Edge detection project
MIF17 - image analysis course

Judith Millet & Emie Lafourcade

Department of Computer Science
University Claude Bernard
43 Bd du 11 Novembre 1918, 69100 Villeurbanne
{judith.millet & emie.lafourcade}@etu.univ-lyon1.fr

Abstract : **This project aims to develop edge detection on any kind of image. It will be possible to choose the method and the thresholding.**

Keywords - Edge detection, thresholding, image.

I. INTRODUCTION

As part of this image analysis project, we had to implement an edge detection method, using the basic differential operators seen in class.

Edge detection is the process used to identify the points of a matrix image which correspond to a sudden change in the intensity. In a grayscale image, a sudden change in value characterizes an edge. The purpose of the operation is to transform this image into another of the same dimensions in which the contours appear by convention in white on a black background.

This project was made in C++ language with the OpenCV library where we used some basic functions.

II. APPROACH

A. *Gradient vectors*

In our approach, an edge is an abrupt transition of the gray levels of an image. As seen in class, the gradient is a directional operator and its analysis will allow us to highlight these contours. Indeed, the amplitude of this vector indicates a more or less large module as well as a direction which will be normal to the contour.

Thus, we calculate the gradient vector for each pixel in the image, except the borders, to determine if it is an outline or not.

We will implement two methods : a bidirectional and a multidirectional method. The bidirectional one is used by default. To use the multidirectional use the *-multi* option when executing the program.

1) The bidirectional method

This method requires calculating the bidirectional gradient from a mask applied along 2 axes. To calculate the norm of the gradient at a point, we perform the convolution product between the neighborhood of this point and the mask along the horizontal axis (called Gx). The convolution is done on the 8 neighbors of the point.

Then the same calculation is performed with the mask along the vertical axis (called Gy). The final gradient norm is sqrt(Gx^2 + Gy^2). We normalize the obtained values to be sure they are between 0 and 255.

After that, we determine the slope at this point with the formula arctan(Gy/Gx) if Gx is different from 0, Pi/2 otherwise.

2) The multidirectional method

This method is almost similar to the previous method, but we do not calculate only two values for the gradient.

At each point, we perform the convolution product for a mask along four potential axes (horizontal, vertical, the two diagonals). In this case, the result is determined by selecting the maximum of the absolute values for the four directions. As in the bidirectional method, this result is normalized between 0 and 255. The slope is thus decided according to the direction of the selected mask only :

- if the maximum comes from the horizontal component, the gradient argument is 0;
- if the maximum comes from the vertical component, the gradient argument is Pi/2;
- if the maximum comes from the first diagonal component, the gradient argument is Pi/4;
- if the maximum comes from the second diagonal component, the gradient argument is 3 * Pi/4;

The gradient and slope of each pixel is stocked a pair. We then obtain a vector of pairs for the whole image.

3) The masks

The 3 masks we used are Prewitt's, Sobel's and Kirsch's.

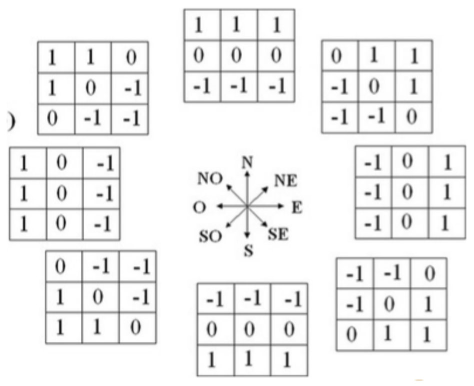

Figure 1 : Mask Prewitt along the 8 axis.

The default one is Prewitt but we can choose which one to use when executing the program with the *-mask* option. *-mask 0* is Prewitt, *-mask 1* is Sobel and *-mask 2* Kirsch.

B. *Thresholding*

Each thresholding method we will use the principle of separating the pixels above the threshold from those below. The result is then limited to a gradient intensity. We will propose three thresholding methods operating according to this principle : a fixed, a global and a hysteresis threshold.

1) Fixed threshold

The first approach consists in applying a fixed threshold given as a parameter. It is editable by the user. If the gradient modulus of each pixel is above this threshold, we assign it the intensity 255, 0 otherwise. So this will give us a binary image.

2) Global threshold

This approach is automatic. It consists in calculating the threshold by taking the average of all the modules of the gradient for each pixel of the image. Thus, we call the fixed threshold method with the mean as a parameter. In this case, the complexity is greater because 2 image scans are performed.

3) Hysteresis threshold

The hysteresis method consists in keeping pixels with a gradient modulus superior to a certain high threshold but also pixels with a gradient modulus that are superior to a low threshold and has at least one neighbor with a gradient modulus superior to the high threshold.
If a fixed threshold is determined we use it as the high threshold and the low threshold is 30% of the high one. If no threshold is indicated, the low threshold is the mean of all the modules of the gradient for each pixel of the image and the high threshold is the mean of all the modules of the gradient for every pixel with a gradient superior to the low threshold.

Foreach threshold method, we will propose a color result by giving a color per gradient orientation. The global thresholding method is used by default. Use the *-threshold {VALUE}* option to use the static thresholding method, and *-hysteresis* for the hysteresis one. These two are cumulable.

C. *Refining*

To make fine outlines we implemented a refining algorithm. It will reduce the outline. This method consists in traversing the image in search of a contour and looking in the neighborhood defined by the direction of the gradient, if its neighbors have a higher gradient norm. If so, we can remove that point because it is not a local maximum for that direction of the gradient. To use this, use the *-refining* option when executing the program.

III. RESULTS

We tested our different approaches on several images. The different masks apply give a similar result with the global threshold when we use the bidirectional method.

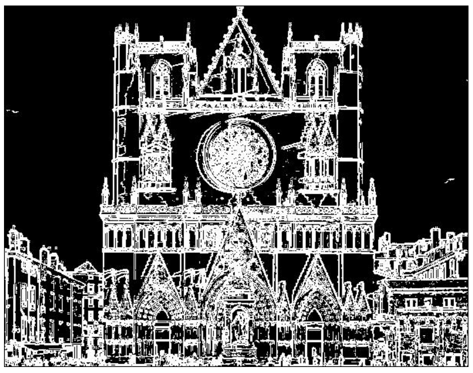

Figure 2 : Result with mask Prewitt in bidirectional method with a global threshold at 24.

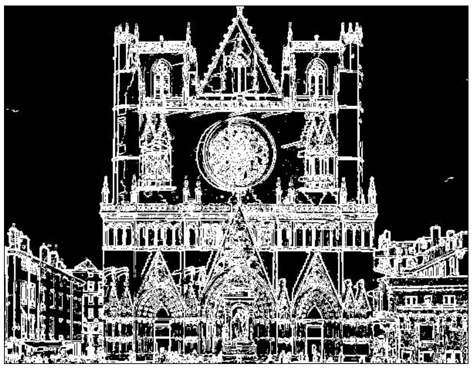

Figure 3 : Result with mask Sobel in bidirectional method with a global threshold at 25.

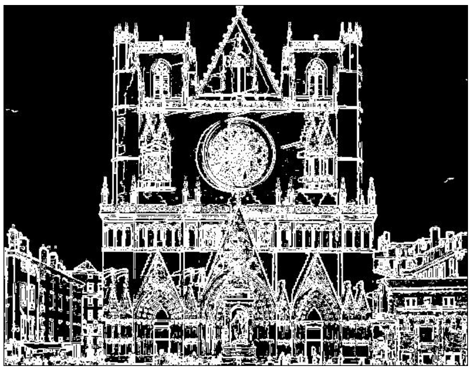

Figure 4 : Result with mask Kirsch in bidirectional method with a global threshold at 21.

There is no best mask. It depends on the entry image.

Moreover, the bidirectional and multidirectional methods produce results that are quite alike. The advantage of the bidirectional method is that only 2 filters are needed to calculate the gradient at one point. However, it may be more sensitive to noise than the multidirectional method. Therefore, we can see more details with the multidirectional method.

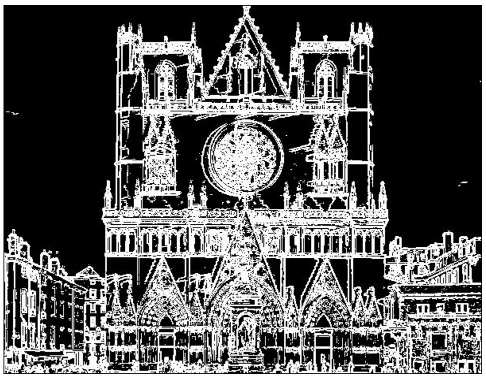

Figure 5 : Result with mask Prewitt in multidirectional method with a global threshold at 24.

The different thresholding processus gave similar results.

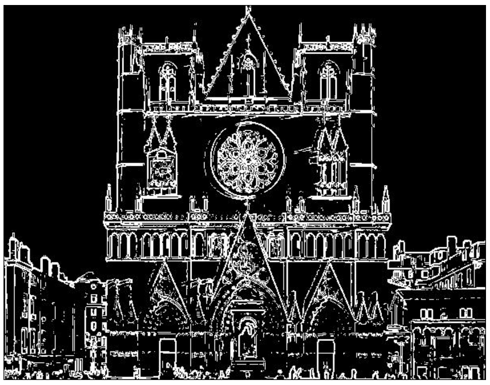

Figure 6 : Result with mask Prewitt in multidirectional method with a fixed threshold at 50.

Depending on the fixed threshold, we obtain a different result. Indeed, several tests are necessary to find an interesting threshold for this case. As we can see cf. Figure 6, fixing the threshold at 50 gives us a less detailed result.

Then, the hysteresis threshold prevents forming too many holes in the outlines and avoids local noise.

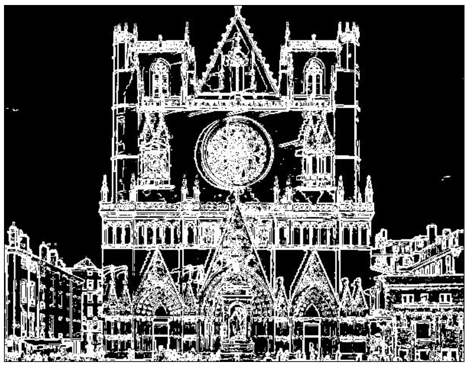

Figure 7 : Result with mask Prewitt in multidirectional method with an hysteresis threshold based on the average.

The hysteresis thresholding has a low threshold at 24 and a high threshold at 60 when it is used with the average method.

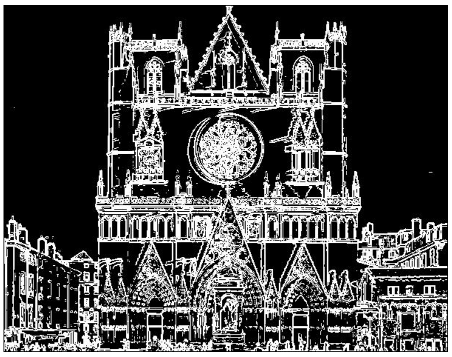

Figure 8 : Result with mask Prewitt in multidirectional method with an hysteresis threshold based on a fixed threshold at 100.

In that case, the hysteresis thresholding has a low threshold at 30 and a high threshold at 100. It produces a more detailed result.

Using various fixed thresholding values is the best way to adjust the outlines. Plus, it's the method with the best complexity (browsing the picture only one time).

The main difference is the coloration. It is better with the multidirectional approach.

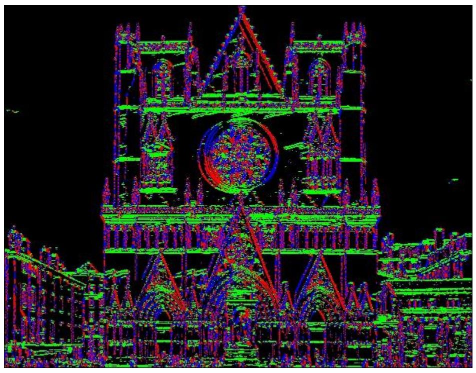

Figure 9 : Result with mask Prewitt in bidirectional method with a global threshold and colored.

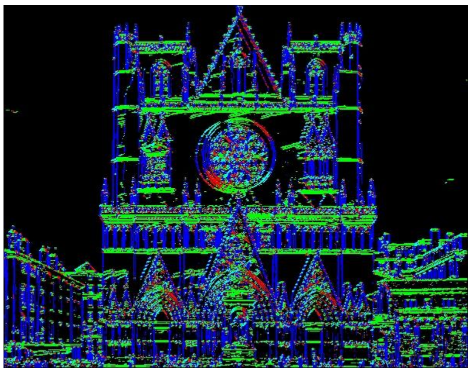
Figure 10 : Result with mask Prewitt in bidirectional method with a global threshold and colored.

Concerning the refining, applying it produces a result with an outline of 1 pixel only.

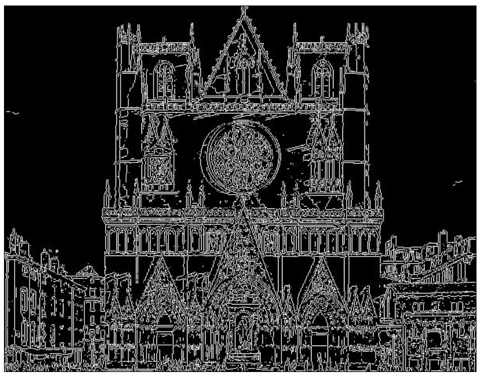
Figure 11 : Result with mask Prewitt in multidirectional method with a global threshold and refining.

This step allows us to obtain a satisfying image, which has relatively little noise with fairly precise outlines.

IV. CONCLUSION

We have developed an edge detection method that has proven its efficiency on the various tests carried out. We tried three different masks, two convolution methods (bidirectional and multidirectional) and three different thresholding. Some imperfections remain, but these operators have already demonstrated their effectiveness.

Suggested improvements would be to make a better bidirectional coloration and refining. Indeed, we use only 3 colors in the bidirectional coloration. We could adjust the color to the calculated slope of each pixel.

Another prospect for improvement could be the implementation of outline closure consisting of advancing little by little trying to select at each step the best pixel to continue the outline.