NCC-based segmentation and Harris corner detection

Authors: Ierardi Ambra, Scorrano Andrea, Trovatello Alessandro, *Università degli Studi di Genova*

There will be only $one\ version$ of the code and the report submitted, since the group worked together over every part of the assignment.

ABSTRACT: This project has the objective to implement two fundamental algorithms of $Computer\ vision$: **NCC-based segmentation** and **Harris corner detection**. $NCC-based\ segmentation$, is an algorithm used for blob detection and template matching. After choosing a template (patch) of the specific desired object, the highest score area is detected basically following the same procedure as a filtration: the result is the highest where the image has the closest features to the template. This way the object will be detected, if the template is properly customized. Another way to detect objects in an image is the $color-based\ segmentation$. $Harris\ corner\ detector$, is an algorithm that has the goal of looking for corners in an image. This algorithm works computing a specified matrix M to get a R score map and searching the points where R>threshold. Using the $MatLab$'s function **regionprops()**, it is possible to detect all the points where the corners are present.

KEYWORDS: object and corner detection

1 INTRODUCTION

This assignment was first focused on the identification of objects, given a template image.
This was made using the **normalized cross-correlation**, a mathematical tool which allows to recognize the features present in a template image inside another image, if existing. The $object\ detection$ can be done also by using the **color-based segmentation**, an algorithm that is based on the subdivision of an image in HSV channels, which are perceptually meaningful dimensions, highlighting the image's characteristics of color changes.
These two methods bring to different results, further analyzed in this paper.
Another topic of interest in this project was the $corner\ detection$, which can be performed exploiting the $2D\ derivatives$ of the image itself, in order to find the local maxima, which can represent a corner, depending on their value.

2 REPORT TASKS

2.1 NCC-based segmentation

In this point it was required to compute the segmentation of the images, based on the Normalized Cross-Correlation (NCC) method.

2.1.1 NCC

The first step was to select a window (template taken from the rows 360:425 and columns 690:770), in which the red car was found, manually from the *"ur_c_s_03a_01_L_0376.png"* image and perform the Normalized Cross Correlation (NCC) in each image by using the built-in **normxcorr2()** *MatLab* function, where the inputs are the coordinates of the template window and the image in gray scale, in order to find the template in all the six provided images. The equation of **normxcorr2** is the following one:

$$\gamma(u,v) = \frac{\sum_{x,y}[\, f(x,y) - \bar{f}_{u,v}\,][\, t(x-u,y-v) - \bar{t}\,]}{\{\sum_{x,y}[\, f(x,y) - \bar{f}_{u,v}\,]^2 \sum_{x,y}[\, t(x-u,y-v) - \bar{t}\,]^2\}^{0,5}} \tag{1}$$

While, in the provided slides, the equation of the Normalized Cross Correlation is a bit different, as follows:

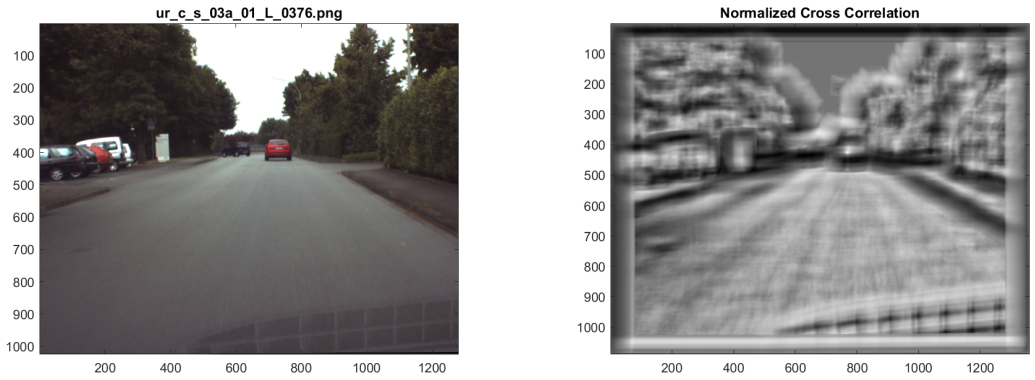
$$NCC(f,g) = C_{fg}(\hat{f}\cdot\hat{g}) = \sum_{[i,j]\in R} \hat{f}(i,j)\hat{g}(i,j) \tag{2}$$

where:

$$\hat{f} = \frac{f-\hat{f}}{\sqrt{\sum(f-\hat{f})^2}} \qquad \text{and} \qquad \hat{g} = \frac{g-\hat{g}}{\sqrt{\sum(g-\hat{g})^2}} \tag{3}$$

The output of **normxcorr2()** is the transformed image. The coordinates of highest score of the image, corresponding to the center of the detected object, are found looking for the indexes corresponding to the highest value of this image. After that, it was requested to do all this procedure also for the black car, whose template was taken from the row 370 to the 405 and the column number 575 to 635.

An example of NCC as below:



(a) (b)

Figure 1: (a) "ur_c_s_03a_01_L_0376.png" image. (b) Norm Cross Correlation computed on the red car of the left image. The "high score" is the white dot at the center of the red car.

2.1.2 Comparison with color-based segmentation

As second point, the previously obtained results were compared with the ones got in the activity of the Color-Based segmentation, fully completed in the function **color_based_seg.m**. With this method, it is possible to detect objects by exploiting the data of the HSV channels: the H one is the $Hue\ value$, which represents the property of a color which switches from red to green; the S channel is the $Saturation$ one, which depicts the characteristic of a color to change from red to pink; the last one, the V, is the $Intensity$ or $Lightness$ value, which stands for the property to transition from black to white. The **Hue component** can be used to detect an object in an image.
The first step to do is to convert the given image from the RGB values to the HSV ones to consider only the H one, for all of the 6 given images. Then, after choosing as reference the first image, which corresponds to the first frame of the moving car, it was necessary to select the area of interest. The target to be recognized was the dark car turning left in the background of the image. The corresponding pixels to be taken were the rows from number 390 up to 400 and the columns from 575 to 595. In this area, it was required to compute the mean value and the standard deviation, performed respectively with the $Matlab$ functions **mean2** and **std2**. These values were used to threshold the $Hue\ components$: all the values inside the range **m**-**s** and **m**+**s** were set to 1, while the others to 0, being **m** the mean value and **s** the standard deviation just computed. This way, only the pixels with values close to the mean of the $Hue\ component$ of the $template$, which was the zoom on the black car, were preserved.
As last point, both the binary images and the colored ones were displayed, highlighting with a star and a rectangular box the maximum area detected in the image thanks to the **regionprops()** $Matlab$ function, which corresponds to the considered car.
All these steps were repeated for the red car, using as Hue range $[0.97; 1]$, due to the fact that it is possible to see from the original picture that this car is close to the green trees, so the $Hue\ component$ is close to the maximum in this region.

2.1.3 Test with three different windows

In the third point of this assignment, it was required to choose three different sizes of the window centered around the black car and to discuss the computation time and accuracy effects.
All these windows were then tested on the initial image recalling the **template_matching.m** function.
To record the timing, the $MatLab$ functions **tic** and **toc** were used at the beginning and at the end respectively of the action to be timed.

- The first window goes from row 365 to 410, and from column 570 to 640, so it's slightly bigger than the original one, just 10 pixels more both for the horizontal and vertical dimension;
- the second one is from the 380^{th} to the 395^{th} row and from the 585^{th} to the 625^{th} column, so it's 20 pixels less than the first size on the rows and the same for the columns;
- the last window goes from the row 345 to the 430 and from the column 550 to the 660, so it's the biggest window among this group, being 50 pixels more both for the rows and for the columns with respect to the original template.

2.2 Harris corner detection

The second point of the assignment has the goal of implementing the *Harris corner detection*. This algorithm can be implemented using three steps:

- Compute the M matrix for all image windows to get their R scores, which is the *cornerness*, the value that identifies the likelihood of a region to be a corner;
- Find the points with a large corner response (R $>$ threshold);
- Take the points of local maxima of R.

The matrix M has the following structure:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \tag{4}$$

where I_x and I_y are the derivatives along x and y of the images. To calculate the derivatives, it has been used the convolution between the image and the *Sobel operator*, which has the following shape, respectively for the x and y direction.

$$\frac{\partial}{\partial x} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{5}$$

$$\frac{\partial}{\partial y} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{6}$$

One of the tasks of the assignment was to show the results of image's derivatives. Once obtained the derivatives, it was necessary to compute the products $I_x I_x$, $I_y I_y$, $I_x I_y$.

To compute the sum of products of derivatives at each pixels was needed to apply the convolution between a gaussian filter and respectively $I_x I_x$, $I_y I_y$, $I_x I_y$.

The following step was to find the points with large corner response. In order to do this, it was necessary to apply this formula:

$$R = det(M) - \alpha \, trace(M^2) \tag{7}$$

This operation has been done for all the elements of the matrix and the result is a R map, namely a matrix mapping the distribution of the R values. To find the corner, it was sufficient to apply a threshold to the R map, as the assignment text suggested, 0.3 times the maximum value of the R map. With this algorithm, it was possible to map a new image on a black background containing the corners, plotted in white, of the original image. Using this *MatLab* function

```
prop=regionprops(corner_reg>0,'Centroid');
```

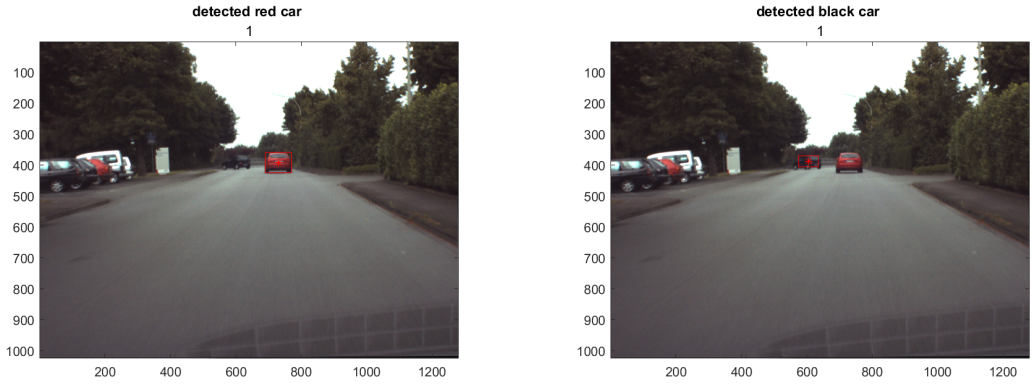
it was possible to detect regions with specific properties, in this case the ones that have a value higher than zero. This function returns a vector containing all regions that satisfy that condition. Iterating with a for loop on the vector, it was possible to plot the points where the corners are present.

3 RESULTS

3.1 NCC-based segmentation

3.1.1 NCC

In this first step of the assignment, we derived the template of the red car from the *"ur_c_s_03a_01_L_0376.png"* image and used that to compute the template matching in all the six images. This was done also for the black car. The following images represent the template matching for the respective cars:



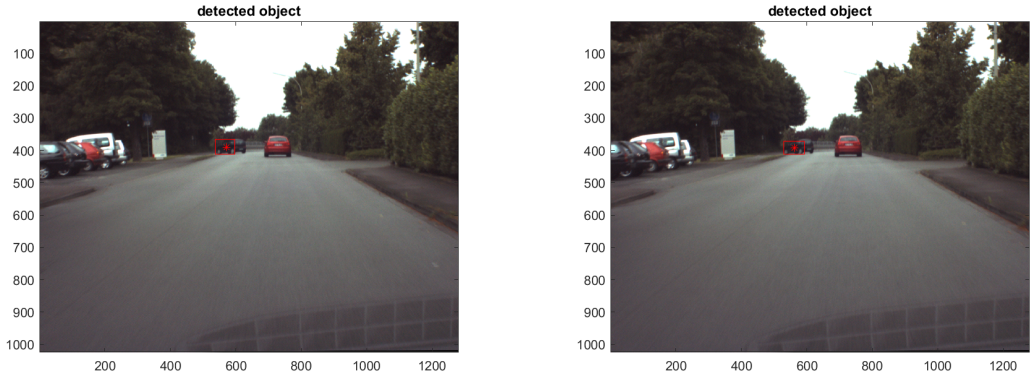
(a) (b)

Figure 2: (a) Template matching of red car. (b) Template matching of the black car.

It is possible to notice that, with the chosen size of the template, both cars are well detected.

3.1.2 Comparison with color-based segmentation

Looking at the images resulting from $NCC - based\ segmentation$, and comparing them with the $color - based\ segmentation$ ones, it is possible to see some differences focused on the **black car**, especially in the last three images: with the second method, these objects are not detected in an accurate way, because the maximum of the area is not exactly in the center of the black car. For example, for the fourth and the sixth images, the plot is as below:



(a) (b)

Figure 3: Color-based segmentation computed on the black car. (a) Fourth image. (b) Sixth image.

It is possible to see on the fourth image that the center is moved a little leftwards and the rectangle is squeezed in the width, while for the sixth image it is moved a little downwards and the drawn centroid has now squeezed height and base. Hence, overall for the black car it is better to use the $template\ matching$ technique, with the $NCC - based\ segmentation$.
Regarding the **red car**, the matching made with the $color - based\ segmentation$ brought results as good as the ones given by the other method.
It is possible to notice just one detail: with the first method in figure 2 (a), the rectangle has the center moved slightly to

the left with respect to the other one as below:

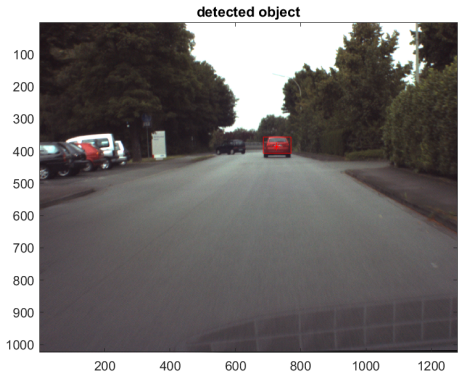
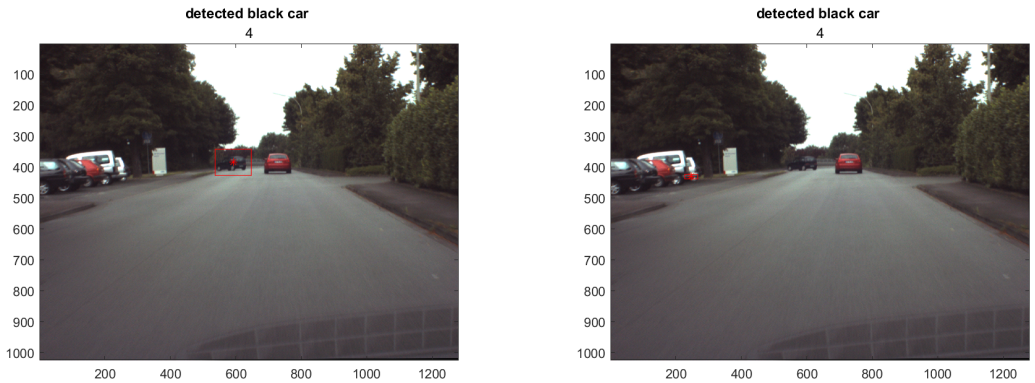


Figure 4: Color-based segmentation compute on the red car in the first image as figure 2 (a).

Also, in this case the rectangle is quite adherent to the car, excluding the wheels, while with the NCC method these ones are included in the rectangle. This is due to the fact that the latter algorithm requires the choice of the size of the template, so it was made by hand, looking at the original image, so also the wheels were taken into account in this step, while the $color-based\ segmentation$ proceeds skipping this action and just relies on the presence or not of the interested intensity of the Hue component of the picture: this is why the black wheels were excluded.

3.1.3 Test with three different windows

In this part of the assignment, the goal, as previously mentioned, was to compare the results obtained with different sizes of the window used as template for the black car. The comparison based on the **computation time** was performed also on the size used at the previous point, so over all there will be four contribution to this analysis.
This first parameter is not very reliable, since every time the code is run the results are different and, above all, the trend changes at almost each trial. Some times it may happen that the algorithm performs faster with the first size, which has a base of 60 pixels and a height of 35 units, or with the second, which is 10 pixels bigger for both dimensions with respect to the first one, or the third, which is the smallest one since it is 20 pixels less in both dimensions, again with respect to the first template: it is quite aleatory. The only stable result is the fourth one, which is always the slowest one, due to the magnitude of the used window: it was chosen from the row number 345 to the 430 and from the column number 550 to the 660. The first and the second window are very similar, speaking of size, so it is acceptable that the computational times are very close. The odd situation is with the third one: since it is the smallest one and covers only approximately half the pixels of the car, it brings problems in the detection of the car as in Figure 5 (b), which is not easy as in the previous cases.



(a) (b)

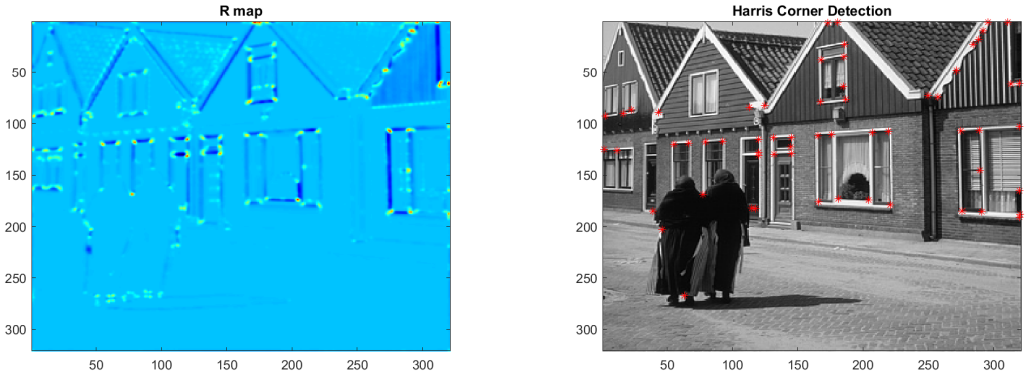
Figure 5: (a) 85x110 window size (largest). (b) 15x40 window size (smallest).

The second parameter examined was the **accuracy of detection**: the first, second and fourth windows work well, since the car is detected in all the frames and, zooming in the images, it is possible to see that the marked centers are approximately matched to the center of the car. The third template does not work well: in the second half of the frames the car

not detected, it commits a mistake and recognizes it in the car headlight in the left of the image as in Figure 5 (b). This happens because the size of the window is too small, so the details in that small part of the image can be found also in other parts of it, as happens in this case. In the first three images, where the algorithm works correctly, the center of the car is approximately overlapped to the marker.

3.2 Harris corner detection

The following images show the R map and the relative image after using the Harris corner detector. It's possible to note that in the R map the corner are highlighted with a yellow color: these regions have significant change in every direction. Once applied a threshold to locate the areas relative to the corners, and used the *regionprop()* function with the appropriate conditions, it is possible to see the results on the other image, that shows the corners of the input image.



(a) (b)

Figure 6: (a) R map. (b) Harris Corner Detection.

4 SUMMARY

This assignment was focused on **NCC-based segmentation** and **Harris Corner Detection**.
First, the normalized-cross correlation was computed, in order to get the coordinates of highest score of the match with the template, which were useful to center the rectangle representing the correspondent object on the car in all the images. About this task, it is possible to remark that, for the *red car*, this algorithm works fine, as much as the *color-based segmentation*, while for the *black car*, it is definitely better the first one.

The second point covered by this laboratory was the corner detection, performed thanks to the *Harris Corner Detector* algorithm, which computes the derivatives along x and y of the image, creates a *R map*, exploiting the *M matrix*. Using a threshold and identifying the points that have a score of the R map higher than the threshold, it's possible to detect the corners.
A possible future application is to use this algorithm to find the correspondences between two images.