Computer Vision Laboratory Report n.3

EDGE DETECTION AND HOUGH TRANSFORM

April 3, 2020

Chiara Terrile ID: 4337786
Giulia Scorza Azzarà ID: 4376318
University of Genoa

Contents

Chapter 1

Introduction

1.1 Defining the objective

The purpose of this laboratory is to analyze an important topic in computer vision, which is objects' boundary detection.
In order to achieve this goal we have performed the so called 'zero crossing' of an image convoluted with the Laplacian of Gaussian.
Another technique that we have used in this laboratory is the so called 'Hough Transform', which is useful to detect straight lines in an image.

Chapter 2

Theoretical Background

2.1 Edge Detection

Since the noise values at each pixel are typically uncorrelated, they would result as edges when applying edge detecting algorithms. Therefore, to achieve the edge detection goal, at first the image is filtered to remove the noise, using, in this case, a Gaussian filter. This algorithm works by considering the second derivative of the filtered image and then looking for the zero-crossing in order to find the local maxima, which corresponds to the edges.

2.2 Hough Transform

The Hough transform is a technique which can be used to isolate features of a particular shape within an image. Since it requires that the desired features are specified in some parametric form, the classical Hough transform is most commonly used for the detection of regular curves, such as lines, circles, ellipses, etc.
A generalized Hough transform can be employed in applications where a simple analytic description of a feature is not possible.

Chapter 3

Implementation and Results

The code is composed of a single **main** function for the three images 'boccadasse.jpg', 'highway1.jpg' and 'highway2.jpg', in which are called four functions in order to perform the Edge Detection and the Hough Transform of the images.
Our functions are the following:

- **laplacian_gaussian.m:** which performs the convolution between the Laplacian of Gaussian Operator and the original image
- **zeroCrossing.m:** which detects the zero-crossings to find the edges
- **hough_transform.m:** which detects the straight lines using the Hough Transform in the image 'highway1.png'
- **hough_transform2.m:** which detects the straight lines using the Hough Transform in the image 'highway2.png'

3.1 laplacian_gaussian

This function takes as input the original image (Figure 3.1) and the standard deviation 's' of the Gaussian and returns as output the filter of the Laplacian of the Gaussian and the convoluted image.

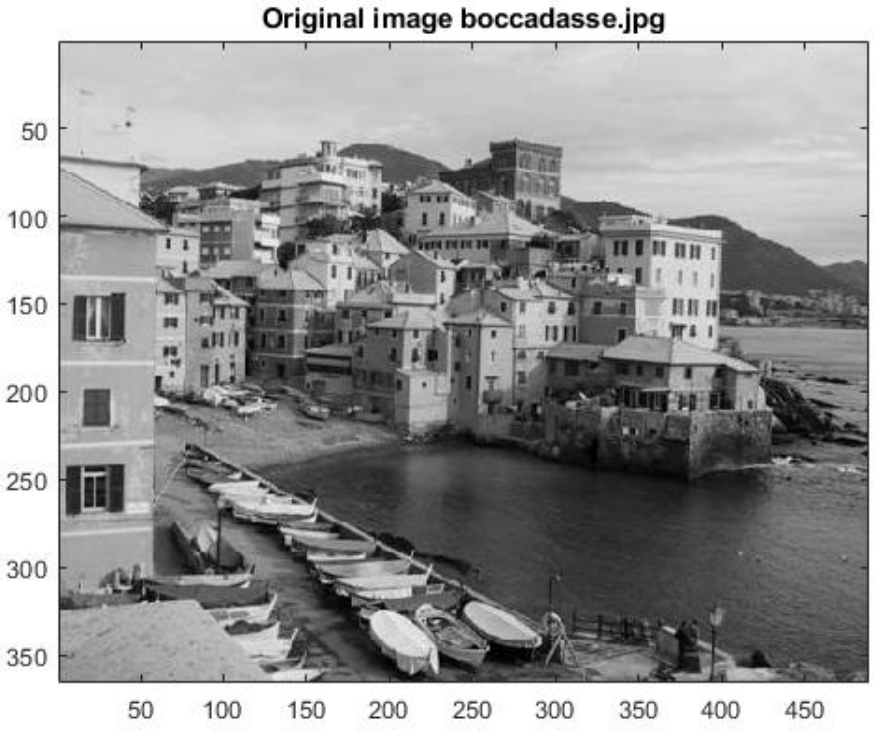

Figure 3.1

Note: In the function we have used the approximation according to which the spatial support is equal to $ceil(3s)$.

After applying this function, we have subplotted the results in the **main** script.
Figure 3.2 and Figure 3.3 in the next page show the filter image, the filter surface and the convoluted image, using two different values for the standard deviation.

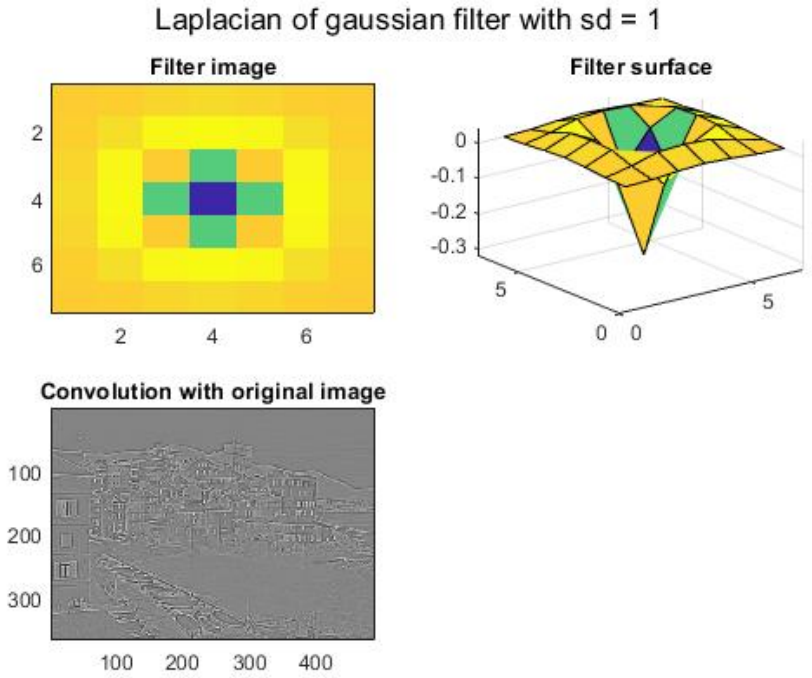

Figure 3.2

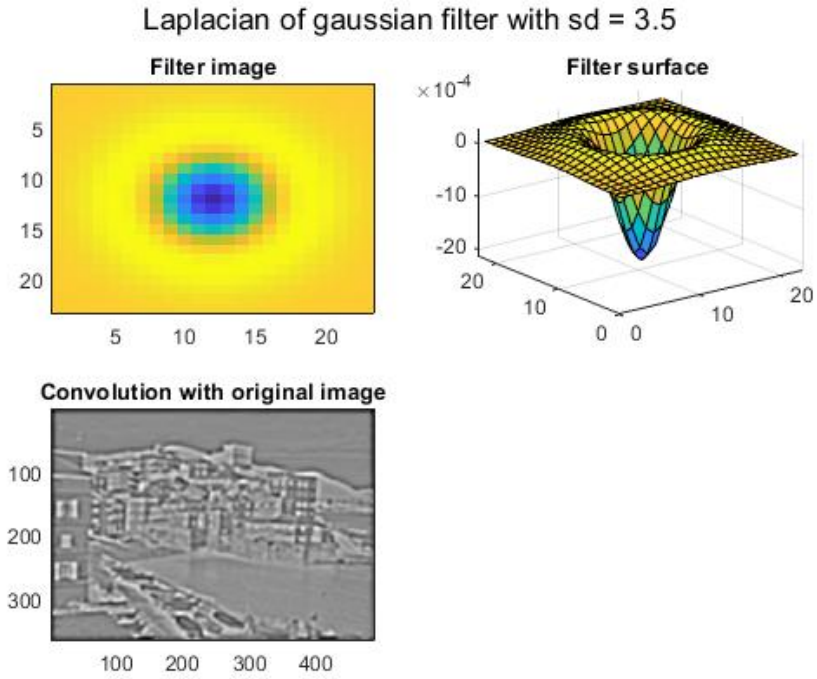

Figure 3.3

3.2 zeroCrossing

This function takes as input the convoluted image (img_conv) and the threshold desired (th) and returns as output an image containing the edges (edges). In this function the edges are detected using the zero-crossing procedure, which is divided into two parts:

- In the first part of the algorithm, all the columns of the image are scanned and the edge is detected only if two adjacent horizontal pixels show a change of sign (+- or -+) and the module of their difference (the slope) is greater than the threshold. It's also possible the case in which the zero-crossing pass for a pixel whose value is actually zero (+0- or -0+), although it's really unlikely; anyway this possibility is also checked. Each time an edge is detected, a corresponding pixel arbitrarily chosen (in this case the left pixel) is set to 1 (white) in a binary image with the same size of the original one.
- The second part is analogous to the first one, but in this case are scanned the rows, so the correspondence to look for is between two adjacent vertical pixels.

At the end of the whole process, is produced an image containing the edges corresponding to a certain threshold. In the **main** script we have called this function four times, using two different standard deviations and two different thresholds. Finally we have subplotted the image obtained with zeroCrossing function and the one obtained with the MATLAB function **edge()**, using the same values of threshold and standard deviation. In the following Figure 3.3 and Figure 3.4 are shown all the results.

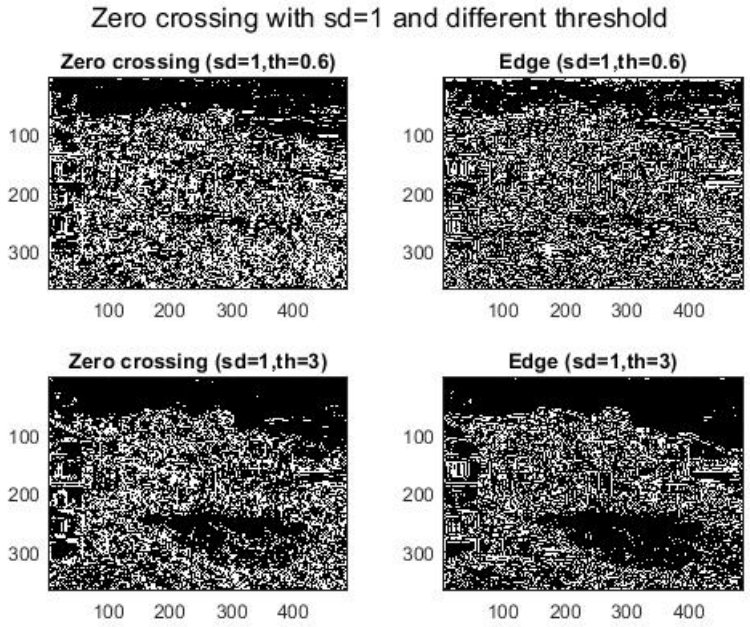

Figure 3.4

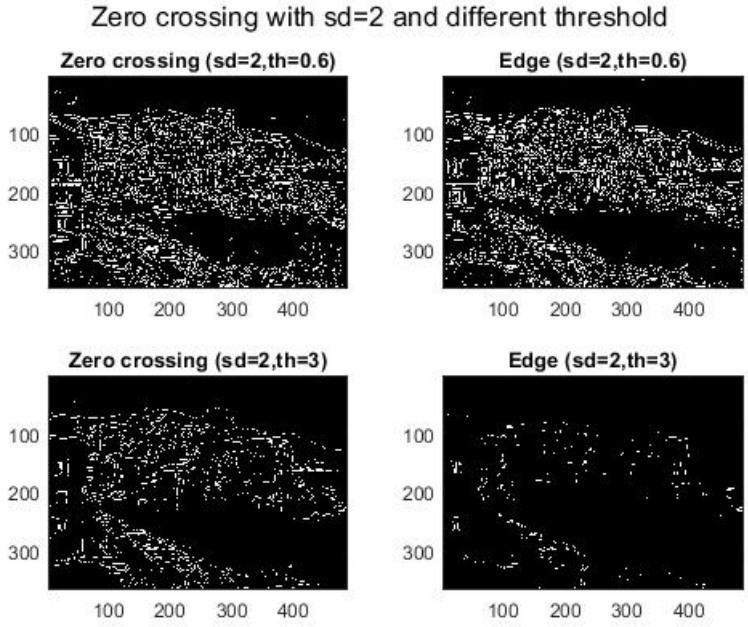

Figure 3.5

Comparing all the figures obtained by using respectively our **zeroCrossing.m** function and the MATLAB function **edge()**, we can conclude that if the standard deviation increases, with the same threshold value, some lines are lost, proportionally to the increasing value of the standard deviation. In the same way, if the threshold value increases, with the same standard deviation, some lines are still lost, proportionally to the increasing value of the threshold. In conclusion, the optimal case is to have a balanced combination of standard deviation and threshold, avoiding too high values for both of them.

3.3 hough_transform

The Hough function is designed to detect lines. The purpose of this technique is achieved by using a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform. This function is applied to the image highway1.jpg and identifies four peaks.
The main steps using this function are:

- Load the image highway1.jpg
- Apply the Edge Detection at the image
- Compute the Hough Transform (H) with the MATLAB function **hough()**
- Identify the peaks using the MATLAB function **houghpeaks()**, which takes as input H and the number of peaks desired
- Record all possible lines on which each edge point lies, using the peak values in Hough Transform, and look for lines that get many votes
- Finally line segments are computed by using the straight lines and the values in the edge map

In Figure 3.6 in the next page are shown the results of the described function.

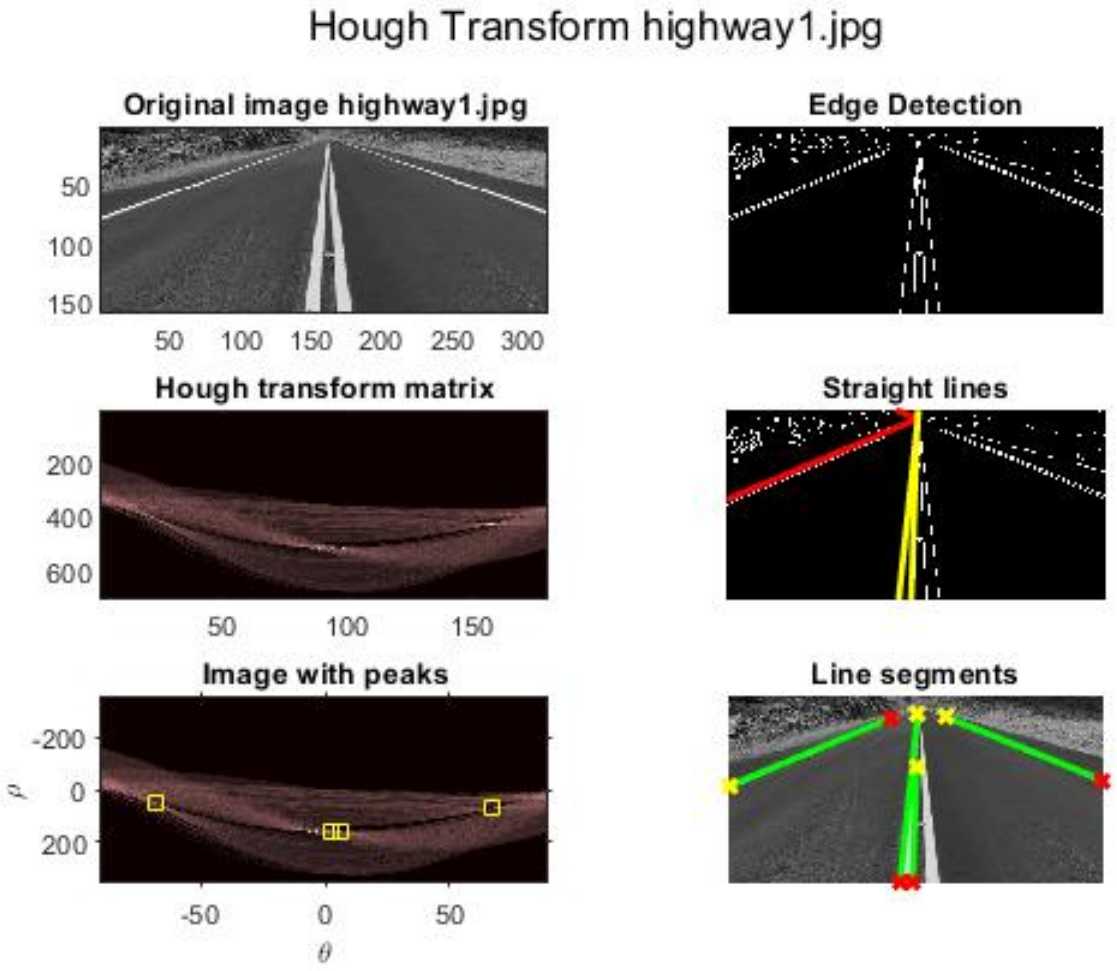
Hough Transform highway1.jpg
Original image highway1.jpg
Edge Detection
Hough transform matrix
Straight lines
Image with peaks
Line segments
ρ
θ

Figure 3.6

3.4 hough_transform2

The last function is equivalent to the **hough_transform.m** function, but in this case it is applied to the image highway2.jpg and it identifies five peaks with the modality 'NHoodSize' [21 21]. In the following Figure 3.7 are shown the results.

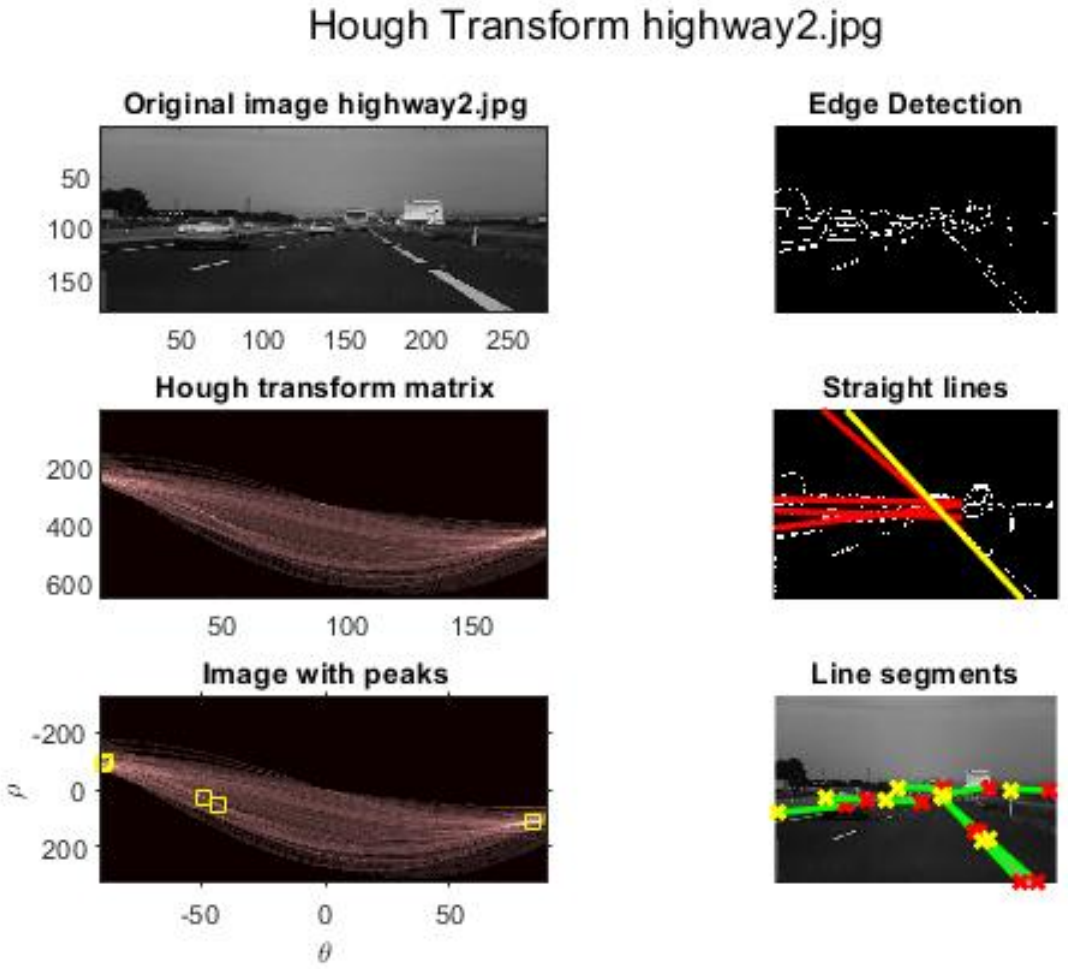

Figure 3.7