TENSOR-DIRECTED SIMULATION OF STROKES FOR IMAGE STYLIZATION WITH HATCHING AND CONTOURS

David Tschumperlé

GREYC Laboratory (CNRS UMR 6072), Image Team, 6 Bd Maréchal Juin, 14050 Caen/France

ABSTRACT

We describe a simple but capable algorithm to generate sketches from color images. The method is based on the drawing of dark strokes on a white background, directed by an image-dependent tensor-valued geometry. It is able to produce various sketching styles with different kind of hatching and contours. Combining these grayscale sketches with the colors of the input images allows to produce a wide variety of image stylization renderings.

Index Terms— Non-Photorealistic Rendering, Image Stylization, Sketching, Tensor Geometry, Stroke Simulation, Streamlines.

1. INTRODUCTION

In the field of non-photorealistic imagery, stroke-based techniques are often used as primary steps to render painterly or sketchy versions of input color images [1]. Many methods have been proposed in the literature, mainly to transform images into paintings [2], sketches [3], stipple drawings [4], comics [5], or pen-and-ink illustrations [6, 7, 8]. Similar ideas have been used as well, for the visualization of vector and tensor fields [9, 10], or medical images [11]. The main differences between these algorithms lie both in the placement and in the shape of the drawn primitives. Here, we propose an original stroke-based rendering algorithm which has the particularity of being able to generate both stylized edges and hatches *at the same time*. This is feasible through the consideration of a 2^{nd}*-order tensor-valued geometry* for the strokes. Such tensors can indeed describe local image structures having both isotropic (flat regions, background) or anisotropic shapes (contours). Thus, we "throw" either straight segments or curved streamlines, randomly placed on the image, in a way that they remain consistent with this prescribed tensor geometry. This makes regular hatches appearing on homogeneous areas while stylizing edges on image contours. Besides, our all-in-one method is particularly simple to implement and runs quickly, while being able to generate various rendering styles due to the high variability of the proposed tensor model (Fig.1). At the end, we show more stylization results obtained by simple combinations between the grayscale output of our sketching algorithm with the colors of the input image.

2. A FIRST NAIVE APPROACH

In order to simulate the artistic drawing of black strokes on a white background, we suggest to determine a finite set of segments (roughly) representing the significant structures of an input color image $\mathbf{I} : \Omega \to \mathbb{R}^n$. Many of the contour informations in $\mathbf{I}$ may be found in the gradient field of its luminance channel $Y : \Omega \to \mathbb{R}$ (or a smoothed version of it, if $\mathbf{I}$ is noisy). Based on this first assumption, we could build a very naive but straightforward image sketching algorithm, by applying the following processing pipeline :

Fig. 1. *A tensor-directed stroke simulation algorithm is proposed, for the rendering of stylized images from an input color photograph (top-left). It is capable of generating various drawing styles.*

1. Compute the luminance field Y of the input color image $\mathbf{I}$, then its smoothed gradient $\nabla Y_\sigma = Y * G_\sigma$, where G_σ is a normalized gaussian kernel with standard deviation $\sigma \geq 0$.
2. Initialize a white background image $S : \Omega \to \mathbb{R}$. It will contain the final grayscale rendering of the sketch.
3. Pick a random location (x, y) in Ω, such that $\|\nabla Y_{(x,y)}\| \geq \epsilon$. The user-defined parameter $\epsilon \geq 0$ determines whether the point (x, y) can be considered located on an edge or not.
4. Draw a (partially opaque) black segment from $(x - u, y - v)$ to $(x + u, y + v)$ in S, where $(u, v) = \frac{L}{2} \frac{\nabla Y^\perp}{\|\nabla Y\|}$. The stroke length $L \geq 0$ may be proportional to $\|\nabla Y_{(x,y)}\|$ (local edge contrast).
5. Loop to step **3.** until enough strokes have been drawn on S.

Yet, this rudimental approach succeeds already in rendering notable sketch-like images (Fig.2). But, some usual characteristics of artistical sketches are still missing : First, the method draws only *straight strokes*, which limits the stylization variety of the contours. Second, it does not generate aligned strokes on homogeneous areas of the image (background), i.e. when choosing $\epsilon \to 0$ (Fig.2a).. Indeed, the gradient orientation $\frac{\nabla I}{\|\nabla I\|}$ has no clear meaning there

(a) $(\epsilon, \sigma) = (0, 0)$ (b) $(\epsilon, \sigma) = (20, 0)$ (c) $(\epsilon, \sigma) = (20, 1)$

Fig. 2. *Results of the naive approach with different parameter sets.*

(since $\|\nabla I\| \approx 0$), and the drawn segments are randomly crossing *without any spatial coherence*, as hatching would typically demand. In the followings, we propose to improve this naive approach by introducing tensor fields to model the strokes geometry, instead of considering a vector-valued one (as the gradient ∇Y). We show that this extension allows to render both stylized contours on high-gradient areas and spatially regular cross-hatches on flat regions. Besides, we propose to naturally handle the drawing of curved strokes (using streamlines) in order to add more stylization diversity.

3. TENSOR-VALUED GEOMETRY OF STROKES

Second-order tensors are symmetric and positive-definite matrices suitable to model geometric features of either anisotropic or isotropic natures. Fields of 2×2 tensors have been successfully used so far in the literature, to model the geometry of local image structures (*structure tensors*) [12, 13], or to describe local behaviors of diffusion processes for image denoising (*diffusion tensors*) [13, 14]. For our purpose of image sketching, we assume similarly that the strokes we would like to draw may have either isotropic (cross-hatch) or anisotropic (contour) configurations, whether the stroke is located on an flat region or on an image edge. Within this context, using a tensor field to locally model this variety of stroke arrangements looks quite natural. Our two-step approach is close to those in [13, 14] for the definition of diffusion tensors for image regularization : We first estimate the smoothed structure tensor field $\mathbf{G}_{\alpha,\sigma} : \Omega \to \mathrm{P}(2)$:

$$\mathbf{G}_{\alpha,\sigma} = \left(\sum_{i=1}^{n} \nabla I_{i\alpha}\ \nabla I_{i\alpha}^T\right) * G_\sigma \ \text{ where } \ I_{i\alpha} = I_i * G_\alpha \qquad (1)$$

I_i stands for the i^{th} channel of the n-valued image $\mathbf{I}$. On each image point $(x, y) \in \Omega$, the positive eigenvalues λ_+, λ_- of $\mathbf{G}_{\alpha,\sigma(x,y)}$ give local informations on the color variation of $\mathbf{I}$. The measure $\lambda_+ + \lambda_-$ can thus be used to determine on which kind of geometrical structure we are located (edge, flat region, corner). The corresponding orthogonal eigenvectors θ_+, θ_- tell about the orientation of these local structures. Actually, $\mathbf{G}_{\alpha,\sigma}$ can be fairly represented by a field of ellipses of radii $\lambda_{+/-(x,y)}$ and orientations $\theta_{+/-(x,y)}$ (Fig.3a). Computing the structure tensor (1) guarantees that all color variations are considered at the same time, contrary to the naive approach where the image geometry is guessed only from the luminance channel Y. Besides, the smoothness parameters (α, σ) allow the estimated image geometry to be more robust to local noise and/or to be estimated at any spatial scale.

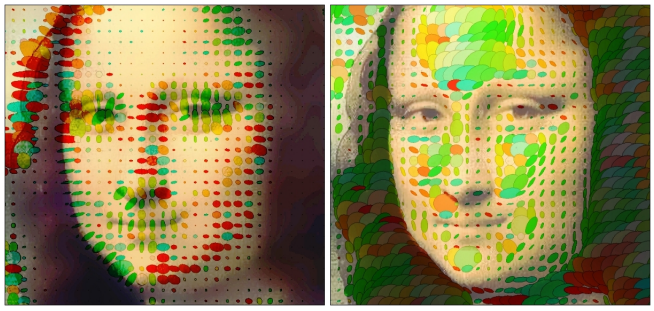
Once this local color geometry $\mathbf{G}_{\alpha,\sigma}$ has been estimated, we model the *strokes geometry* by another tensor field $\mathbf{T} : \Omega \to \mathrm{P}(2)$, defined as : $\forall (x, y) \in \Omega, \quad \mathbf{T} = c_+ \theta_+ \theta_+^T + c_- \theta_- \theta_-^T$

$$\text{with} \quad c_+ = \frac{1}{(1+\lambda_+ + \lambda_-)^{p_1}} \quad \text{and} \quad c_- = \frac{1}{(1+\lambda_+ + \lambda_-)^{p_2}} \qquad (2)$$

(where $p_1 \geq p_2 \geq 0$). This field $\mathbf{T}$ has the following properties :

- Where $\mathbf{T}_{(x,y)}$ is anisotropic, it will be oriented mainly along θ_- ($p_1 \geq p_2$). Then, (x, y) is likely on an image contour (higher $\lambda_{+/-}$) and all strokes thrown from there must follow the edge direction θ_-.
- Where $\mathbf{T}_{(x,y)}$ is isotropic, it has no major orientations. Then (x, y) lies on an almost-flat image area, and all strokes thrown from there must generate a spatially coherent cross-hatching arrangement.
- The in-between configurations (swelled tensors) are expressed in a continuous manner and must generate semi-contours/hatches.

The eigenvalues $c_{+/-}$ of $\mathbf{T}_{(x,y)}$ are related to the length of the stroke at (x, y), both for contours and hatches. Satisfyingly, two parameters p_1, p_2 for $\mathbf{T}$ suffice to achieve a wide variety of sketching styles, since they are able to globally set at the same time the preferred tensor profiles (isotropic everywhere with $p_1 \approx p_2$, anisotropic everywhere with $p_1 \gg p_2 \gg 0$, or a continuous mix of the two), as well as their way of depending on the local image contrast $\lambda_+ + \lambda_-$. One example of such a stroke tensor field $\mathbf{T}$ is illustrated on Fig.3b, with $(p_1, p_2) = (1.2, 0.5)$ and $(\alpha, \sigma) = (0.5, 1.2)$. Thereafter, we show how to draw a set of strokes that precisely coincides with this prescribed tensor-valued stroke geometry $\mathbf{T}$.



(a) Structure tensors $\mathbf{G}_{\alpha,\sigma}$ (b) Stroke tensors $\mathbf{T}$.

Fig. 3. *Structure and stroke tensor fields inlaid on an input image.*

4. THROWING TENSOR-DIRECTED STROKES

Whereas the vector-directed stroke drawing algorithm (section 2) always uses the same orientation $\frac{\nabla Y_{(x,y)}}{\|\nabla Y_{(x,y)}\|}$ to draw a segment at one point (x, y), we want tensor-directed strokes to be rendered as single- or multiple-oriented segments (contours or hatches), whether $\mathbf{T}_{(x,y)}$ is anisotropic or not. For this purpose, we decompose the tensor field $\mathbf{T}$ through several vector fields $\mathbf{w}_\gamma : \Omega \to \mathbb{R}^2$, each of them representing the *amount of* $\mathbf{T}$ *along the orientation* $\gamma \in [0, \pi]$ of the plane. As $\int_{\gamma=0}^{\pi} a_\gamma a_\gamma^T \, d\gamma = \frac{\pi}{2}\, \mathbb{I}_d$, with $a_\gamma = (\cos\gamma \ \ \sin\gamma)^T$, we have $\mathbf{T} = \frac{2}{\pi}\, \sqrt{\mathbf{T}} \left(\int_{\gamma=0}^{\pi} a_\gamma a_\gamma^T \, d\gamma\right) \sqrt{\mathbf{T}}$, leading to the expression :

$$\mathbf{T} = \frac{2}{\pi} \int_{\gamma=0}^{\pi} \mathbf{w}_\gamma \mathbf{w}_\gamma^T \, d\gamma\,, \quad \text{with} \quad \mathbf{w}_\gamma = \sqrt{\mathbf{T}} a_\gamma$$

$\sqrt{\mathbf{T}}$ stands for the usual matrix square root. Each $\mathbf{w}_\gamma \mathbf{w}_\gamma^T$ is a field of purely anisotropic unit tensors, oriented along $\mathbf{w}_\gamma$. Then, we get :

- Where $\mathbf{T}_{(x,y)}$ is fully anisotropic ($c_+ \approx 0$), $\mathbf{w}_{\gamma(x,y)}$ is equal to $(\theta_-.a_\gamma) c_-^{\frac{1}{2}} \theta_-$ (parallel to θ_-). Then, on image edges, the strokes will be necessarily drawn along the edge directions, whatever γ is.
- Where $\mathbf{T}_{(x,y)}$ is fully isotropic ($c_+ \approx c_-$), $\mathbf{w}_{\gamma(x,y)} = c_- a_\gamma$ stays parallel to a_γ. Then, on a flat region, multiple-oriented strokes with different angles γ will be generated from the same point (x, y). It enables the rendering of various hatching styles, depending on the way γ is discretized (i.e by the 3 parameters $\gamma_{min}, \gamma_{max}$ and $\delta\gamma$).

• An in-between tensor will attract more or less the plane orientations a_γ into its eigenvectors $\theta_{+/-}$, in accordance with its anisotropy. Such tensors allow to generate a smooth transition of the strokes arrangements, from contours to hatches.

The splitting of the tensor field $\mathbf{T}$ using several vector fields intuitively suggests to extend the stroke rendering algorithm by drawing streamlines of $\mathbf{w}_\gamma$, instead of straight lines (as in section 2). A streamline of length L is an integral curve $\mathcal{C} : [-\frac{L}{2}, \frac{L}{2}] \to \mathbb{R}^2$, implicitely defined by :

$$\frac{\partial \mathcal{C}_{(l)}}{\partial l} = \mathbf{w}(\mathcal{C}_{(l)}) \ , \quad \text{where} \quad \mathcal{C}_{(0)} = (x, y) \tag{3}$$

Considering such curves as stroke primitives is interesting, since they are able to stick to the image contours, whatever their curvatures are. Consequently, it create less harsh stylizations of the edges.

Finally, our tensor-directed stroke simulation algorithm reads :
1. Compute the stroke tensor field $\mathbf{T}$ (2) of the input color image $\mathbf{I}$.
2. Split it to several vector fields $\mathbf{w}_\gamma = \sqrt{\mathbf{T}} a_\gamma$, for γ discretized in $[\gamma_{min}, \gamma_{max}]$ with an angular step δ_γ.
3. Initialize a white background image $S : \Omega \to \mathbb{R}$. It will contain the final grayscale rendering of the sketch.
4. Pick a random location (x, y) in Ω.
5. For each available $\mathbf{w}_\gamma$, draw a (partially opaque) black segment from $(x, y) - L\ \mathbf{w}_{\gamma(x,y)}$ to $(x, y) + L\ \mathbf{w}_{\gamma(x,y)}$, or a streamline (3) of $\mathbf{w}_\gamma$ starting from (x, y), with length $||L\mathbf{w}_{\gamma(x,y)}||$.
6. Loop to step **4.** until enough strokes have been drawn on S.

Fig.4 illustrates the results of this algorithm on a synthetic image (top-left), with streamlines as primitives. Note how many different hatching styles can be generated whereas the parameters $\gamma_{min}, \gamma_{max}, \delta_\gamma$ change, The stroke tensors $\mathbf{T}$ are displayed as colored ellipses in the background of the generated sketches.

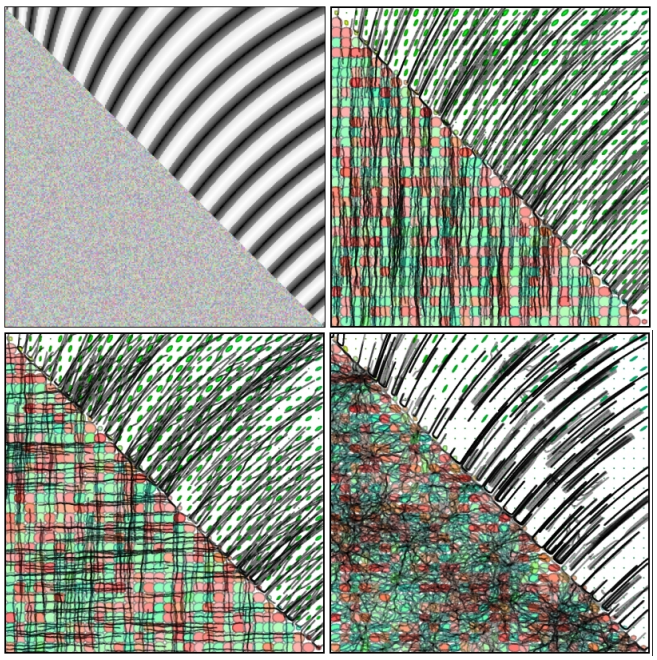


Fig. 4. *Tensor-directed simulation of strokes from a synthetic image (top-left) containing both isotropic (noise) and anisotropic (stripes) structures. Corresponding* $(\gamma_{min}, \gamma_{max}, \delta_\gamma)$ *are* $(90^o, 90^o, 0)$ *(top-right),* $(0^o, 90^o, 90^o)$ *(bott.-left) and* $(15^o, 165^o, 20^o)$ *(bott. right).*

5. COLORIZATION AND CONCLUSION

Once the grayscale image of strokes have been generated, we can compose it with the colors of the input image in order to render the final sketch. To generate our results, we used only simple image composition techniques (multiplications, overlay, hard light,...), also sometimes blurring and boosting colors of the stylization results a little bit to make them more cartoonish. Sketches in Fig.5,6 have been obtained with a single pass of our stroke-based rendering method, while Fig.7 exhibits two results where several passes of our algorithm (from the same color input image, but with different parameters) have been merged. Additional paper texture have been added in the background to finalize the sketch rendering.

The tensor-directed stroke simulation algorithm we proposed is quite flexible and generic, and succeeds in creating sketches where the most important structures of the input images are extracted and displayed, thanks to the use of local tensor-valued geometric models. Yet, the algorithm is simple enough to be implemented quickly, and can be potentially parallelized for real-time applications. It opens interesting perspectives in the field of automatic and real-time conversions from videos to cartoon movies.

6. REFERENCES

[1] A. Hertzmann, "A survey of stroke-based rendering," *IEEE Comp. Graphics and Applications*, vol. 23, pp. 70–81, 2003.

[2] A. Hertzmann, "Painterly rendering with curved brush strokes of multiple sizes," in *SIGGRAPH*, 1998, pp. 453–460.

[3] F. Durand, V. Ostromoukhov, M. Miller, F. Duranleau, and J. Dorsey, "Decoupling strokes and high-level attributes for interactive traditional drawing," in *Eurographics Workshop on Rendering Techniques*, 2001, pp. 71–82.

[4] O. Deussen, S. Hiller, C. van Overveld, and T. Strothotte, "Floating points: A method for computing stipple drawings," *Computer Graphics Forum*, vol. 19, pp. 40–51, 2000.

[5] D. DeCarlo and A. Santella, "Stylization and abstraction of photographs," in *SIGGRAPH*, 2002, pp. 769–776.

[6] O. Deussen and T. Strothotte, "Computer-generated pen-and-ink illustration of trees," in *SIGGRAPH*, 2000, pp. 13–18.

[7] M. Salisbury, C. Anderson, D. Lischinski, and D.H. Salesin, "Scale-dependent reproduction of pen-and-ink illustrations," in *SIGGRAPH*, 1996, pp. 461–468.

[8] G. Winkenbach and D.H. Salesin, "Computer-generated pen-and-ink illustration," in *SIGGRAPH*, 1994, pp. 91–100.

[9] B. Cabral and L.C. Leedom, "Imaging vector fields using line integral convolution," in *SIGGRAPH*, 1993, vol. 27.

[10] D. Tschumperlé and R. Deriche, "Tensor field visualization with pde's and application to dt-mri fiber visualization," in *IEEE Workshop on Variational and Level Set Methods*, 2003.

[11] S. Bruckner and M. Eduard Gröller, "Style transfer functions for illustrative volume rendering," vol. 26, no. 3, 2007.

[12] S. Di Zenzo, "A note on the gradient of a multi-image," *Comput. Vision Graph. Image Process.*, vol. 33, pp. 116–125, 1986.

[13] J. Weickert, "Coherence-enhancing diffusion filtering," *Int. Journal. Comput. Vision*, vol. 31, pp. 111–127, April 1999.

[14] D. Tschumperlé and R. Deriche, "Anisotropic diffusion partial differential equations for multichannel image regularization: Framework and applications," vol. 145 of *Advances in Imaging and Electron Physics*, pp. 149 – 209. Elsevier, 2007.

Fig. 5. *Various image stylization renderings using our proposed tensor-directed stroke simulation method.* Images courtesy of Lyle Kroll (Skull image), and Geoff Livingston (Eiffel tower image, licensed under CC-SA).

Fig. 6. *Simulation of chronological sketching steps towards a final painting (last image is the only input to our proposed algorithm).*

Fig. 7. *Generation of drawings from photographs. Several instances of the proposed tensor-directed stroke simulation algorithm have been combined here via simple layer compositions.* Images courtesy of Tom Keil (licensed under CC-SA-NC).