Flow-Centric Painterly Rendering

Omar Hesham
Carleton University
Ottawa, ON, Canada
ohesham@connect.carleton.ca

David Mould
Carleton University
Ottawa, ON, Canada
mould@scs.carleton.ca

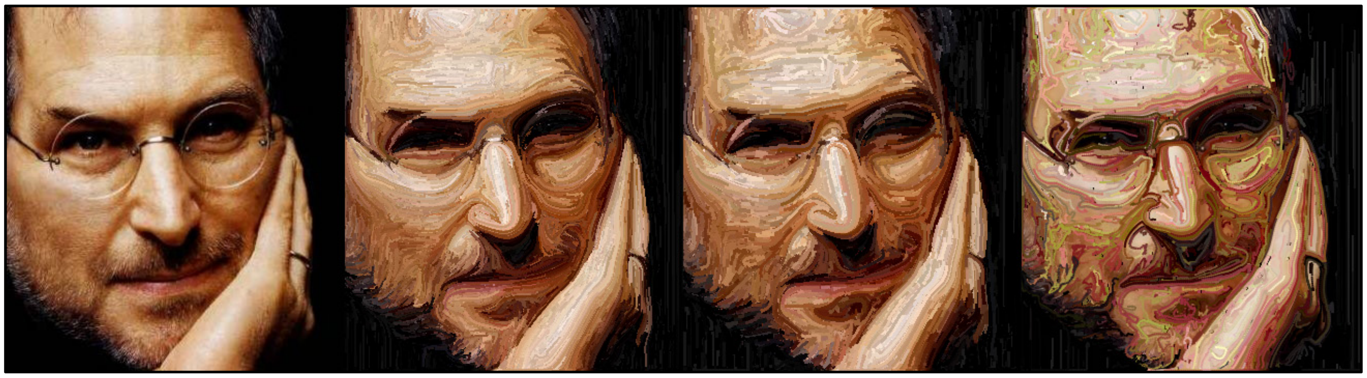


Figure 1. *Our method applied to (a) an input image, with (b) & (c) different brush sizes, and (d) using a harmonic color theme.*

Abstract

We present a painterly rendering technique that emphasizes smooth line flows to describe important shapes in the image, as well as contributing in an essential part to the final stylistic artwork. Our method uses the smoothed edge tangent flow (ETF) [20] for stroke pathing, and provides an artist-friendly way to control the general look and feel of the image using familiar brush stroke properties and colour theme palettes. We achieve a highly stylized expressionistic look that can be applied to still photographs and portraits.

Key words: non-photorealistic rendering, edge flow filtering, painterly rendering.

1. Introduction

Developing painterly rendering techniques that imitate or expand on traditional art styles, has been at the heart of non-photorealistic rendering (NPR) and a driving factor in its popularity. A common source of inspiration comes from impressionistic art, for its pleasant visual dynamics and abstraction qualities that make it suitable for NPR research.

Traditionally, edge flows have been used to guide those painterly algorithms. Many consider flow-like structures to be pleasant, harmonic, or at least interesting [11]. Those lines, however, tend to remain behind-the-scenes for the currently available methods, and their appearance is usually subtle, if not unintended, at best. This is again motivated by the *impressionistic* art style which doesn't announce those lines very often. This motivated us to develop an automated algorithm for stylistic paintings where smooth contour lines are a prominent feature in the artwork. Thus, those edge flows become as much part of the art, as they are computational guides for the brush strokes. Our goal in this paper is to produce a wide variety of painterly renderings, similar in spirit to *expressionistic* art, like van Gogh's work, but not limited to his style. The user should be able to comfortably choose any variation between van Gogh's short and visually arresting strokes, and Edvard Munch's soothing yet tempestuous long sweeps. The variation extends to the way color varies across neighboring strokes, be it subtle or announced.

One of the challenges that come with our goal of artistic flexibility is stroke length management. A painting's canvas is a limited 2-dimensional space, and giving free range to the length of the strokes to be long and prominent, should also be balanced with a control method, so that the strokes don't end up in unwelcomed regions. Quite often, these control methods result in shortening of those strokes, effectively canceling the flexibility we were aiming for.

Another challenge is determining a user-friendly way to manage the colour variation in the painting, without necessarily forcing a procedural rule. This invited a study of how expressionistic artist used colour, just to realize that there is in fact no common theme, except uncommonness.

This allows us to break the painting challenge into three specific parts: i) Brush stroke path, ii) Brush properties, and iii) Brush coloring. Our method relies on finding a smooth edge flow, and allowing the brush stroke to flow freely along its path, then relying on harmonic colour themes for the final render.

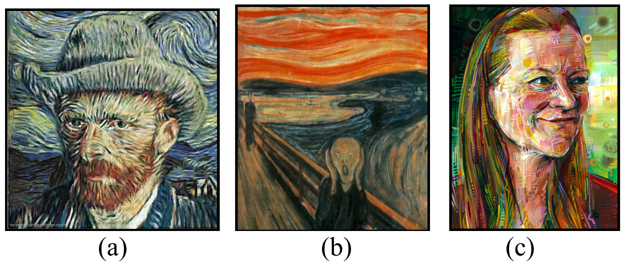


Figure 2. *Artwork by (a) Vincent van Gogh, (b) Edvard Munch and (c) Gwenn Seemel, illustrating the stylistic impact of flow lines; which inspired our work.*

The rest of this paper is organized as follows: In Section 2, we review the existing painterly rendering methods and briefly compare their results to our goals. In Section 3, we detail our method for flow-based painting, and discuss the results in Section 4. In conclusion, Section 5 reflects on the development of this project and discusses our future goals and research direction.

2. Related Work

There are several categories of techniques in the NPR literature to achieve a stylistic painterly effect. There is the stroke-based rendering (SBR), which started with Haeberli [28], and has gone through improvements ever since, the most prominent of which are the introduction of image gradient to control stroke lengths and using optical flow to achieve inter-frame coherence, which were presented in [1] and later improved by [4]. In addition, Hertzmann [25] introduced a layered approach to simulate how an artist would paint a broad sense of the picture first, and then add in the details at a finer level. The reader is referred to Shwartz [8] for a historic account of the development of SBR methods. In general, SBR techniques have focused on improving the realistic look of the paint strokes, and faithfully reproducing the input image's hues and tone.

Olsen[3], also inspired by the works of van Gogh, proposed a fluid-dynamics system which allows the user to place vortices around the image. It then traces particles as they get advected in their container. The results were pleasant with long turbulent strokes. However, color-space segmentation was required in order to prevent the fluids from different colored regions from mixing.

Recent work in texture transfer in [2] has also been successful in producing an expressionistic look, but their technique was more suitable for hatching and pen illustrations, than their image vector flow results, as they looked as if the flow visualization was simply pasted on top of the original image, and was not very painterly.

Particle based diffusion methods have been central to vector field visualization, using methods like line-integral convolution (LIC) [18][19], which have found their way into image processing in hopes of achieving a smooth painterly look [11][10]. Image pixels, as particles, are diffused along a smoothed tensor field representing the image edge flow. The results are visually appealing and smooth, but they look more like blurred images and washed regions than a painting made by an artist's strokes.

The method we propose, is an SBR renderer that is capable of turning images into highly stylized expressionistic paintings, and doesn't restrict the stroke style or color to the original input image. It is also compatible with the aforementioned SBR techniques, allowing it to mimic their impressionistic painting results if desired.

3. Method

Our method first generates a smooth vector field from the input image, traces the brush stroke paths along those vectors according to user-selected painting variables (brush size, stroke length, opacity, etc.); and finally consults the colour theme palette before painting each stroke's final output. The major steps are detailed in this section and a sketch of the method is seen in Figure [4].

3.1 General Image Flow

To determine lines suitable for the brush strokes, we intuitively need a vector field that considers the silhouettes of objects and takes into account image topology. Previous SBR methods have relied on the vectors orthogonal to the image gradient field, which were later smoothed using bilinear filtering [1][4][3]. While they provided satisfactory stroke guidance, the fields suffered from poor temporal coherence, as the authors point out, and required the development of elaborate and memory-consuming techniques to achieve temporal coherence and smooth videos. We elected to go with the edge tangent flow (ETF) smoothing method proposed by Kang[20] which uses bilateral filtering on the image gradient vector field to produce a noise-free and smooth field that preserves sharp edges and can be extended to 3D filtering [9], producing temporally coherent flow fields. We don't attempt to tackle video in this paper; however, it is always good to have a solid base for future development.

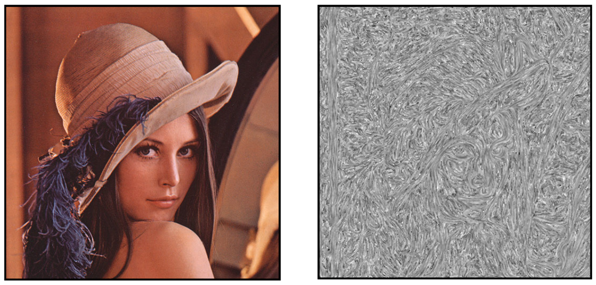


Figure 3. *Edge Tangent Flow (ETF)*

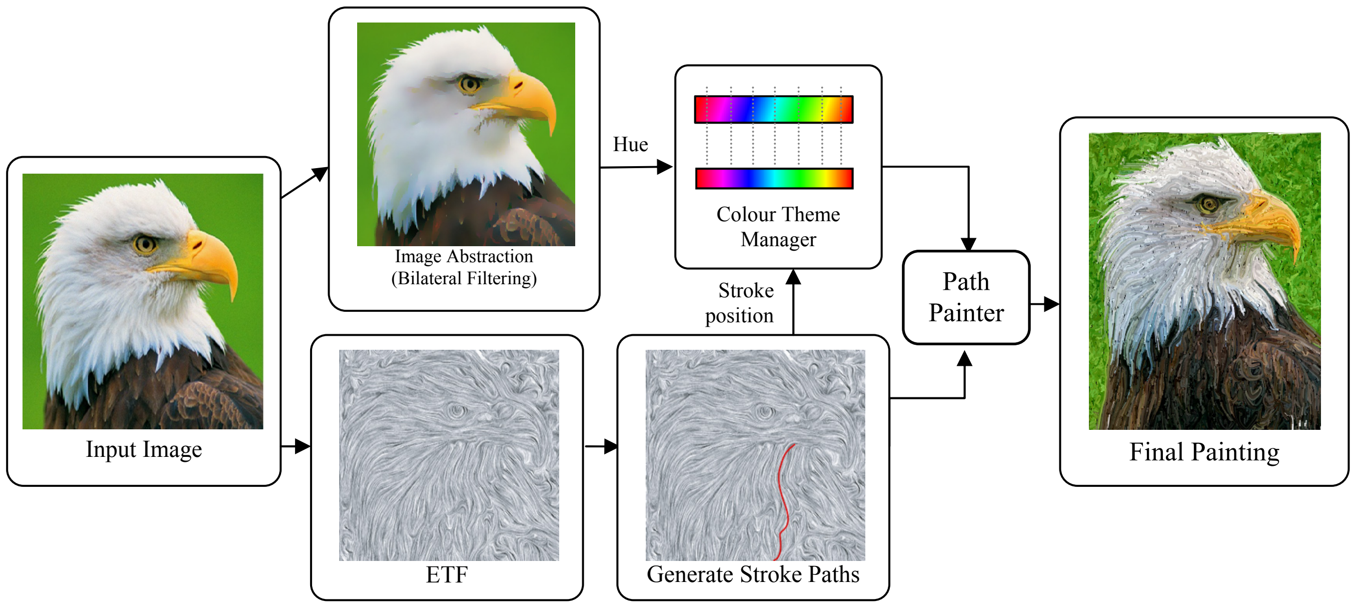


Figure 4. *Method overview.*

3.2 Brush Strokes

The first step in this section involves brush stroke placement. It would be ideal to come up with an energy minimization scheme to assess the best stroke placement. This is not easy, however, due to the complex and seemingly random nature of the expressionistic style we're going for. The next step involves tracing a brush stroke path along the vector field lines. Such sophisticated methods to approach these problems can be seen in Hertzmann[25] using cubic splines, and Salisbury [14] using differential equations to find the smooth curve. LIC methods have seen recent popularity in tracing strokes in images[10][11] and video[6][9], but they are computationally expensive, essentially image processing methods that diffuse the image along those lines rather than painting strokes. This makes it challenging to extend them to textured strokes, for example.

Any of the above methods would work with our system, albeit for tracing only not painting. We empirically found, however, that they are not needed to achieve a great painterly result. Our current implementation is a major simplification of the previous techniques, yet the results are more than satisfactory. For stroke placement, we distribute the stroke points randomly, and to trace a path, we simply travel in the direction of the closest vector by an amount equal to the radius of the brush. This seems to work primarily because of the smoothness of the ETF field, especially at the local level, where usually a brush would want to sample a local area of vectors to decide where to go next, but the smoothed ETF, in a sense, already made that decision and stored it in the one pixel underneath it.

Every stroke stores the following properties:

- *Stroke Path* – a set of control points describing the stroke path.
- *Stroke Radius* – defines the brush footprint size, and the stroke path tracing increment.
- *Stroke Length* – at least as long as the stroke radius. The length is usually set in the 100s.
- *Stroke Colour* – see Section 3.3

If a user wanted to control the amount of stroke "spill-over" between regions, we implemented a version of Litwinocz's[1] control method, where the brush stroke path would be interrupted if it is perceived to have crossed a boundary. We do this by making sure that the Euclidean distance in the perceptually uniform CIELab color space, between the center of the stroke path so far and the current candidate path point, to be less than a certain amount, controlled by the user. If that's not the case, the pathing is interrupted and stored as it is at that stage. In our implementation, we call this global parameter the *edge protection* value.

3.3 Color Themes

The final element of our method deals with delivering a vibrantly colored painting, with a naturally random look, and coherent hues. Before the brush gets painted in the previous step, it sends out the 2D image location of the center point along its current path, to our color theme manager. The manager then samples the colour from the bilaterally filtered image and then processes this input color according to user-chosen rules and returns the new color which is then used to render the stroke. Note that this is not a post-process, but an active element of every brush stroke's rendering.

By default, we use the bilaterally filtered image instead of the source image as input to the colour manager, because it eliminates input noise which can produce unwanted stroke variation. For example, bumps on the skin usually appear with shadow noise and variations that could, at the pixel level, result in dramatic stroke color shifts, which is undesirable if the user hasn't asked for it. Such variations in our paintings should originate from the user specified color theme settings. We prefer the bilateral filter over the Guassian blur, for its ability to smooth out those details while maintaining important edges. The user can specify not to use the bilateral filter at all, if that's to their liking.

We introduce the notion of a "colour theme" for fast artistic manipulation, which allows the user to easily change the general look and feel of the output painting. In the current implementation, a theme is stored as a cyclic image file whose height represents the input color's hue in the HSV space, and the width contains all the possible hues for the theme manager to randomly choose from. In addition, we perform a luminance perturbation step, which introduces the randomized variance amongst strokes, usually seen in expressionistic paintings. Instead of uniformly randomly varying the luminance however, we use the inverse logistic cumulative distribution function, the logit [Figure 5]. It is, in our experience, better than the probit function (inverse Gaussian), as it has heavier tails, allowing it to maintains focus on the mean, while still affording a healthy amount of extreme values to appear. The probit is also difficult to compute and is usually approximated, while the logit function is efficiently computed as:

$$\text{logit}(\mathbf{q}) = \log(\mathbf{q}) - \log(1 - \mathbf{q})$$

, where $\mathbf{q}$ is a uniformly random number $\in [0,1]$. The user specifies the amount of luminance variance and we use that to obtain the final luminance. An example computation is:

$$p = \text{random}[0,1]$$
$$\sigma = \sqrt{v}$$
$$\mu' = \mu + \sigma\,\text{logit}(p)$$

where p is a uniformly random number, v is the user-input value for variance, usually $\in [8, 30]$, μ is the input CIELab luminance value, and μ' is the outcome luminance, which is now a random value with the desired logistic probability distribution of variance v and mean μ.

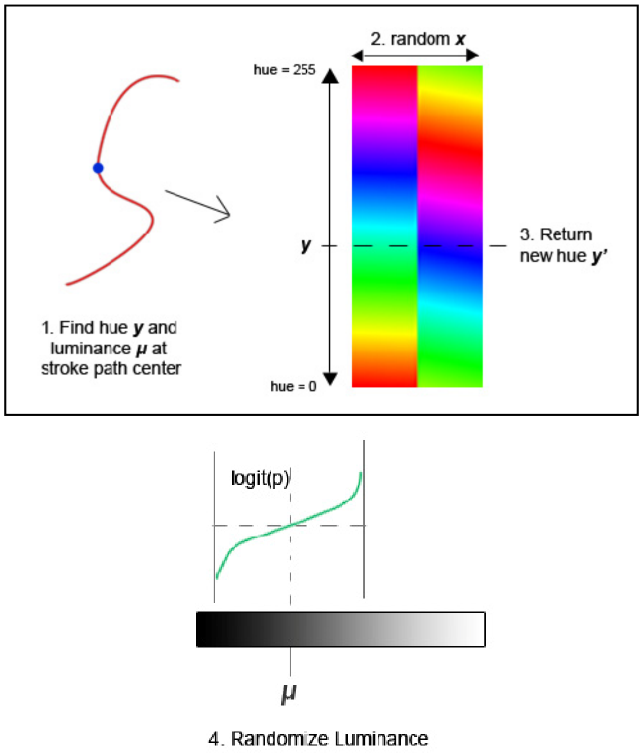


Figure 5. *Color theme and stroke hue/luminance management.*

In order to achieve a pleasant result, we consult the aid of colour harmonies [29] to define a few color themes around the hue spectrum of the HSV space, dictating how each color should be processed. Our implementation comes with a handful of predefined color harmonies, including Type i, Type V, Type L and hue opponency.

We also allow adventurous users to extend the colour theme sampling process to the entire HSV colour definition, instead of just the hue. This means that an artist could specify exactly the set of colours to be used for a given input hue, no matter what that input hue's brightness and saturation were. Examples of this creative flexibility can be seen in [Figure 6].

3.4 Stroke Rendering

For the bottom most layer, the user can choose between black, white, and the bilaterally filtered image as a starting canvas. The color theme manager returns a single RGB colour to the brush stroke that invoked it, and that colour is used to paint the entire stroke, by drawing straight lines between every point along a stroke's path. We do not tackle bump mapping and other canvas enhancement methods as we view them as post-processing elements that are out of the scope this project. We decided to keep it constrained to single colored solid strokes to showcase that sophisticated textures are not always necessary, and to bring attention to the underlying methodology, instead of nifty after-effects.

4. Discussion

The gallery [Figures 10-13] showcases a range of painterly renderings obtained through our system. The performance is reasonably fast, with the main bottleneck being the image and ETF bilateral filtering steps, taking up to 2 minutes with very high parameter values and number of iterations. Once past the filering stage, our painter is very fast, accommodating anything between 500 and 100000 strokes, while only taking around 5 seconds to paint at extreme values. This is partly attributed to the simplistic tracing method we used (just straight lines), and the method would probably become slower if something more expensive like stroke splines were calculated. A GPU implementation, however, would eliminate these problems as the painter is trivially parallelizable by nature.

At the moment, all parameters are defined globally, making it difficult to correct problem areas like the eyes, without requiring some user input, like weight maps or spatial interaction. Increasing the edge protection parameter fixes the issue with the eyes, but at the cost of what might have been a pleasant interplay of strokes in the background. In addition, setting a very high edge protection value might prevent a large portion of the image from being painted at all, because the strokes keep getting interrupted early on. Although not always necessary, using the bilateral image as the startup canvas can afford the use of high edge protection values [Figure 7] without losing the painterly look. These observations call for examining better localization of parameters using mostly automated and optionally user-interactive methods.

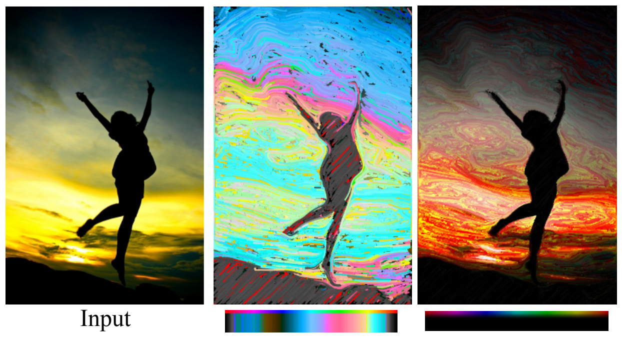


Figure 6. *All HSV channels being processed by the theme. Paintings shown with the color theme that produced them.*

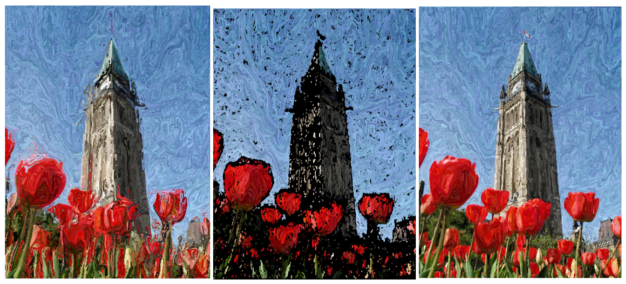


Figure 7. *From left to right: Edge Protection = 8; Edge Protection = 95 with black canvas; Edge Protection = 95 with bilateral image as canvas.*

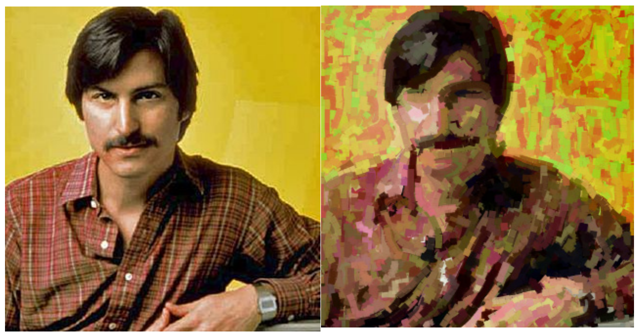


Figure 8. *Our painter is not limited to thin long strokes. We can use short broad strokes and still achieve the expressionistic look we're after, and our stylistic edge flow prominence remains intact*

5. Conclusion

Through our implementation, we have met our goal of producing an automated algorithm that leverages Kang's ETF to achieve a flow-centric painterly effect, with user input to control artistic preferences like brush properties and color theme, and we feel enthusiastic moving forward.

In this section, I go over a few of the possible directions this project can benefit from:

Tensor Fields – On the whole, the ETF vector field has exceeded my expectations, especially given my crude stroke tracing method. However, tensor fields can be very suitable for localization work, by sampling various readily accessible properties about the field like curvature and eigen values [16][17]. An investigation into how these properties can be utilized in our algorithm is needed, but similar previous applications like the highly coherent video abstraction in Kang[23] and the beautifully illustrated pattern wrapping on 3D surfaces by Li[7], look very promising already. In addition, manipulating singularities and other features in these tensor fields produces interesting user-interaction possibilities [2010].

Extension to video – to achieve temporal coherency, 2D SBR methods have mainly relied on storing strokes in memory between frames, and performing optical-flow calculation or other similar metrics to move and distort their strokes[1][3][4]. Another recently published technique is particle flow in a 3D ETF field[9], requiring no optical flow calculation, and no intermediate stroke storage required, however the results don't look very painterly but rather like blurred images (similar to my discussion in Section 2 of LIC diffusion methods). Our current method is extensible using both approaches and I'd like to investigate a happy medium, hoping for an online greedy method that gives the video the same rich painted look we have right now with images, but without the optical-flow storage overhead, and maybe also allow animated strokes (think light-cycles on a Tron suit).

Richer Strokes – I'm happy with this initial development of the colour theme manager and the way it easily transforms the image. On the other hand, if we wanted to arrive at something rich and colorful like Semeel's in [Figure 2(c)] and other similarly sophisticated paintings, we propose an adjustment to the brush painting model. First, a layered approach would be great, adapting Hertzmann's curved strokes, taking into consideration the edge flow topology. Second, allowing for and randomizing the stroke texture. Third, exploring a more painter friendly color-spaces, like Red-Yellow-Blue (RYB) space, as suggested by Schwartz[8].

Initial Stroke Placement – random placement worked great in our implementation, but that limits our ability to parameterize it. For example, we would like to spread the strokes uniformly along the field lines, and be able to control the spacing and randomness of the placement. This can be achieved using the techniques described by Hansen[27] and Jobard[30].

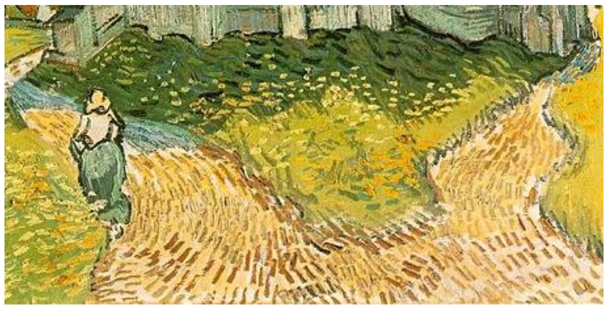


Figure 9. *The strokes in this detail of a van Gogh illustrate the concept of uniform distribution along and amongst the vector stream lines.*

These changes are not aimed at replacing human artists, but rather providing a practical implementation that leverages computational power, in order to empower their creative process and allow them to efficiently apply it to, say, video or 3D surfaces.

Acknowledgements

For personal academic use only. I have not yet obtained permissions for some of the images present here. Please do not distribute, submit or share this document publicly.

References

[1] Peter Litwinowicz. 1997. Processing images and video for an impressionist effect. *(SIGGRAPH '97).*

[2] Hochang Lee, Sanghyun Seo, and Kyunghyun Yoon. Directional texture transfer with edge enhancement. *(NPAR '10)*

[3] Sven C. Olsen, Bruce A. Maxwell, and Bruce Gooch. 2005. Interactive vector fields for painterly rendering. *In Proceedings of Graphics Interface 2005 (GI '05).*

[4] James Hays and Irfan Essa. 2004. Image and video based painterly animation. *In Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering (NPAR '04), Stephen N. Spencer (Ed.).*

[5] Barbara J. Meier. 1996. Painterly rendering for animation. *In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96).*

[6] Huang, H., Zhang, L. and Fu, T.-N. (2010), Video Painting via Motion Layer Manipulation. *Computer Graphics Forum, 29: 2055–2064*.

[7] Yuanyuan Li, Fan Bao, Eugene Zhang, Yoshihiro Kobayashi, and Peter Wonka. 2011. Geometry Synthesis on Surfaces Using Field-Guided Shape Grammars. *IEEE Transactions on Visualization and Computer Graphics 17, 2 (February 2011)*

[8] Martin Schwarz, Tobias Isenberg, Katherine Mason, and Sheelagh Carpendale. 2007. Modeling with rendering primitives: an interactive non-photorealistic canvas. *(NPAR '07).*

[9] Yoon, J.; Lee, I.; Kang, H.; , "Video Painting Based on a Stabilized Time-Varying Flow Field," Visualization and Computer Graphics,

[10] Holger Winnemöller. Oilpaint Effect in Pixel Bender Plugin for Photoshop CS5. *Adobe Systems.*2010

[11] Joachim Weickert, Coherence-enhancing diffusion of colour images, Image and Vision Computing, Volume 17, Issues 3-4, March 1999, Pages 201-212.

[12] Jonathan Palacios and Eugene Zhang. 2007. Rotational symmetry field design on surfaces. ACM Trans. Graph. 26, 3, Article 55 (July 2007)

[13] Palacios J, Zhang E. Interactive Visualization of Rotational Symmetry Fields on Surfaces. *IEEE Trans VisComput Graph. 2010 Sep 10.*

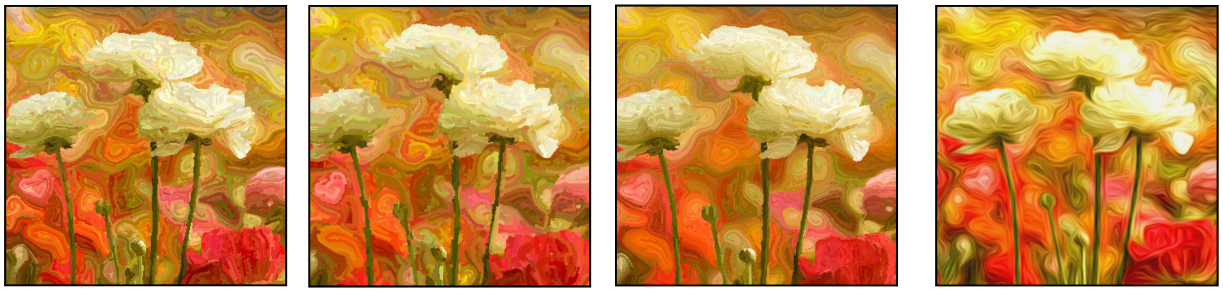


Figure 10. *First three: our method with decreasing luminance variance and increasing edge protection; Last image: Adobe Photoshop CS5 OilPaint filter*[10].

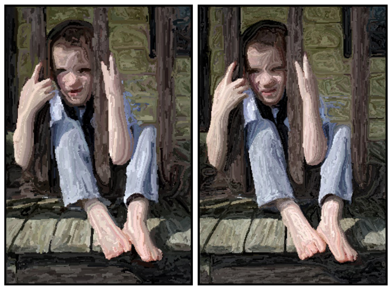
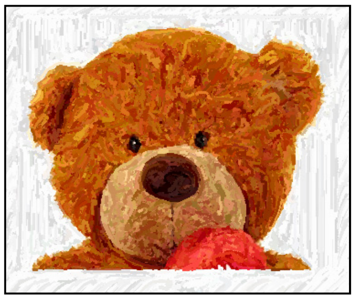


Figure 11. [Left] *The same degree of edge protection works for different brush sizes and varying levels of detail.*

Figure 12. [Right] *Even when furry details get lost through bilateral filtering, our color theme variation can bring them back in a harmonic way.*



[14] Michael P. Salisbury, Michael T. Wong, John F. Hughes, and David H. Salesin. 1997. Orientable textures for image-based pen-and-ink illustration. *(SIGGRAPH '97).*

[15] Lee, Jaewook. Volume painting: incorporating volumetric rendering with line integral convolution (LIC). *Texas A&M Theses and Dissertations Collection.*

[16] Eugene Zhang, James Hays, and Greg Turk. 2007. Interactive Tensor Field Design and Visualization on Surfaces. *IEEE Transactions on Visualization and Computer Graphics 13, 1 (January 2007)*

[17] Eugene Zhang, Harry Yeh, Zhongzang Lin, and Robert S. Laramee. 2009. Asymmetric Tensor Analysis for Flow Visualization. *IEEE Transactions on Visualization and Computer Graphics 15, 1 (January 2009),*

[18] Brian Cabral and Leith Casey Leedom. 1993. Imaging vector fields using line integral convolution. *In Proceedings of the 20th annual conference on Computer graphics and interactive techniques (SIGGRAPH '93).*

[19] Detlev Stalling and Hans-Christian Hege, "Fast and Resolution Independent Line Integral Convolution," *Proceedings of ACM SIGGRAPH 95.*

[20] Henry Kang, Seungyong Lee, and Charles K. Chui. 2007. Coherent line drawing. *In Proceedings of the 5th international symposium on Non-photorealistic animation and rendering (NPAR '07).*

[21] Henry Kang, Seunyong Lee. Shape-simplifying Image Abstraction. *Pacific Graphics 2008.*

[22] Henry Kang, Seungyong Lee, and Charles K. Chui. 2009. Flow-Based Image Abstraction. *IEEE Transactions on Visualization and Computer Graphics 15, 1 (January 2009)*

[23] Jan Eric Kyprianidis, Henry Kang. Image and Video Abstraction by Coherence-Enhancing Filtering. *Proceedings Eurographics 2011.*

[24] William Baxter, Jeremy Wendt, and Ming C. Lin. 2004. IMPaSTo: a realistic, interactive model for paint. *In Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering (NPAR '04),*

[25] Aaron Hertzmann. 1998. Painterly rendering with curved brush strokes of multiple sizes. *(SIGGRAPH '98).*

[26] Yutaka Goda, Tsuyoshi Nakamura, and Masayoshi Kanoh. Texture transfer based on continuous structure of texture patches for design of artistic Shodo fonts. *In ACM SIGGRAPH ASIA 2010 Sketches (SA '10).*

[27] Charles Hansen, Chris R. Johnson. Visualization Handbook. 2005 Elsevier Inc. ISBN: 978-0-12-387582-2

[28] Paul Haeberli. 1990. Paint by numbers: abstract image representations. *(SIGGRAPH '90).*

[29] Daniel Cohen-Or, Olga Sorkine, Ran Gal, Tommer Leyvand, and Ying-Qing Xu. 2006. Color harmonization. *In ACM SIGGRAPH 2006 Papers (SIGGRAPH '06).*

[30] B. Jobard and W. Lefer, "Creating Evenly-Spaced Streamlines of Arbitrary Density," Proc. Eurographics Workshop Visualization in Scientific Computing. 1997.

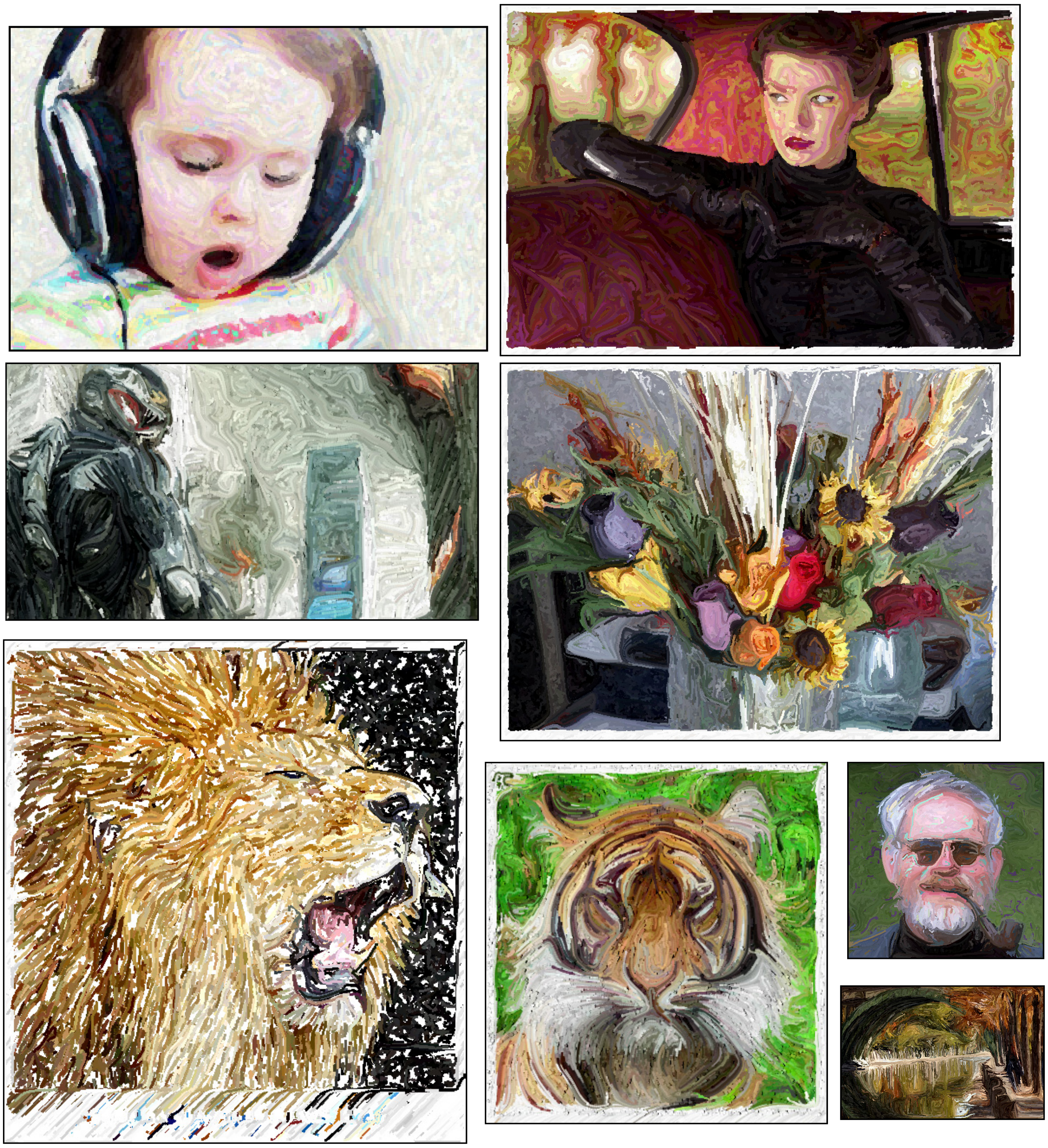


Figure 13. *Various styles achievable using our painter (source images available in the Processing project folder)*

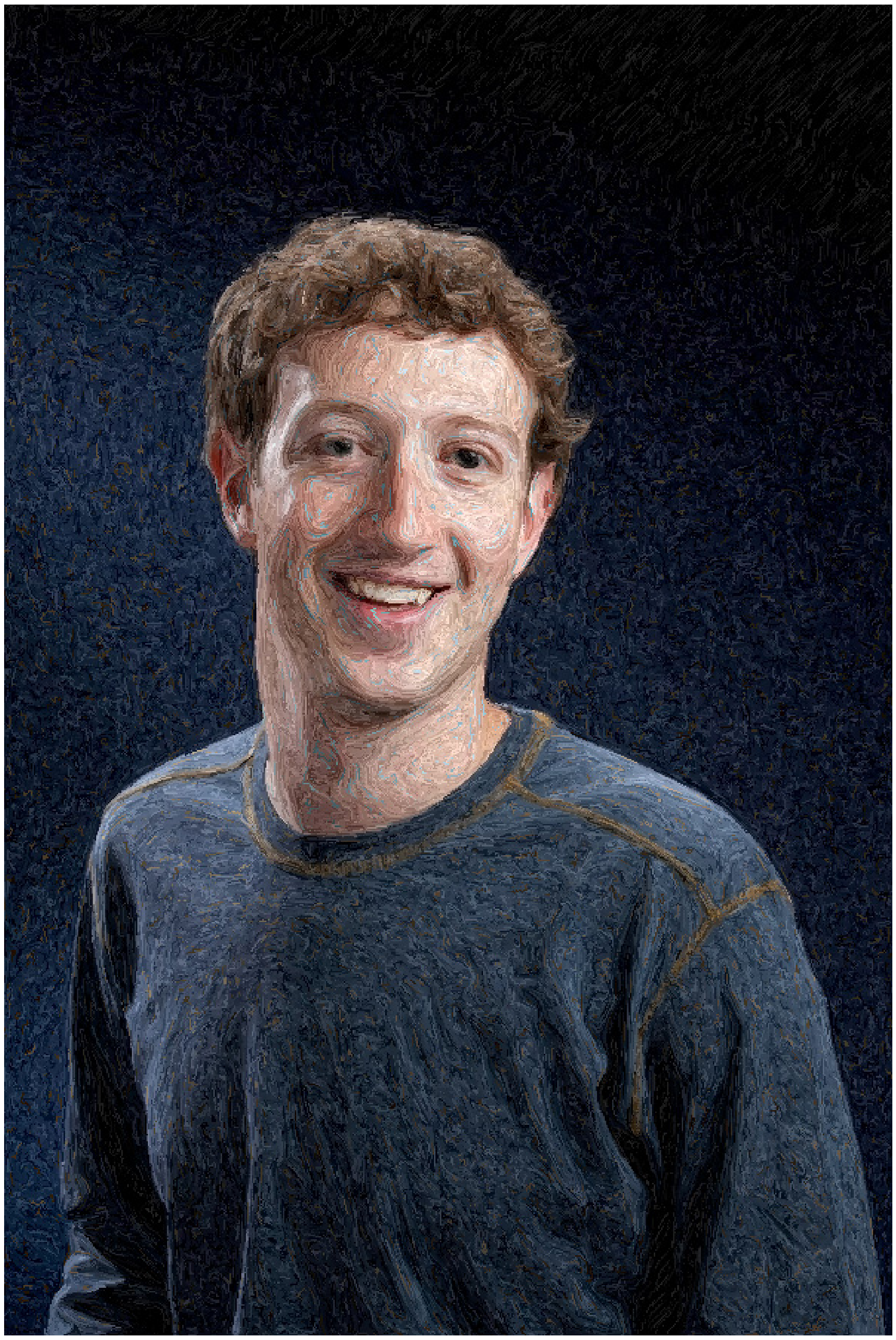


Figure 14. *High detail painting with the color opponency theme and moderate edge protection.*