Università degli studi di Genova

Computer Vision

Assignment 4: NCC - based segmentation and Harris Corner detection .

Aurora Bertino, Chiara Saporetti

April 26,2020

Introduction

0.1 Harris Corner Detection

The basic idea of Corner Detection is given by a recognition of the point by looking through a small window and through the shifting of the window in any direction in order to recognize large change in intensity. Harris corner detector gives a mathematical approach for determining that. We distinguish 3 regions:

- *Flat region, in which there is not a change in all directions.*
- *Edge region, in there is not a change along the edge direction.*
- *Corner region, in which there is a significant change in all direction.*

The algorithm steps consist on:

- *Compute M matrix within all image windows to get their R scores.*
- *Find points with large corner response ($R >$ threshold).*
- *Take the points of local maxima of R.*

M is the second moment matrix computed from image derivatives and it's very important for corners detection. We also need to diagonalize M because classification of image points is made using eigenvalues of M. Infact, the eigenvectors encode edge directions, the eigenvalues edge strength. In the Lab we are going to implement the Harris corner detector. Then we apply this metod to the "image i235.png" and we show the results.

0.2 NCC - based segmentation

To find an image patch in a given picture, we can use template matching, which is a simple intensity matching, given that the image and the template are similar enough. A way to apply template matching is through normalized cross correlation. Given an image f and a template g:

- *Substract from both of them the mean value of the template intensity. Without this step, the correlation score would increase with the brightness of the patch.*
- *Divide for the standard deviation. Without this step, the correlation score would be highest in the local neighborhood of the object.*

The result is a matrix that has the maximum value in the element corresponding to the pixel that has the highest correspondence score.

0.3 Matlab Resolution

0.4 Harris Corner detection

Firstly, we implement the Harris Corner detector algorith in order to find the max value of R map; The steps are :

- *Compute x and y derivative (dx,dy) of the image and perform a 2D convolution between the InputImage and dx, dy.*
- *Compute products of derivatives at every pixel.*
- *Compute the sum of products of derivatives at each pixel and construct the structure tensor M.*
- *We start the features detection, we choose a constant K=0.5 and we compute trace(M) as a sum of diagonal elements in order to find R.*
- *We compute the response of the detector at each pixel.*
- *We find the max value of R map.*

These are the steps of the following function:

$$[valore_max] = Harris_detector\_value_max(InputImage) \tag{1}$$

*Subsequently, with this value "valore_max" we can compute again the Harris corner detection with the threshold "0.3*value_max" using this "valore_max" in the following function:*

$$[edge\_reg, flat\_reg, corner\_reg, R\_map] = Harris\_detector\_threshold(InputImage, value\_max) \tag{2}$$

As second step, we use the function regionprops() to get the centroids of the blobs and with a for loop we put them in the corner regions map.

0.5 applyTemplate

This code applies the normalized cross correlation and computes the elements useful to build the rectangle around the object we are looking for.
The correlation is applied with the matlab function normxcorr2(template, image). The maximum correspondence is at the maximum point of the matrix returned by the function.
From the coordinates of the maximum value we can then define bottom left corner, width and height of the of region of the object we wanted to find. These values are returned in output.

Results

0.6 Harris Corner detection and applyTemplate

We are at the beginning of the implementation of Harris corner detector on the image "image i235.png". First of all we show the partial derivatives of the image (lx, ly) - Figure 1.
The we display the Gaussian filter and after the construction of the M matrix, we can show the Flat region of the image (Figure 2) in which both eigenvalues of this matrix will be small, because all terms will be small.
We also compute and display the "edge" region in which we expect to see one large eigenvalue associated with gradients at the edge and one small eigenvalue, because few gradients will run in other directions.
After the right threshold 0.3 valore_ max we compute the corner reagion. This is the reagion in which both eigenvalues will be large is shown and we can notice that the corners of the image are marked as white (Figure 3).*
Then we show the R map region (Figure 3).
As last result, using the function regionprops() we get the centroids and with a for loop we get them in the right position in order to detect properly the corners in the image (Figure 4).
The results of the NCC are tested by selecting templates of different sizes on image "ur_ c_ s_ 03a_ 01_ L_ 0376.png". We chose to use three rectangles of dimensions that are increased by a factor of 1.2 at each loop. Then we searched for the templates on the set of the given (six) pictures.
When the result is found, the code plots a rectangle around the result, and a cross over the point of maximum correnspondence. It is possible to also show the original coordinates of the cars obtained in the previous lab.
Moreover, we computed the mean of the computational time used by the cpu to run the applyTemplate function. We did it through the Matlab functions tic and toc. To better understand the correlation between the template size and the time spent we plotted the results in a graph that had, as axis, the template dimensions (in Pixels) and the time (in seconds). The time seems to slowly increase with the template size.
The results are good since the similarity between the images is very high. In fact, since this computation depends on the patch itself and not on the color channels, it is more precise. In the images are the results for two different template sizes. See the results in Figure 5-6-7.

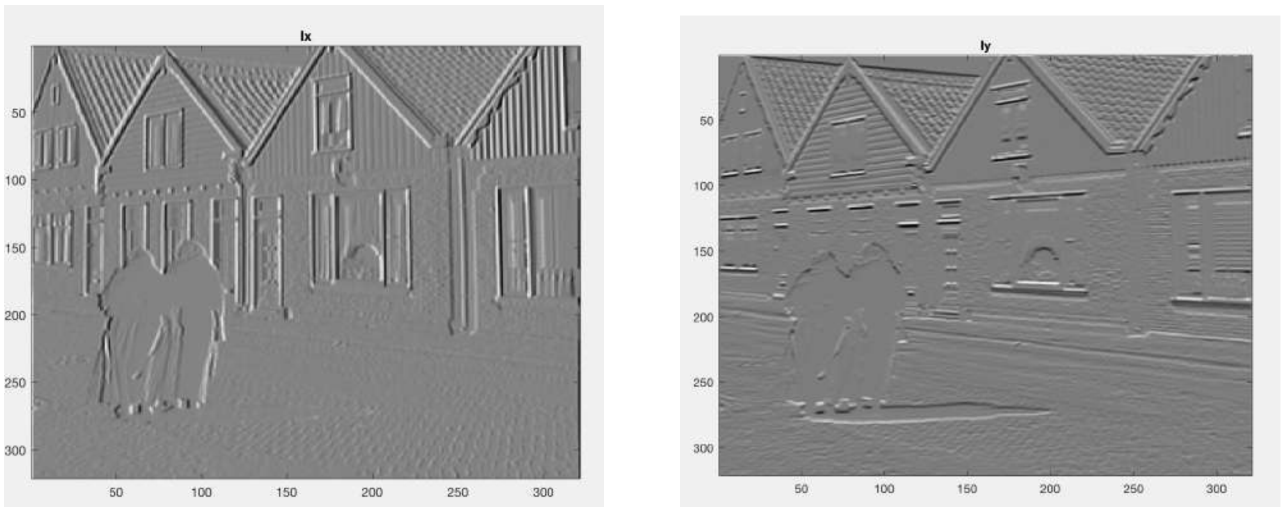


Figure 1

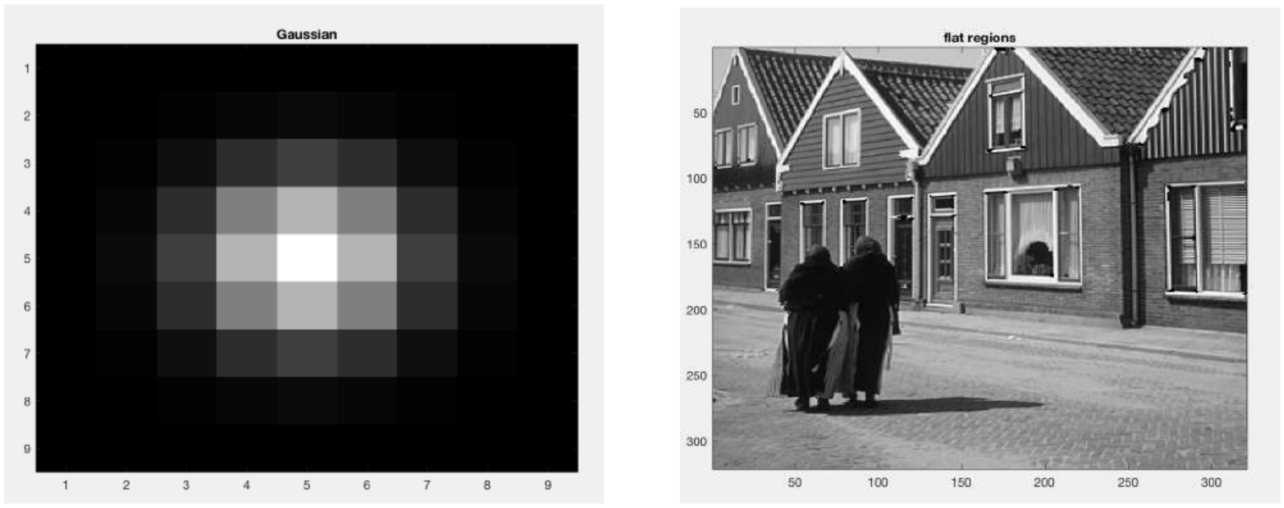
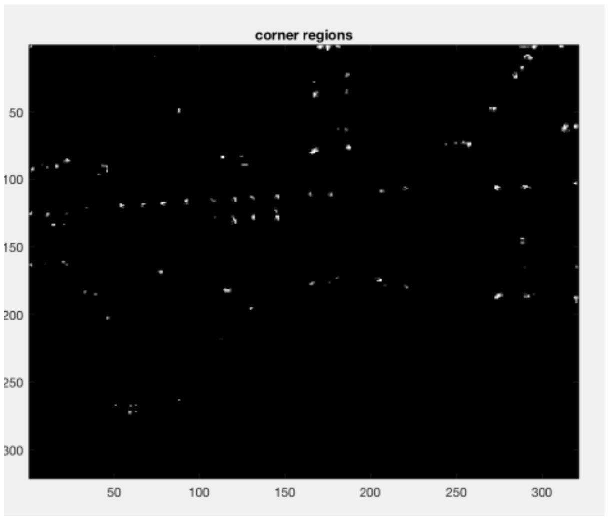


Figure 2



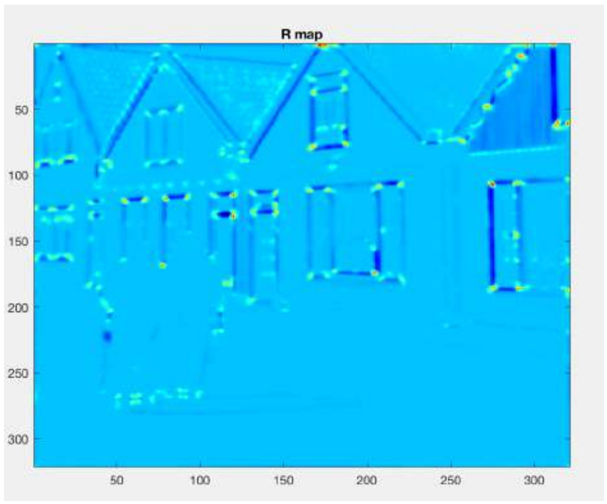


Figure 3

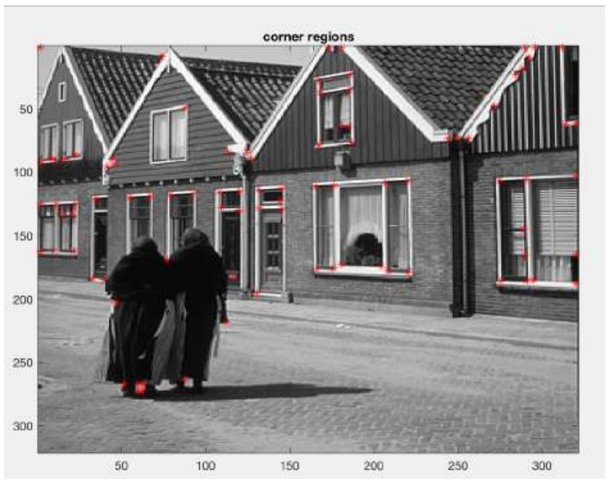


Figure 4

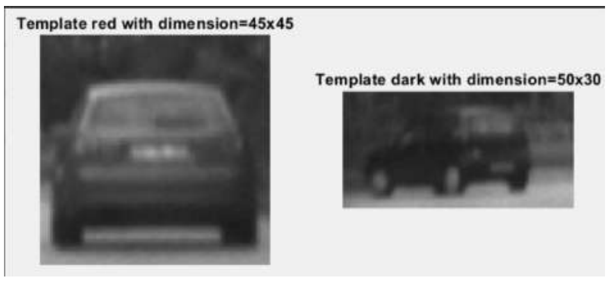

Template red with dimension=45x45
Template dark with dimension=50x30

Figure 5

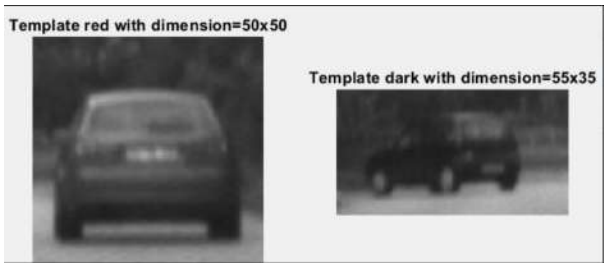
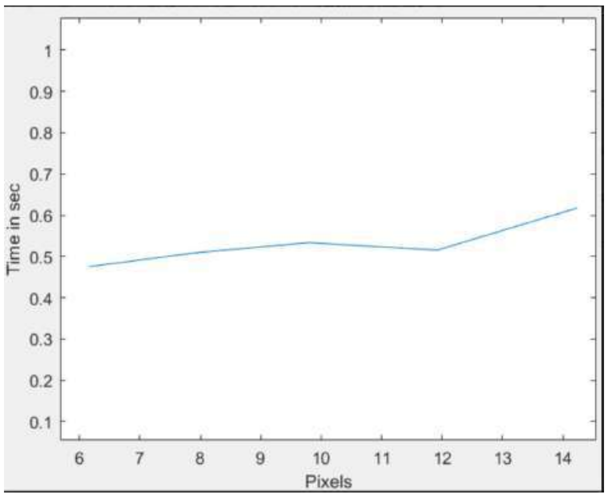

Template red with dimension=50x50
Template dark with dimension=55x35

Figure 6


Time in sec
Pixels
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
6
7
8
9
10
11
12
13
14