Tensor-Directed Simulation of Strokes for Image Stylization with Hatching and Contours

David Tschumperlé

Image Team, GREYC / CNRS (UMR 6072)

IEEE ICIP'2011, Brussels/Belgium, September 2011

Context & Motivation

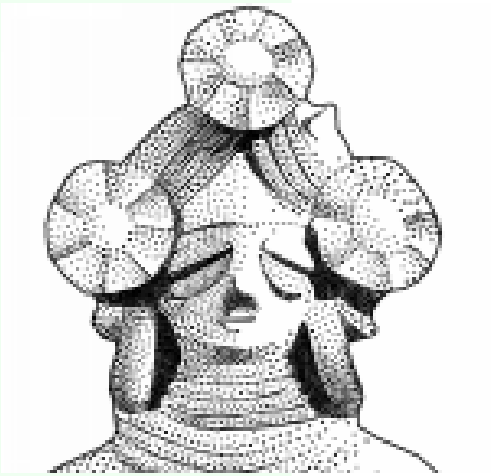
- Context : **Non-Photorealistic Rendering** (NPR).
- Motivation : Render a photograph as a **sketch** or a pencil **drawing**.

- Method : **B&W stroke-based algorithm**. Place (black) graphic primitives on an white canvas, according to the **geometry of the initial image**. Then, colorize the result.

Original color image

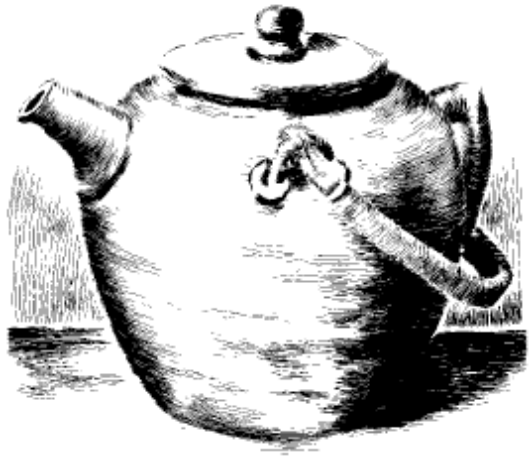
Final rendering

Overview : State of the art of B&W NPR



Deussen-Hiller-etal
(*stipple drawing*)

Durand-Ostromoukhov-etal
(*interactive method*)



Salisbury-Wong-etal
(*oriented textures*)

Hertzmann-Zorin
(*from 3d models*)

Winkenbach-Salesin
(*from 3d models*)

Salisbury-Anderson-etal
(*interactive method*)

Contributions of this work

We propose (yet) another **stroke-based algorithm** with :

- Straight or curved lines as graphic primitives.
- 2^{nd}-order tensors for analyzing the local image geometry **and** modeling the pencil paths.
- Ability to render image **contours** and **hatching** within the same single process.
- Use the flexibility of the tensor model to allow multiple drawing styles with the same algorithm and from the same input image.

$\Longrightarrow$ Try to simulate pencil strokes as they would be drawn by an artist !

Illustration of the proposed method

Same input image leads to multiple styles of pencil rendering

A first naive approach

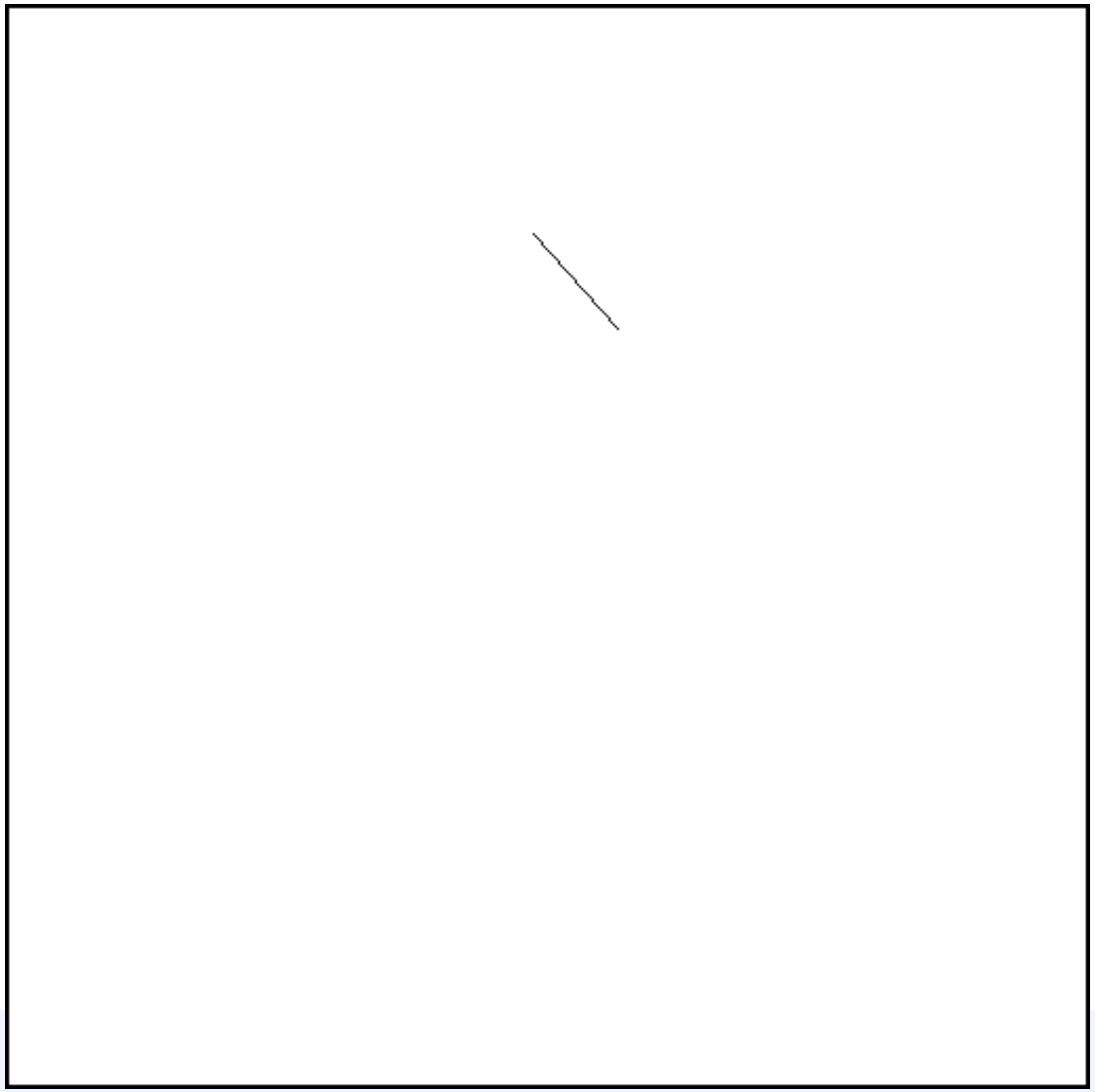
1. Compute the image luminance $Y : \Omega \to \mathbb{R}$, and its smoothed gradient $\nabla Y_\sigma = \nabla Y * G_\sigma$.
2. Create an empty (white) canvas $S : \Omega \to \mathbb{R}$.
3. Repeat N times :
   1. Pick one random position $\mathbf{X} = (x, y)$ in Ω.
   2. Draw a semi-transparent straight line $\mathbf{X} - L\xi_{(\mathbf{X})} \to \mathbf{X} + L\xi_{(\mathbf{X})}$ on S, where $\xi_{(\mathbf{X})} = \frac{\nabla Y^{\perp}_{\sigma(\mathbf{X})}}{\|\nabla Y_{\sigma(\mathbf{X})}\|}$ (contour direction at $\mathbf{X}$).

Color image I

Luminance Y

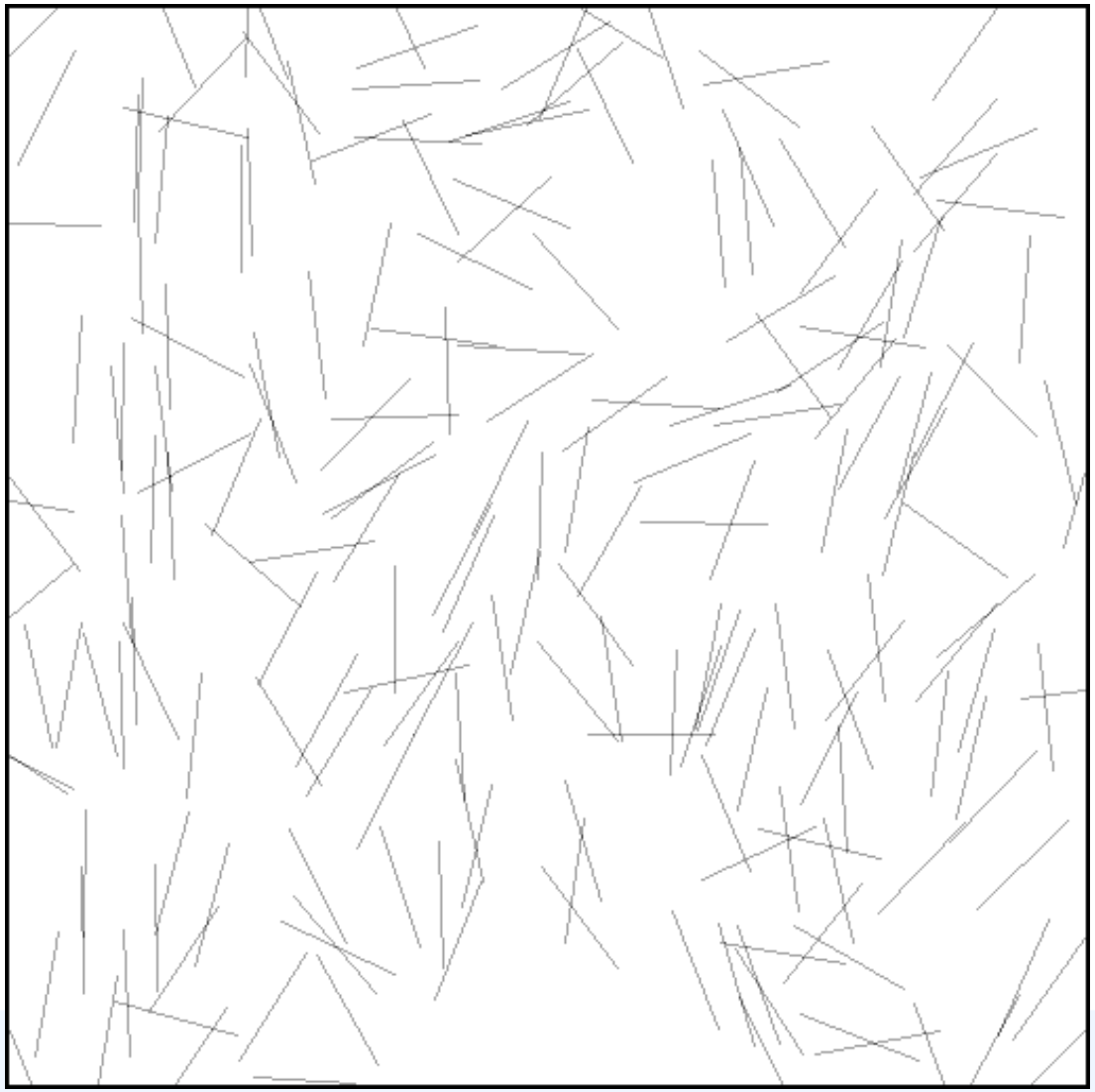
Smoothed gradient ∇Y_σ

After 1st iteration



Canvas S

After 200th iterations



Canvas S

After 600th iterations

Canvas S

After 1200th iterations

Canvas S

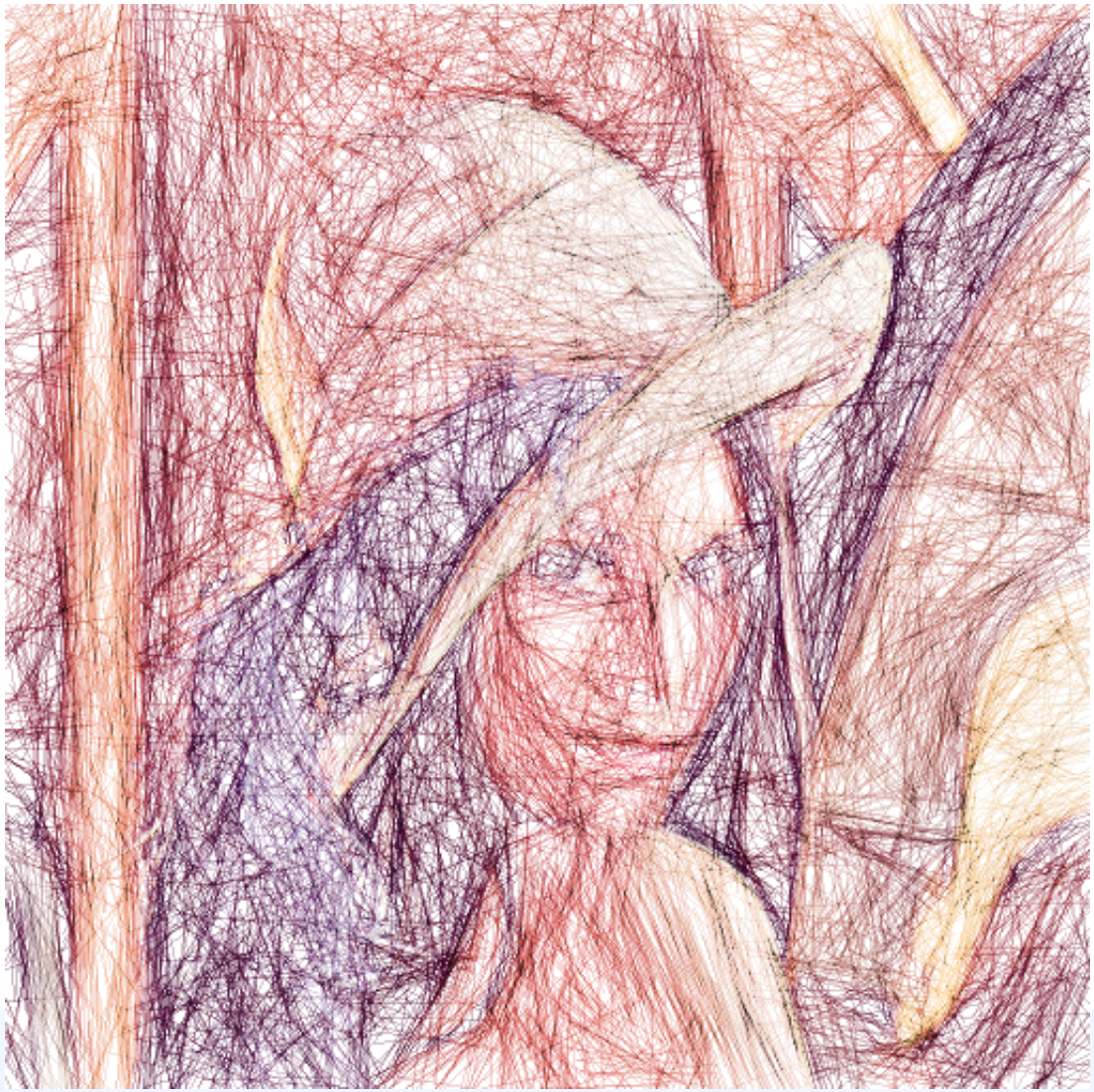
After 4800th iterations

Canvas S

After 10000th iterations

Canvas S

After 10000th iterations (colorized)



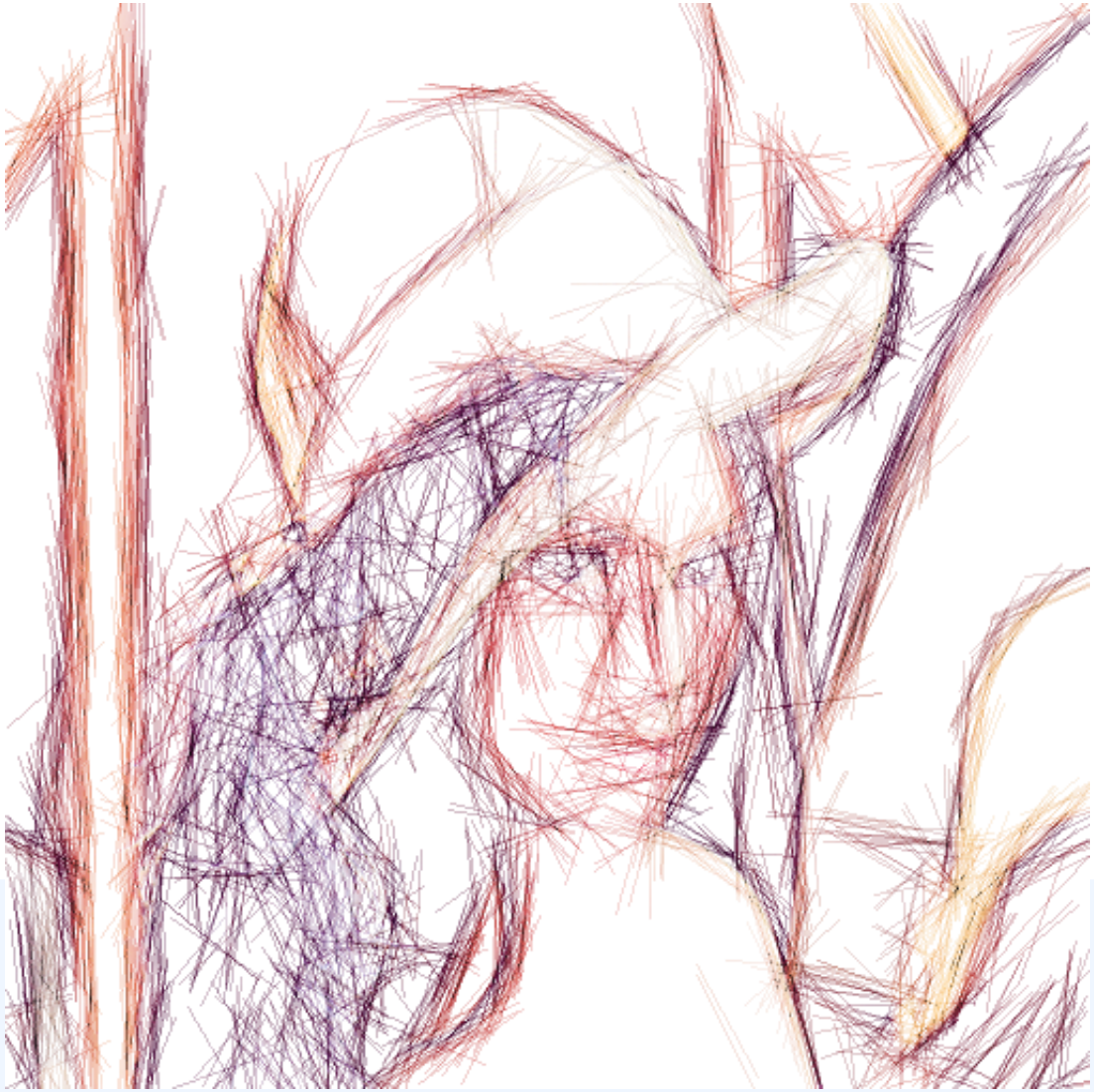
Canvas S + Colorization

Edge-focused modification

- Draw a pencil stroke at **X** only when $\|\nabla Y_{\sigma(\mathbf{X})}\| > \epsilon$ (i.e. **X** is on a significant edge).

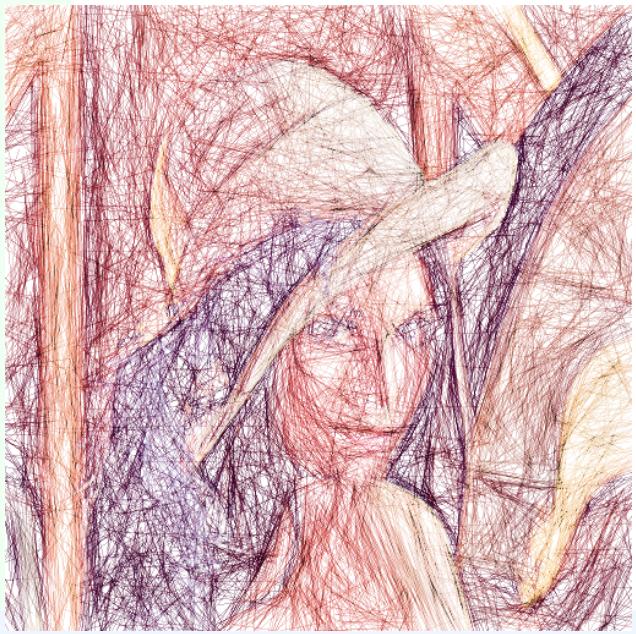
Edge-focused modification (colorized)

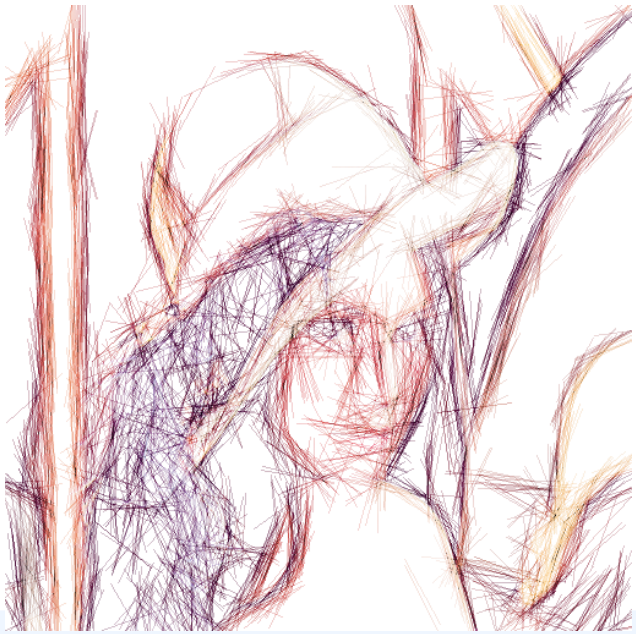
- Draw a pencil stroke at **X** only when $\|\nabla Y_{\sigma(\mathbf{X})}\| > \epsilon$ (i.e. **X** is on a significant edge).



Things that are missing

1. Only straight lines.
2. Poor analysis of the image geometry (simple gradient, no analysis of the color variations).
3. No spatial coherence (or no drawing) on flat image regions.





$\Longrightarrow$ **Can be easily improved by considering tensor models !**

From diffusion tensors...

- Inspired by the formalism of diffusion tensors, used for directing PDE-based anisotropic smoothing processes.

  [Weickert:98,Tschumperle-Deriche:03,...]

$$\frac{\partial I}{\partial t} = \mathrm{div}\,(\mathbf{D}\nabla I) \qquad \text{or} \qquad \frac{\partial I}{\partial t} = \mathrm{trace}\,(\mathbf{DH})$$

- Based on the analysis of the local image geometry through the computation of the structure tensor [DiZenzo:86].

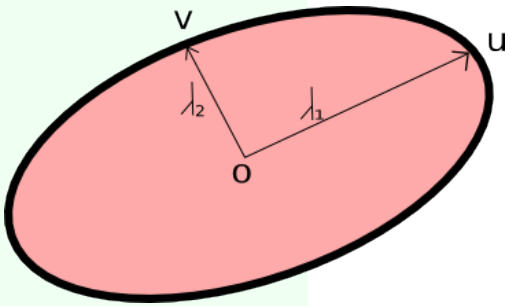
$$\mathbf{G}_{\alpha,\sigma} = \left(\sum_i \nabla \mathbf{I}_{i\alpha}\ \nabla \mathbf{I}_{i\alpha}^T\right) * G_\sigma$$

- Eigenvalues/eigenvectors of $\mathbf{G}_{\alpha,\sigma}$ are **robust local geometric estimators** of the image structures. Diffusion tensors $\mathbf{D}$ are built upon $\mathbf{G}_{\alpha,\sigma}$ and direct the smoothing process
  $\longrightarrow$ *Anisotropic diffusion*.

From diffusion tensors (illustrated)...

$$T = \lambda_1 \, \mathbf{u}\mathbf{u}^T + \lambda_2 \, \mathbf{v}\mathbf{v}^T$$

$$= \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$



2nd order tensor **T**

Structure tensors field $\mathbf{G}_\sigma$

Diffusion tensors field **D**

$\Longrightarrow$ Diffusion tensors model the way a digital painter would apply the "smudge tool" to smooth the noise and remove image artefacts.

...to stroke tensors

- Structure tensors are able to locally distinct between **different types of image regions** :
  1. Flat regions : Small isotropic tensors.
  2. Contours : Anisotropic tensors, oriented $\perp$ to the contour.
  3. Small-scale texture : Large isotropic tensors.

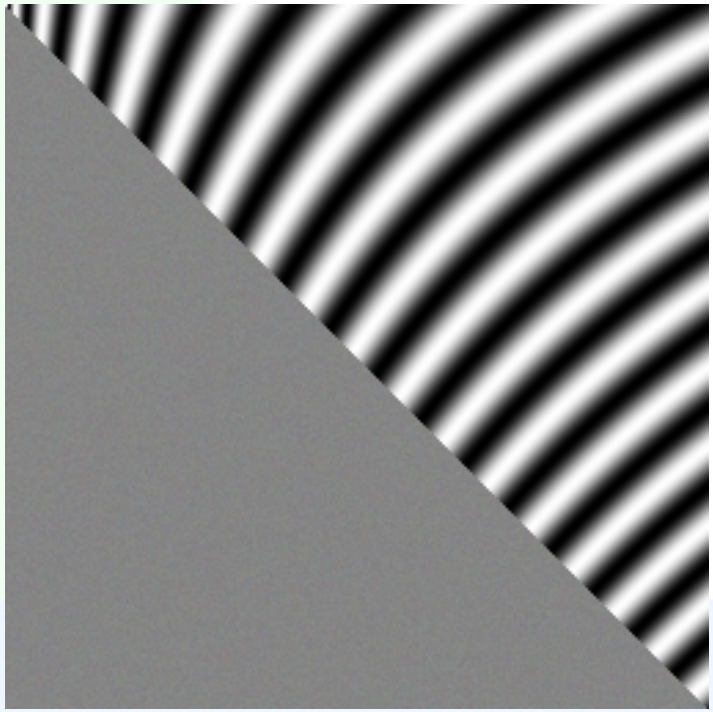
- From an "sketching" point of view, this would correspond to differents **drawing behaviors** :
  1. Flat regions or textures : Regular strokes forming a coherent hatching pattern.
  2. Contours : Oriented strokes, along the contour direction.

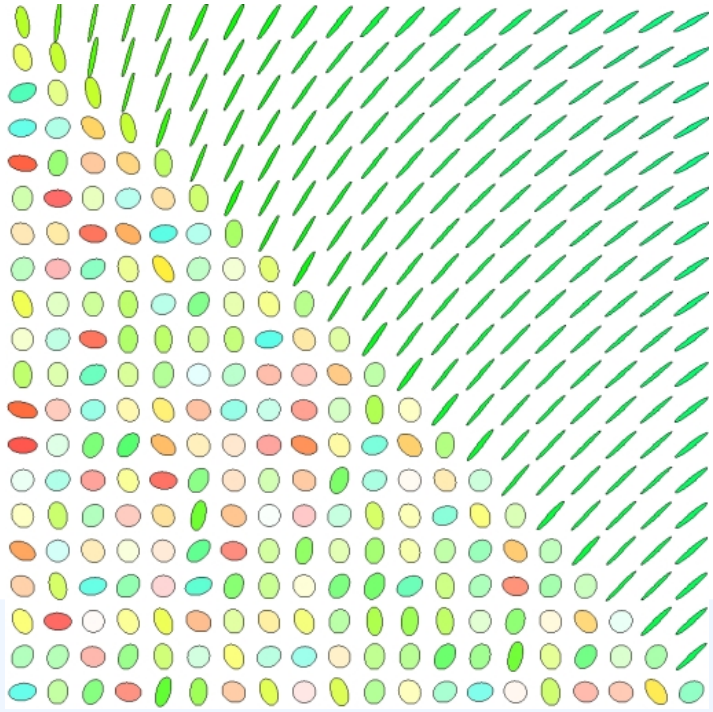
$\Longrightarrow$ **Strokes geometry can be successfuly modeled by a tensor field T that depends on $G_{(\alpha,\sigma)}$!**

[See paper content for more details about our proposal]

...to stroke tensors

- From an “sketching” point of view, this would correspond to differents **drawing behaviors** :
  1. Flat regions or textures : Regular strokes forming a coherent hatching pattern.
  2. Contours : Oriented strokes, along the contour direction.

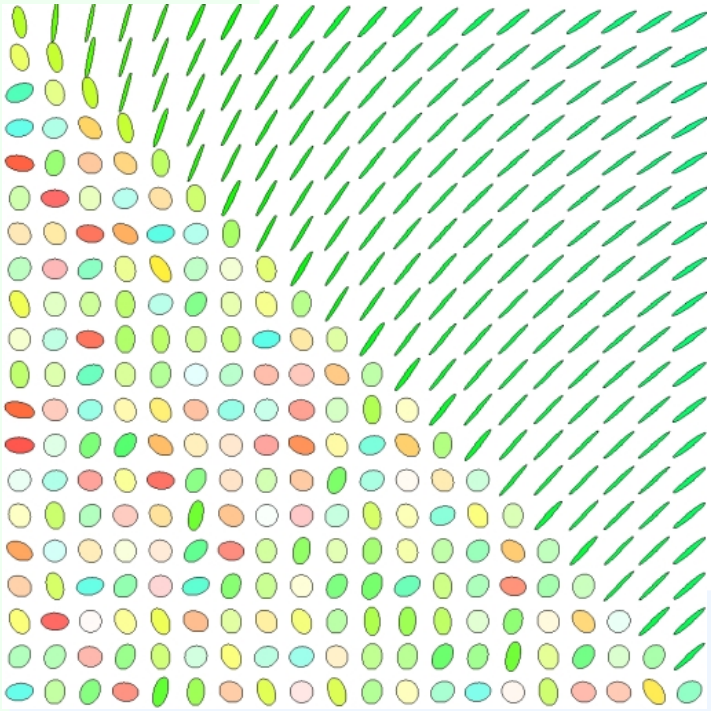




$\Longrightarrow$ **Stroke tensors variability will lead to different drawing styles.**

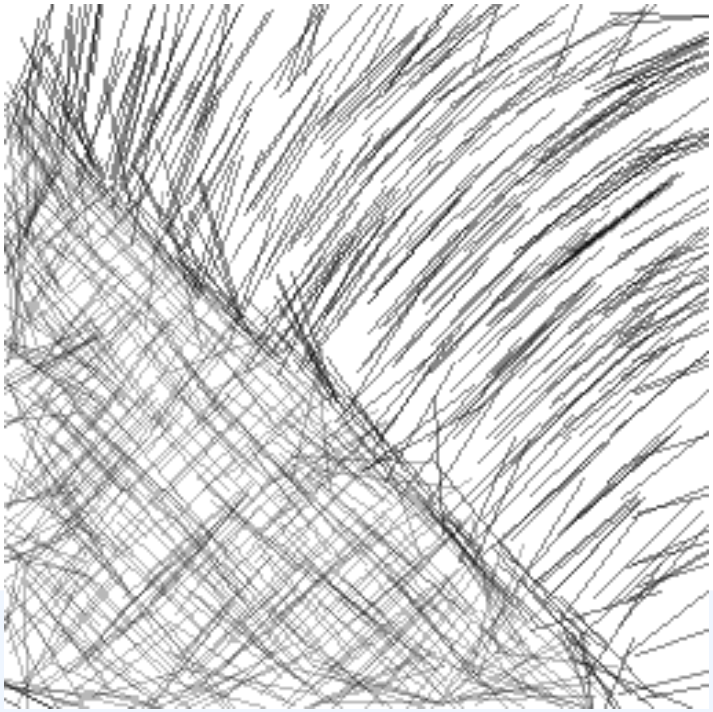
From stroke tensors...

- **Question :** How to draw strokes on canvas S with respect to the stroke tensor field **T**, such that :
  1. Small isotropic tensors locally render very few strokes.
  2. Large isotropic tensors locally render hatches.
  3. Anisotropic tensors locally render oriented contour strokes.



?

$\longrightarrow$



...To stroke drawing

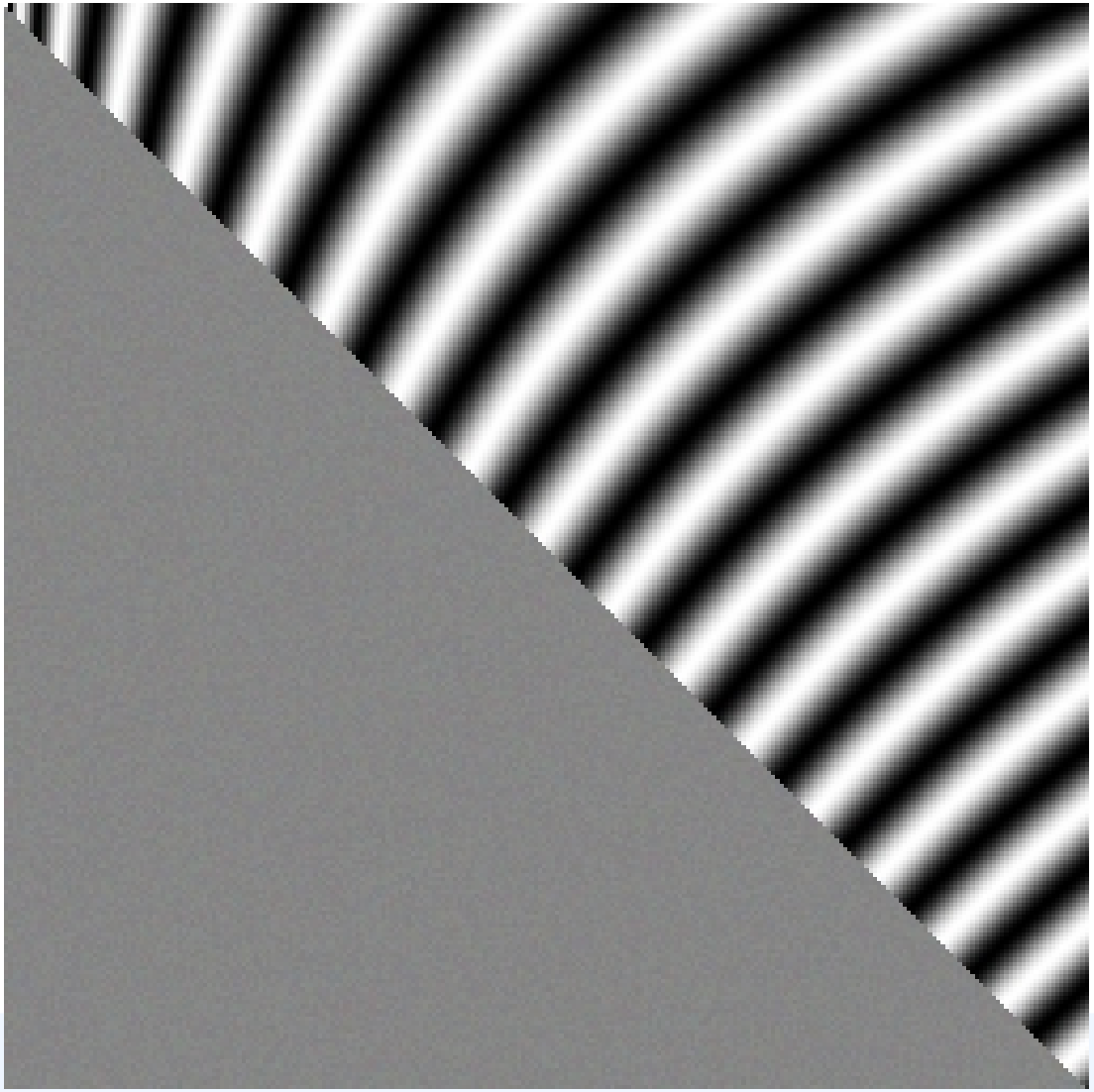
- **Stroke throwing algorithm :** Initialize an empty canvas S and compute the stroke tensors $\mathbf{T}$ from the input image $\mathbf{I}$.

  For each angle γ from a given set $[\gamma_0, \gamma_1, ..., \gamma_N]$, do :

  1. Compute vector field $\mathbf{w}_\gamma = \sqrt{\mathbf{T}} a_\gamma$ with $a_\gamma = (\cos\gamma \quad \sin\gamma)^T$.
  2. Repeat N times :
     1. Pick one random position $\mathbf{X} = (x, y)$ in Ω.
     2. Draw a semi-transparent straight line $\mathbf{X} - L\xi_{(\mathbf{X})} \rightarrow \mathbf{X} + L\xi_{(\mathbf{X})}$ or a streamline of $\mathbf{w}_\gamma$ starting from $\mathbf{X}$ and of length L, on S.

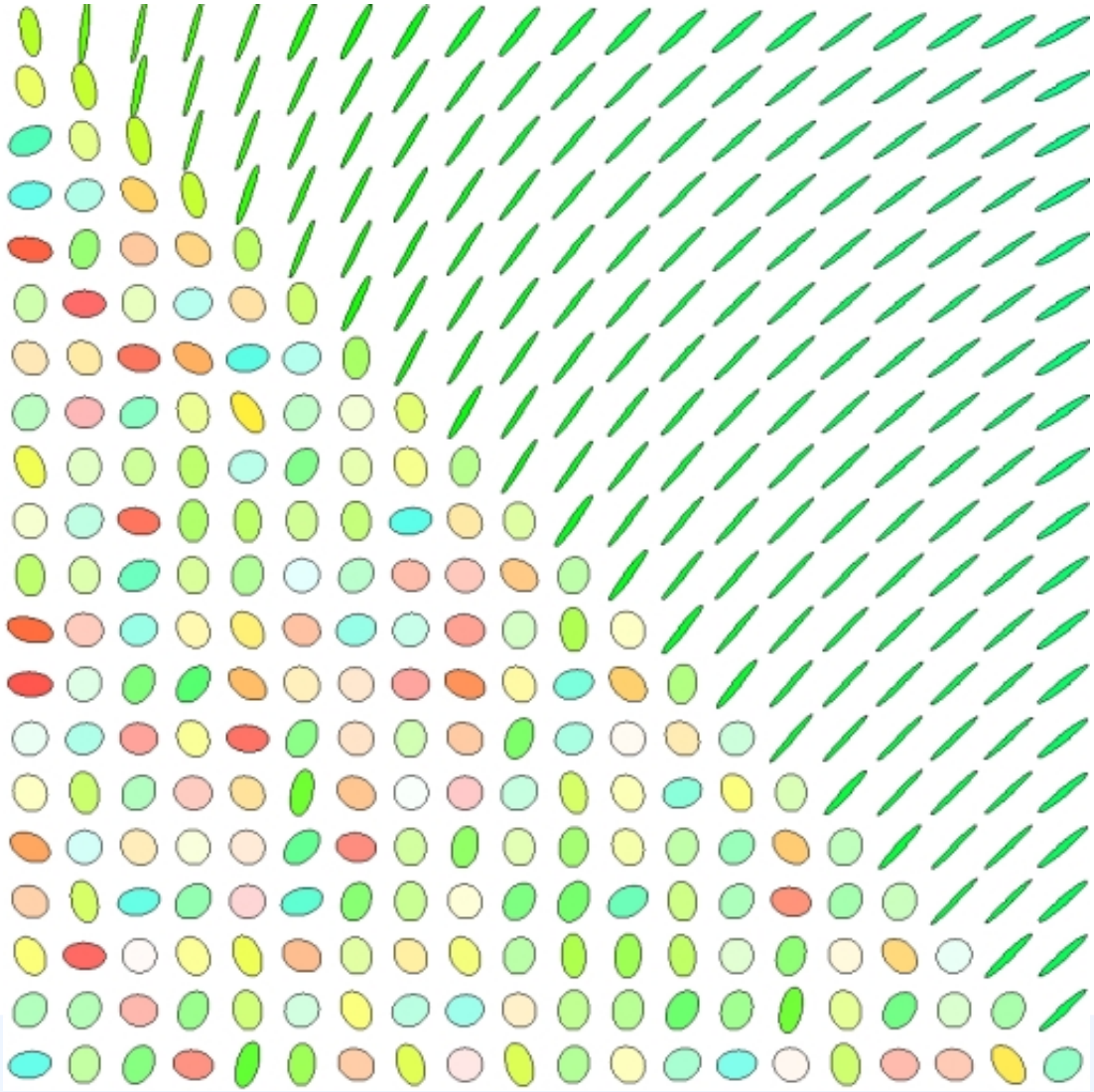
- The stroke tensors are used to **distort** each direction a_γ, according to the geometry of the input image.
- The choosen number of angles γ set the style of the hatches.
- Streamlines allow drawn strokes to be curved.

Illustration of the algorithm



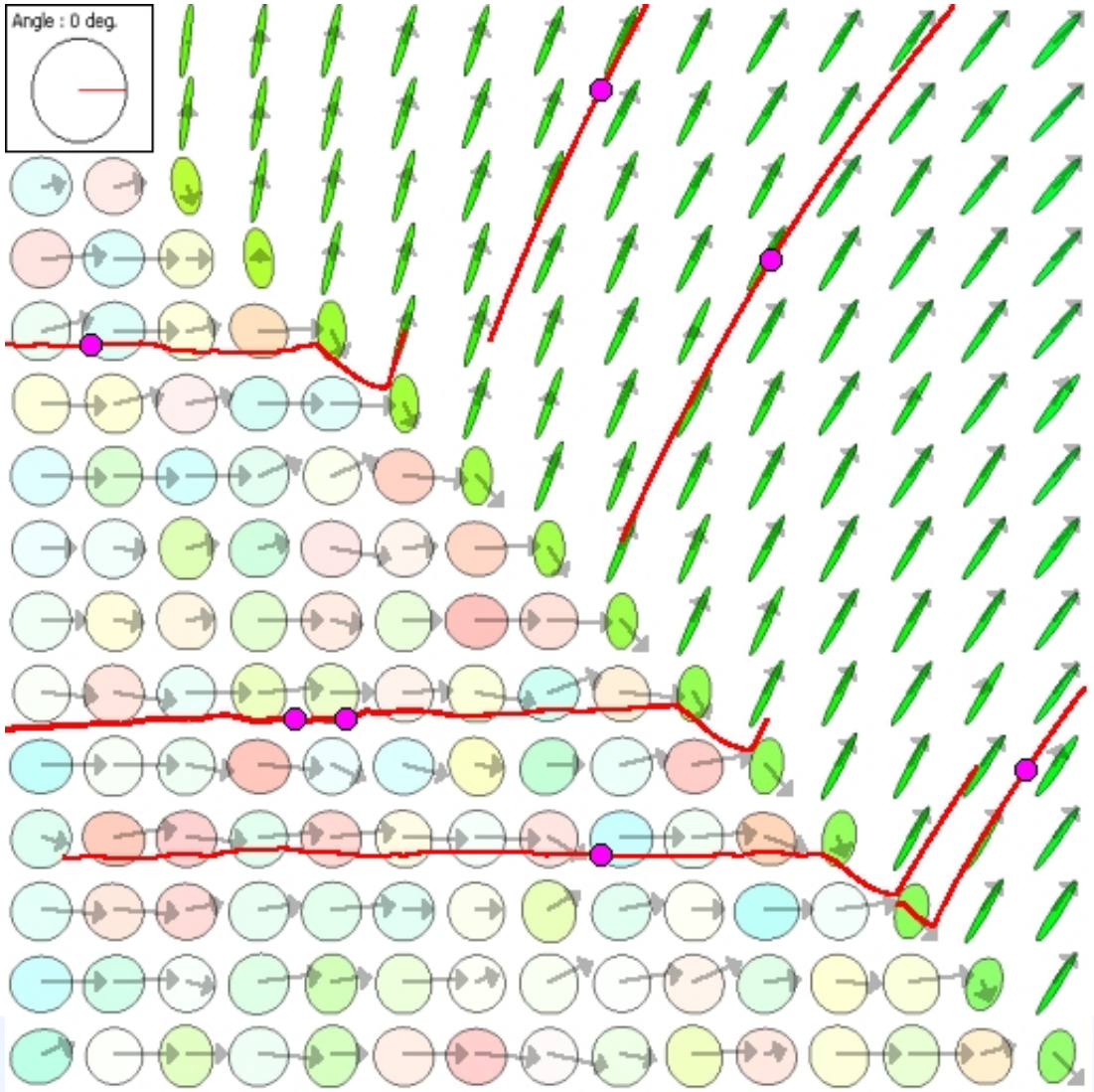
Input image (synthetic).

Illustration of the algorithm



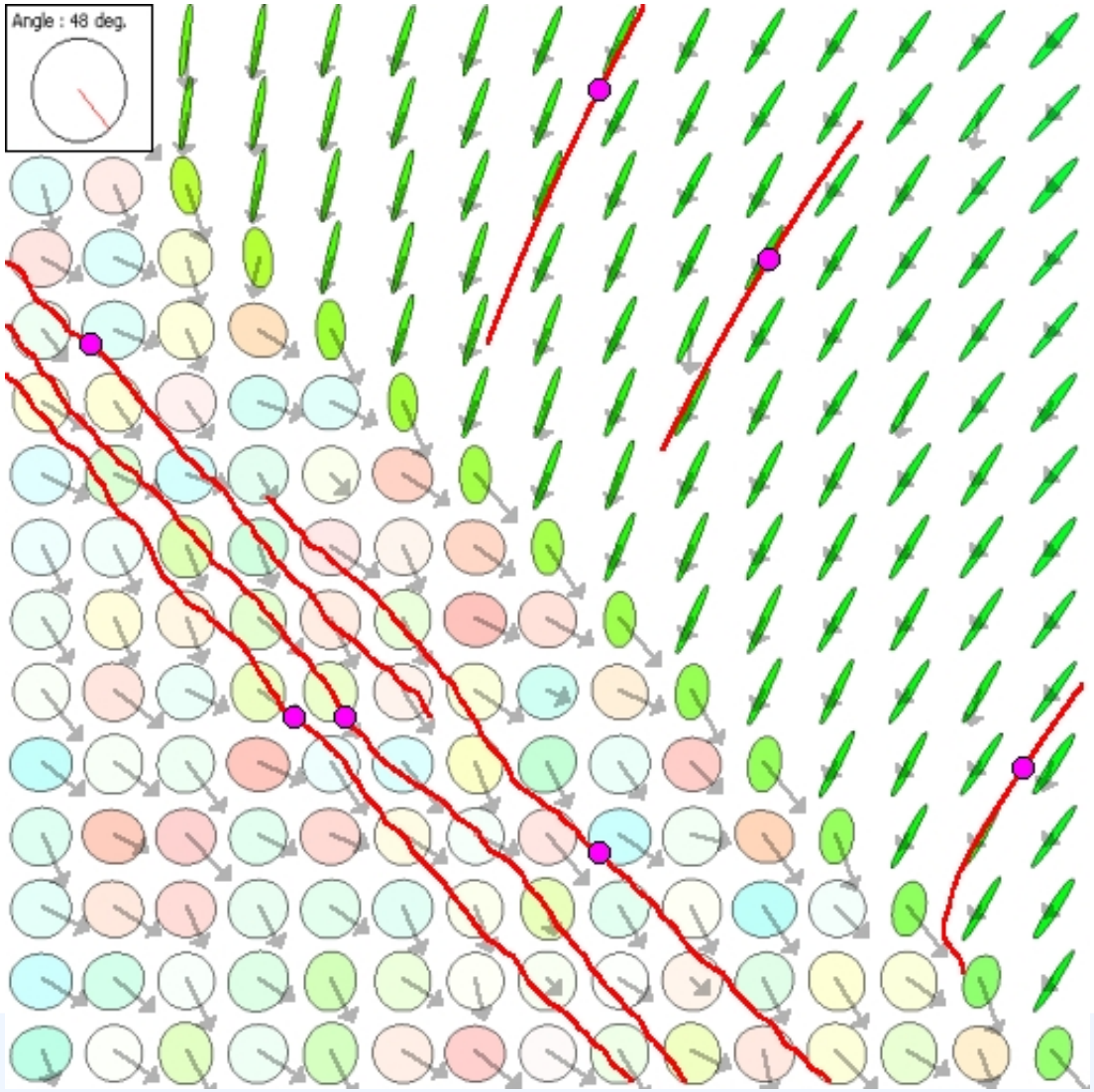
Computed field of stroke tensors.

Illustration of the algorithm



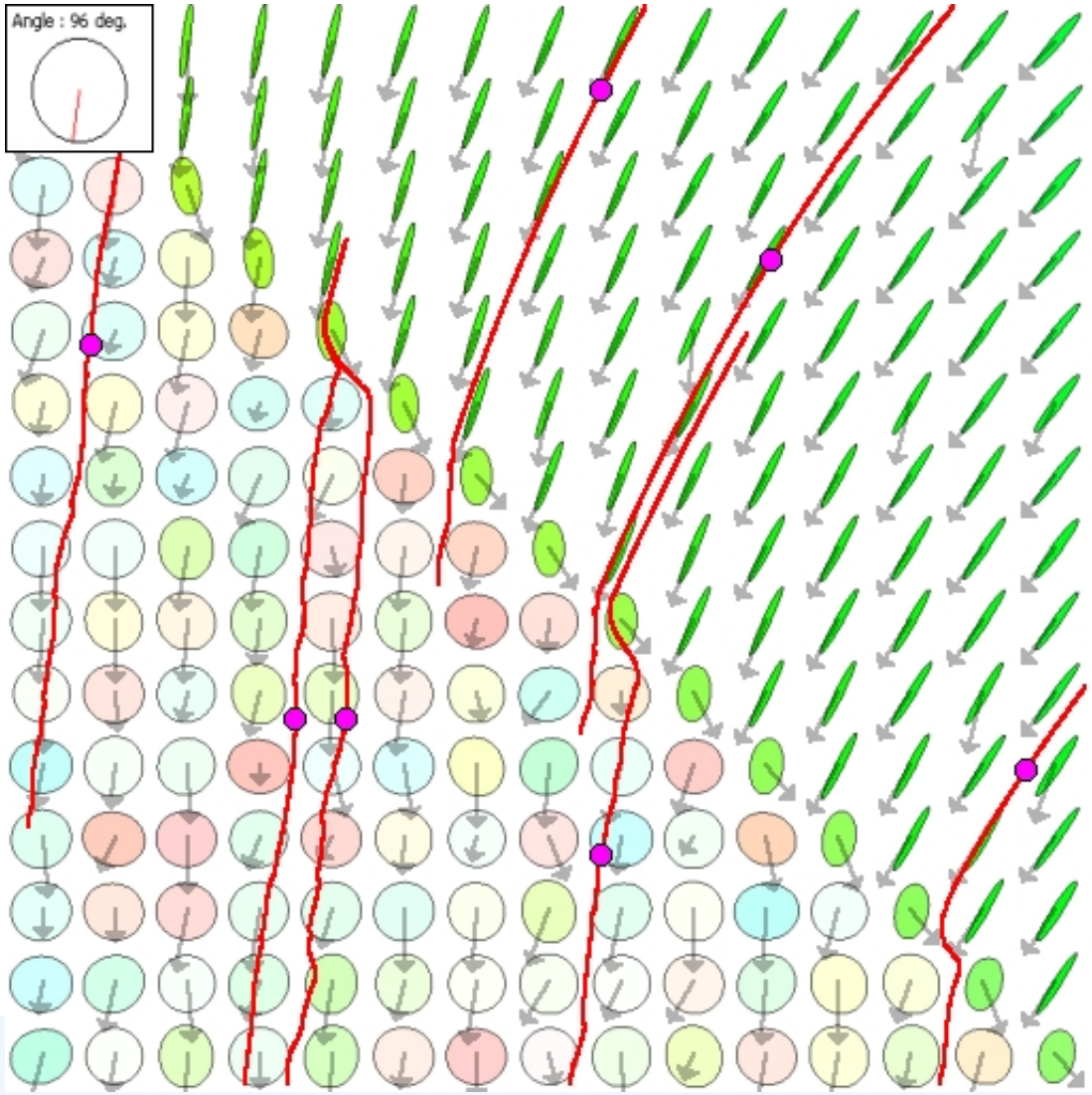
Some streamline strokes drawn for $\gamma = 0^o$.

Illustration of the algorithm



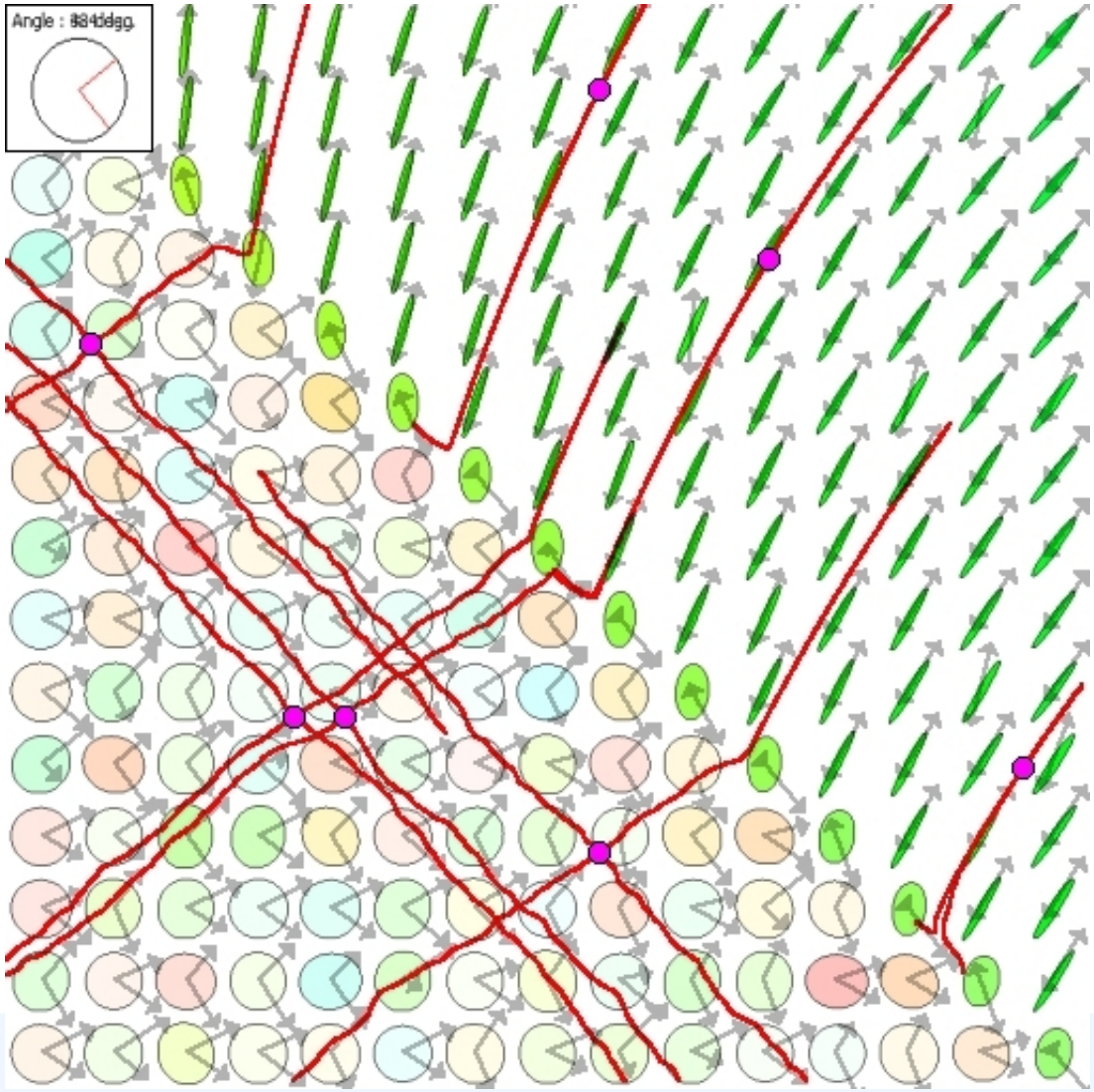
Some streamline strokes drawn for $\gamma = 48^o$.

Illustration of the algorithm



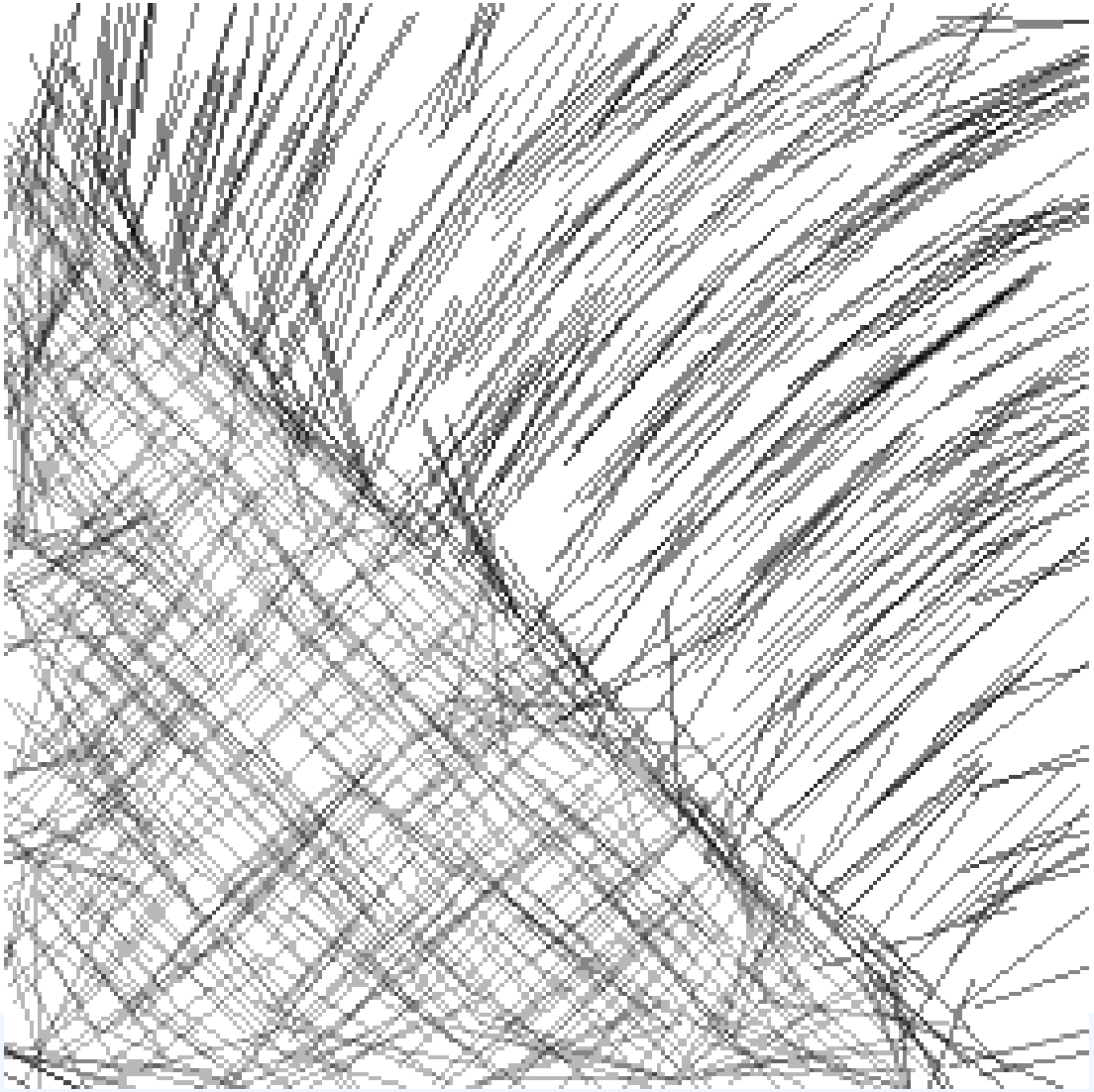
Some streamline strokes drawn for $\gamma = 96^o$.

Illustration of the algorithm



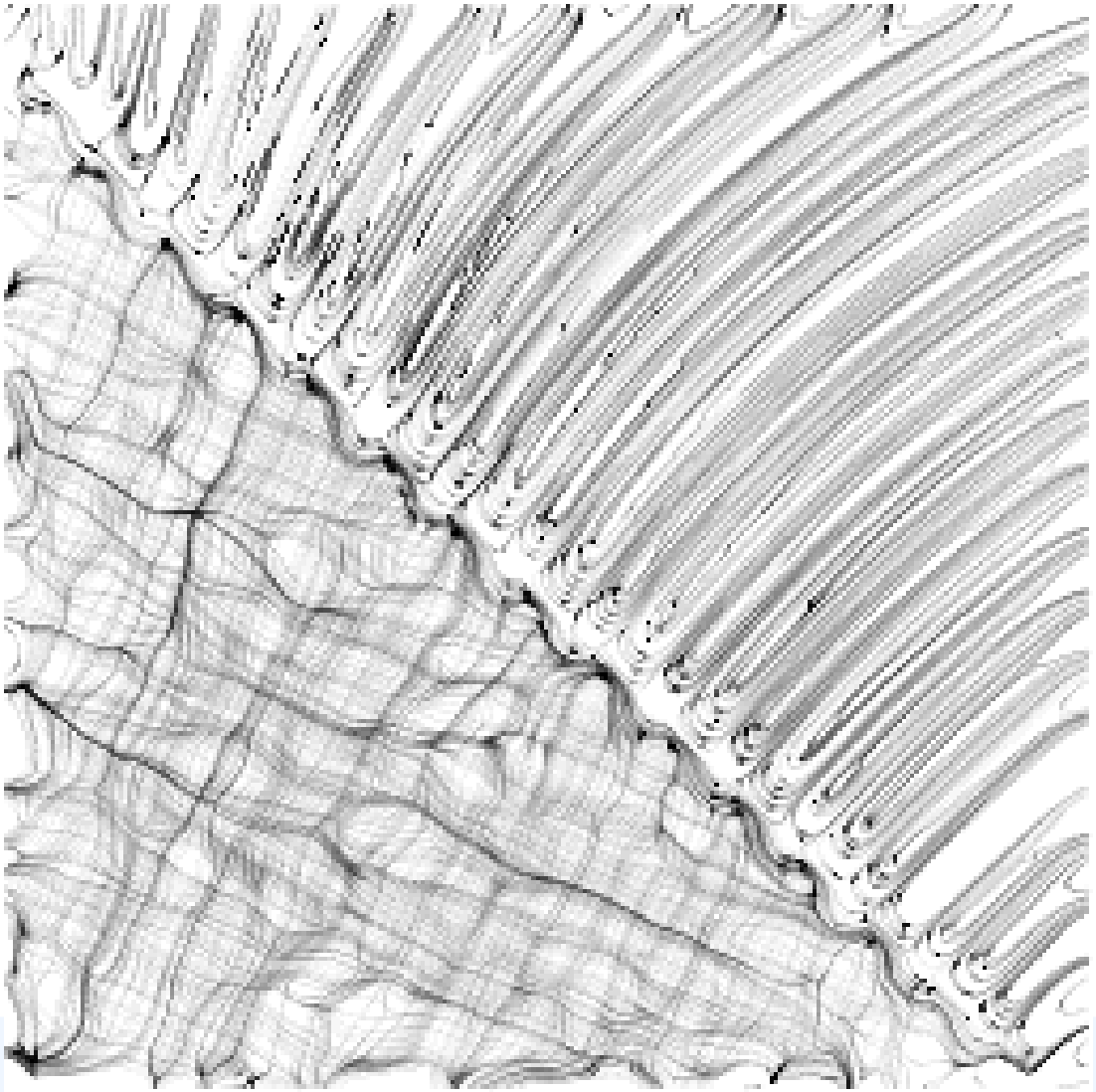
Some streamline strokes drawn for $\gamma = 45^o$ and $\gamma = 135^o$.

Illustration of the algorithm



Final result with straight lines, $\gamma = 45^o$ and $\gamma = 135^o$.

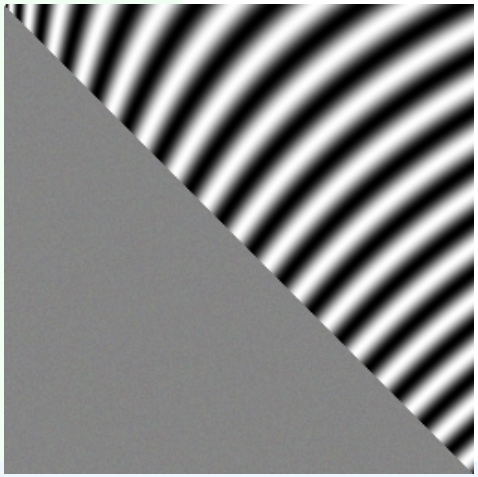
Illustration of the algorithm



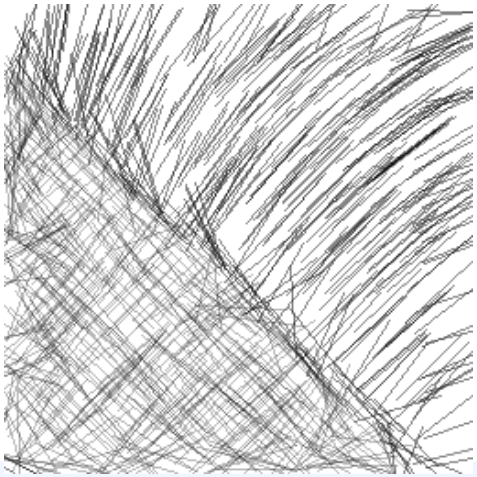
Final result with curved lines, $\gamma = 45^o$ and $\gamma = 135^o$.

Colorization step

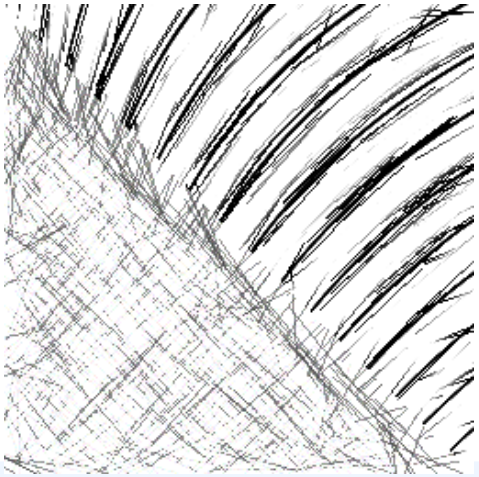
- Colorization combines the B&W sketch canvas and the colors of the input image, to add colors to the generated sketch.
- Here, we tried very simple combinations only, also known as layer blending modes : Soft light, hard light, overlay,....
- All results shown afterwards are using these simple coloring techniques.



Input image



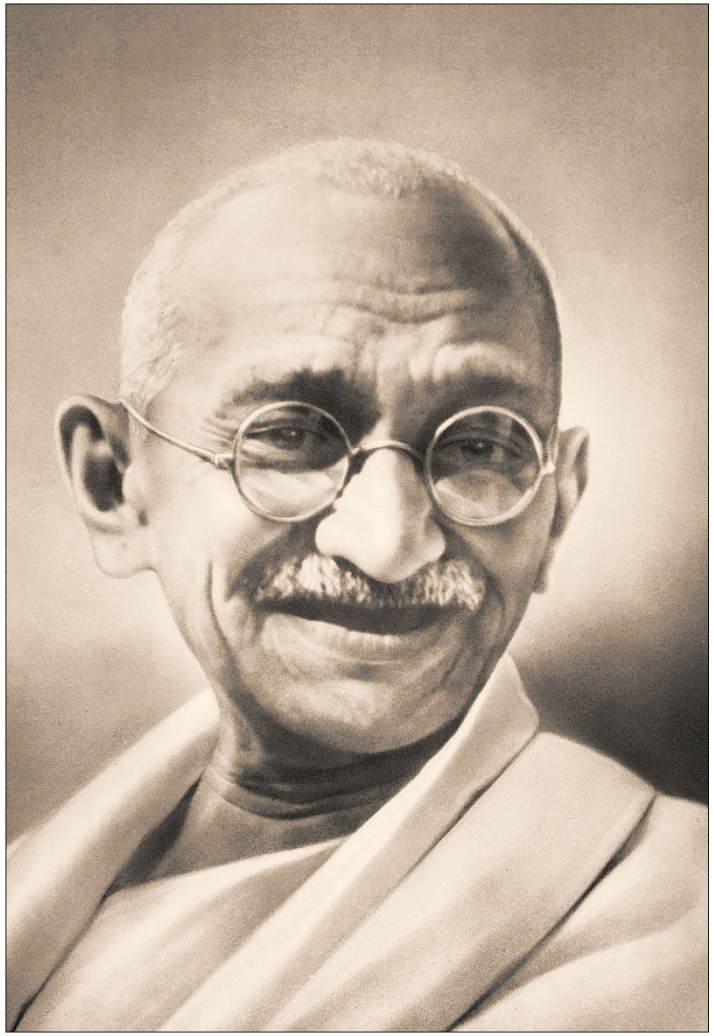
B&W sketch

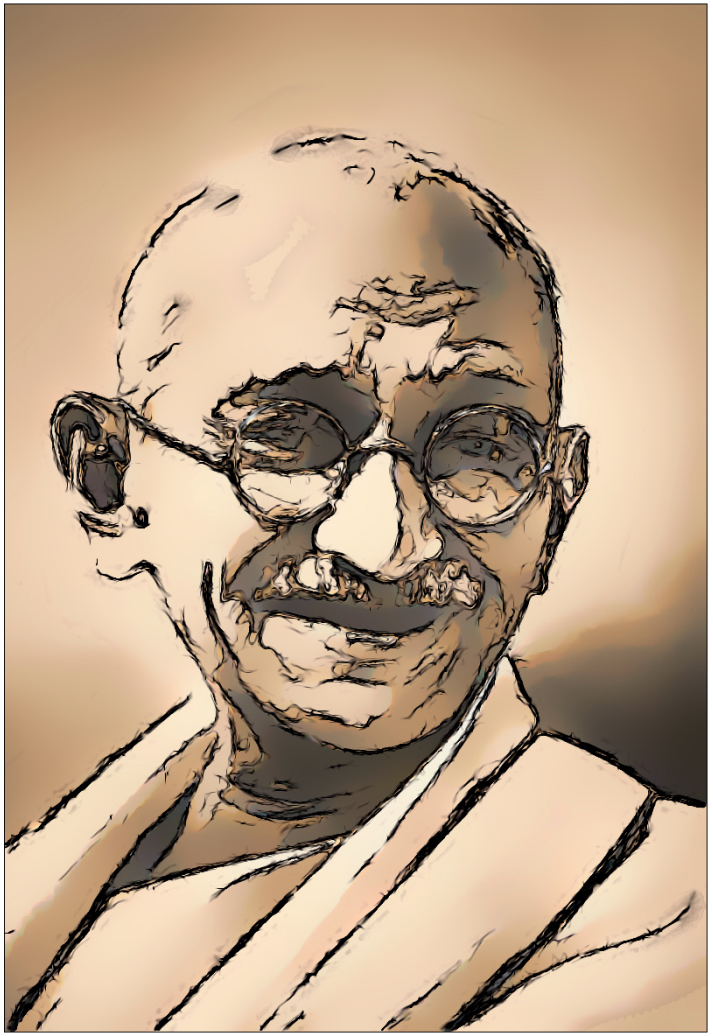


“Colorized” result

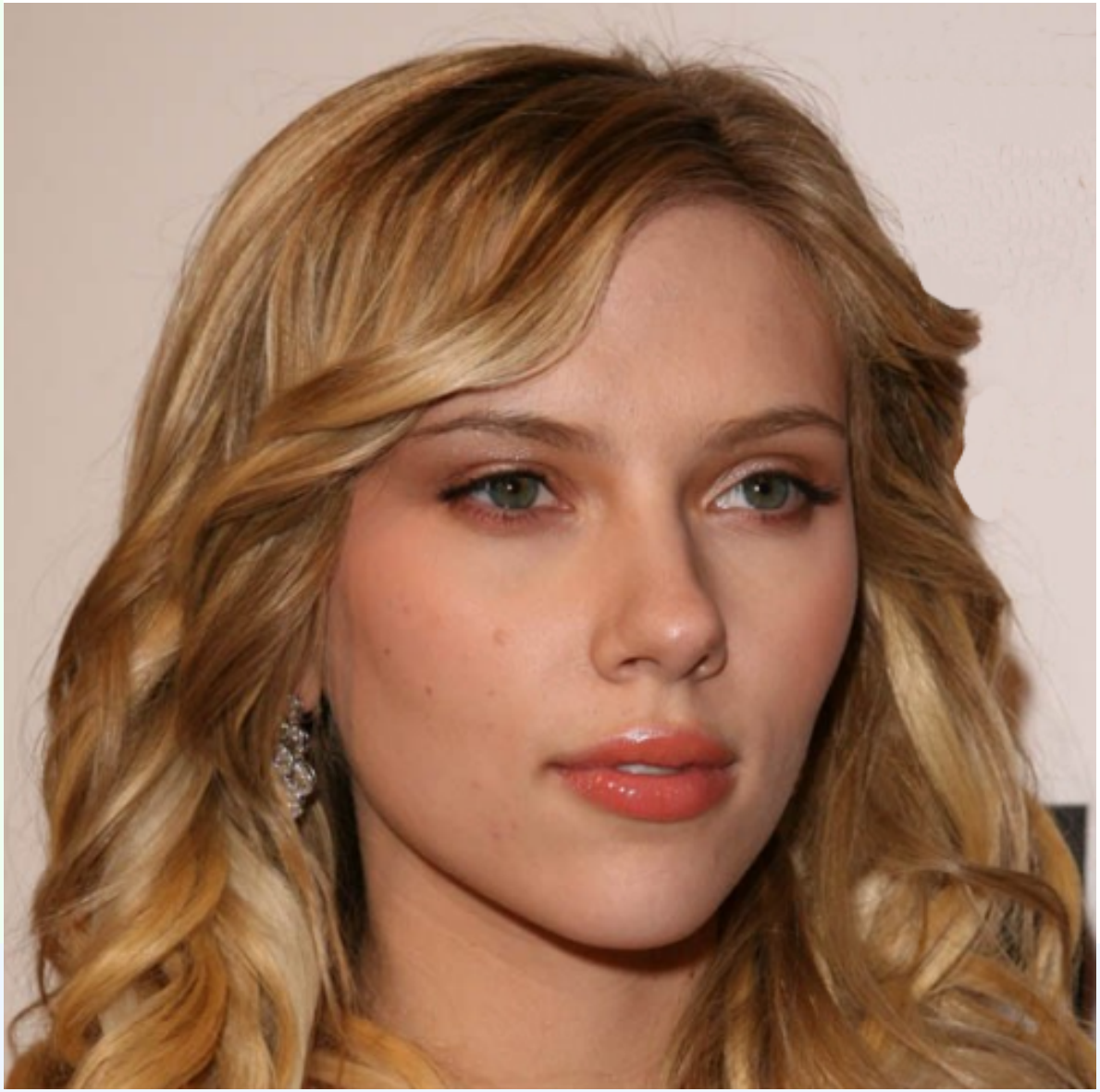
Some results

- Some straightforward applications of the sketch algorithm, with simple colorization steps...

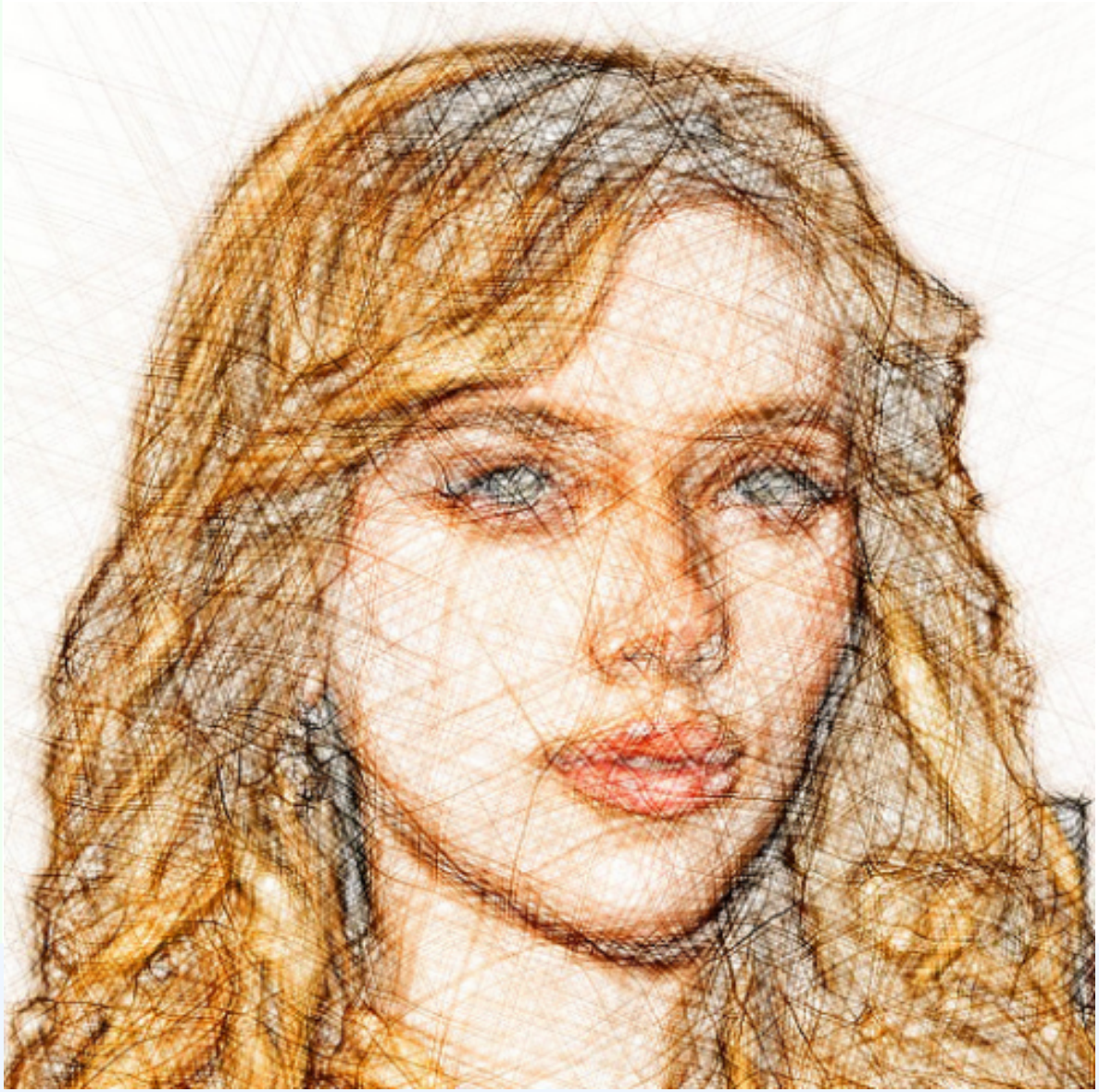




Some results



Some results



Some results

- Can be used to simulate several different drawing steps towards a final painting.

Step 1 (straight strokes)

Some results

Step 2 (curved strokes)

Some results

Step 3 (curved strokes + colorization)

Some results

Step 4 (curved strokes + colorization)

Some results

Final painting (the only input image of the algorithm !)

Some results

(Colors do not have to match the input image)

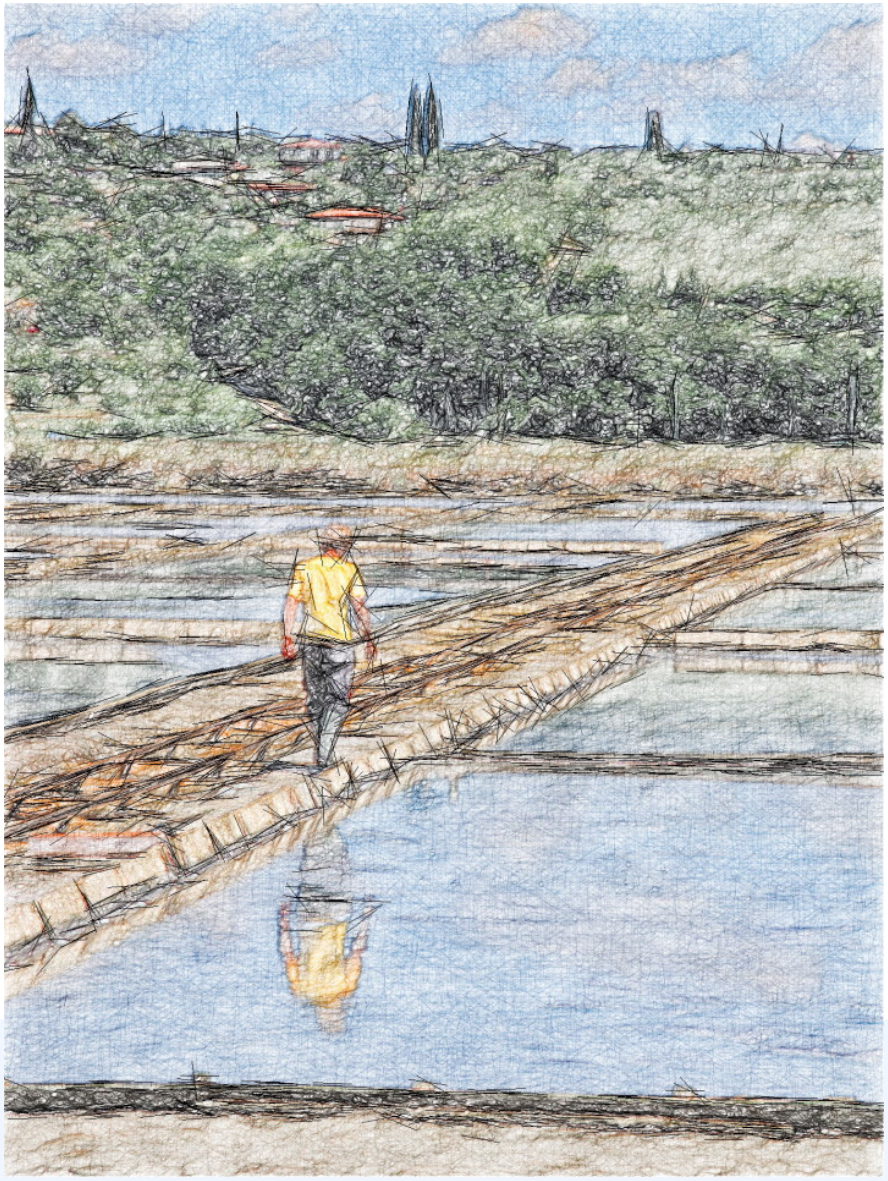
Advanced results

- Several sketch results can be also merged together to render even more complex rendering (courtesy of Tom Keil).

Advanced results

- Several sketch results can be also merged together to render even more complex rendering (courtesy of Tom Keil).

Advanced results



Advanced results

Advanced results

Advanced results

Advanced results

Conclusions and perspectives

- **Conclusions :**
  We have proposed a B&W stroke-based algorithm for simulating the drawing of lines, to convert photographs into sketchs.
  1. It uses tensor fields to deal with the geometry both of the image and the strokes.
  2. It defines a way to draw strokes according to the geometric shape of the tensor field.

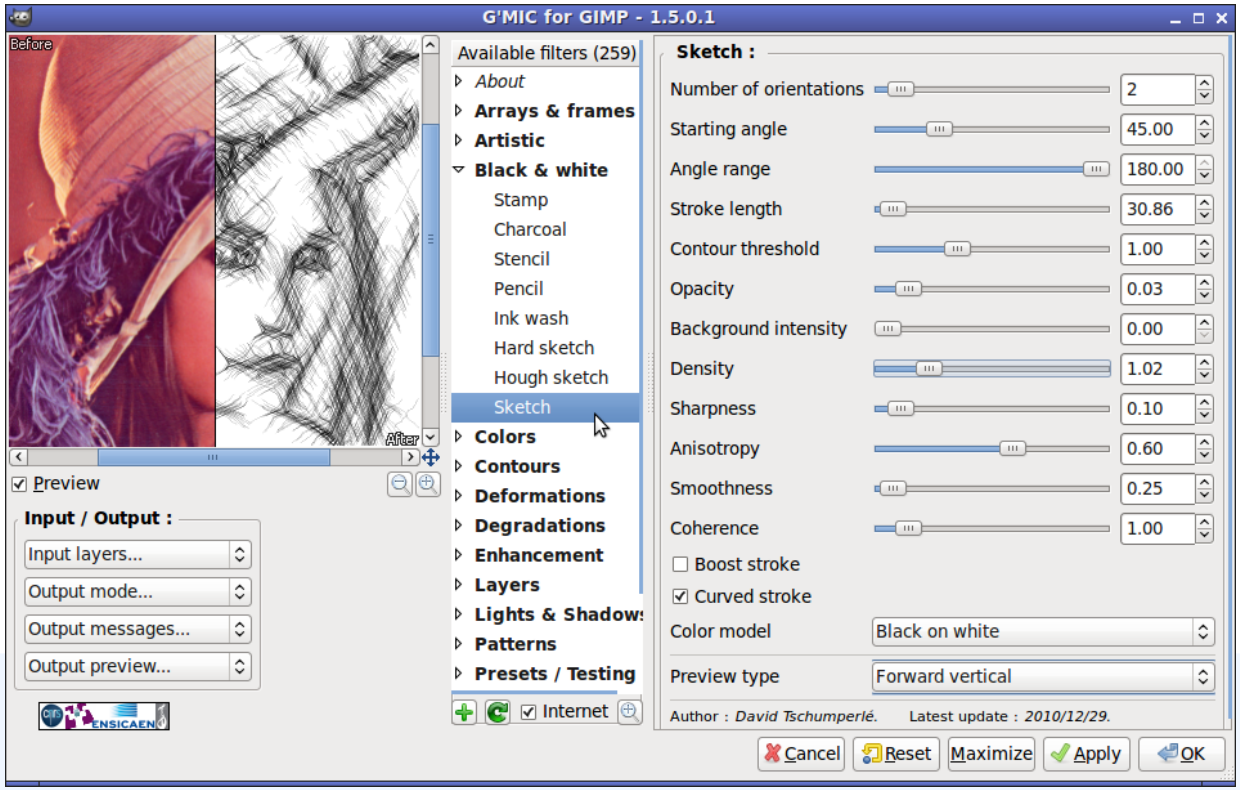
- **Perspectives :**
  1. Find the best ways to model the stroke tensors according to a given artistic style.
  2. Improve the colorization step.
  3. Model the thickness of strokes.
  4. Deal with image sequences (temporal coherence).

Do it yourself !

- The algorithm code is available in **G'MIC**, a powerful framework for image processing, developed at the GREYC/Image lab.

http://gmic.sourceforge.net

- Try our sketch algorithm, with the **G'MIC** plug-in for GIMP.



Questions ?

(Don’t shoot the speaker !)