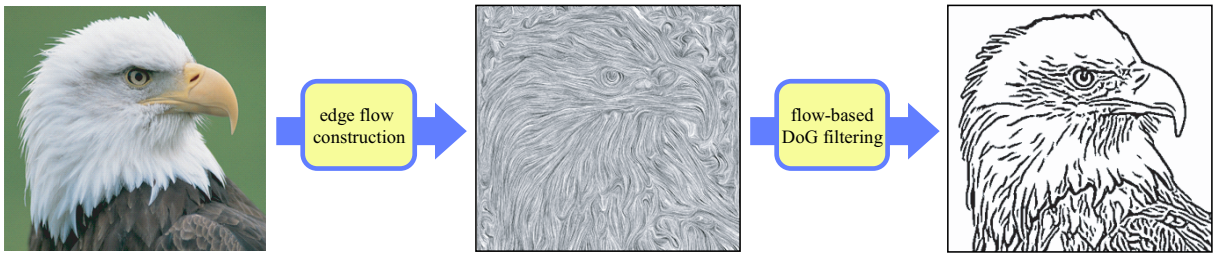
We present an automatic technique that generates a high-quality line drawing from a photograph, where the meaningful structures in the scene are captured and displayed with a set of clean, smooth, coherent, and stylistic lines, with little or no clutter. The main contribution of this paper lies in the introduction of a novel *flow-driven anisotropic filtering* framework. We modify an existing edge detection filter so that it adapts to a highly anisotropic kernel defined by the 'flow' of salient image features, and thereby significantly enhance the coherence of the lines while suppressing noise. The resulting filter response directly serves as the targeted line illustration and does not require any post-processing.

1.1 Related work

In the community of non-photorealistic rendering (NPR), a variety of methods have been reported on the line drawing of 3D models [Markosian et al. 1997; Hertzmann and Zorin 2000; DeCarlo et al. 2003; Kalnins et al. 2003; Sousa and Prusinkiewicz 2003; Isenberg et al. 2003]. 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. Many of these issues are addressed by existing edge detection or image segmentation techniques, which, however, do not deal with aesthetic aspects as they are not intended for creating stylistic illustrations. Also, a crude edge map often fails to qualify as a good illustration, on account of spurious lines and missing lines.

Many image-based NPR techniques thus 'partially' use lines to help create illustrations other than pure line drawing, that is, the ones dealing with not only the outlines of regions but also their interior properties such as color, tone, material, etc. For example, Salisbury et al. [1994] developed a system for interactive pen-and-ink illustration, where they allow the use of an edge detector [Canny 1986] to construct the outline strokes and also to clip the interior strokes. Litwinowicz [1997] employed the same edge detector to similarly clip the paintbrush strokes and preserve feature lines. Ostromoukhov [1999] used interactively specified outlines as a basis for digital facial engraving. Similarly, algorithms for tile mosaics [Hausner 2001], jigsaw mosaics [Kim and Pellacini 2002] and stipple drawing [Deussen et al. 2000] all take advantage of user-

*e-mail: kang@cs.umsl.edu
†e-mail: leesy@postech.ac.kr
‡e-mail: chui@arch.umsl.edu



(a) Input image (b) Edge Tangent Flow (c) Line drawing

Figure 2: *Process overview*

defined outlines to enhance the look of the illustration.

Certain NPR styles such as 'photograph tooning' require more explicit display of lines. DeCarlo and Santella [2002] employed both Canny edge detector and mean-shift segmentation algorithm [Comaniciu and Meer 2002] to obtain cartoon-style image abstraction. Wang et al. [2004] and Collomosse et al. [2005] presented off-line video abstraction techniques based on mean-shift segmentation, focusing on achieving good spatio-temporal coherence. Wen et al. [2006] also use mean-shift segmentation for producing a rough sketch of the scene. Fischer et al. [2005] applied Canny edge detector in conjunction with bilateral filter to obtain stylized augmented reality. Canny's method was again used by Kang et al. [2006] in producing a wide variety of artistic styles, ranging from region-based to outline-based. As shown in [DeCarlo and Santella 2002], image segmentation is useful for abstracting regions (and color) but less ideal for line drawing, because each segmented region inevitably forms a closed boundary which does not necessarily match the accurate outline.

While Canny's method [1986] is generally considered the *de facto* standard for edge detectors, one may choose other edge detection method for line drawing. Gooch et al. [2004] presented a facial illustration system based on a difference-of-Gaussians (DoG) filter, originated from Marr–Hildreth edge detector [1980]. They used this filter in conjunction with binary thresholding to produce a black-and-white illustration. Winnemöller et al. [2006] recently extended this technique to general color images and video. Compared with Canny's method, their DoG edge model has proven to be more effective for artistic illustrations in the following respects: It captures interesting structures better (as shown in [Gooch et al. 2004]), and it automatically produces a set of stylistic edges (in non-uniform thickness).

This DoG edge model, however, is not without limitations. Due to the nature of isotropic filter kernel, the aggregate of edge pixels may not clearly reveal the sense of 'directedness' (and thus may look less like lines). Also, the thresholded edge map may exhibit some set of isolated, scattered edge components that clutter the output, especially in an area with image noise or weak contrast (see Fig. 1b). Although we 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 anisotropic filtering framework (see Fig. 1c).

1.2 Contributions and Overview

The main idea behind our approach is to take into account the 'direction' of the local image structure in DoG filtering, rather than looking in all directions. Especially, we apply DoG filter only in a direction perpendicular to that of the local 'edge flow', that is, the direction in which there is supposed to exist the biggest contrast. We then collect the evidence (filter response) along this flow to finally determine the edge strength. As will be shown in this paper, this flow-guided filtering approach not only enhances the coherence of the lines, but also suppresses noise.

Our technical contributions are two-fold. For computing a smooth, feature-preserving local edge flow (called *edge tangent flow*), we develop a kernel-based nonlinear vector smoothing technique (described in Section 2). Also, we present a flow-based anisotropic DoG filtering technique (described in Section 3) that directly produces the line illustration. Fig. 2 shows the overview of our method.

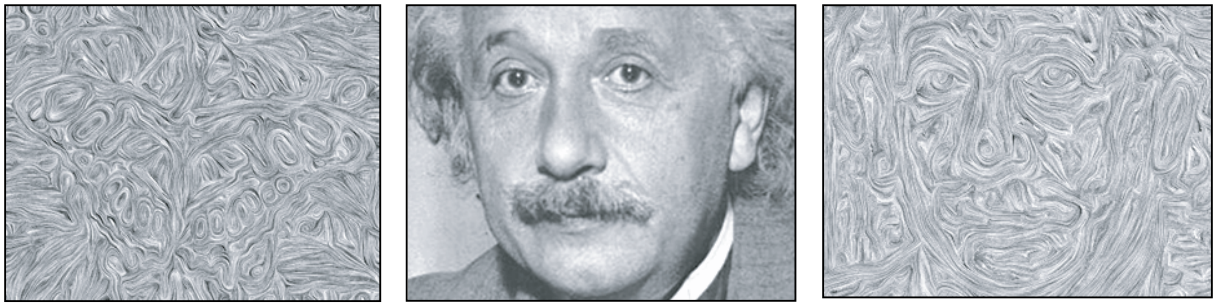
Our line drawing framework thus gives advantages in the following respects:
Line coherence: Our technique differs from conventional edge detectors in that it uses a highly anisotropic, curve-shaped filter kernel in order to maximize the line coherence. It is even capable of constructing a line from a set of isolated edge points, by adjusting the kernel size.
Robustness: Our method is robust and less susceptible to image noise. Thus, it reduces spurious lines and produces smooth line strokes.
Quality: With the properties above, the filter output often directly results in a good-quality line illustration.
Simplicity: Our method is straightforward and easy to implement.
Generality: Our flow-based filtering framework is general. Other filters may be applied similarly to improve their performance (in terms of feature preservation).

2 Flow construction

2.1 Edge Tangent Flow

We first construct the edge flow field from input image $I(\mathbf{x})$, where $\mathbf{x} = (x, y)$ denotes an image pixel. To facilitate the production of high-quality line drawing, this vector field must satisfy the following requirements: (1) The vector flow must describe the salient edge tangent direction in the neighborhood; (2) The neighboring vectors must be smoothly aligned except at sharp corners; (3) Important edges must retain their original directions. Here, we define the *edge tangent*, denoted $\mathbf{t}(\mathbf{x})$, as a vector perpendicular to the image gradient $\mathbf{g}(\mathbf{x}) = \nabla I(\mathbf{x})$. We use the term 'tangent' in a sense that $\mathbf{t}(\mathbf{x})$ may be viewed as the tangent of a curve representing the local edge flow. Accordingly, we call this vector field an *edge tangent flow (ETF)*.

We present a novel technique for constructing ETF that meets all the requirements stated above. Our method uses a kernel-based nonlinear smoothing of vector field, inspired by the bilateral filtering framework [Tomasi and Manduchi 1998]. In each pixel-centered kernel, we perform a nonlinear vector smoothing, such that the salient edge directions are preserved while weak edges are directed to follow the neighboring dominant ones. Also, to preserve



(a) ETF (Butterfly – Fig. 1a) (b) Einstein (c) ETF (Einstein)

Figure 3: *ETF construction*

sharp corners and avoid undesirable 'swirling' artifact, we encourage smoothing among the edges with similar orientations. This also prevents weak vectors from being affected by strong but irrelevant vectors, and thus results in more tightly aligned vectors. Our ETF construction filter is thus defined as follows:

$$\mathbf{t}^{new}(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{y} \in \Omega(\mathbf{x})} \phi(\mathbf{x}, \mathbf{y}) \mathbf{t}^{cur}(\mathbf{y}) w_s(\mathbf{x}, \mathbf{y}) w_m(\mathbf{x}, \mathbf{y}) w_d(\mathbf{x}, \mathbf{y}) \tag{1}$$

where $\Omega(\mathbf{x})$ denotes the neighborhood of $\mathbf{x}$, and k is the vector normalizing term. For the *spatial weight function* w_s, we use a radially-symmetric box filter of radius r, where r is the radius of the kernel Ω:

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

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

$$w_m(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \left(1 + \tanh[\eta \cdot (\hat{g}(\mathbf{y}) - \hat{g}(\mathbf{x}))]\right) \tag{3}$$

where $\hat{g}(\mathbf{z})$ denotes the normalized gradient magnitude at $\mathbf{z}$, and η controls the fall-off rate. Note that this weight function is monotonically increasing, 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. A bigger value of η means more strict obedience to the dominant vectors. We set $\eta = 1$ throughout.

Finally, we define w_d, the *direction weight function*, as follows:

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

where $\mathbf{t}^{cur}(\mathbf{z})$ denotes the 'current' normalized tangent vector at $\mathbf{z}$. Note that this weight function increases as the two vectors are closely aligned (that is, the angle θ between two vectors gets close to 0° or 180°), and decreases as they become perpendicular (that is, θ approaches 90°). In addition, we reverse the direction of $\mathbf{t}^{cur}(\mathbf{y})$ before smoothing (in Eq. 1) using the sign function $\phi(\mathbf{x}, \mathbf{y}) \in \{1, -1\}$, in case θ is bigger than 90°:

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

This induces tighter alignment of vectors while avoiding swirling flows.

The initial ETF, denoted $\mathbf{t}^0(\mathbf{x})$, is obtained by taking perpendicular vectors (in the counter-clockwise 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 Sobel operator. Our filter may be iteratively applied to update ETF incrementally: $\mathbf{t}^i(\mathbf{x}) \rightarrow \mathbf{t}^{i+1}(\mathbf{x})$. Note 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. 3 shows ETF fields obtained from sample images. Notice ETF preserves edge directions well around important features while keeping them smooth elsewhere.

2.2 Discussion

It is worth noting that there are other ways to construct ETF. In non-photorealistic painterly rendering, scattered data interpolation is often used to create a rough ETF, based on radial basis functions [Litwinowicz 1997; Hays and Essa 2004]. The resulting ETF is used to guide the placement of brush strokes. Typically, a small number of basis points (with strong gradients) are selected for vector interpolation, and as a result the flow can be easily misguided in an area where some meaningful basis points are omitted by the selection process (such as gradient magnitude thresholding). This in general does not pose a huge problem in painterly rendering, where features are often deliberately obscured for an aesthetic purpose. When applied to line drawing, however, the quality of illustration can be degraded noticeably. Although one may attempt using a large number of basis points to resolve this, it could result in a noisy direction field (let alone the increased computational cost) as all the selected basis vectors cannot change the original directions.

It is possible to construct a more sophisticated ETF by taking into account the entire set of pixels. Xu and Prince [1998] formulated this problem as a partial differential equation, and presented an iterative *vector diffusion* algorithm to create a gradient vector flow (GVF), that is, a perpendicular version of ETF. They use GVF as an external force field to attract active contours. While ETF can also be obtained in this way, we find it less suited for our line-drawing application in that it does not preserve the salient edge tangent directions well, and produces undesirable swirling flows (see Fig. 4-a2). This is attributed to the fact that the diffusion process takes into account only the magnitude of the vectors but not their directions. As shown in Fig. 4-a3, our method successfully removes such artifact. The second limitation of the vector diffusion method is that it lacks control over the size of diffusion kernel (only the immediate neighbors are considered) and thus it is difficult to establish coherence between isolated edges (see Fig. 4-b2). Our method addresses this problem by providing a size-controllable smoothing kernel (see the originally isolated components are connected by the flow in Fig. 4-b3).

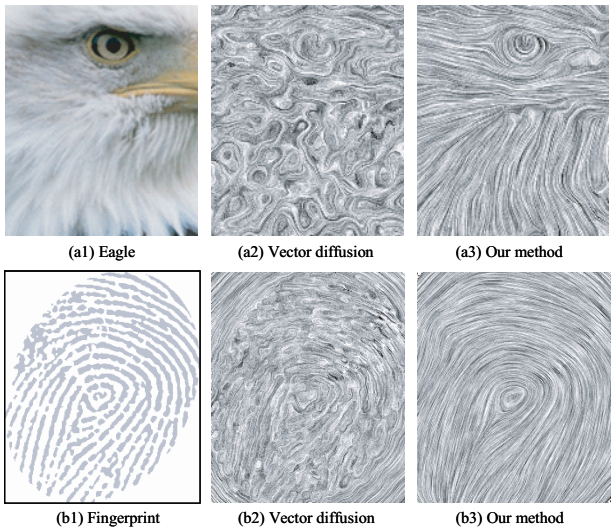


Figure 4: *Our method vs. Vector diffusion*

3 Line construction

3.1 Flow-based Difference-of-Gaussians

Given $\mathbf{t}(\mathbf{x})$, namely the ETF constructed by Eq. 1, we apply a flow-guided anisotropic DoG filter using the kernel whose shape is defined by the local flow recorded in ETF. Note $\mathbf{t}(\mathbf{x})$ represents the local edge direction, which means we will most likely have the highest contrast in its perpendicular direction, that is, the gradient direction $\mathbf{g}(\mathbf{x})$. Thus, 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'). As a result, we can 'exaggerate' the filter output along genuine edges, while we 'attenuate' the output from spurious edges. Therefore, this not only enhances the coherence of the edges, but also has the effect of suppressing noise.

Fig. 5 illustrates our filtering framework. Let $c_{\mathbf{x}}(s)$ denote the integral curve (also called stream line) 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, $c_{\mathbf{x}}(0) = \mathbf{x}$. Our filtering framework is then described as follows. As moving along $c_{\mathbf{x}}$, we apply a 1-dimensional filter f along the line l_s that is perpendicular to $\mathbf{t}(c_{\mathbf{x}}(s))$ and intersecting $c_{\mathbf{x}}(s)$:

$$F(s) = \int_{-T}^{T} I(l_s(t)) f(t) dt \tag{6}$$

where $l_s(t)$ denotes the point on the line l_s at parameter t. Again t is an arc-length parameter, and we assume l_s is centered at $c_{\mathbf{x}}(s)$, that is, $l_s(0) = c_{\mathbf{x}}(s)$. Note l_s is parallel to the gradient vector $\mathbf{g}(c_{\mathbf{x}}(s))$. $I(l_s(t))$ represents the value of the input image I at $l_s(t)$.

As for f, we employ the edge model suggested by Winnemöller et al. [2006] based on difference-of-Gaussians (DoG):

$$f(t) = G_{\sigma_c}(t) - \rho \cdot G_{\sigma_s}(t) \tag{7}$$

where G_σ denotes a 1-dimensional Gaussian function of variance σ:

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{8}$$

The two variances, σ_c and σ_s, are for center and surrounding sections, respectively. We set $\sigma_s = 1.6\sigma_c$ to make the shape of f closely resemble that of Laplacian-of-Gaussian [Marr and Hildreth 1980]. Therefore, once σ_c is given by the user, it automatically determines σ_s and thus the size of T in Eq. 6. It also directly affects the resulting line width. ρ controls the level of noise detected, and is set to a default value of 0.99.

The individual filter responses $F(s)$ are then accumulated along $c_{\mathbf{x}}$:

$$H(\mathbf{x}) = \int_{-S}^{S} G_{\sigma_m}(s) F(s) ds \tag{9}$$

where F is convolved with a Gaussian function to assign a variable weight to each response according to s. The user-provided parameter σ_m automatically determines the size of S. Note σ_m controls the length of the elongated flow kernel, and hence the degree of line coherence to enforce.

Once we obtain H from Eq. 9, we convert it to a black-and-white image by binary thresholding, as suggested in [Winnemöller et al. 2006]:

$$\tilde{H}(\mathbf{x}) = \begin{cases} 0 & \text{if } H(\mathbf{x}) < 0 \text{ and } 1 + \tanh(H(\mathbf{x})) < \tau, \\ 1 & \text{otherwise.} \end{cases} \tag{10}$$

where τ is a threshold in $[0, 1]$. This binary output $\tilde{H}$ serves as our targeted line illustration.

Since our anisotropic DoG filter is driven by the vector flow, we name it Flow-based Difference-of-Gaussians (FDoG) filter.

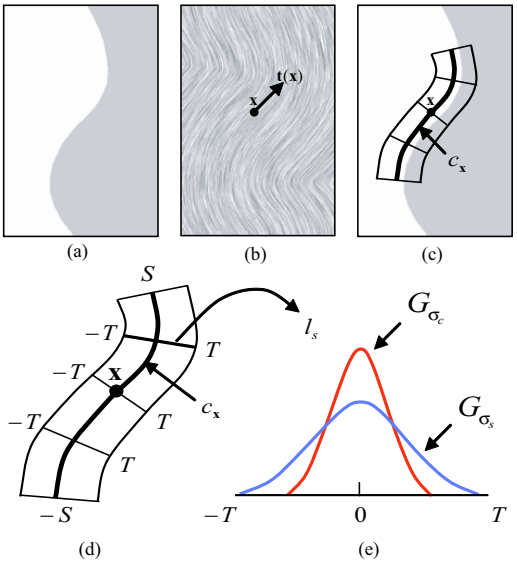
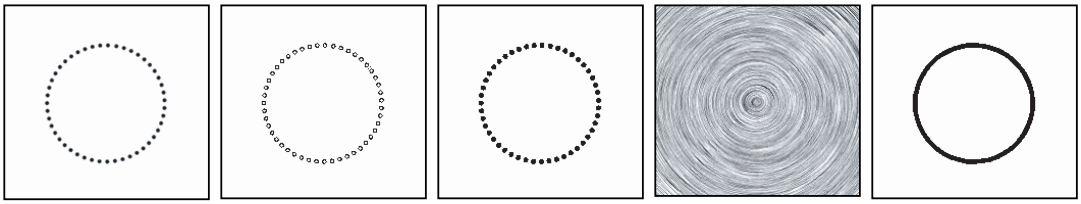


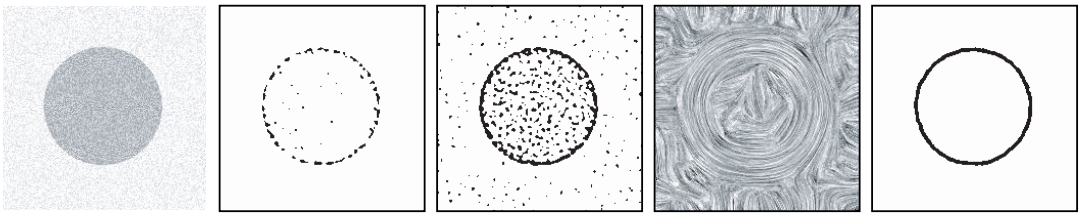
Figure 5: *Flow-based DoG filtering: (a) Input (b) ETF (c) Kernel at* $\mathbf{x}$ *(d) Kernel enlarged (e) Gaussian components for DoG*

For implementation, we sample $p \times q$ points from the kernel and discretely approximate Eq. 6 and Eq. 9. We first sample p points along $c_{\mathbf{x}}$ by bidirectionally following the vector flow starting from $\mathbf{x}$. 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 δ_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 q points along the line perpendicular to $\mathbf{t}(\mathbf{z})$, similarly with the step size of δ_n. In our experiments, we set $\delta_m = \delta_n = 1$. Note p and q are automatically determined by σ_m and σ_c, respectively.



(a) Input (b) Canny (c) Isotropic DoG (d) ETF (e) FDoG

Figure 7: *Line construction from a set of isolated points*



(a) Input (b) Isotropic DoG (c) Isotropic DoG (d) ETF (e) FDoG

Figure 8: *Noise suppression by FDoG*

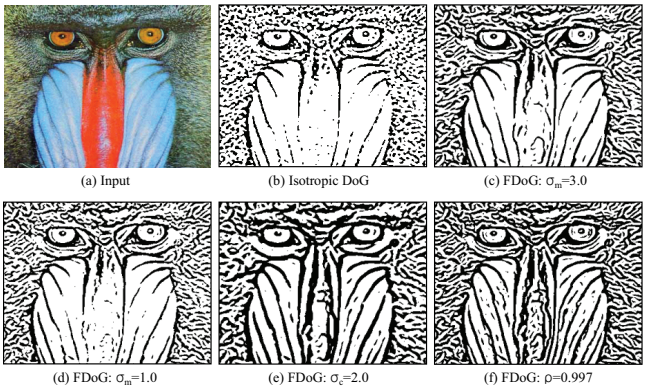


Figure 6: *FDoG filtering with parameter control*

Fig. 6 shows applications of our FDoG filter with differing parameter values. Each caption gives the modified parameter value from the default setting: $\sigma_m = 3.0$, $\sigma_c = 1.0, \rho = 0.99$. Notice the improved line coherence compared to that of the isotropic DoG filter. Fig. 7 shows that unlike conventional edge detectors, it is even possible to find a line from a set of disconnected points, by simply setting the ETF kernel (Ω in Eq. 1) bigger than the distance between neighboring points. Fig. 8 demonstrates the robustness of FDoG against noise. Fig. 8a is an image corrupted by Gaussian noise. Fig. 8b is an output of isotropic DoG filter followed by binarization with a low threshold ($\tau = 0.2$). Notice the weak line coherence due to noise. While higher threshold ($\tau = 0.7$) as in Fig. 8c improves the coherence, the added noise clutters the output. On the other hand, Fig. 8d shows that we can still construct a relatively smooth ETF around the target shape, resulting in the extraction of a clean, coherent line (Fig. 8e) at a low threshold ($\tau = 0.2$).

3.2 Iterative FDoG filtering

Our FDoG filter may be iteratively applied to further enhance the filter response. In fact, we find that iterative FDoG filtering is often more effective in improving line coherence in a consistent manner than simply adjusting parameters in a single FDoG application. After each application of FDoG, we re-initialize the filter input by superimposing the black edge pixels of the previous binary output $\tilde{H}$ (obtained by Eq. 10) upon the original image I, then re-apply 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 FDoG filter, we may optionally Gaussian-blur the input to further smooth out the line strokes. Fig. 9 shows the iterative FDoG filtering (applied to Fig. 4-b1) that successively enhances the coherence of lines. Notice many of the originally disconnected components of the fingerprint are re-connected.

(a) 1st iteration (b) 2nd iteration (c) 3rd iteration

Figure 9: *Iterative FDoG*

3.3 Discussion

The idea of using directional filter kernel is not new. Canny [1986] discussed the use of directional Gaussian derivative for improving the reliability in straight line detection. Gabor filters [Grigorescu et al. 2002] and other steerable filters [Freeman and Adelson 1991] typically employ a set of oriented, elliptic kernels to help analyze parallel structures or texture patterns. In anisotropic mean-shift segmentation [Wang et al. 2004], a data-dependent elliptic kernel is used to capture thin, elongated structures with less regions. Note, however, conventional anisotropic kernels still use a rigid, straight-line axis, while we use a 'curved' axis representing the local edge flow, and thus make it more effective in capturing the actual local

structure, whether it is a straight line or a curve with nonzero curvature.

Since our filtering approach is based on the flow of vectors, it is more closely related to the line integral convolution (LIC) framework [Cabral and Leedom 1993], which is often used for the visualization of vector field (and it is actually used for visualizing ETF in this paper). The main difference is that we add another dimension to the conventional LIC in defining the filter kernel (that is, we expand the curvilinear kernel sideways as shown in Fig. 5). It should also be noted that the quality of the underlying vector field (ETF in our case) is crucial in the success of flow-based filtering.

4 Results

Fig. 11 shows line drawing results obtained from the test images in Fig. 10. Our method performs consistently well on a variety of images with different characteristics and subjects, such as humans, animals, plants, buildings, still objects, outdoor scenes, etc. Note each filter output consists of lines that are clean, smooth, and also coherent, with little or no dispersement. Also, due to the inherent property of DoG filtering, the captured lines are stylistically depicted such that the thickness of the line automatically reflects the significance of the edge. Fig. 12 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 is not only capable of capturing 'perceptually interesting' structures but also capable of depicting them with smooth and coherent lines.

Figure 10: *Test images*

The line drawings in Figs. 1, 2, 11, and 12 were obtained by setting $r = 5$ (ETF kernel size), $\sigma_m = 3.0, \sigma_c = 1.0, \tau = 0.5$ (filtering parameters) and with 3 FDoG iterations, while other parameters were set as default values. This shows that our method does not require sophisticated parameter tuning based on the input data. The performance mainly depends on the image size and the filter kernel size. Note that our ETF construction method is an $O(n \times r^2)$ algorithm where n is the number of image pixels, while FDoG filtering is of $O(n \times p \times q)$ where $p \times q$ is the number of sample points in a kernel, determined by σ_m and σ_c. In our current implementation, no attempt for acceleration has been made. For a 512×512 image and with default parameters, an application of ETF construction filter (Eq. 1) typically takes $2 \sim 4$ seconds, and FDoG filter (Eq. 9) takes $4 \sim 6$ seconds on a 3GHz dual-core PC.

5 Discussion and Future work

Line drawing is generally considered the cornerstone of non-photorealistic rendering, and we believe the lines constructed by our technique (and the ETF itself) can be used as the basis for enhancing other image-guided NPR effects such as cartooning, pen-and-ink illustration, stippling, engraving, mosaics, pencil drawing, painting, and so on. Our flow-based anisotropic filtering framework is general and independent of the underlying filter, and thus it is possible to similarly adapt other filters to obtain improved results. In fact, this suggests the possibility of developing a family of flow-based filters, which we are currently exploring.

Since our FDoG filter builds on the DoG filter, they share some of the limitations too. For example, a high-contrast background will be filled with a dense set of lines although this area may be perceptually unimportant. Also, like DoG, the lines are formed as pixel aggregates rather than well-defined strokes (although it is possible to extract a set of thin strokes from the thick lines by morphological processing). While FDoG filter is capable of connecting isolated edge segments, it still operates on a local kernel and thus a global-scale subjective contour may be hard to detect.

Possible future research directions include the development of acceleration schemes for the proposed filters (Eq. 1 and Eq. 9). Since our ETF construction filter is essentially a modified bilateral filter, it may be possible to employ existing acceleration schemes [Durand and Dorsey 2002; Weiss 2006]. The FDoG filtering may be similarly accelerated since there is a huge overlap between neighboring kernels. Also, the local nature of our filters suggests a (GPU-based) parallel implementation, which could even lead to a real-time performance.

Acknowledgements

This research is supported in part by ARO Grant #W911NF-04-1-0298 and DARPA/NGA Grant #HM-1582-05-2-2003. It is also supported by UM Research Board and the ITRC support program.

References

CABRAL, B., AND LEEDOM, L. 1993. Imaging vector fields using line integral convolution. In *Proc. ACM SIGGRAPH 93*, 263–270.

CANNY, J. 1986. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, 6, 679–698.

COLLOMOSSE, J. P., ROWNTREE, D., AND HALL, P. M. 2005. Stroke surfaces: Temporally coherent non-photorealistic animations from video. *IEEE Transactions on Visualization and Computer Graphics 11*, 5, 540–549.

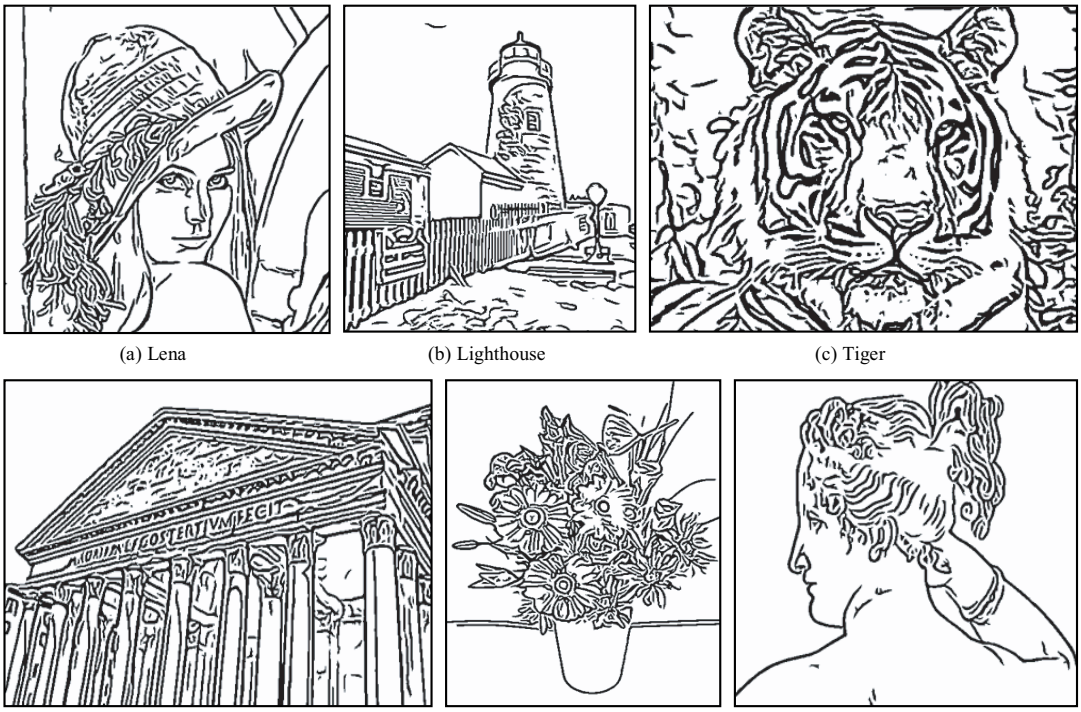
COMANICIU, D., AND MEER, P. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Machine Intell 24*, 5, 603–619.

DECARLO, D., AND SANTELLA, A. 2002. Stylization and abstraction of photographs. In *Proc. ACM SIGGRAPH 02*, 769–776.

DECARLO, D., FINKELSTEIN, A., RUSINKIEWICZ, S., AND SANTELLA, A. 2003. Suggestive contours for conveying shape. In *Proc. ACM SIGGRAPH 03*, 848–855.

DEUSSEN, O., HILLER, S., VAN OVERVELD, K., AND STROTHOTTE, T. 2000. Floating points: A method for computing stipple drawings. *Computer Graphics Forum 19*, 3, 40–51.

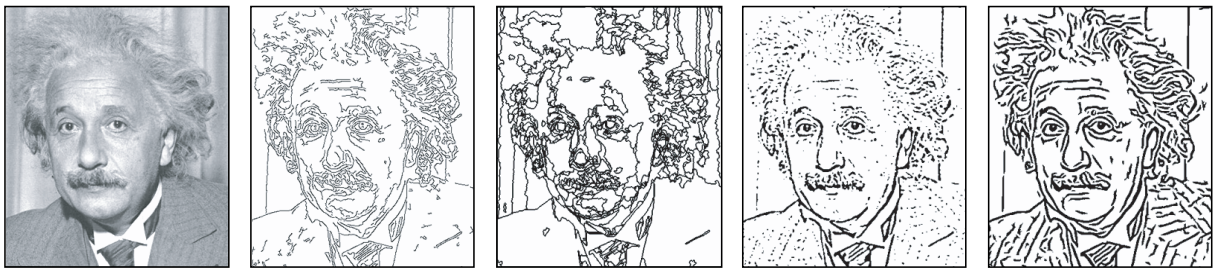
DURAND, F., AND DORSEY, J. 2002. Fast bilateral filtering for the display of high-dynamic-range images. In *Proc. ACM SIGGRAPH 02*, 257–266.



(a) Lena (b) Lighthouse (c) Tiger

(d) Pantheon (e) Flowers (f) Paolina

Figure 11: *Results*



(a) Input (b) Canny (c) Mean-shift (d) Isotropic DoG (e) FDoG

Figure 12: *Comparison with other techniques*

FISCHER, J., BARTZ, D., AND STRASSER, W. 2005. Stylized augmented reality for improved immersion. In *Proc. IEEE VR*, 195–202.

FREEMAN, W., AND ADELSON, E. 1991. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 9, 891–906.

GOOCH, B., REINHARD, E., AND GOOCH, A. 2004. Human facial illustrations. *ACM Transactions on Graphics 23*, 1, 27–44.

GRIGORESCU, S. E., PETKOV, N., AND KRUIZINGA, P. 2002. Comparison of texture features based on gabor filters. *IEEE Transactions on Image Processing 11*, 10, 1160–1167.

HAUSNER, A. 2001. Simulating decorative mosaic. In *Proc. ACM SIGGRAPH 01*, 573–578.

HAYS, J., AND ESSA, I. 2004. Image and video-based painterly animation. In *Proc. Non-Photorealistic Animation and Rendering*, 113–120.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. In *Proc. ACM SIGGRAPH 2000*, 517–526.

ISENBERG, T., FREUDENBERG, B., HALPER, N., SCHLECHTWEG, S., AND STROTHOTTE, T. 2003. A developer's guide to silhouette algorithms for polygonal models. *IEEE Computer Graphics & Applications 23*, 4, 28–37.

KALNINS, R. D., DAVIDSON, P. L., MARKOSIAN, L., AND FINKELSTEIN, A. 2003. Coherent stylized silhouettes. *ACM Transactions on Graphics 22*, 3 (July), 856–861.

KANG, H., CHUI, C., AND CHAKRABORTY, U. 2006. A unified scheme for adaptive stroke-based rendering. *The Visual Computer 22*, 9, 814–824.

KIM, J., AND PELLACINI, F. 2002. Jigsaw image mosaics. In *Proc. ACM SIGGRAPH 02*, 657–664.

LITWINOWICZ, P. 1997. Processing images and video for an impressionist effect. In *Proc. ACM SIGGRAPH 97*, 407–414.

MARKOSIAN, L., KOWALSKI, M. A., TRYCHIN, S. J., BOURDEV, L. D., GOLDSTEIN, D., AND HUGHES, J. F. 1997. Real-time nonphotorealistic rendering. In *Proc. ACM SIGGRAPH 1997*, 415–420.

MARR, D., AND HILDRETH, E. C. 1980. Theory of edge detection. In *Proc. Royal Soc. London*, 187–217.

OSTROMOUKHOV, V. 1999. Digital facial engraving. In *Proc. ACM SIGGRAPH 99*, 417–424.

SALISBURY, M. P., ANDERSON, S. E., BARZEL, R., AND SALESIN, D. H. 1994. Interactive pen-and-ink illustration. In *Proc. ACM SIGGRAPH 94*, 101–108.

SOUSA, M., AND PRUSINKIEWICZ, P. 2003. A few good lines: Suggestive drawing of 3D models. *Computer Graphics Forum 22*, 3, 381–390.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proc. ICCV*, 839–846.

WANG, J., XU, Y., SHUM, H.-Y., AND COHEN, M. F. 2004. Video tooning. *ACM Transactions on Graphics 23*, 3, 574–583.

WEISS, B. 2006. Fast median and bilateral filtering. In *Proc. ACM SIGGRAPH 06*, 519 – 526.

WEN, F., LUAN, Q., LIANG, L., XU, Y.-Q., AND SHUM, H.-Y. 2006. Color sketch generation. In *Proc. Non-Photorealistic Animation and Rendering*, 47–54.

WINNEMÖLLER, H., OLSEN, S., AND GOOCH, B. 2006. Real-time video abstraction. In *Proc. ACM SIGGRAPH 06*, 1221–1226.

XU, C., AND PRINCE, J. L. 1998. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing 7*, 3, 359–369.