Computer Vision Laboratory Report n.5

NCC-BASED SEGMENTATION

May 14, 2019

Claudio Curti ID: 4216203
Nicola De Carli ID: 4198668
Alberto Ghiotto ID: 4225586
University of Genoa

Contents

Chapter 1

Introduction

1.1 Defining the objective

This project is the direct continuation of the previous one, its aim is to carry on the examination of object detection.
In order to achieve what mentioned above this experiment will extract a template from an image to apply the normalized cross correlation, with the intent of evaluating the correlation between the template and the image from which it was extracted along with some similar images.
With the purpose of highlighting the effects given by using different sized templates, this experiment will discuss the obtained computational time and the accuracy in each different case.
Eventually, the results will be compared with the one achieved in the previous laboratory.

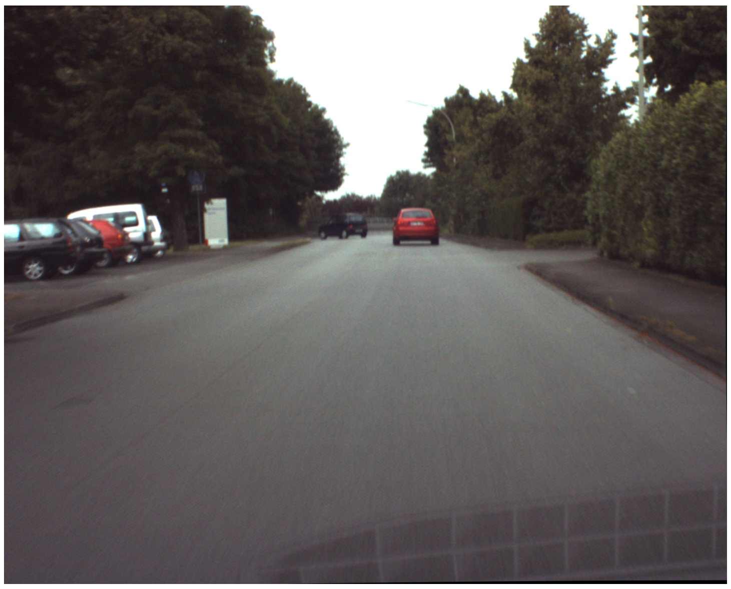

Figure 1.1: Main image used for the experimentation

Chapter 2

Theoretical Background

Convolution (and cross-correlation) with a filter can be viewed as comparing a little “picture” of what you want to find (in this case a window around the car) against all local regions in the image; such procedure is called template matching. In order to reach this goal, normalized cross-correlation is applied, this is obtained by subtracting the mean value of both the image and the template to themselves, and by dividing them for their standard deviation. The normalization is very important, thank to this, the correlation score is higher only when the darker parts of the template overlap the darker parts of the image and the brighter parts of the template overlap the brighter parts of the image, furthermore, it counteracts affine intensity change. Therefore it also enforces robustness to illumination changes. In order to apply the normalized cross correlation the MATLAB function ”*normxcorr2*” was used. This function takes as inputs the template and the image on which to apply the algorithm and returns a matrix containing the correlation values in the range [-1,1].

Chapter 3

Implementation and Results

The code is organized using a script, which loads the RGB images in the workspace and then calls all the developed functions. In order to carry out all the required manipulation the following functions were developed:

- **corrDetection**: which computes the cross-correlation between the image and a template passed as parameters. It then plots a rectangle corresponding to the size of the given template by finding the indexes of the highest value of correlation to use it as the top-left vertex of the rectangle.
- **compTime**: which computes the time taken by the function "*corrDetection*" to perform the normalized cross correlation between a given image and a template of the size specified by the input parameters.

3.1 corrDetection

This function takes as inputs the image and a template and gives as outputs the indexes of the pixel presenting the greatest similarity between the template and the image. It works by computing the normalized cross-correlation between the template and the image to then find the indexes of the greatest value in the resulting image. However, this image is bigger than the starting one due to padding, a technique used to compute the result of correlation (or convolution) at the margin of the image, for this reason to find the corresponding indexes on the original image it is needed to translate the indexes of half the sizes of the template. Eventually, the original image is showed with a star pointer identifying the location of best matching and a bounding box of the same size as the template. In the following images are displayed the results of the recognition based on NCC choosing templates with different dimensions.

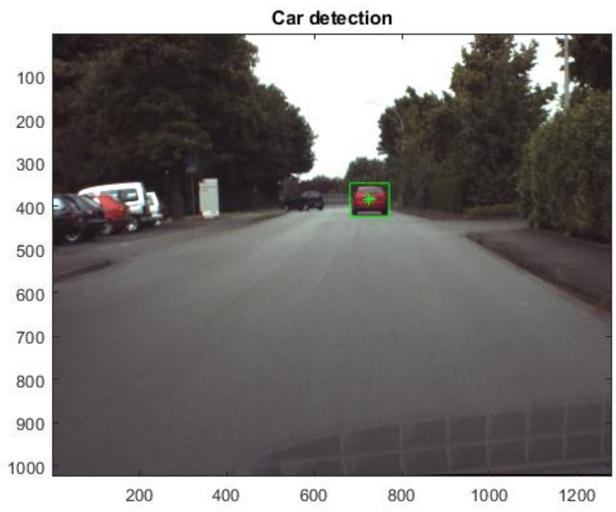

Figure 3.1: Object detection using a small template

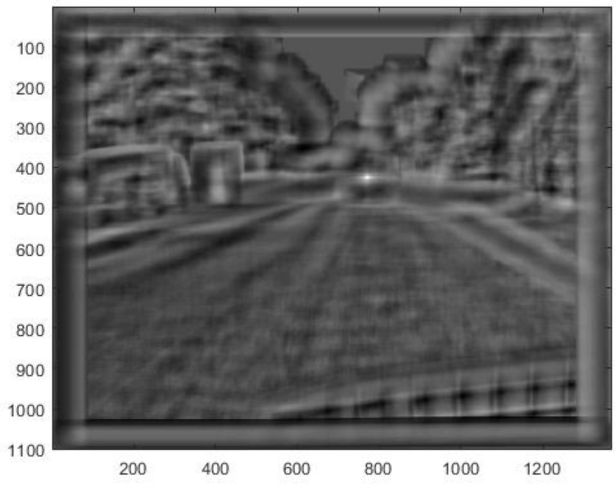

Figure 3.2: Normalized cross-correlation [90x79]

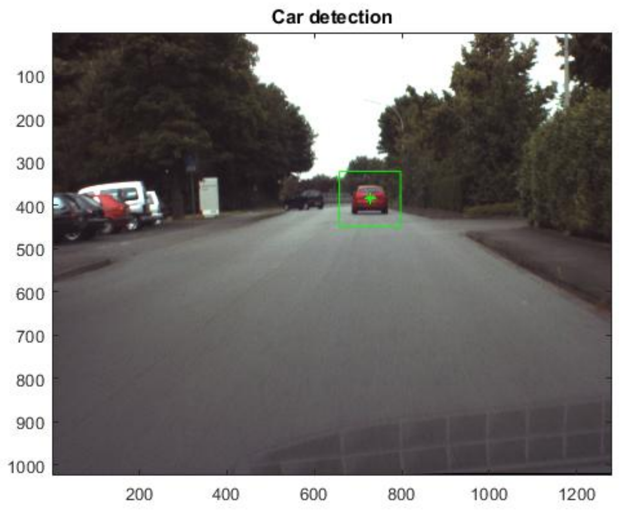

Figure 3.3: Object detection using a medium template

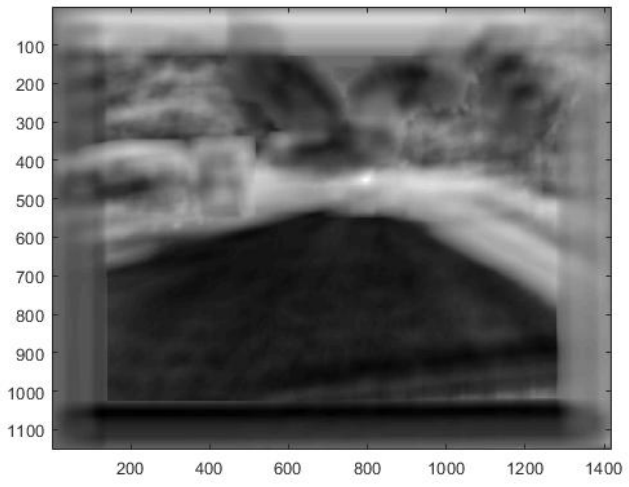

Figure 3.4: Normalized cross-correlation [290x279]

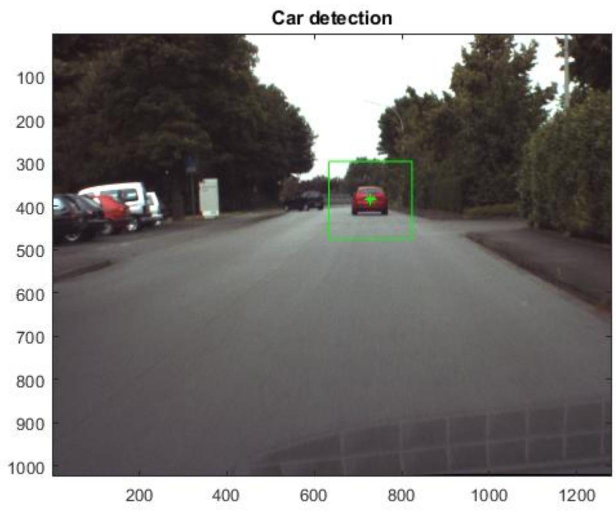

Figure 3.5: Object detection using a big template

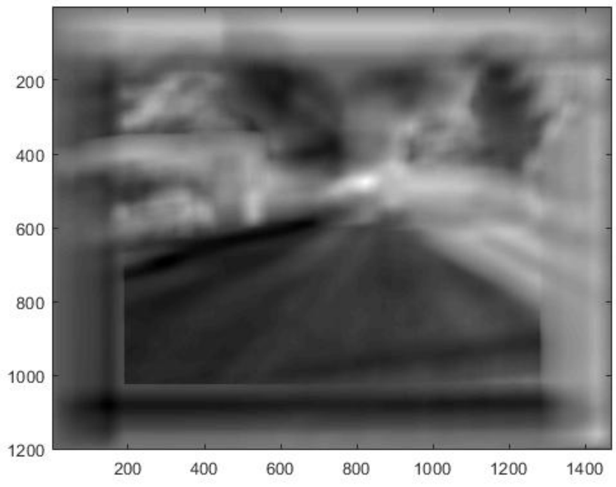

Figure 3.6: Normalized cross-correlation [390x379]

3.2 compTime

This function simply calls the "*corrDetection*" function a chosen number of times in order to measure the time of each execution and then average it. This function is used to compare the execution time for each different size of the template.

3.3 Results

Concerning computational times for different template sizes the expected results are that the larger the size, the longer it takes to calculate the result. This is caused by the higher number of operations required for the cross-correlation and also, to find the maximum of the result of cross-correlation more pixels are checked as a result of the effect of the padding. As a matter of fact, the effort due to this second reason, is probably quite negligible when compared to the first one. The results highlight that, as might be expected, the larger the size, the less accurate it is the result.

Hereafter the results are showed:

- The computational time for a template 90x79 is 1.196875e+00
- The computational time for a template 290x279 is 1.415625e+00
- The computational time for a template 390x379 is 1.664063e+00
- The pixel found by a window 90x79 is [726,385.5]
- The pixel found by a window 290x279 is [729,384.5]
- The pixel found by a window 390x379 is [729,384.5]

By using the NCC-based segmentation for object detection only one object is detected, instead, using the color-based segmentation the result is composed by all the objects in the image with a similar value of hue. Therefore to detect a particular object a specific environment is needed (without other objects with similar color and geometric shape in the background). Furthermore the NCC-based segmentation is less oversensitive to illumination changes but it is more susceptible to variations in the orientation of the object. The color-based segmentation has revealed to be much faster than the NCC-based one. Below the results of the color-based segmentation are showed:

- The computational time is 3.281250e-01
- The pixel found is [727.3825,385.7628]