Edge detections methods

1st Mikel-ange Barros
computer science department
University Claude Bernard Lyon 1
Lyon, France
mikel-ange.barros@etu.univ-lyon1.fr

2nd Zacharia Beddalia
computer science department
University Claude Bernard Lyon 1
Lyon, France
zacharia.beddalia@etu.univ-lyon1.fr

3rd Thomas Scouflaire
computer science department
University Claude Bernard Lyon 1
Lyon, France
thomas.scouflaire@etu.univ-lyon1.fr

***Abstract*—Image analysis is a field based on a set of tools made for object recognition in images. One of the few areas of expertise of image analysis is edge detection. It counts as one of the most important and most used methods. This document presents our edge detection methods.**

***Index Terms*—image analysis, edge detection**

I. Introduction

Image analysis is a branch of computer science with a growing number of different use, will it be in the field of video games, medecine or data protection. All of these technologies use one or more image analysis method, especially edge detection methods. Implementing different methods of edge detection for each of these applications is an important part of the research work. Our study goes in that direction, with the purpose of acquiring a set of convincing edge detection methods working on most of the use cases.

II. Edge detection for an image

Edge detection is a set of segmentation methods based on corner detection and splitting line. It requires several criterias for it to be considered efficient, especially the detection speed and the precision of the found edges. As said before, edge detection is a set of methods of image analysis and therefore, it is important to state what is image segmentation to understand it, since the are linked. Image segmentation is the first level of image analysis and the most important. Indeed, image segmentation methods will allow the extraction of the main characteristics of the picture, such as the color, the edges, etc. Depending on its uses and how it operates, a segmentation methods will be catagorized as one of four area:

- Edge detection segementation : This type of segmentation tries to detect the edges of objects in the picture with the purpose of identifying them. One of the most notable image segmentation using edge detection is the Canny filter.
- Thresholding : This type of segmentation will detect the colors of the objects in the image to identify them. Using this method, we try to highlight objects having a different color spectrum than the rest of the objects in the picture. Adaptative thresholding is one of the most used thresholding methods.
- Cluster-based image segmentation : this methods consist of defining clusters of pixels having similar characteristics. The diffences between this method and the region segemation one is that a cluster, unlike a region, can be made of pixels that are not connected. The K-means algorithm is an implementation of this method.
- Region growing segmentation : This method consist on finding region of neighbouring pixels having the same characteristics.

In this report, we will study the edge detection method. We will explain our implementation and dissect its results depending of the defined parameters, such as the filtering methods and threshold values.

A. Gradient-based edge detection principles

This edge detection methods is based on the convolution of a filter with the image. This convolution returns the value of the gradient of each pixels in the picture and the direction of that gradient. Using this gradient, we can create a greyscaled image showing the color shifts around a pixel. By finding the right threshold value, we can highlight the gradient values characteristic of edges.

This algorithm is usually applied on greyscaled images.

III. Our algorithm

A. Data structures

To manage to do the edge detection, we used two data structures :

- A structure matrix that contains a 3x3 matrix that will allow the operations between the filter and the image.
- A structure result that contains the set of the connected component of the image and the resulting image.

B. Preprocessing

Before any operation on the image, we apply a bilateral filter on it to reduce potential noise without altering the sharpness of the edges.

C. Filtering

Our algorithm is based on the method presented above and implements it for a wide set of filters and thresholding methods. This way, we can integrate the filter we want into this image. For each of these filters we have the possibility to apply it in three different ways:

- Unidirectional: the filter is applied in one direction and the gradient is calculated from this simple filter.

- Bidirectional: The filter is rotated 90 degrees and the gradient is calculated using both the rotated and the base filter. The general gradient of the pixel is determined by the square root of the gradients squared. And the direction is calculated as the arctangent of the general gradient divided by the gradient according to the second filter.
- Multidirectional: The filter is rotated 45, 90 and 135 degrees and the gradient is calculated from all filters. The general gradient of the pixel is determined by the square root of the gradients to squares. And the direction is calculated as the arctangent of the general gradient divided by the gradient according to the first three filters.

D. Thresholding

In order to obtain the best possible result we define different thresholding methods:

- Threshold by average: In order to perform this thresholding, the average value of the image gradient is calculated. All gradient values above this average are considered to be an outline.
- Threshold by local average: In order to perform this thresholding we calculate the average value of the gradient on N neighbors around the treated pixel. If the average value around the pixel is lower than the value of the gradient of the pixel then the pixel is considered by a contour.
- Hysteresis threshold: two thresholds are defined manually.
  - If the value of the gradient of the pixel is lower than the first threshold then it is rejected.
  - If the value of the gradient of the pixel is higher than the first threshold then it is an edge.
  - if the value of the gradient of the pixel is higher than the first threshold AND lower than the second threshold and the pixel is in contact with a contour, then it is a contour.
- Hysteresis threshold based on a k-mean algorithm: This method is the same as the previous one except that both thresholds are determined by a k-mean algorithm. To do this, we split our image into two sets and take the value of the two centers of our sets as the threshold value.

E. Edge refinement

The detection set up here works but creates double borders because it fills the border on both sides of the border. In order to avoid this problem, we will reduce the border so that it is only one pixel. The border reduction method used here is based on the previously calculated gradient direction. Indeed, depending on the direction of the gradient, we check whether the gradient value of the pixel we are testing is the largest relative to its direct neighbors in the direction of its gradient. If it is the case then we test if it is a border, otherwise we remove it from the list of possible borders.

F. Postprocessing

- Detections of related components : In order to be able to work on contours it seemed logical that we needed a larger data structure. For this, we define this structure as a related component. A related component is defined as the set of pixels belonging to a contour. To detect a related component, we analyze the set of pixels and look for the direct neighbors of each pixel, being a border. Thus, if a pixel is a border and the neighbors of a border, then it is added to its related component.
- Contour Closure : In order to carry out the closing of the contours, two methods were set up:
  Une première méthode basée sur le concept de dilatation et d'erosion. Pour obtenir notre résultat on va procéder en trois étapes :
  - First of all a dilatation. This magnifies the contours of the image in order to merge the related components separated by a single black pixel.
  - Then an erosion. We reduce the contours of the image in order to remove duplicate borders on the image.
  - Finally, an erosion is subtracted from the previous image to eliminate unwanted artifacts.

  Expansion and erosion are made from a 3x3 averaging filter. For each pixel of the image we take it and its neighbors and we proceed as follows :
  - If we dilate, we set the value of the pixel to the highest value among its neighbors and itself.
  - If an erosion is done, the pixel value is set to the lowest value among its neighbors and itself.

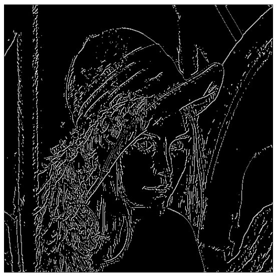
  A second method based on a search for the best path. This is done in several steps.
  - For each of our related components found above we take one of the ends.
  - From this extremity we determine a probable direction in which our contour will go.
  - We test the three pixels in the direction previously found, the pixel in the direction and the two pixels around it.
  - We assign a weight to each of the paths thus defined. 1 if the path passes through the pixel in direction 2 otherwise.
  - We repeat this step for each path. At each step we create three new paths, until we have reached a related component or the weight of the path has reached a maximum, 5 in this case.
  - Once all the paths have been tested, we look for the path with the lowest weight and add it to the image.

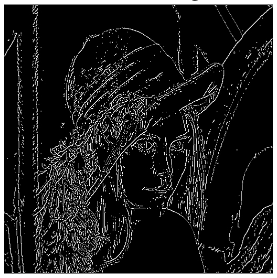
G. Comparative results

As explained above we have implemented many processing methods for contour detection. We will now compare the results according to different filters and parameters. In order for the results to be consistent, they will be obtained on the same image.

First, we will compare the results by just changing the filter. We can thus notice that the filter used has little influence on the shape of the contours but on the other hand will have a huge influence on the continuity of the contours (fig 1).

(a) Base image


(b) Using Prewitt filter


(c) Using Sobel filter

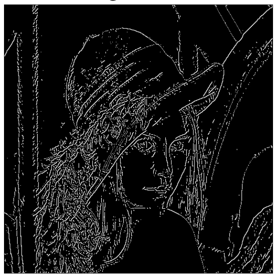
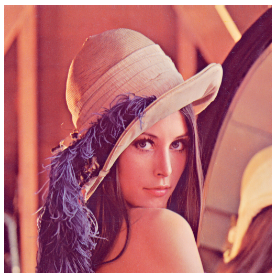
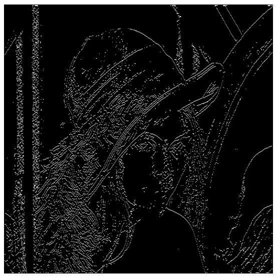
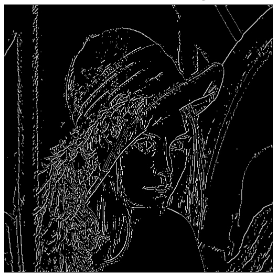

(d) Using Kirsh filter

Fig. 1. Comparison of the results of the different filters

For the following tests we will use the Kirsch filter that gave the best results on the continuity of the contours and we will try to look at the influence of the direction on the filters.


(a) Base image


(b) Unidirectional filter result


(c) Bidirectional filter result

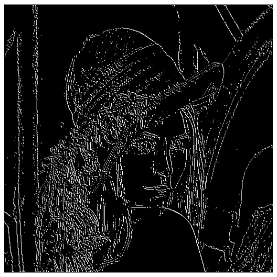
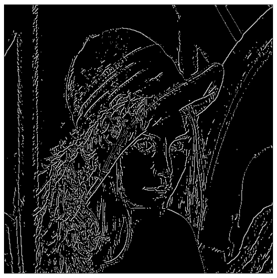

(d) Multidirectional filter result

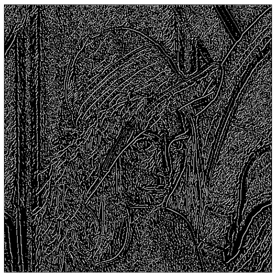
Fig. 2. comparisons of directional results

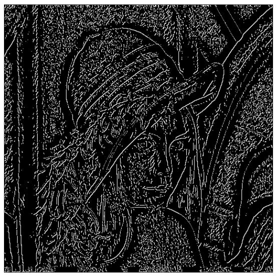
We notice that a unidirectional filter (b) produces much more artifacts than a bidirectional filter (c) while the result of the multidimensional filter (d) is very similar to that of the bidirectional filter (c). It can be concluded that the number of directions greatly influences the quality of the result.

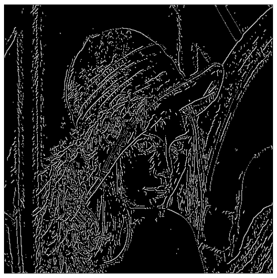
For further tests we will use a bidirectional filter. We are now going to compare the different thresholds presented above. With this analysis, we can notice that the different thresholds have a great influence on the result obtained. And even if the different thresholds seem to tend towards a common result, none of them is precise enough to have a net result. We can however note that the thresholding by average of the neighbors is slower than the others for very uninteresting results.

(a) Base Image


(b) Gradient average result


(c) Neighbour average result (1)


(d) Neighbour average result (10)


(e) Hysteresis result

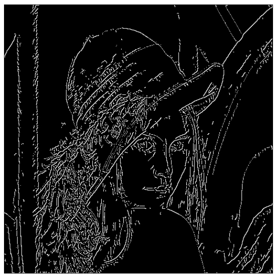
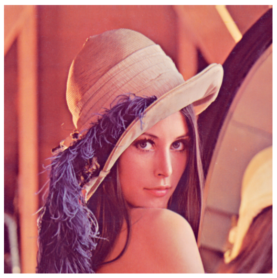

(f) hysteresis using k-means result

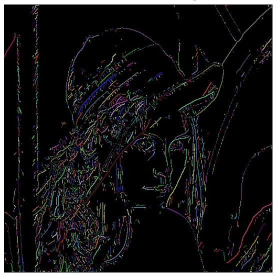
Fig. 3. comparisons of directional results

H. Postprocessing results

The detection of related components not bringing significative changes, we can just notice that the results obtained seem coherent on two images.



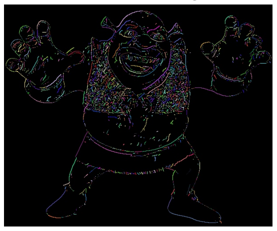
(a) Base image



(c) connexe

Fig. 4. Connected component

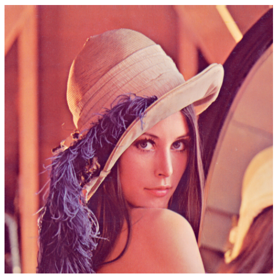
(a) Base image



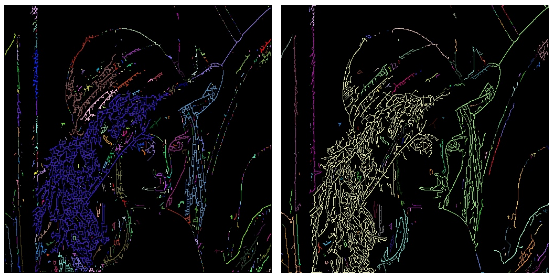
(c) connexe

Fig. 5. Connected component

Both methods of contour closure give similar results. However we can note that the better path method, although longer than the morphing, has a tendency to impact the contours much more and to better respect the shapes.



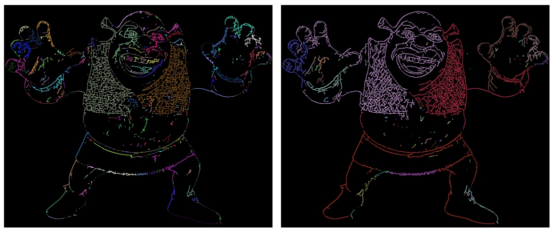
(a) Base image



(c) contour morphing (c) contour better path

Fig. 6. Contour closing

(a) Base image



(c) contour morphing (c) contour better path

Fig. 7. Countour closing

I. Limit of our algorithm

The algorithm put in place has two major problems:

- First of all, it is far too dependent on user settings which makes it difficult to use in a general case and often requires a lot of testing and tweaking to find the right settings.
- The extracted data are in their current state not very exploitable by an algorithm that would use it as input.

IV. Conclusion

To conclude, we can notice that each parameters influences the detection of contours and that it is difficult to find parameters to obtain a perfectly smooth and usable result. However, edge detection as presented has the advantage of being inexpensive and giving a consistent result on the shapes of the image. However, a method using a morphological gradient would probably give better results.

Acknowledgement

Thanks to Mrs. Saida for this project and for her help.