Flow Based Image Abstraction

Aditya Garg

IMT-2021-005

ABV-Indian Institute of Information Technology and Management, Gwalior

imt_2021005@iiitm.ac.in

Apoorv Jain

IMT-2021-015

ABV-Indian Institute of Information Technology and Management, Gwalior

imt_2021015@iiitm.ac.in

Harsh Dalwadi

IMT-2021-026

ABV-Indian Institute of Information Technology and Management, Gwalior

imt_2021026@iiitm.ac.in

***Abstract*—A non-photorealistic rendering method is implemented that generates a stylized abstraction of a photograph automatically. The method relies on shape/color filtering, which is directed by a vector field characterizing the image's prominent feature flow. The abstraction performance is much enhanced in terms of feature enhancement and stylization by this flow-based filtering. The process is quick, straightforward, and simple to use. The outcomes of the experiments show how well the approach works to create visually appealing and feature-rich graphics from photos.**

***Keywords*—Non-photorealistic Rendering, Image Abstraction, Flow-Based Filtering, Line Drawing, Bilateral Filter**

I. INTRODUCTION

Non-photorealistic rendering (NPR) means abstraction and stylizing the image, which simplifies visual details and more effectively conveys features of the image. Line are very useful in getting information about shape. To further help in object identification need to paint items with a limited palette of colors. In this project, we present an automatic method that takes a photograph and creates a stylistic visual abstraction.

A. Problem Statement

There are two tasks in image abstraction: line extraction and region smoothing.

B. Contribution and Overview

A flow-driven approach has been presented for both line extraction and region smoothing. This technique uses the bilateral filter for region smoothing and the DoG filter for line extraction but it considers the "direction" of the local image to fit a curved kernel that resembles to the "edge flow" in the area. The final image abstraction is created by combining the the result of line extraction and region smoothing.

Also it has been shown that how different value of kernel radius, number of iterations of ETF filter affect the edges produced. Apart from the Difference of Gaussians filter that is used in FDoG, Extended Difference of Gaussians Filter and Laplacian of Gaussians filter were tried as alternatives in the edge detection process and the advantages and disadvantages of these filters over the Difference of Gaussians filter were noted. Different values of constants were taking during thresholding and their outputs were observed. Constants such as the standard deviations σ_m ,σ_c ,σ_s were changed and their results were observed.

II. FLOW CONSTRUCTION

A. Edge Tangent Flow

The edge flow field for input image I(x) is constructed, where x =(x,y) is an image pixel. Following points were taken into consideration for the generation of lines: (1) The salient edge's tangent direction in the neighborhood must be described by the vector flow; (2) The neighbouring vectors must be smoothly aligned except at sharp corners; and (3) significant edges must maintain their original directions. t(x) is curve's tangent which is a vector perpendicular to the image gradient, g(x) = ΔI(x).

B. Formulation

To generate the t(x) first g(x) need to be calculated. Sobel operator is used to calculate the g(x).

The initial ETF, is normalized. t(x) is created by taking perpendicular vectors from the initial gradient map g(x) of the input image. Then, it is normalized. Now the goal is to get the tangents as per the conditions stated in section A.

The ETF construction filter is thus defined as follows:

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

Ω_μ(x) means the kernel of radius μ at x, and k is the vector normalisation term.

w_s is spatial weight function to ensure the neighbours in radius μ are selected

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

w_m the magnitude weight function, which is defined as

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

where $\hat{g}(z)$ is the normalized gradient magnitude at z. Greater weights are given to the dominant neighbours.

w_d the direction weight function, to promote smoothing among similar orientations:

$$w_d(x,y) = |t(x).t(y)|$$

Direction wight increases as both are aligned similarly and decreases if both are aligned oppositely.

ϕ(x,y) is used to change the direction of tangent to ensure that negihbours with similar orientation but opposite direction can contribute in the edge flow

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

Filter can be iteratively applied to get desired results. See Fig (2.1).

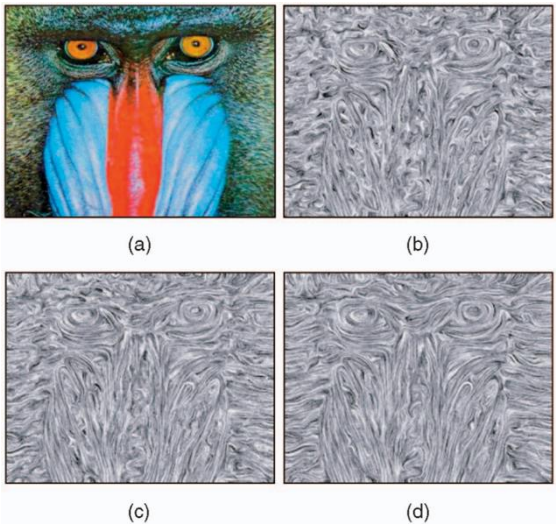


Fig 2.1. (a) input (b) ETF iteration-1
(c) ETF iteration-2 (d) ETF iteration-3

C. Comparision with vector diffusion

To produce a gradient vector flow (GVF), or a perpendicular variant of ETF, Xu and Prince [2] treated this problem as a partial differential equation and presented an iterative vector diffusion algorithm. As an external force field, they draw active outlines with GVF. This method can also provide an ETF, but the resulting whirling flows are not desired and the conspicuous edge tangent directions are not well preserved, making it less suitable for line-drawing applications. This is explained by the fact that the diffusion process only considers the vectors' magnitudes, not their directions. By introducing the direction weight function into the formula, this technique effectively eliminates these artifacts. The vector diffusion method's inability to regulate the diffusion kernel's size (only the immediate neighbors are taken into account) is its second drawback. This approach provides a size-controllable kernel to solve this issue.

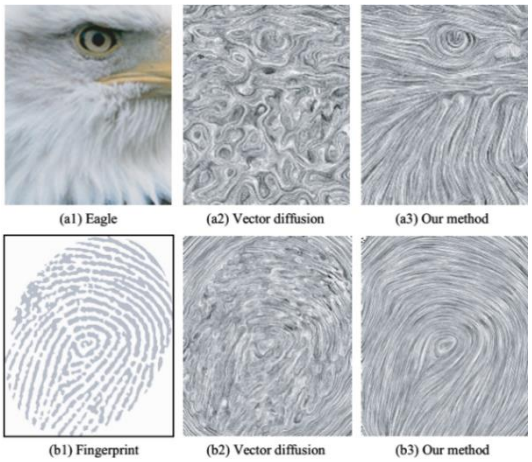


Fig 2.2.

D. Analysis of the effect of ETF on edge detection

The value of μ is affected by image resolution. For high-resolution images require larger value of μ is required. Fig (2.3) and Fig (2.4) shows the effect of the ETF filter in edge detection of a 512x512 Lenna image.

It is observed that for the same value of threshold if μ is increased from 3 to 5 the coherence of edges is getting increased, whereas if μ is further increased from 5 to 7 the coherence of edges starts decreasing. This happens because in μ = 3 filter the neighbors supporting the edge flow were not included entirely, in the μ=7 filter the excessive neighbours that were not supporting the edge flow were also included. Hence choosing μ is a critical factor in getting the desired output.

By keeping other factors same, if the iterations of ETF filters are increased then the dominant edges become more dominant. This is because the goal of the filter is to highlight the salient edge direction, due to the w_d weight function with each iteration more tangents will align in the direction of their nearby dominant edges.

Figure 2.3. Figure shows images generated with different values of kernel radii and different number of iterations of the ETF filter (all these images have same threshold value of 0.9)

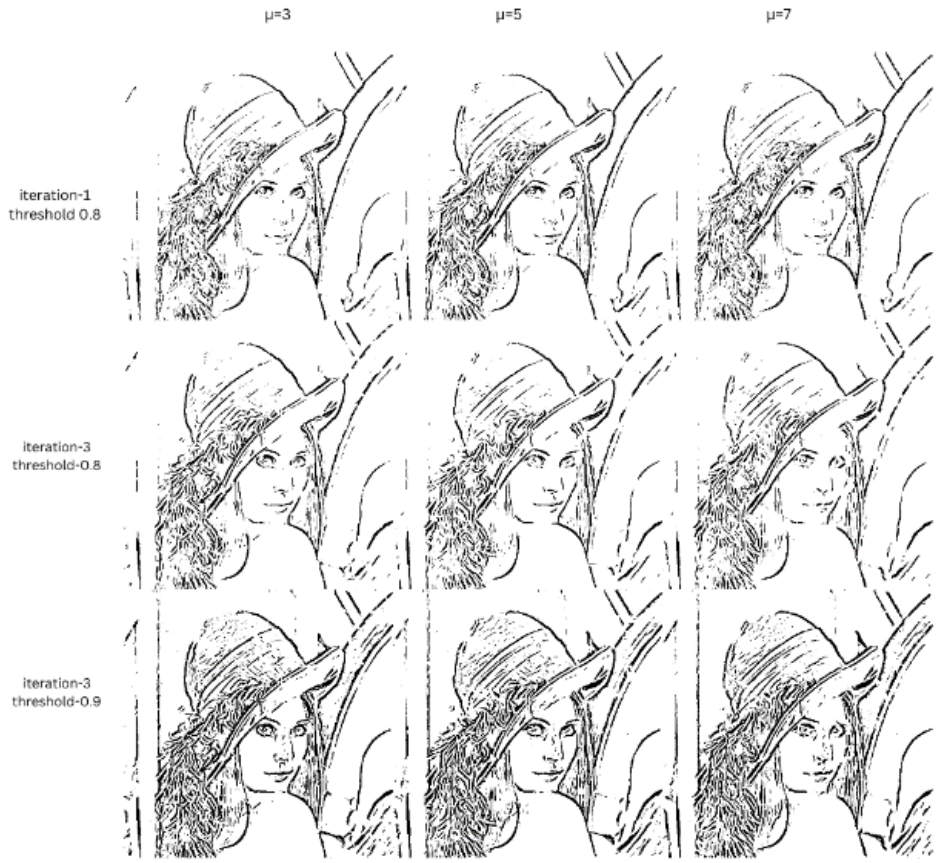


Figure 2.4. Figure shows images generated with different values of kernel radii and different number of iterations of the ETF filter (all these images have same threshold value of 0.9)

III. LINE EXTRACTION

Conventional edge detectors such as Canny Edge, Difference of Gaussian Filtering, etc. are used in line extraction traditionally. The DoG edge detection model published by Winnemoller et al. is the basis for the proposed FDoG Filter [3] due to its appropriate stylistic fit and simplicity. The main objective is to guide the DoG filter along the Edge Tangent Flow curve in order to improve line quality.

The fundamental idea is to apply a linear DoG filter in this gradient direction while maintaining the edge flow. Subsequently, the responses obtained from every filter are combined, and the edge is correctly generated. This improves the spatial coherence of the edges, minimises noise, and highlights the genuine edges.

A. Flow-Based Difference of Gaussians

One blurry version of a picture is subtracted from another, less blurry version of the same image using the difference Gaussian filter. The original image is convolutioned using Gaussian filters whose kernel weights are set by two distinct Gaussian distributions (i.e., separate standard deviations) to produce blurred images. Image blurring with a Gaussian blur only mutes high-frequency spatial information. The spatial information between the range of frequencies preserved in the two blurry photos is preserved when one image is subtracted from the other. Gaussian difference functions as a bandpass filter, meaning it only allows frequencies within a specific range to pass through. The result is more elegant with varying edge widths than Canny Edge. Collectively, the edge orientations determined by ETF at every position.

A method known as the flow-based difference-of-Gaussians (FDoG) involves computing a DoG response across edges by applying one-dimensional Gaussian blur along lines perpendicular to the edge tangents. This approach also incorporates edge-aligned smoothing via a line integral convolution pass that tracks the flow of edge tangents. Consequently, FDoG utilizes a two-pass flow-guided technique to achieve its results.

To convert the standard DoG into an FDoG, three distinct parameters must be used in place of the parameter σ:

- σc: Indicates how much blurring has been applied to the structural tensor. Higher values may distort finer details, while lower values may boost edge noise.

- σs: Controls the Gaussians filter's gradient-aligned difference's spread. Wider edge lines and finer details may be lost when σs increases.

• σm: Regulates the line integral convolution's breadth in relation to the edge tangents. Greater values combine shorter segments into longer segments, improving the coherence of edge lines. Excessive σm in comparison to σc, however, could cause noise to enter the edges.

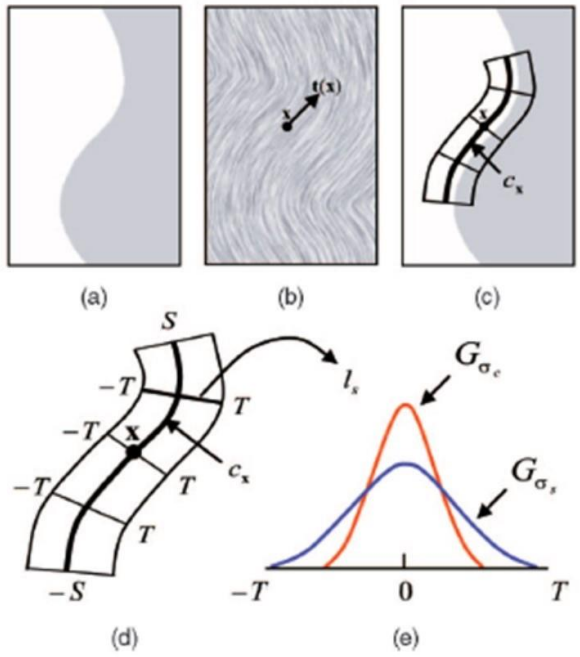


Figure 3.1. FDoG filtering. (a) Input. (b) ETF. (c) Kernel at x. (d) Enlarged kernal. (e) Gaussian components for DoG

Notations:

1) $C_x(s)$: flow curve at x, where s is an arc-length parameter that may take on positive or negative values. It is assumed x serves as the curve centre, that means $c_x(0) = x$.
2) $l_{x,\,s}$: A Line segment that is perpendicular to $t(c_x(s))$ and intersecting $c_x(s)$.
3) $l_{x,s}(t)$: A point on the line $l_{x,\,s}$ at t distance from the curve.

The notation t(x) in ETF represents the local edge direction. According to the notations and assumptions it can be interpreted that, $c_x(0) = x$, $l_{x,s}(0) = c_x$ and, $l_{x,\,s}$ is parallel to the gradient vector $g(c_x(s))$.

The filtering scheme is formulated as:

$$H(x) = \int_{-S}^{S} \int_{-T}^{T} I\left(l_{x,s}(t)\right) f(t) G_{\sigma m}(s) dt ds$$

Where, $I(l_{x,s}(t))$ represents the value of the input image I at $l_{x,s}(t)$.
A one-dimensional filter f is applied to the gradient line l_x, s as we move along c_x. The individual filter responses are then accumulated along c_x using a weight function of s, represented as $G_t(s)$, where G is a univariate Gaussian Function of variance t^2. This is how the filter operates in its basic form.

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

The underlying filter f is the edge detection model based on Difference of Gaussians

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

Where the two parameters control the sizes of the intervals and we set $\sigma_s = 1.6\sigma_c$ for better edge detection. [5]

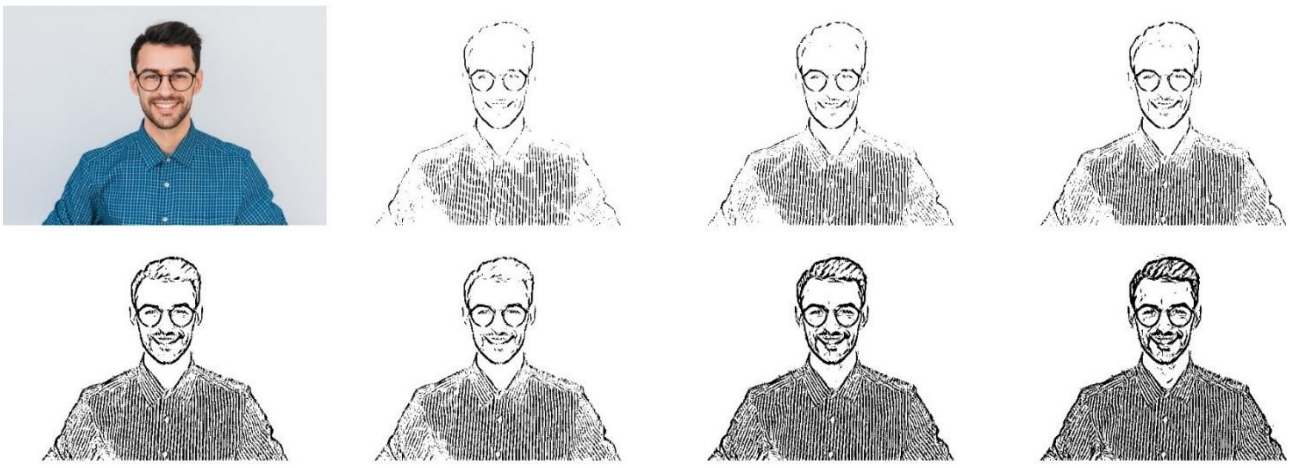


Figure 3.2. Iterative Thresholding using 0.4, 0.5 0.6, 0.7, 0.8, 0.95 values as thresholds.

Figure 3.3. Comparison with other techniques. (a) Input. (b) Canny. (c) Mean Shift. (d) Isotropic DoG. (e) FDoG.

After obtaining H(x), we convert it to a black and white image by binary thresholding as:

$$\overline{H}(x) = \begin{cases} 0, & if\ H(x) < 0\ ,1 + \tanh\big(H(x)\big) < r \\ 1, & otherwise \end{cases}$$

This binary output serves as the targeted line illustration.

B. Alternative Methods to Difference of Gaussians

1) Laplacian of Gaussians

The Laplacian of Gaussians for a pixel (x, y) is given as:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

The filtering scheme remains the same only the function f(t) changes to LoG (x, y).

The Gaussian kernel convolution of the picture is followed by the Laplacian operator in the LoG filter. This procedure requires two distinct convolution procedures, which makes it computationally costly, particularly for large images. The DoG filter uses merely two convolutions with Gaussian kernels and a subtraction operation, making it computationally more efficient than the LoG filter.

One of LoG filter's best-known features is its precise edge localization. Applications needing accurate edge detection can use it because of its sub-pixel accuracy in edge detection. Although the DoG filter can identify edges as well, it might not be as accurate at localising edges as the LoG filter. It is still capable of efficiently detecting edges with good localization, though. In addition, this filter is less sensitive to parameter selection than the DoG filter. The behaviour of the LoG filter is reasonably stable if the Gaussian kernel's standard deviation is known. The DoG filter's performance can be affected by the selection of parameters, specifically the Gaussian kernels' standard deviations.

2) Extended Difference of Gaussians

The Difference of Gaussians Filter is given as:

$$XDoG(x) = G_\sigma(x) - pG_\sigma(x)$$

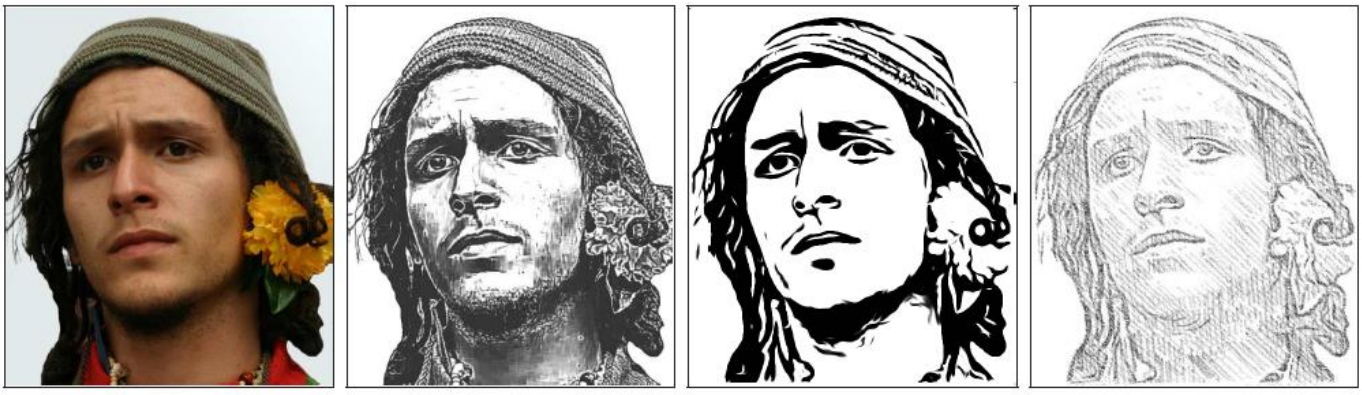
But the only way to make the edge accent lines heavier is to increase p [6]. Therefore, any variation to p must be implemented as follows in order to provide XDoG outputs with varying edge accent strengths but the same average brightness:

$$XDoG(x) = (1+p)G_\sigma(x) - pG_\sigma(x)$$

The new formulation (XDoG) and flow-alignment (FDoG) are mutually independent extensions to the DoG operator, and may therefore be combined, as desired.

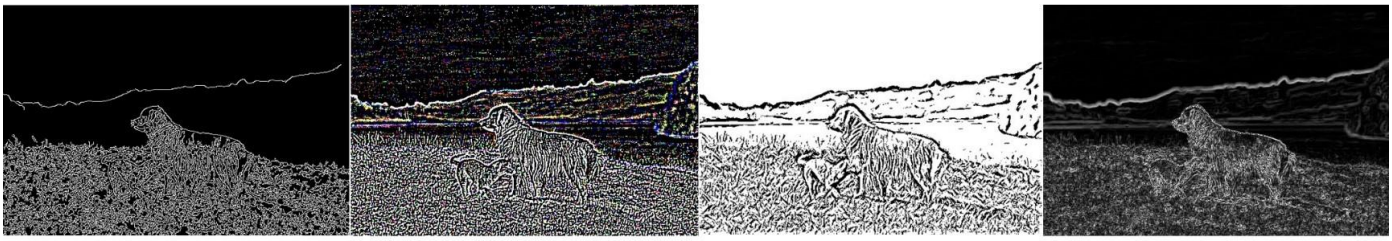
C. Comparison of FDoG and DoG filters

The DoG filter is called the FDoG Filter since it is driven by the vector flow. When compared to the DoG filter, the line coherence is better with FDoG filtering. More lines are included in the picture as the ρ value is increased. In contrast to traditional DoG filters, the FDoG filter uses a large ETF kernel size to extract ETF from a Gaussian-smoothed input



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

Figure 3.4. (a) Source. (b) XDoG. (c) XDoG Thresholding. (d) XDoG-Hatching



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

Figure 3.5. (a) Canny Edge. (b) DoG. (c) FDoG. (d) Sobel Filter.

image, allowing lines to be constructed from a collection of disconnected points. The FDoG Filter has limits when it comes to handling small-scale non-directional structures, but it is excellent for safeguarding directed structures.

Since the FDoG filter depends on ETF, it is crucial to create an ETF that accurately depicts the shape. This can be challenging for texture patterns or small-scale features, though.

Using the smooth ETF vectors surrounding the target shape, the FDoG filter is also resistant to noise. In comparison to the FDoG filter, the DoG filter conveys the "tone" information more effectively. Second, noise pixels that were caught in earlier rounds may be sent down to and stabilised by the FDoG filter programme that comes after. These line segments are boosted and stabilised; whether they constitute noise or not is unknown.

IV. REGION SMOOTHING

Region smoothing is a process that is used in image processing to smooth or blur specific regions or segments of an image while preserving the overall structure and edges. The objective of this technique is to remove unimportant details from the region's interiors while maintaining important information.

There are multiple purposes of region smoothing technique, some of them are –

1. Enhancing Features
2. Noise Reduction
3. Edge Preservation
4. Selective Smoothing
5. Image Segmentation

Both linear and nonlinear filters are used in image processing for region smoothing.

Linear Filters:

1. Gaussian Filter
2. Mean Filter

Nonlinear Filters:

1. Bilateral Filter
2. Median Filter

Among the filters mentioned above for region smoothing in image processing, the bilateral filter[5] is often considered one of the most prominent and widely used techniques. Its popularity comes from its effectiveness in smoothing images while preserving edges and fine details.

A. Bilateral Filter

The bilateral filter is a nonlinear filtering technique used in image processing for smoothing images while preserving edges. It is designed to reduce noise and blur images while retaining important structural information, such as edges and boundaries. The bilateral filter achieves this by considering both the spatial proximity and the intensity similarity of neighboring pixels when computing the weighted average for each pixel.

Working of Bilateral Filter-

- Spatial Proximity Weighting: The filter considers the spatial distance between the center pixel and its neighboring pixels. Pixels that are closer to the center pixel are given higher weights in the averaging process, while pixels farther away receive lower weights. This spatial proximity weighting ensures that the filter focuses more on nearby pixels during the smoothing process.
- Intensity Similarity Weighting: In addition to spatial distance, the filter also considers the similarity in intensity or color between the center pixel and its neighbors. Pixels with similar intensity values to the center pixel are given higher weights, while pixels with significantly different intensity values are given lower weights. This intensity similarity weighting helps preserve edges and fine details in the image.

Limitations of Bilateral Filter-

- Ignored Direction of Color Contrast: The bilateral filter does not consider the direction in which color contrast is formed when smoothing minor color differences in the image. This lack of direction awareness can result in the removal of subtle but meaningful shape boundaries that are aligned with the direction of color contrast. As a consequence, important features may be smoothed out instead of being preserved.
- Loss of Feature Directionality: Due to the circular nature of the spatial kernel, the bilateral filter may fail to preserve feature directionality in the image. Features such as edges, textures, or patterns that have a specific orientation or direction may not be adequately protected during the smoothing process. This can result in a loss of visual coherence and fidelity, particularly in images with prominent directional features.

- Rough Surviving Edges: The lack of direction awareness in the bilateral filter may lead to surviving edges appearing rough or jagged after the smoothing process. Instead of preserving smooth and coherent edges, the filter may introduce irregularities or artifacts along edge boundaries, which can detract from the overall stylization and aesthetic quality of the image.

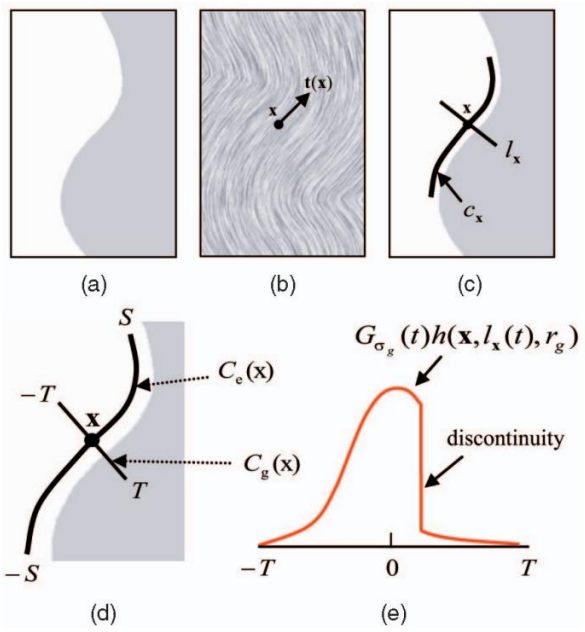


Fig. 4.1. FBL filtering. (a) Input. (b) ETF. (c) Two linear kernels at x. (d) Kernels enlarged. (e) Bilateral weight function for C_g.

B. Flow Based Bilateral Filter (FBL)

C_e (Edge Direction Filter)

$$C_e(x) = \frac{1}{v_e}\int_{-S}^{S} I\big(c_x(s)\big)G_{\sigma_e}(s)h(x, c_x(s), r_e)ds$$

Notations:
$C_e(x)$: Output of the linear bilateral filter at pixel x
v_e: Weight normalization term.
S: Domain over which the integral is calculated.
$c_x(s)$: Flow curve of the Edge Tangent Flow (ETF).
$I\big(c_x(s)\big)$: Input image or signal.
$G_{\sigma_e}(s)$: Spatial weight function along the flow axis $c_x(s)$.
$h(x, c_x(s), r_e)$: Color weight function.
r_e: Parameter related to color similarity.
ds: Infinitesimal element of integration over the spatial domain S.
$x(s)$: Point in the spatial domain within the filter kernel at location s.

$$h(x, y, \sigma) = G_\sigma(||I(x) - I(y)||)$$

- C_e operates primarily along the edge directions, focusing on preserving and enhancing shape boundaries in the image.
- It protects and cleans up the shape boundaries by emphasizing edges and reducing noise along edge directions.
- The spatial weight function $G_{\sigma_e}(s)$ along the edge direction helps in preserving edge information, while the color weight function $h(x, c_x(s), r_e)$ ensures that edges are maintained by considering color similarity.

C_g Gradient Direction Filter

$$C_g(x) = \frac{1}{v_g}\int_{-T}^{T} I\big(l_x(t)\big)G_{\sigma_g}(t)h\big(x, l_x(t), r_g\big)dt$$

Notations:

The application of this filter can extend to color images by employing a distance metric within the color space. Likewise, the subsequent equation outlines another linear bilateral filter aligned with the gradient direction.
where $l_x(t)$ is again an abbreviated notation for $l_{x,0}(t)$, which is the gradient axis at x.

- It operates along the gradient direction, reducing color variations within homogeneous regions and promoting smoother color transitions.
- The spatial weight function C_g along the gradient direction helps in smoothing out color transitions, while the color weight function $h\big(x, l_x(t), r_g\big)$ suppresses color differences within regions.

FBL Filter – Combination of C_e and C_g

The FBL (Flow Bilateral) filter is a combination of C_e and C_g applied iteratively.
By alternately applying C_e and C_g in an iterative fashion, the FBL filter effectively balances the preservation of edges and smoothing of regions.
The iterative application of C_e and C_g allows for efficient edge preservation and region smoothing, making it faster than the full-kernel bilateral filter.
The FBL filter is particularly useful in scenarios where both edge preservation and region smoothing are desired, such as in image abstraction, stylization, or denoising tasks.
In summary, C_e and C_g play distinct roles in edge preservation and region smoothing, respectively. Their combination as the FBL filter allows for efficient and effective image filtering, balancing the preservation of shape boundaries and the smoothing of region interiors.

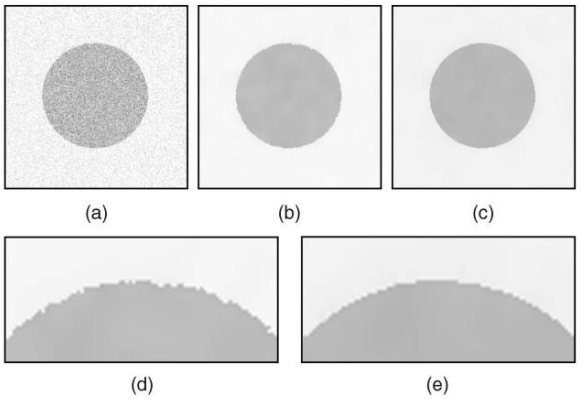


Fig. 4.2. FBL: Boundary restoration. (a) Input. (b) Bilateral filter. (c) FBL filter. (d) Bilateral filter (enlarged). (e) FBL filter (enlarged).

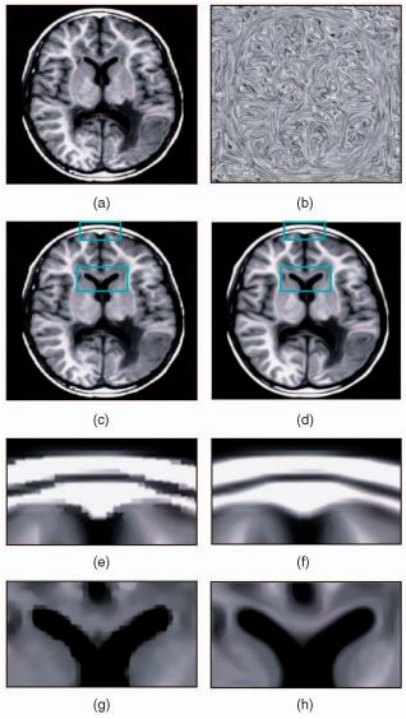


Fig. 4.3. FBL: Shape enhancement. (a) Input image. (b) ETF. (c) Bilateral filter (σ_d = 2.0, σ_r =10, three iterations). (d) FBL (σ_e =2.0, r_e = 50, σ_g =0.3, r_g = 10, three iterations). (e) Bilateral filter (enlarged). (f) FBL (enlarged). (g) Bilateral filter (enlarged). (h) FBL (enlarged).

V. RESULT

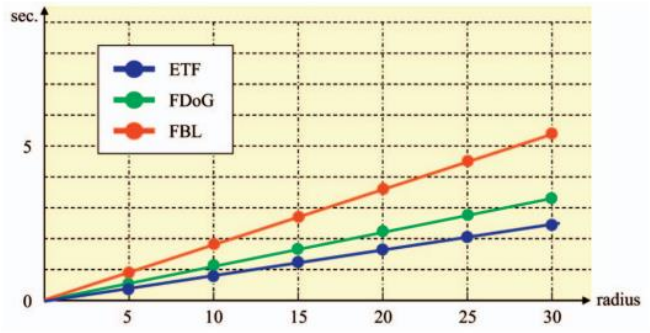


Fig. 5.1. Execution time for filters

The feature lines are effectively depicted to convey shapes, while simplified colors efficiently represent tonal information without being overly distracting.

Our accelerated Edge Tangent Flow (ETF) construction filter operates with a complexity of O(n x μ), where n represents the number of pixels and μ signifies the kernel radius. The FDoG filter and the FBL filter have complexities of O(nxα + nxβ). In Figure 5.1, timing data for these filters is presented on a 512 X 512 RGB image, with varying kernel radius values. For the ETF filter, "kernel radius" corresponds to μ, while for the FDoG and FBL filters, it denotes both α and β, with α=β for this experiment. All three filters demonstrate linear time complexity regarding the kernel radius.

VI. CONCLUSION

We presented an automated technique for image abstraction using a flow-based filtering framework. Specifically, we introduced the FDoG and FBL filters as novel solutions to address two critical aspects of image abstraction: line drawing and region smoothing. By harnessing the Edge Tangent Flow (ETF) for feature preservation, these filters outperform conventional methods by improving features and stylization, resulting in high-quality image abstractions. This method effectively conveys shapes and colors, ensuring a superior level of abstraction quality.

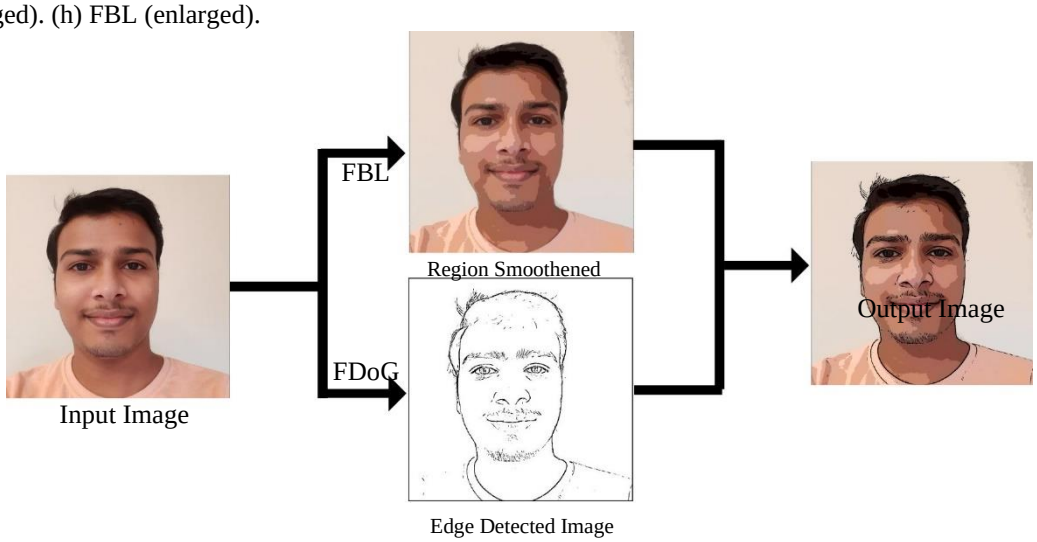


Fig 5.2

[1] H. Winnemo ¨ller, S. Olsen, and B. Gooch, “Real-Time Video Abstraction,” Proc. ACM SIGGRAPH ’06, pp. 1221-1226, 2006
[2] XU, C., AND PRINCE, J. L. 1998. Snakes, shapes, and gradient vector flow. IEEE Transactions on Image Processing 7, 3, 359 369.
[3] D. DeCarlo and A. Santella, “Stylization and Abstraction of Photographs,” Proc. ACM SIGGRAPH ’02, pp. 769-776, 2002.
[4] “XDoG: An eXtended difference-of-Gaussians compendium including advanced image stylization”, Holger Winnemoller a,Jan Eric Kyprianidisb,Sven C. Olsen
[5] 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.

Plagiarism Scan Report

Report Generated on: Apr 08,2024

Plagiarised

Unique

Total Words:	976
Total Characters:	6382
Plagiarized Sentences:	4.9
Unique Sentences:	44.1 (90%)

Plagiarism Scan Report

Report Generated on: Apr 08,2024

Plagiarised

Unique

Total Words:	1000
Total Characters:	6709
Plagiarized Sentences:	0
Unique Sentences:	48 (100%)

Plagiarism Scan Report

Report Generated on: Apr 08,2024

Plagiarised

Unique

Total Words:	933
Total Characters:	5877
Plagiarized Sentences:	0.98
Unique Sentences:	48.02 (98%)