Laboratory Session 3

Computer Vision - Università Degli Studi di Genova

Filip Hesse
S4889393
filip_hesse@yahoo.de

Justin Lee
S4885003
leej1996@gmail.com

Steven Palma Morera
S4882385
imstevenpm.study@gmail.com

***Abstract*—A Laplacian of Gaussian was used to detect edges on a test image and a Hough transform was used to detect straight lines on two other test images.**

***Index Terms*—Laplacian of Gaussian, Hough Transform, Edge Detection, Voting method**

I. Introduction

The purpose of this lab [2] is to implement a Laplacian of Gaussian edge detection algorithm on Fig 1 and compare its efficiency with the MATLAB version of the algorithm. In addition, a Hough transform is used to detect straight lines on Fig. 2 and Fig. 3.

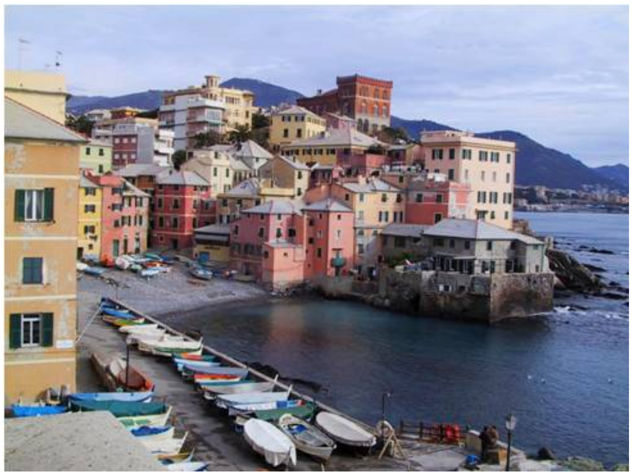

Fig. 1. Test Image of Bocadasse. Source: [2]

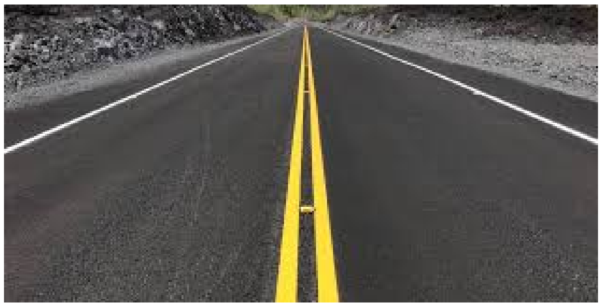

Fig. 2. Test Image of Hwy 1. Source: [2]

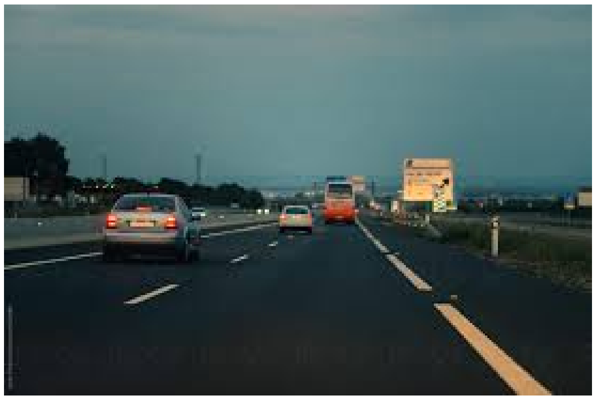

Fig. 3. Test Image of Hwy 2. Source: [2]

II. Procedure

A. Task 1

The Laplacian of Gaussian Operator was implemented in MATLAB. It samples and displays the filter with different standard deviations (between 1 and 3.5) and spacial supports (always 3 times larger than the standard deviation). This filter was convolved with the boccadasse test image. This LoG script was then used for edge detection by finding zero crossings. The effectiveness of this was tested by varying the standard deviation and the threshold. This script was then compared to the MATLAB edge() script.

B. Task 2

The Hough Transform was used in order to detect lines in two different highway test images. The basis of the Hough Transform script was taken from the example done in class [1].

III. Results and Analysis

A total of 105 images were obtained after running the Lab3.m script in the MATLAB workspace as follows:

Lab3();

The following images and analysis shown are the most useful ones for understanding the effect of each task from the procedure. However, the reader can access the full set of images in the folder called ”plots”, automatically generated when the script is executed.

A. Task 1

In the Fig.4 and Fig.5 a Laplacian of Gaussian filter with a standard deviation of 1 and 3.5, respectively, is shown. Notice their similarities to the ones presented in [1].

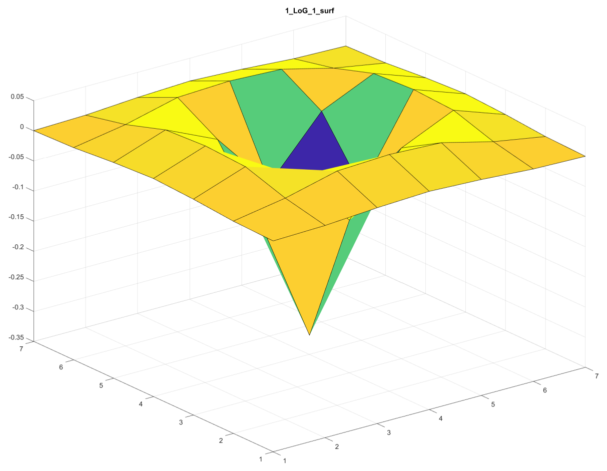

Fig. 4. Laplacian of Gaussian filter with sd=1. Source: Own elaboration

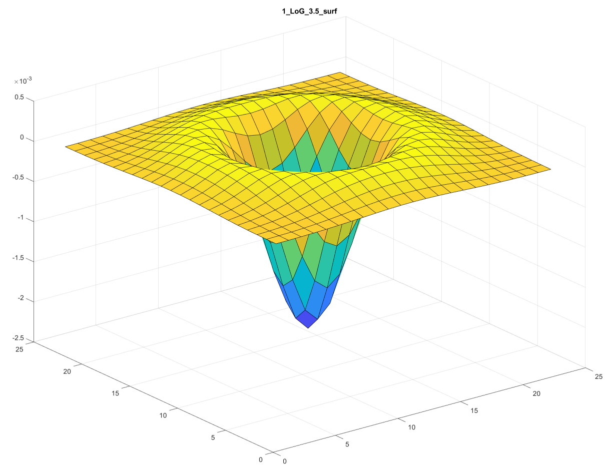

Fig. 5. Laplacian of Gaussian filter with sd=3.5. Source: Own elaboration

This same filters were applied to the Fig.1, resulting in Fig.6 and Fig.7. As it is already known, in a Gaussian filter the standard deviation plays a role in how much the image is smoothed, in both Fig.6 and Fig.7 this effect is still true. Since the filter from Fig.5 has a bigger standard deviation, then it can be expected that the Fig.7 will lose detail and therefore only big and large scale features are obtained, while in the Fig.6 the details are preserved because it has a smaller standard deviation, and therefore more small and fine features are detected.

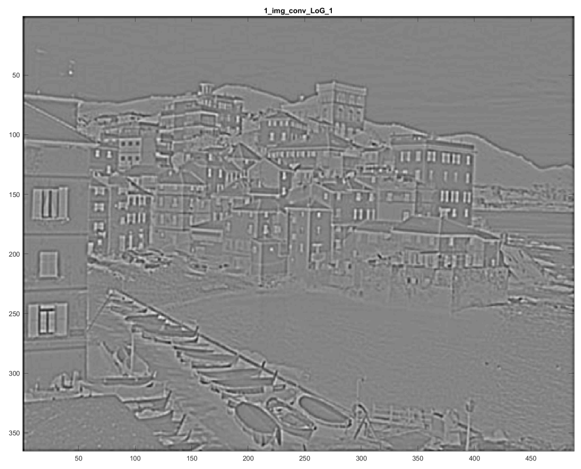

Fig. 6. Landscape image convoluted with the LoG (sd=1) filter. Source: Own elaboration

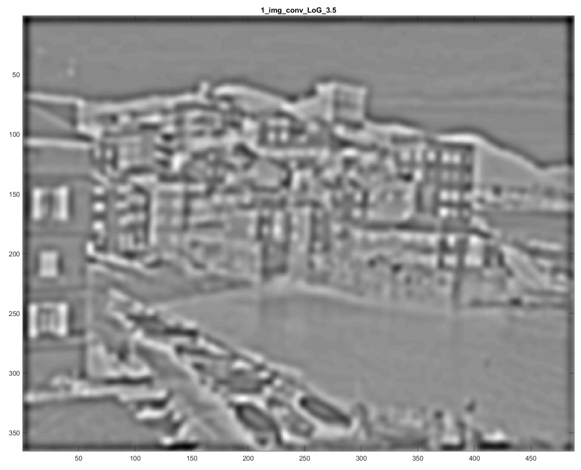

Fig. 7. Landscape image convoluted with the LoG (sd=3.5) filter. Source: Own elaboration

In the Fig.8 and Fig.9 the edges of the landscape source images are displayed using different values for the threshold of the edge detection algorithm based on LoG. The Fig.8 uses a very small threshold for the edge detection, 0.1, and the Fig.9 uses a bigger threshold, 10. It can be noticed that when a small threshold is used, more detailed and fine edges are detected, this is because the algorithm's admission criteria of edges is softer so almost any zero-crossing in the convoluted image will be treated. On the other hand, the Fig.9 shows a different case where the edges identified are the most defined ones from the source image. Both approaches of the threshold have their advantages and drawbacks. For instance, a small threshold could be useful for identifying fine edges and details of the image, while a big threshold can be used for identifying the most noticeable edges of an image. However, a small threshold could show an edge where there isn't any just because of the noise, or it can show so many edges that the real shape of the image isn't distinguishable, as it happens in Fig.8; while a big threshold could lose valuable information of the edges present in the source image.

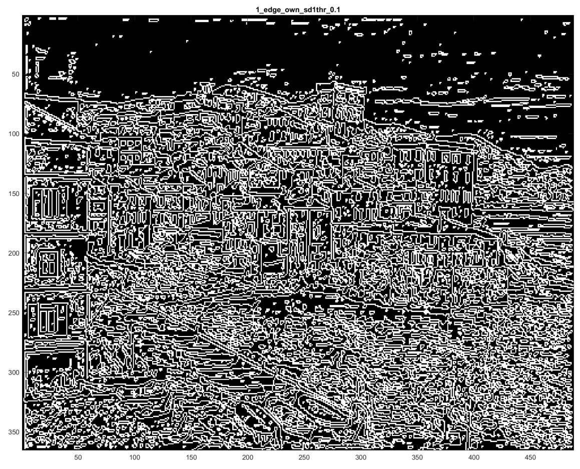

Fig. 8. Edges detected from image 6 using 0.1 as threshold. Source: Own elaboration

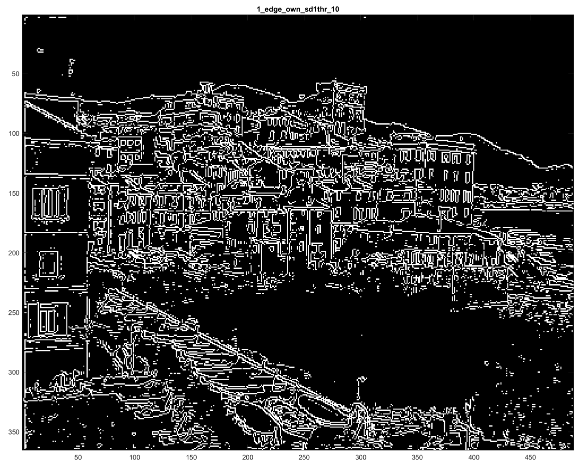

Fig. 9. Edges detected from image 6 using 10 as threshold. Source: Own elaboration

In the Fig.10 is shown the result of the MATLAB function edge() using the same image from Fig.6 and a standard deviation of 10. Comparing it to the Fig.9, it can be said that both of them are pretty similar in form and detail obtained for the edges, valitading the edge detection algorithm based on LoG developed by the team for the current lab. However, to get a deeper insight of the similarities and differences between the developed algorithm and the MATLAB function, the Fig.11 is shown.

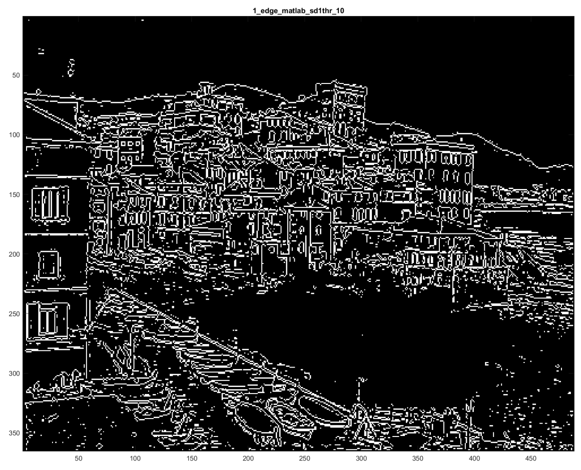

Fig. 10. Edges detected from image 6 using 10 as threshold and the MATLAB function edge(). Source: Own elaboration

The Fig.11 is the result of a XOR operation between the image of the Fig.9 and Fig.10. The XOR operation between two images will result in a new image that highlights the differences between the two images. It can be seen, that even if -as said and shown before- both of the images look pretty similar, there are still some differences between them. However, it is interesting to notice that in the Fig.11 the edges of the source image can still be seen, this means that the real difference between the developed algorithm and the MATLAB function is actually the location of the resultant pixel of the algorithm. It could be happening that the developed script is saving the result of the algorithm in the pixel (i,j) while the MATLAB function edge could be saving the result in a different pixel, for instance (i-1,j-1). Even though this happens, it can be said that the developed script satisfies enough the implementation of the LoG edge finder algorithm with respect to the MATLAB function edge().

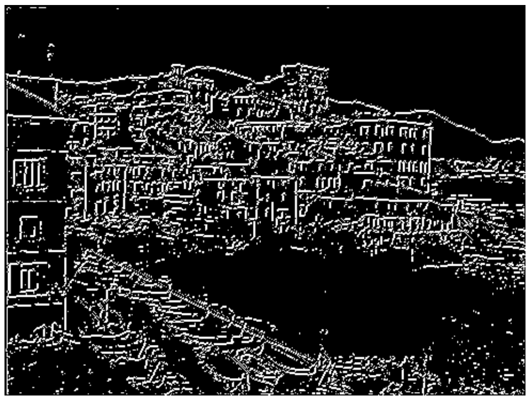

Fig. 11. XOR between the images obtained with the developed algorithm of LoG and the MATLAB function edge(). Source: Own elaboration

B. Task 2

In Fig. 12 and Fig. 14, the results of the line detection via Hough transform is shown. The peaks detected can be seen in Fig. 13 and Fig. 15. As can be seen from looking at the figures, the line detection on the Hwy 1 image worked very well as it detected the lines of the road on the ground quite clearly. For the Hwy 2 image, it did not work as well, possibly because the lines on the road were dotted and were not completely connected. Instead, it detected some lines in the middle left of the image. This can be attributed to several factors. First of all, the noise and quality of the image can cause the algorithm to not work as well. In addition, the algorithm isn't perfect so it can detect a line when there isn't one just because the intensity of the pixels are similar.

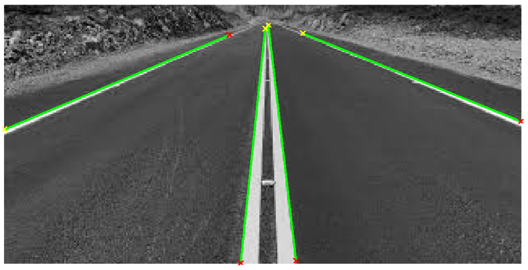

Fig. 12. Image of Hwy 1 with detected lines. Source: Own Elaboration

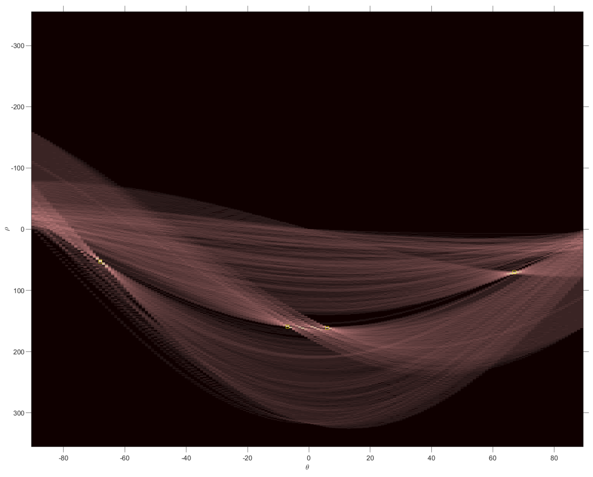

Fig. 13. Image of Hwy 1 Hough transform peaks. Source: Own Elaboration

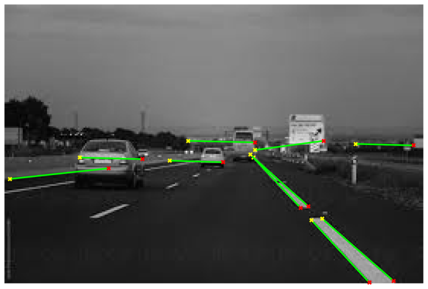

Fig. 14. Image of Hwy 2 with detected lines. Source: Own Elaboration

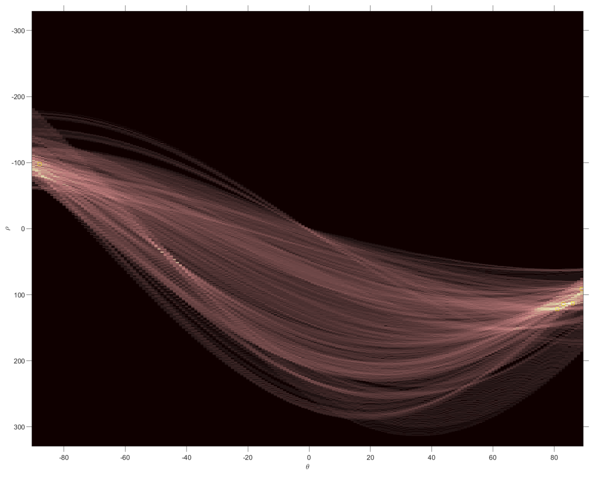

Fig. 15. Image of Hwy 2 Hough transform peaks. Source: Own Elaboration

IV. Conclusions

- With the Laplacian of Gaussian filter there is a trade-off between the smoothing and identification of edges
- The threshold used for identifying edges in an image using the Laplacian of Gaussan filter should be chosen considering the detail and accuracy of the desired information of the edges form the source image.
- The voting technique used by the Hough transform worked well with clearly defined lines but it not with noisy and dashed lines.

References

[1] Solari, F. Part 3 - edge detection and Hough transform, 2020. Retrieved from: https://2019.aulaweb.unige.it/pluginfile.php/209255/mod_resource/content/2/lecture3_edge_HoughT.pdf

[2] Solari, F. Lab Session n.3, 2020. Retrieved from: https://2019.aulaweb.unige.it/pluginfile.php/209257/mod_resource/content/3/Lab3.pdf