Università degli studi di Genova

Computer Vision

Assignment 3: Edge detection .

Aurora Bertino, Chiara Saporetti

April 4,2020

Introduction

The goal of Edge detection is to identify visual changes in an image. The most semantic and shape information from the image can be encoded in the edges.Edges are discontinuities in intensity such as boundaries of material properties, boundaries of objects, boundaries of lighting.Discontinuities can be detected by computing the derivative of the signal and the edges corresponds to the extrema of derivative. Since we deal with images that are functions in f(x,y), the derivative is a gradient. One of the most import algorithm in this fild is "Marr and Hildreth" edge detector which detects edges by considering :

- *The second derivative of the filter*
- *Zero-crossings mark edge location*

Firstly, in the Lab we compute edge detection with a Gaussian Laplacian filter (LOG) on the image "boccadasse.jpg". The Gaussian Laplacian operator (LOG) is the second derivative of the Gaussian and the formula is:

$$G(x,y) = \frac{1}{2\pi\sigma^2}\frac{x^2+y^2-2\sigma^2}{\sigma^4}e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Subquently, we compute the edge function to compare the edge detection as the result of the algorithm implemented. It's based on the detection of zerocrossings doing two for-loop, one to scan along each row and the other to scan along each column. Then we introduce the Sobel operator:
At each point of the image, the result of the operator is the gradient vector corresponding to the pixel taken into consideration.The Sobel algorithm uses two 3x3 kernels, both being separable filters that can be decomposed as the product of an averaging kernel and a differentiation kernel, in order to compute the gradient by also smoothing.
As second step, we detect straight lines in the test images ('highway1.jpg' and 'highway2.jpg') through the Hough transform. When we apply the Hough transform algorithm we switch from the image space to a parametric space that represents a set of features and we apply the voting technique to look for a model that represents them. Here we use the Hough transform for straight lines detection. In order to do so, we initialize a matrix H(d, theta), which is the accumulator array that contains the votes. It can be seen as a discrete set of bins for the votes, and is initially set to zero. Then, for each edge point (x,y) in the image we find the value (d,theta) in the parametric space that defines the line corresponding to the given point. This is done through a for loop that applies sampling, quantization and saves votes in H at each cycle for all the pixels. Then we look for the maximal values (d, tetha*) in H(d, tetha). The line in the image is the one that has, as parameters, the two values. It is also important to understand where the extremities of the straight line segments are.*

0.1 Matlab Resolution

0.2 Laplacian Gaussian Operator

The Laplacian of Gaussian operator is a method used to find edges by looking for zero crossings after filtering the image with a Laplacian of Gaussian filter. The algorithm is:

- *Compute a grid with meshgrid () function.*
- *Defines above that the Gaussian Laplacian Operator function.*
- *Plot the Laplacian of Gaussian operator with the surf() function.*

We implemented these steps computing different spatial supports (sp=3 and sp=9) and standard deviations (sd=1 and sd=3.5). We implements Edge detection with Laplacian of Gaussian Operator by complete the image's filtering through a convolution and then we plot them with the imagesc() function.

0.3 Zerocrossing

We test the edge detection algorithm based on LoG by varying the standard deviation of the kernel and the threshold. The method consists on computing the gradient vector at each pixel by convolving image with horizontal and vertical derivative filters. To detect zerocrossings: we scan along each row, record an edge point at the location of the zerocrossing considering each transition: +,-, -,+, +,0,-, -,0,+. Then we applied a threshold on the slope of the zerocrossings and we display it. We set properly threshold in order to make the result more similar to matlab function's result. At the end we compare the results with the edge detection matlab function. The function in which in the code we implement zerocrossing algorithm is the following:

$$edges = detectZeroCrossings(convImg, threshold) \tag{1}$$

0.4 Hough Trasnsform

In order to detect straight lines we use the following function:

$$straightLines = detectStraightLines(inputImg, peaksNumber, NHoodSize, threshold) \tag{2}$$

It allows to detect straight lines in the input image inputImg. It is possible to specify the number of desired hough peaks to find in the matrix (peaksNumber), the neighborhood around each peak that is set to zero after the peak is identified (NHoodsize) and the threshold to use when plotting the lines. Firstly, the function finds the edges of the images by convolving it with the Sobel operator. The result is a binary image. Then the standard Hough transform matrix is computed from applying the matlab function hough() to the binary image obtained in the previous step. The function returns the Hough matrix H, rho (R) and theta (T), these last two being respectively the y and x axis of the space parameterized by H. In the image space, R is the distance from the origin to the line along a vector perpendicular to the line, and T is the angle in degrees between the x-axis and this vector. So, R and T are used to identify the straight lines of the image:

$$R = xcos(T) + ysin(T) \tag{3}$$

Then, the peaks in the H matrix are identified with the Matlab function houghpeaks(), and the desired number of

peaks to look for must be set. The function gives the possibility, as previously stated, to set the neighborhood size. After doing so, the straight lines are plotted by using the values obtained by the previous computations. Lastly, the line segments are computed by the Matlab function houghlines() by using all the previous information, and the value is returned by our function so that in the main we can plot the resulting image.

Results

As first result we can notice the Laplacian Gaussian Operator combined with the smoothing (Gaussian) to reduce the noise. We compute this operator with different spatial support and standard deviations. Figure 1 displays this operator with standard deviation 1 and spatial support 7x7 while figure 2 displays it with standard deviation 3.5 and spatial support 21x21. Then we convolve the test image 'boccadasse.jpg' with those LoG filters. At this point, we can notice that the image filtered with the first filter is better than the other because the blurring effect is directly proportional to the standard deviation.
Subsequently we start to test the edge detection algorithm based on LoG with the image 'boccadasse.jpg', detecting first zerocrossing and applying a threshold on the slope of them. The "computed edge image 1" is the result of the image implemented through the Edge detection algorithm, in which a Laplacian operator with sd=1 is used. Instead, the "computed edge image 2" is implemented through a Laplacian operator with sd=3.5. Then implement Edge detection through the matlab function for the two images and we compare the results.
Lastly, we used the "detectStraightLines" function tested on two different images. The first one is "highway1", where we set the peaks number to 4. The first result is shown in the fig 6 (peaks in H). The second image shows the binary image made of the edges, computed by Sobel operator, and the straight lines found by the function. By setting a low threshold, the lines on the right are not plotted, while if we set the threshold to approximately 70, we are able to detect the border line of the road on the right. The final result (Fig 7) shows that the algorithm is able to detect most of the straight lines in the image a part from the right central line. By setting a higher number of desired peaks (in our case six) in the hough transform we can also detect this last line. However, we set the peaks to 4 as suggested by the assignment. In general, however, the function has a good detection rate.
The second image we tested our function on is "highway2", where we set the peaks number to 5 and the neighborhood size to a 21 by 21 area. With respect to the previous one, this image is more complex, since it holds more subjects and more straight lines. By trying different values we noticed that the algorithm fails in recognizing the central line of the road or the one of the street sign, while it sees the straight lines of the cars or some lines of the horizon. The results are shown below. Lastly, we plotted the straight segments on the image by only plotting the straight continuous lines up to their edges, specified in point1 and point2 by houghlines().

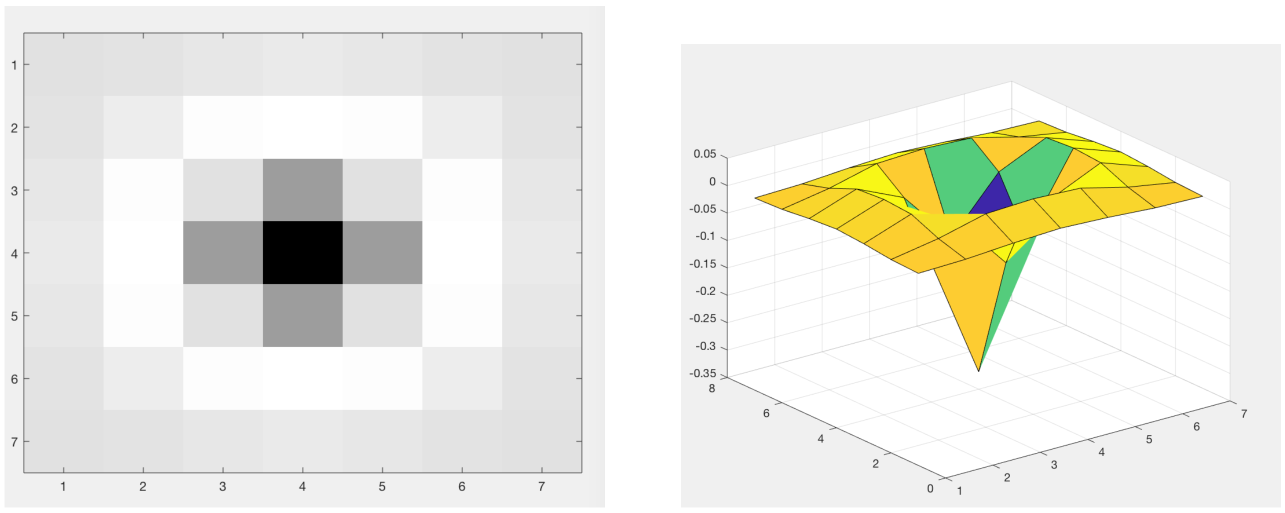


Figure 1: : Laplace Gaussian Operator filter (sd=1)

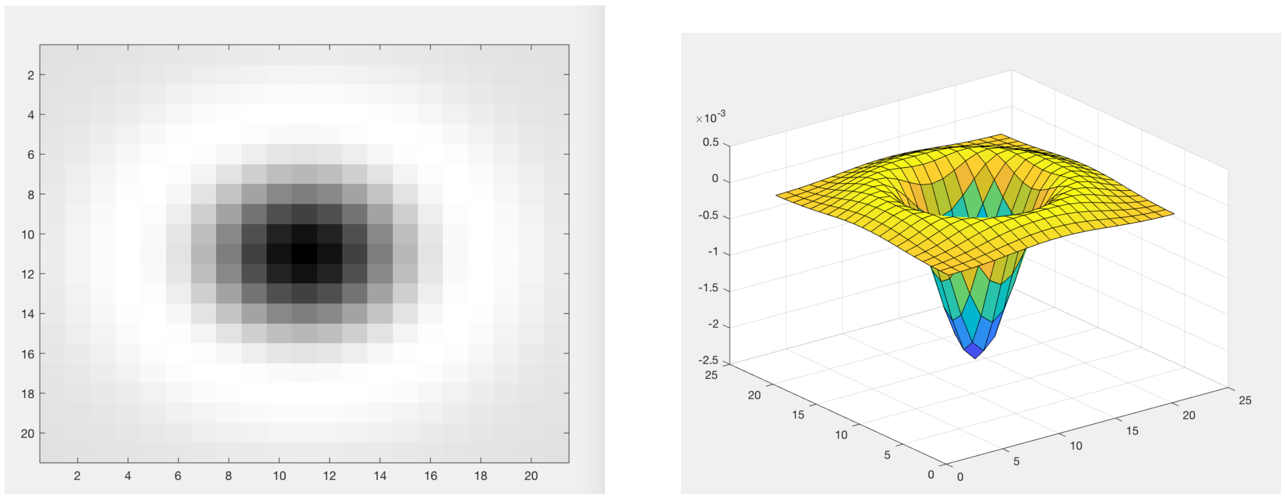
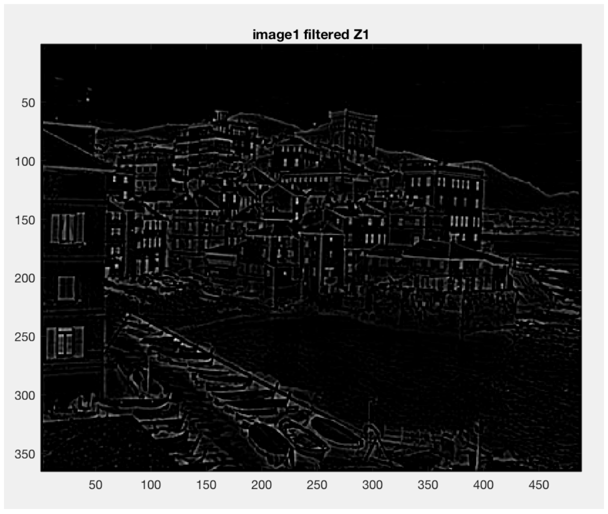


Figure 2: : Laplace Gaussian Operator filter (sd=3.5)



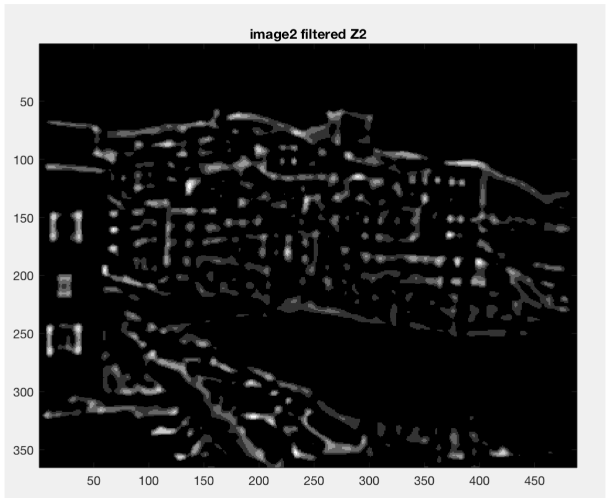
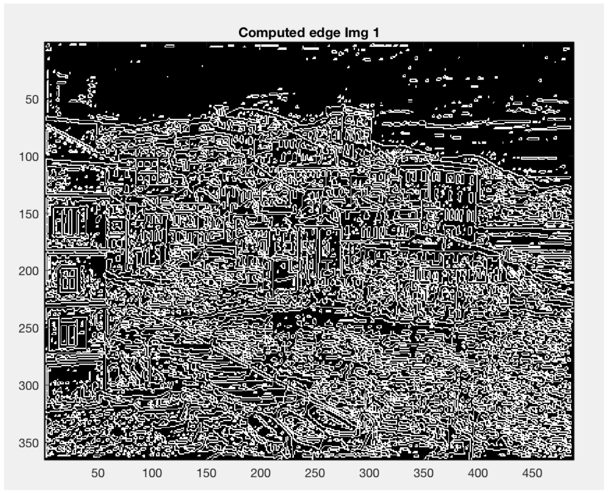


Figure 3: : Comparison between images filtered with different Laplacian Operators



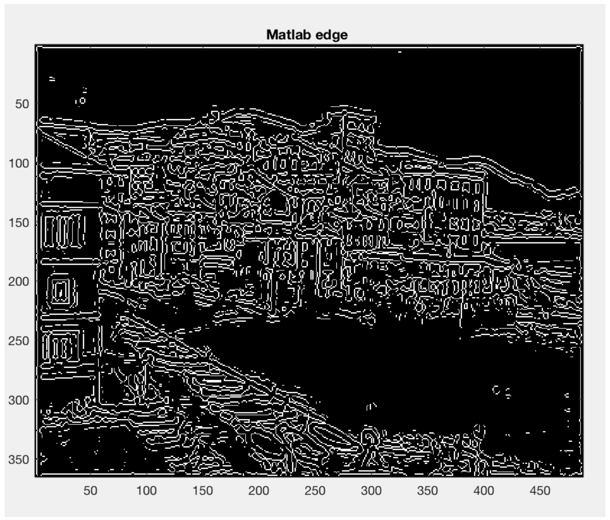
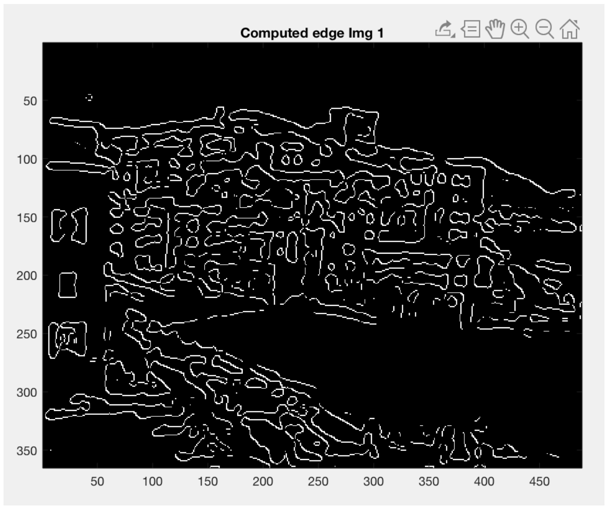


Figure 4: : Comparison between "computed edge image 1"and "matlab image 1".



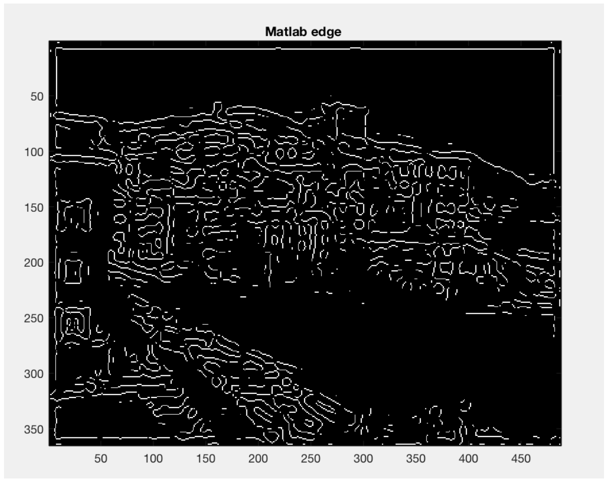
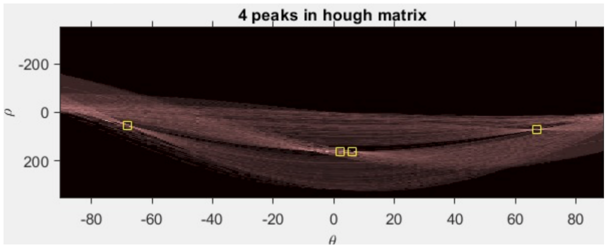


Figure 5: : Comparison between "computed edge image 2"and "matlab image 2."



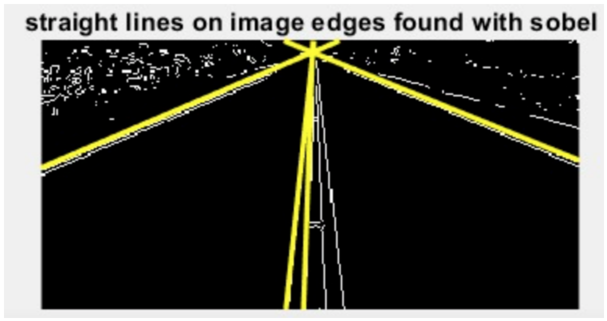


Figure 6: : Binary image computed by Sobel operator and 4 "Peaks".

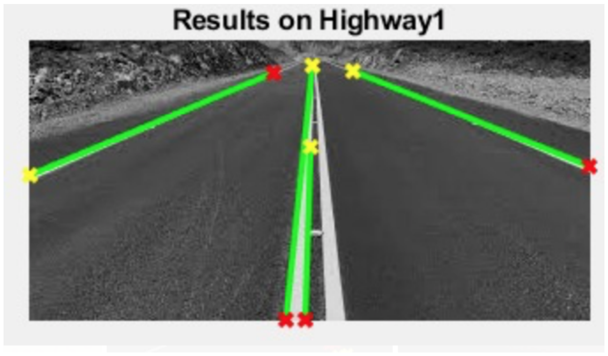
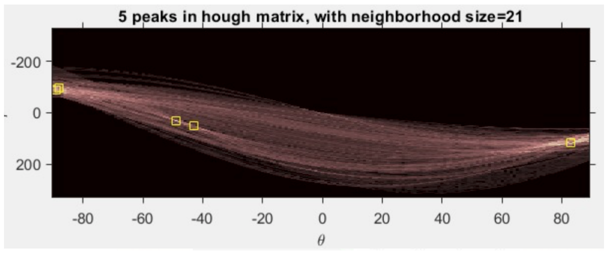


Figure 7: : Straight lines detected in "Highway1".



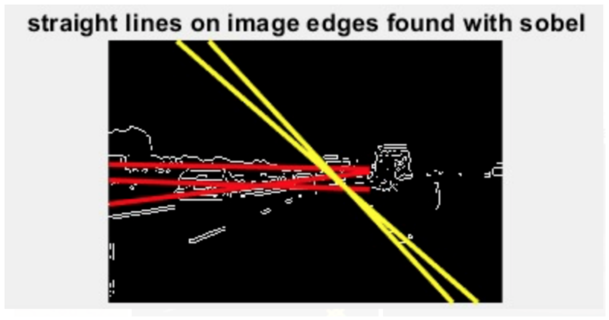


Figure 8: : 5 "Peaks" image "Highway2" and straight lines detected in "Highway2".

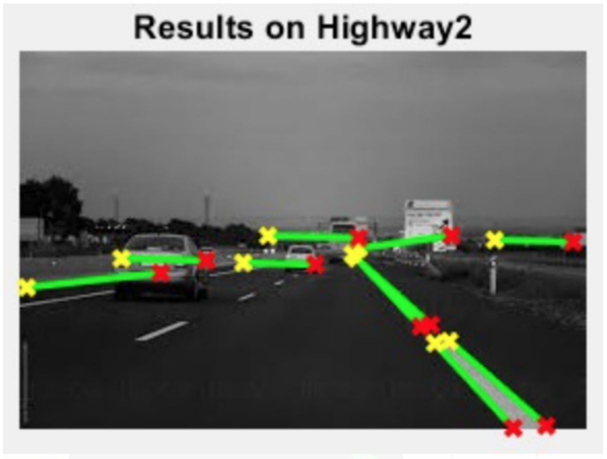


Figure 9: : Final result in "Highway2".