Stylization of pictures : ESIR third year project

Students : Maxime Bichon, Maxime Rivière
Tutor : Olivier Le Meur

***Abstract*—Within our audiovisual communication lesson, we worked on a project about the stylization of pictures. Also known as Non-photorealistic rendering, the aim of this subject is to process a photograph in order to create a drawing-like with pencils strokes or a paint-like. The main purpose of our work is the research of several way to automatically delivers a stylized abstraction of a picture. We are free to implement any techniques to realize it. The software produced in the end is splitted in two main steps, the edge detection, the most important part of the work, and the rendering part, which draw the picture with different effects.**

***Index Terms*—Stylization, non-photorealistic rendering, edge detection, structural tensor, bilateral filter, color quantification.**

I. Introduction

A. Context

Picture stylization is a very common purpose of research. Many way to do it have already been suggested to get the best edge detection [1] and to render it with originals effect. Of course one of the main advantage of this research field is to provide a new way to produce art item through the digital methods. But an other interest is also to extract and highlight the salient elements of a photograph because it is the main step of most solutions proposed to stylized pictures.

B. Objectives

The main objectives in this problematic involves the abstraction and stylization of the target scene. The two steps in order to get this kind of results are the line detection and the artistic effects. The line detection try to conserve the shapes inside the original picture, then the artistic effects applied to it will render a subjectively beautiful result. The first step, the line detection, have the objective to capture and display the high discontinuities without losing significant features. The artistic effect will mainly introduce a loss of information in order for the result to look like a drawing or a paint, whose have both less data and accuracy than a real photo.

C. State of the art

An overview of different techniques already studied was done by Christian Richardt [2]. An german team also produced a software able to stylized pictures and videos [3]. In another hand, many researchers worked on the edge detection problem using various methods and there are articles summering them [4].

II. Tools (Edge Detection)

In order to compute the edge detection previously quoted, we tried several methods. Two basics methods were the Sobel filter and the structural tensor. We then try to implement a following line filter and a flow-based edge detection.

A. Naive Edge detection

In the naive edge detection we only compute gradients of the picture. We use a Sobel filter to get the edge and put it in a new picture. This way is not a good detection because we can only detect sharp edges, thin edges are not detected. That is why we decided to use the structural tensors which is more efficient. An example of the Sobel filter is shown Figure 1.

Figure 1. Edges with a Sobel filter

B. Structural tensors

The structural tensor is a matrix derived from the gradient. It summarizes the predominant directions of the gradient in a specific region of a pixel. The structural tensor that we used is the following :

$$S_\sigma = (\sum_{i=1}^{3} \bigtriangledown_i I \times \bigtriangledown_i^T I) * G_\sigma \tag{1}$$

Where $\nabla_i I$ is the gradient of the ith component of the picture. We use RGB so we must do it for the 3 components and we obtain this structure tensor :

$$S = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \tag{2}$$

Or :

$$S = \begin{bmatrix} R_x^2 G_x^2 B_x^2 & R_x R_y G_x G_y B_x B_y \\ R_x R_y G_x G_y B_x B_y & R_y^2 G_y^2 B_y^2 \end{bmatrix} \tag{3}$$

With R_x the partial derivative of component R in x direction

To get the edge direction we decided to use eigenvalues and eigenvectors of the matrix S. The eigenvalues obtained are non-negativ and correspond to an orthogonal system o f eigenvectors. We have Sv = λv where v is the eigenvector and λ the eigenvalue defined as :

$$\lambda_{\pm} = \frac{g_{11} + g_{22} + \sqrt{\Delta}}{2} \tag{4}$$

$$\theta\pm = \begin{bmatrix} 2g_{12} \\ g_{22} - g_{11} \pm \sqrt{\Delta} \end{bmatrix} \tag{5}$$

Where $\Delta = (g_{11} - g_{22})^2 + 4g_{12}^2$

Using those values θ and λ we can determine the nature of edges.

- θ_+ is the gradient, so the edge to detect
- θ_- is the direction of the isophote, the direction which is normal to the gradient
- λ_+ is the amplitude of the gradient
- λ_- is the amplitude of the isophote

If we analyze λ values, we obtain different cases :
- both λ are small, then we are in a uniform region, there is no variation
- both λ are great, then there is a big variation in the two direction so we are in a corner
- λ_+ is larger than λ_-, we are on an edge

Finally, to solve the problem of the edge detection, we have the choice between using two factors : the coherence norm C or the DPx. In each case the factor have a value in [0 1].

The coherence norm is thus defined as follows :

$$C = (\frac{\lambda_+ - \lambda_-}{\lambda_+ + \lambda_-})^2 \tag{6}$$

Then we draw edges only if C is higher than a chosen threshold value s. The results depending on various s are shown Figure 2.

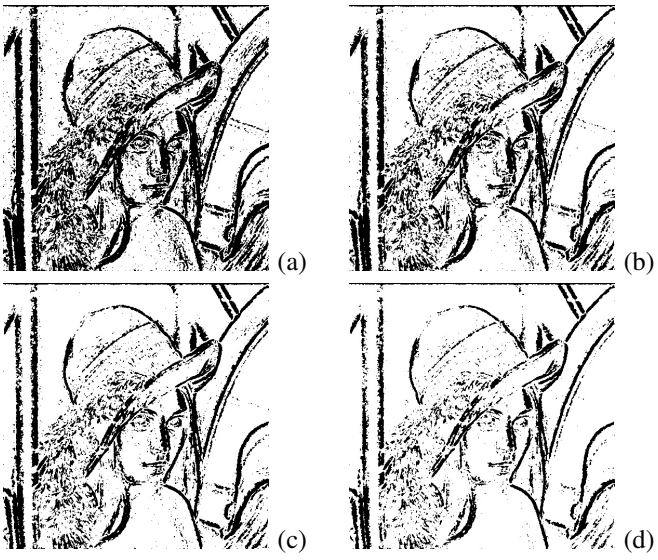


Figure 2. Coherence Norm factor. (a) s = 0,7. (b) s = 0,8. (c) s = 0,85. (d) s = 0,9.

The DPx is defined by the following expression :

$$DPx = \alpha + (1 - \alpha) * e^{\frac{-1}{\eta} / \frac{\lambda_+ - \lambda_-}{\lambda_+ + \lambda_-}} \tag{7}$$

With α and η both user parameters. For all the next parts we used DPx computed with α = 0.1 and η = 20000.

In the same way than for the coherence norm factor, we use a threshold value s. The results depending of s are shown Figure 3.

Figure 3. DPx factor. (a) s = 0,7. (b) s = 0,8. (c) s = 0,9.

C. Following lines

One of the problem noticed with the previous methods is the blanks appearing in straight lines. In order to correct it, we try to follow the tangent direction of the edges already detected and try to reach the next point of the line in order to reconstruct the real gradient. Unfortunately there are a problem with this method, if the tangent direction is not accurate enough, some lines will appear even if there is no edge. This problem explain why we did not use this technique in our final result and is shown Figure 4.

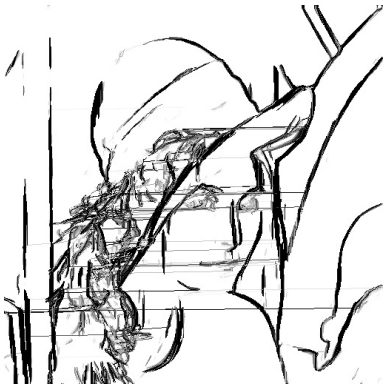


Figure 4. Edges following lines

D. Edge Tangent Flow

Another way to detect the edge is to extract lines through a flow-based abstraction of the picture. In order to implement this, we refers to the approach of [5]. The principle is to use a basic detection of the edges, as a Sobel filter or a Structural Tensor, and to process it in order to obtain the Edge Tangent Flow. From this flow we can compute the new gradient of the picture by extracting lines from the flow. Unfortunately the Edge Tangent Flow was not fully working in the final software, that is why there are no result to show using this method. Nevertheless we believe this track was interesting to follow.

III. Stylization

Stylization is the rendering part of the project. We merge several tools seen previously with different pictures to get an original picture with a comic effect. Most of our work was related to the edge detection, so the different results are mainly different about the shapes preservation, and not the artistic effect applied next.

A. Using original picture

This type of stylization is a step after the basic stylization. We get the original picture and we apply one of the edge detection tools seen previously. We use a loop to draw edges on the original picture. Here the color of each pixels detected as an edge is decreased per laps. So if we have a sharp edge, the color will be very much decreased and we will draw a black edge on the picture. We choose s = 0.85 for the coherence norm factor and s = 0.7 for the DPx. Those choice are based on are subjective feelings. Some examples depending of the edge detection tool used are shown Figure 5.

(a) (b) (c)

Figure 5. Original picture with edges accentuated. (a) Tensor with coherence norm factor. (b) Tensor with DPx factor. (c) Sobel.

B. Non-photorealistic rendering

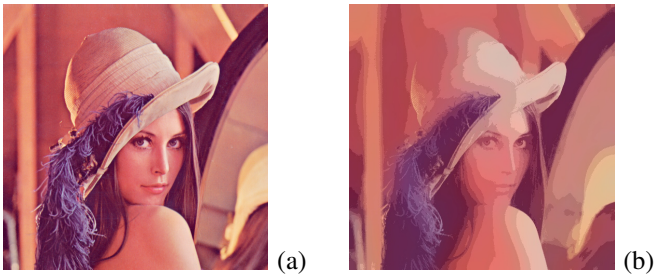
- Bilateral filter
  To preserve a better edges detection and to reduce the noise, we use bilateral filter of OpenCV. With this way we can get better edges and will preserve sharp edges adjusting weights to the adjacent pixels. In the bilateral filter, each pixels has a weight. It is based on a Gaussian distribution of the spatial distance and the color difference between pixels

$$If_{(x)} = \frac{1}{k} \cdot \sum_{y \in Nx} G\sigma_r(\Delta E(x,y)) \cdot G\sigma_s(\|x-y\|) \cdot I_{(y)} \quad (8)$$

$$k = \sum_{y \in Nx} G\sigma_r(\Delta E(x,y)) \cdot G\sigma_s(\|x-y\|) \quad (9)$$

  Here $If_{(x)}$ is the filtered color of the pixel x, for all the neighborhood of this pixel (sum of y) we compute the color differences Gr and the spacial distance Gs. Then we perform a convolution with the color each nearest pixel $I(y)$. The equation is normalized with a constant k which is equation following :
  We decided to use a 60 neighborhood and for an effective smoothing the value 150 for sigma color and sigmaSpace. If sigma values are small, the filter will not have many effect on the original picture but with 150, the picture have more a cartoon-like effect.

- Luminance quantification
  In order to create a cartoon-like effect, we decided to use luminance quantification. While the whole project, we used picture with BGR norm. For the color quantification, we need to convert pictures in the color space CIE-Lab. This space uses every perceivable colors and each attributes is independent so we can easily use the attribute needed which is L* bounded between 0 and 100. This component is the lightness of the picture, L* = 0 represents the darkest black whereas L* = 100 represents the brightest white. After conversion from BGR to CIE-Lab, we put all of the L* values in a vector which will be used in the K-mean. K-mean quantification is the next step, luminance values are splitted in 8 clusters [6]. We finally match each pixel of the picture to its quantized value to obtain the result shown in Figure 6.



(a) (b)

Figure 6. Colors Quantification. (a) Original Picture. (b) Picture after quantification

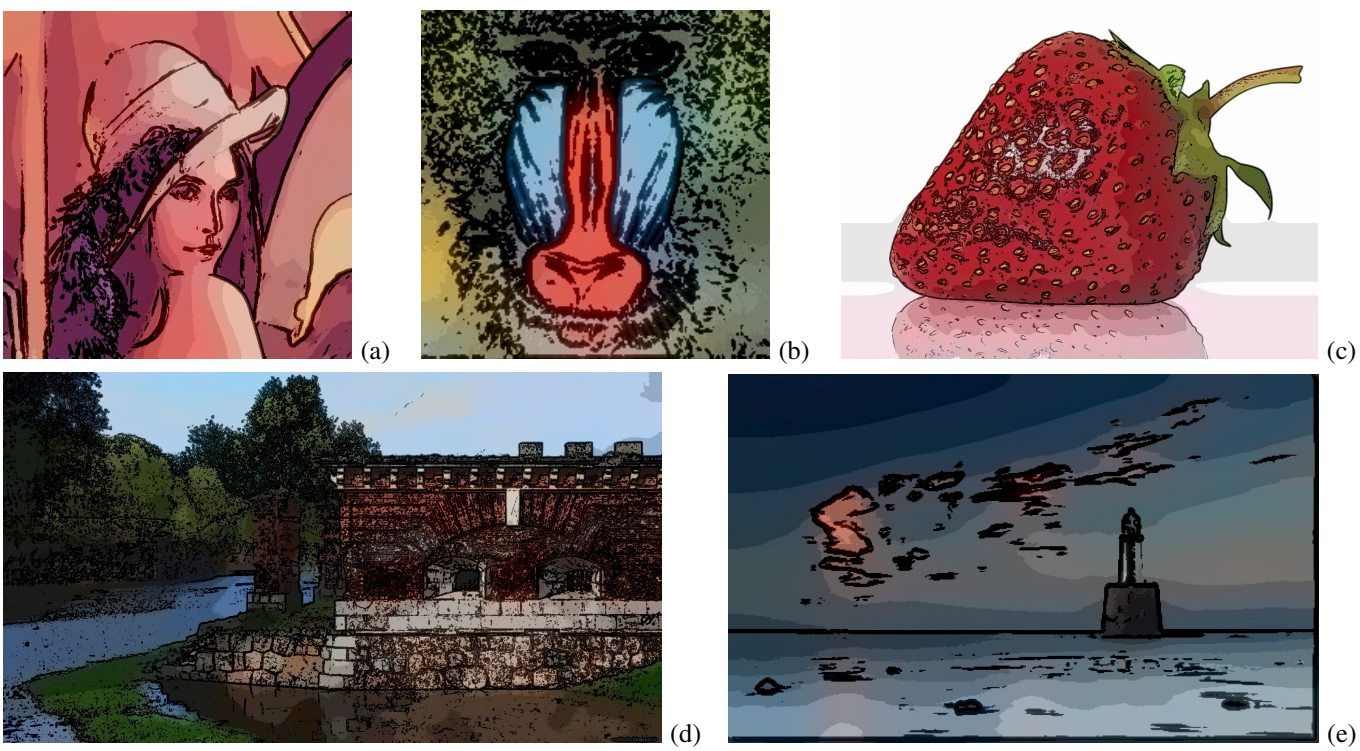


Figure 7. Stylized pictures. (a) Lena. (b) Baboon. (c) Strawberry. (d) Wall. (e) Lighthouse.

- Merging
  For the comic effect we need to merge all of the previous techniques. Firstly we compute the bilateral filter before the edge detection, it will smooth the original picture to get better edges. Secondly we apply the color quantification previously explained, with this representation we get a quite good representation of a comic, but we need pencil stroke effect. To realize it we simply apply the tensor structure to get the edge detection. We finally draw edges on the quantized picture to get the final picture.

IV. Results

After all we used the bilateral filter and quantification of colors using a K-means algorithm for the artistic effect. The structural tensor with a DPx factor (s = 0.7) was the most accurate and non-noisy compromise, that is why we choose it for the edge detection. Some results are shown Figure 7. We finally observe that the result are mostly paint-like, especially for the picture $(e)Lighthouse$. There are also some pictures where the results does not look good because of noisy edge detection like $(b)Baboon$ and $(d)Wall$.

V. Conclusion

We presented in this paper our work about non-photorealistic rendering for our lesson of audiovisual-communication. As we said to the beginning of this paper we used different way to detect edges. We obtained many results different form each other, even if we could, with more time, increase the efficiency of the edge detection. Moreover, we decided to use the original picture to get a better stylized picture. We are glad to have had the opportunity to work on this stylization project, the discover of the different way to create a cartoon-like picture from a photograph was an interesting subject. It allowed us to find out mathematical tools like structural tensors of flow based images. This was a very interesting project with many way to do it.

References

[1] Marr Hildreth, "Approximation to human edge detection," .
[2] Christian Richardt, "Non-photorealistic rendering," .
[3] Jan Eric Kyprianidis and Jürgen Döllne, "Image abstraction by structure adaptive filtering," .
[4] Raman Maini Dr. Himanshu Aggarwal, "Study and comparison of various image edge detection techniques," .
[5] Seungyong Lee Henry Kang and Charles K. Chui, "Flow-based image abstraction," .
[6] Arthur and Vassilvitskii, "Center initialization," .