University of Genova

Department of Computer Science, Bioengineering, Robotics and System Engineering (DIBRIS)

COMPUTER VISION

Lab Session n.5

7852527 - Mattia Tinfena
6504193 - Elisa Martinenghi
7687956 - Chiara Buono

Introduction

In computer vision, recognizing an object is a key feature. In this report, we will analyze some of the most common techniques to detect objects in 2D images, starting with the blob detection through the Normalized Cross-Correlation (NCC). After that, we will analyze how different sizes of template will affect bolb recognition and the computation time. Then, we will compare these results with the ones obtained while detecting blobs with the multi-resolution pyramid. We will use this technique to detect a red car and a dark car in different images, analyzing as different algorithms detect these cars in different ways.

In the second part of the report, we will evaluate a technique to detect keypoints in images, in particular corners. These are important because they are invariant for some properties such as: translation, rotation and partially color's intensity change. To do this, we will use the Harris corner detection algorithm.

Chapter I: NCC-based segmentation

In this first part of the lab, it was requested to apply the Normalized Cross Correlation (NCC) to six images representing two moving cars.

First of all, we convert the images in a gray scale; then, we extract three templates of the black car (T_{11}, T_{12}, T_{13}) and one of the red car (T) from the first image (*ur_c_s_03a_01_L_0376.png*), allowing us to detect them. The templates are created specifying the coordinates of the points which allows to build a bounding box around the two cars. Once the templates and have been acquired and a vector t is created, containing them all, we upload the whole set of images.

For each car, the implemented procedure is the same. Firstly, we acquire the vector containing all the images and, on each of them, the NCC is applied; this passage is implemented in a specific function called *ncc*. Here, the input image is converted to gray scale and the *normxcross2* function is applied, giving as output a binary image, called **score map**, which shows a white spot in the region that best matches the input template and the image. Then the *ncc* function returns the coordinates of the point with the highest intensity in the score map.

The original images are plotted overlaying an asterisk that correspond to the specific point mentioned before and a rectangle around it, as we can see in figure 1.1.

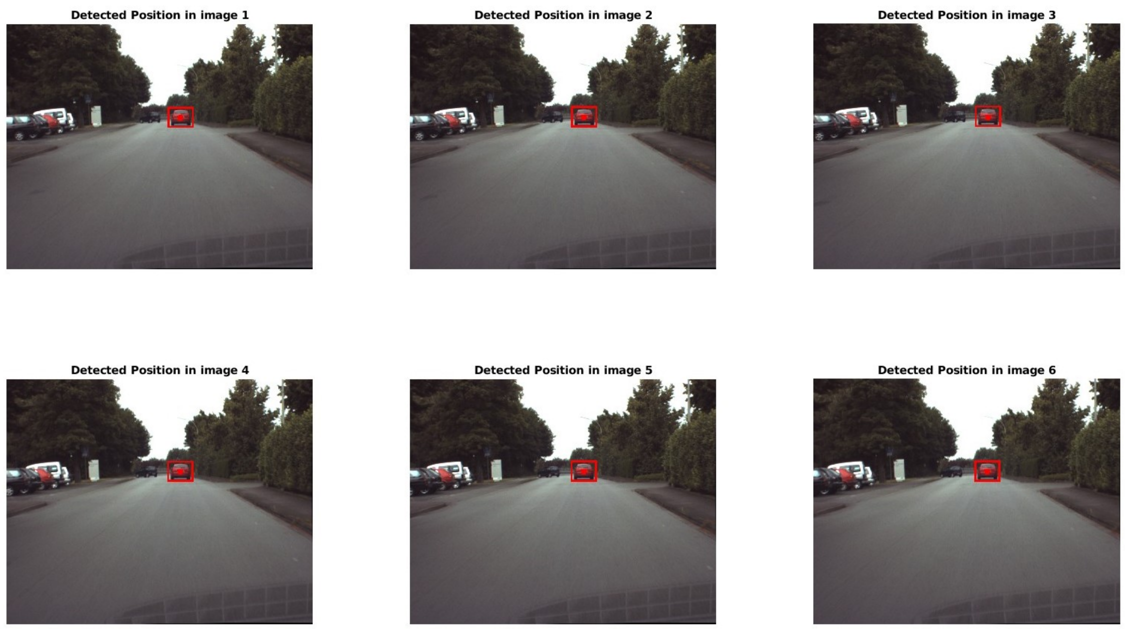
Detected Position in image 1
Detected Position in image 2
Detected Position in image 3
Detected Position in image 4
Detected Position in image 5
Detected Position in image 6

Figure 1.1
Red car detection with NCC

Chapter II: Harris corner detection

The Harris Corner Detector is a method used to detect corners by identifying points with significant gradient changes in all directions. Specifically, a corner is a point whose local neighborhood exhibits two distinct and dominant edges. Accordingly, the Harris Corner Detector finds these differences in intensity for a given shift (u, v) in all possible directions, which can be expressed mathematically in terms of both shift and derivatives for small motions as follows:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2 \approx \sum_{x,y} I_x^2u^2 + 2I_xI_yuv + I_y^2v^2 \quad (2.1)$$

In fact, estimating derivatives is generally important, since sharp changes in an image can be associated with objects boundaries. Using Sobel filters, we can compute the partial derivatives of the image in the x and y directions, as well as their second derivatives (see figure 2.1).

$$dx = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad (2.2) \qquad dy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (2.3)$$

After applying a Gaussian filter to reduce noise, the structure tensor *M* and the response of the Harris detector *R* can be computed for each pixel. Specifically, the matrix *R* has been computed from *M* using the formula:

$$R = det(M) - k \cdot tr(M) \quad (2.4)$$

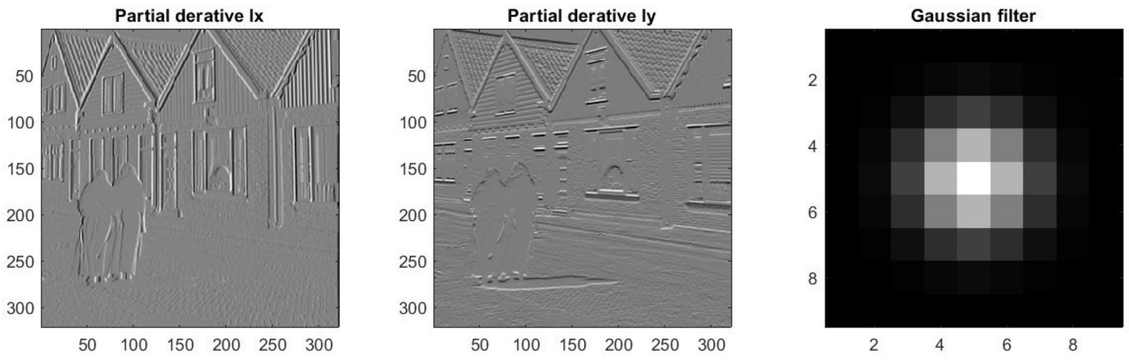

Figure 2.1
Partial derivative and Gaussian filter

After obtaining these matrices, the threshold used to detect the corner regions was set to 30% of the maximum value of the R map. Then, using the ”regionprops” function, the centroids of each corner region were successfully identified and plotted over our image, as we can see in figure 2.2.

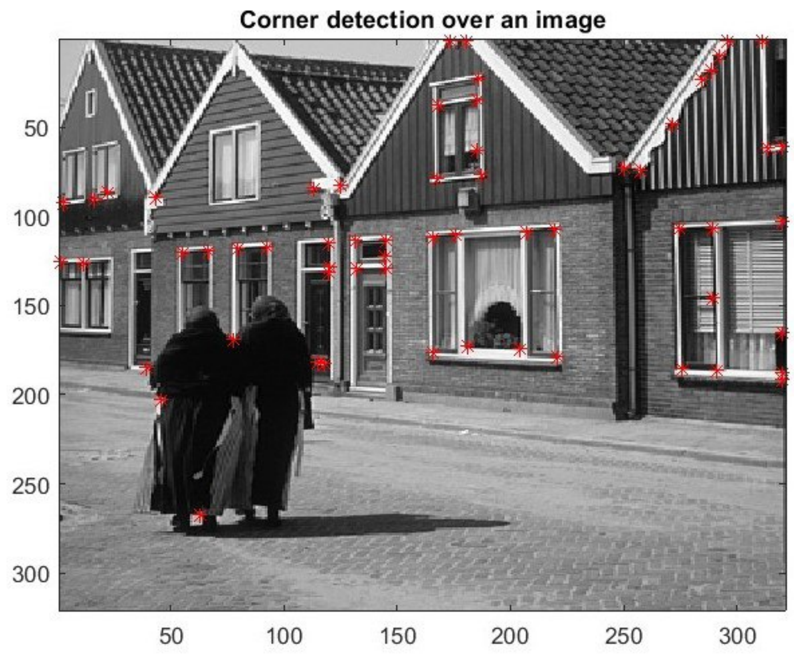

Figure 2.2
Harris corner detection

Conclusions

For both the red and the black car we saw that the templates used worked correctly; in fact, the bounding box and the centroid are placed correctly on the original image.
We also plotted the results obtained using different sizes of templates with both NCC and color-based segmentation in order to be able to compare them.

It was observed that NCC works nicely every time the template contains all the object to be identified; in particular, we observed that the bigger the template gets, the more precise it becomes. Talking about the computational time, we noticed that it grows as the template gets bigger, as we can see in table 2.1.

Iteration	Computational Time
Regular template	1.0748 seconds
Small template	0.88257 seconds
Large template	1.1577 seconds

Table 2.1
Black car calculation time with model size

On the other hand, the color-based segmentation works fine with smaller templates, since they are more likely to present a more uniform pattern in terms of color; in fact, having a template with less variations of intensity makes the computation more precise, since this method is based on color recognition. For the reasons reported above, we can affirm that, while CBS works better with small template, when we have to work with bigger ones, NCC is the best option (figure 2.3).

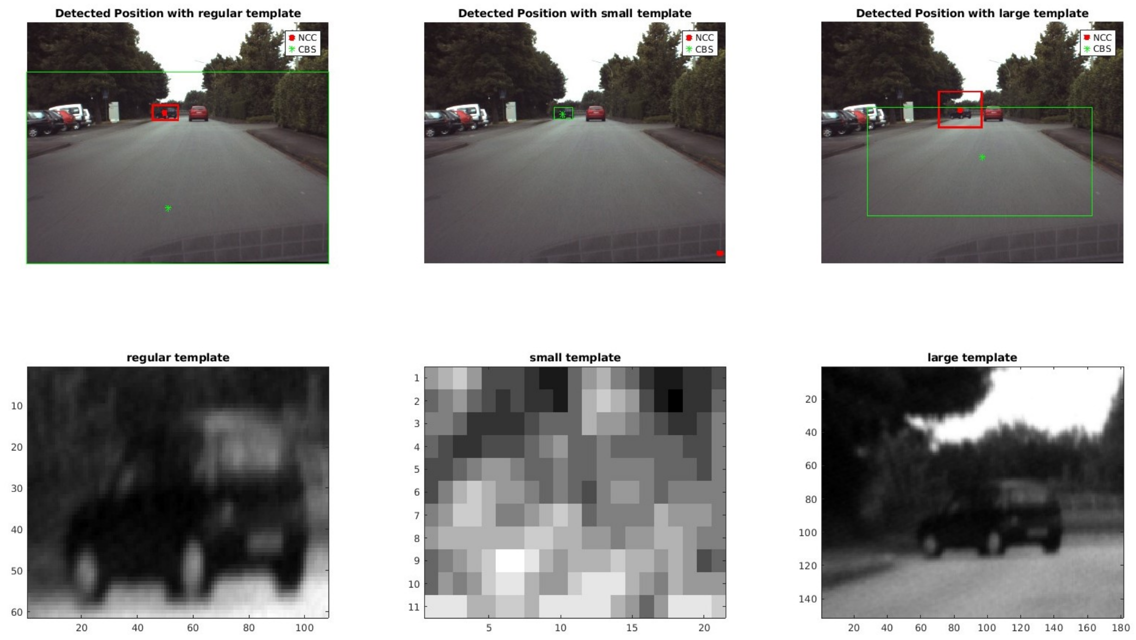
Detected Position with regular template
NCC
CBS
Detected Position with small template
NCC
CBS
Detected Position with large template
NCC
CBS
regular template
small template
large template

Figure 2.3
Comparison of NCC and CBS with different template sizes

Passing now to the computational time, we analyzed both algorithms with the same templates. A particular focus should be paid when considering the template with the red car in it. In this case, the car is easily detectable with both methods, being the only red object in the image; however, the computational time of the two algorithms is very different: NCC takes just 0.09 seconds to compute the result, while CBS requires 0.22 seconds.

Eventually, we can affirm that Harris-corner detection works fine in general. Nevertheless, some points detected do not correspond to any corner, as can be seen around the two people in the image, while there are some corners that have been omitted. This behavior of the Harris-corner detection can be traced back to the way the algorithm works; in fact, it tends to detect more the points whose detected edges follow the directions of the two derivatives, which in this case are along the x and y direction.