COMPUTER VISION LABORATORY

SESSION 5: NCC-based segmentation

June 17, 2020

Arlotta Andrea 4089306
Cantoni Francesca 4698289

Contents

Chapter 1

Abstract

The aims of this laboratory session are to detect a given template inside an image, using the NCC[1]-based segmentation, and compare the results with the ones obtained using the color-based segmentation (performed in Lab4).

Specifically it was required to:

1. apply the NCC operation between each one of the input image[2] and a specific template, obtained by cutting from the first image a window around the red car on the street

2. show the position of the maximum on the score map and display on the original input image a box with the same size and position of the detected template

3. perform the previous steps using three different sizes of template (each one focuses on the red car on the street) and discuss the results in term of computation time and accuracy detection

4. compare the results with the ones obtained using the color-based segmentation

[1]NCC is the acronym of normalized cross correlation operation.

[2]The considered input images was previously converted into gray levels in order to execute the NCC operation.

Chapter 2

Theoretical information

2.1 Template matching

The main idea of template matching is to find a specific template inside a bigger image. This operation can be view as a convolution between an image and a filter.

Recalling from the previous laboratory session the convolution can be performed as follow:

$$O(u,v) = I \circledast H = \sum_k \sum_l I(k,l) * H(u-k, v-l) \tag{2.1}$$

where I is the input image, O the output one and H the spatial filter.

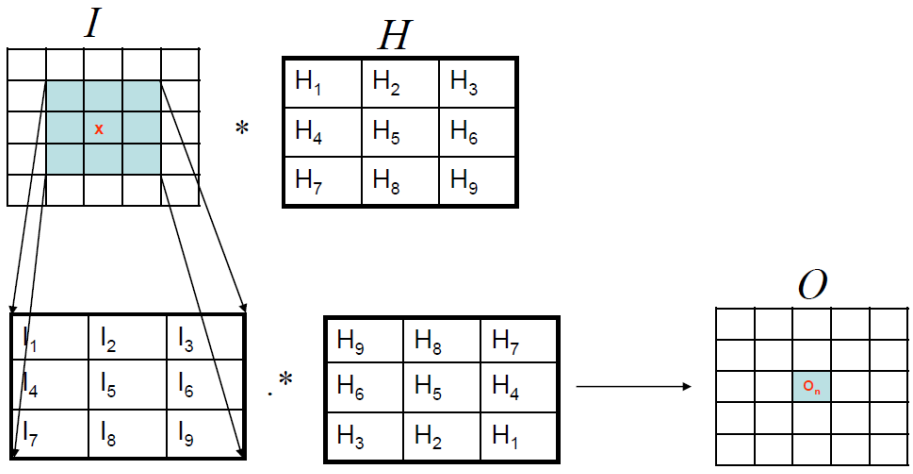


Figure 2.1: Visual representation of one step of the convolution process

For sake of clarity we will proceed to explain the template matching through an example[1].

Considers the following generic image and extract from it a small region that, from now on, will be our template.

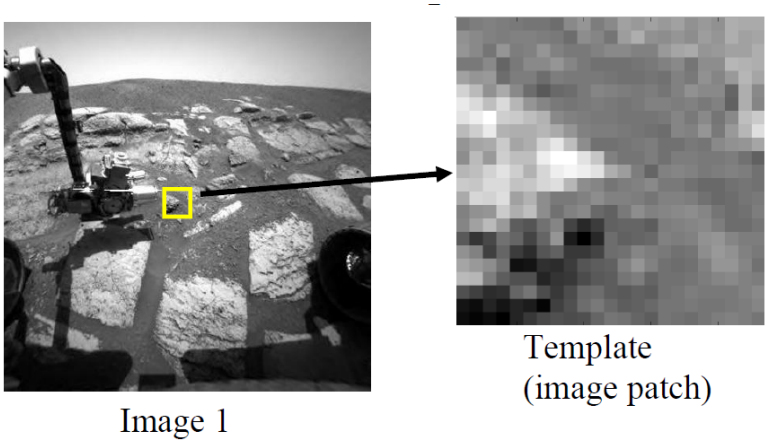
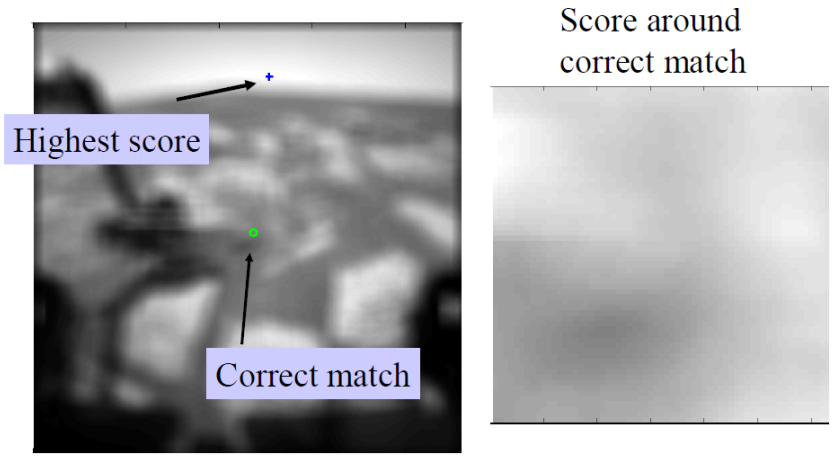


Figure 2.2: Extracted template

Performing a pure cross-correlation between the two images leads to an incorrect result as can be seen from the score map[2] below in which the *highest score*, that represents the position of the perfect match, is depicted in a different places wrt the real *correct match*.



It is important to highlights that the score map is depicted as follow:

- **1** → perfect match
- **-1** → completely anti-correlated

[1] From now on we will use the cross-correlation instead of the convolution remembering that the only difference between the two relation is that the offset in H is reversed wrt to the other.

[2] The score map looks like a blurry version of the original image due to the cross-correlation operation.

In order to explain the reason of the above result it is necessary to consider two different generic gray value filters, as follow:

a	b	c
d	e	f
g	h	i

ⓧ

v	v	v
v	v	v
v	v	v

a	b	c
d	e	f
g	h	i

ⓧ

2v	2v	2v
2v	2v	2v
2v	2v	2v

Figure 2.3: Two generic example of raw correlations between a small region of the original image and a filter with constant gray values

By performing these two operations it is possible to deduce that the second one leads to an higher result regardless the template.

$$2v*(a+b+c+d+e+f+g+h+i) \;>\; v*(a+b+c+d+e+f+g+h+i) \qquad (2.2)$$

This issue can be partially solved by subtracting off the mean value of the template from the original input image in such a way to obtain that the score map presents high values only when darker parts of the template overlap darker parts of the image, and brighter parts of the template overlap brighter parts of the image.

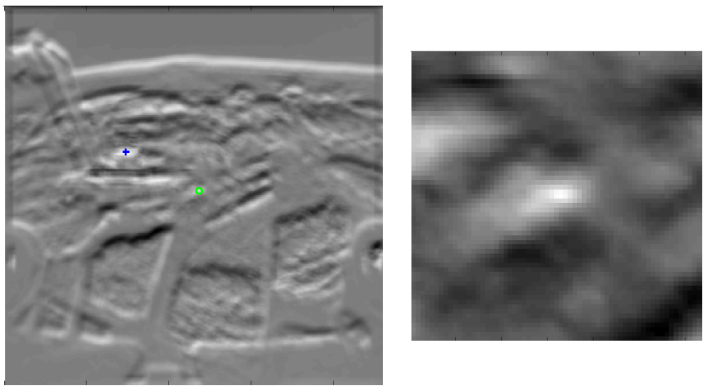


Figure 2.4: Zero-mean template

From the previous figure it is possible to notice that the highest score map still not match the real correct match but the distance between the two has decreased.

2.1.1 SSD method

The **SSD**[3] method allows us to fully solve the previous issue by considering these new relationships:

$$SSD = \sum_{[i,j]\in R} [f(i,j) - g(i,j)]^2 = \sum_{[i,j]\in R} f(i,j)^2 + \sum_{[i,j]\in R} g(i,j)^2 - 2C_{fg} \tag{2.3}$$

$$\text{and} \quad C_{fg} = \sum_{[i,j]\in R} f(i,j)\ g(i,j) \tag{2.4}$$

in which f function represents the template, while g function represents the considered window inside the original input image (that has the same size as the considered f).

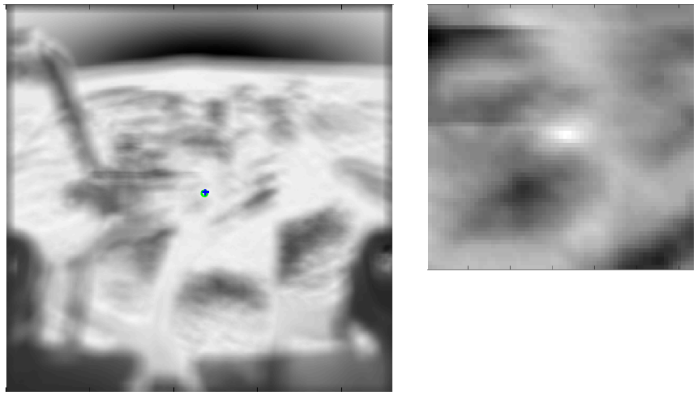


Figure 2.5: Score map using SSD method

This method is not invariant to intensity changes and this could be an issue in case of scene imaged by different sensors, or under different illumination intensities. This is due to the fact that the *SSD* and the C_{fg} can be large for windows representing the same area in the scene.

However a solution of that problem can be provided by the next method.

[3]SSD is the acronyms of Sum of Squared Differences.

2.1.2 NCC method

This method is based on the fact that, before comparing the pixels in the windows, they are normalized by subtracting the mean of the patch intensities and dividing by the std.dev as follow.

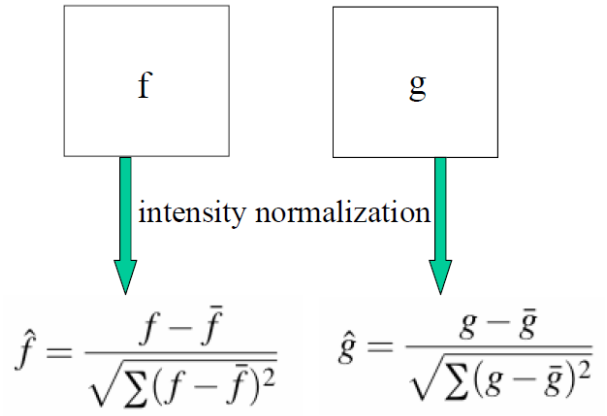


Figure 2.6: Normalized cross-correlation

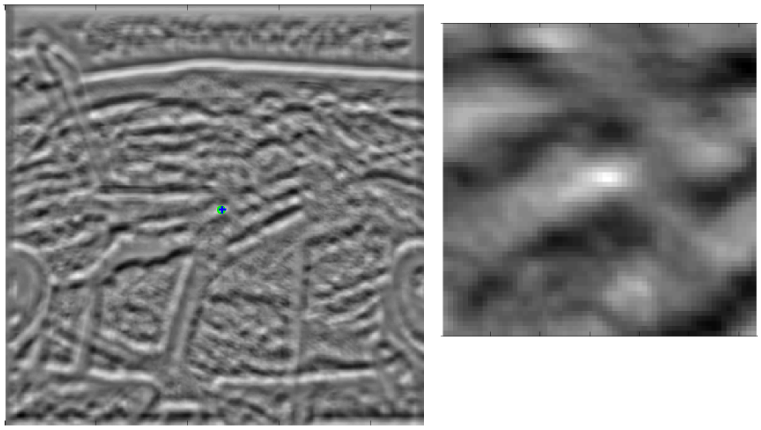


Figure 2.7: Normalized cross correlation

It's now possible to conclude that the NCC method leads to:

- keep the coincidences between the highest score and the correct match
- less chances of getting a wrong match

Chapter 3

Matlab

3.1 Function tree

The code consists of a *main.m* script that runs all the following subfunctions:

1. *BIG_template.m*
   performs the NCC-based segmentation considering a template that keeps into account the red car plus some external information around the latter. Inside this subfunction it is also computed the accuracy matrix as the difference between the given template and the correspondence real window detected inside each one of the original input images

2. *MED_template.m*
   executes the same operations of NCC-based segmentation and accuracy computation. The considered template is more focused on the red car

3. *SMALL_template.m*
   same as the previous two subfunctions but the template contains only information about the red car

Chapter 4

Conclusion

4.1 Execution time

Profile Summary

Generated 12-May-2019 20:26:55 using performance time.

Function Name	Calls	Total Time	**Self Time***	Total Time Plot (dark band = self time)
...inerFactory$FigurePanelContainerLight (Java method)	62	5.863 s	5.863 s	
big_template	5	22.066 s	5.618 s	
Text.Text>Text.get.SubplotAdjustment	62	5.135 s	4.974 s	
med_template	5	21.013 s	4.970 s	
small_template	5	19.926 s	4.820 s	
...ut>LocalCallgenerateGraphicsOutput	31	3.829 s	3.768 s	
fft2	30	3.445 s	3.445 s	
normxcorr2	15	13.046 s	3.419 s	

Figure 4.1: Performance time

The table below shows how much time is spent to execute the three main subfunctions that are inside the script. We can observe that, by changing the dimension of the template employed, the execution time varies proportionally to the number of the template image's pixel; this because the more are the pixel the more are the comparisons (pixel-by-pixel) made inside the NCC process.

4.2 Results

Figure 4.2 shows all the five input images used to test our code, while **Figure 4.3** represents the three templates obtained by cutting the first image with three different sizes. These latter show the same subject, a red car, but each one takes into account different levels of information about the surrounding environment.

Figure 4.2: All original input images

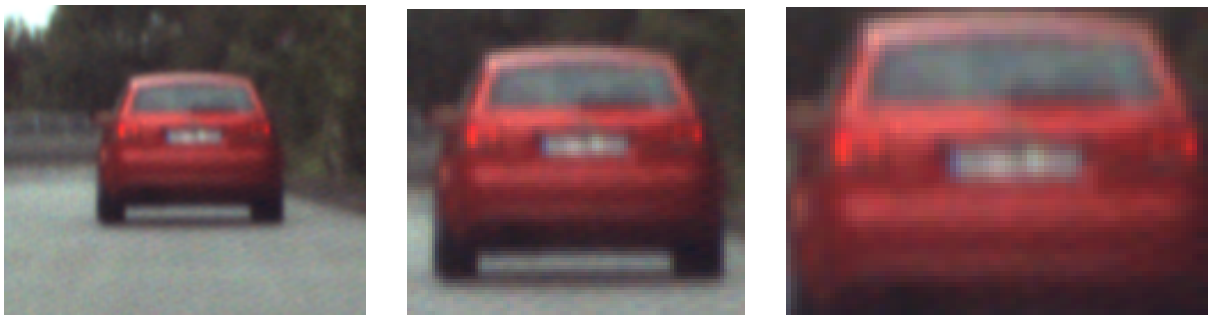


Figure 4.3: All templates from the big one (left) to the small one (right)

The implemented code correctly detects the position and the dimension of the big template, as it can be seen from the figure below. This is based on the fact that the position of the maximum inside the score map (depicted inside the subplot as *NCC operation*), appears in the correct position so in the up-left corner of the desired template.

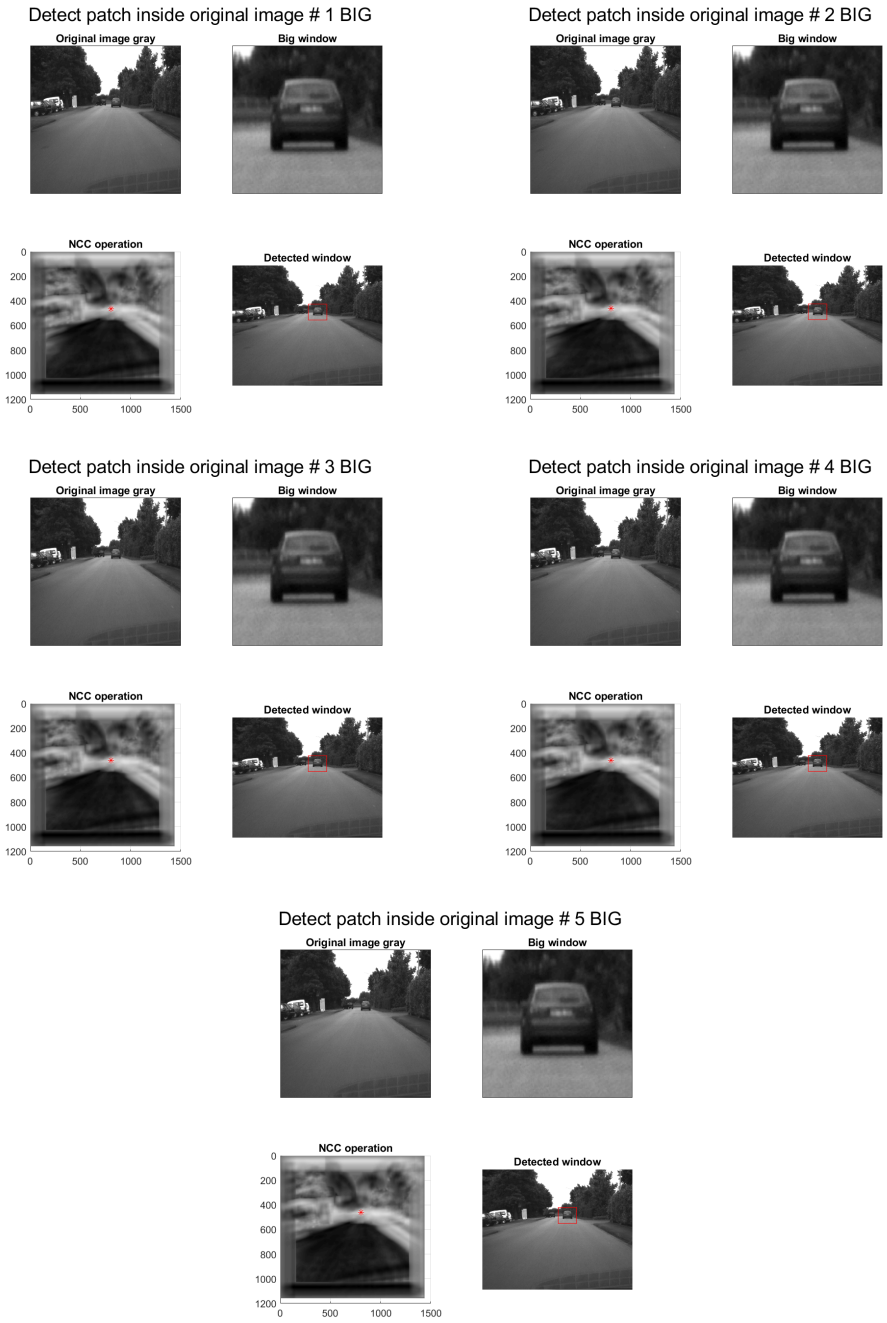


Figure 4.4: Big template detected inside each input image

Considering the accuracy aspect results, we can subdivide it into two main categories:

1. **perfect match:** the detected patch, obtain by cutting the recognize template[1] inside the original image[2], compared with each one of the template gives a constant zeros value matrix because the two considered matrices have exactly the same values.

2. **similarity image match:** the detected patch compared with each one of the template gives a non zero matrix due to the fact that the two considered matrices have slightly different values

Representation of *perfect match*:

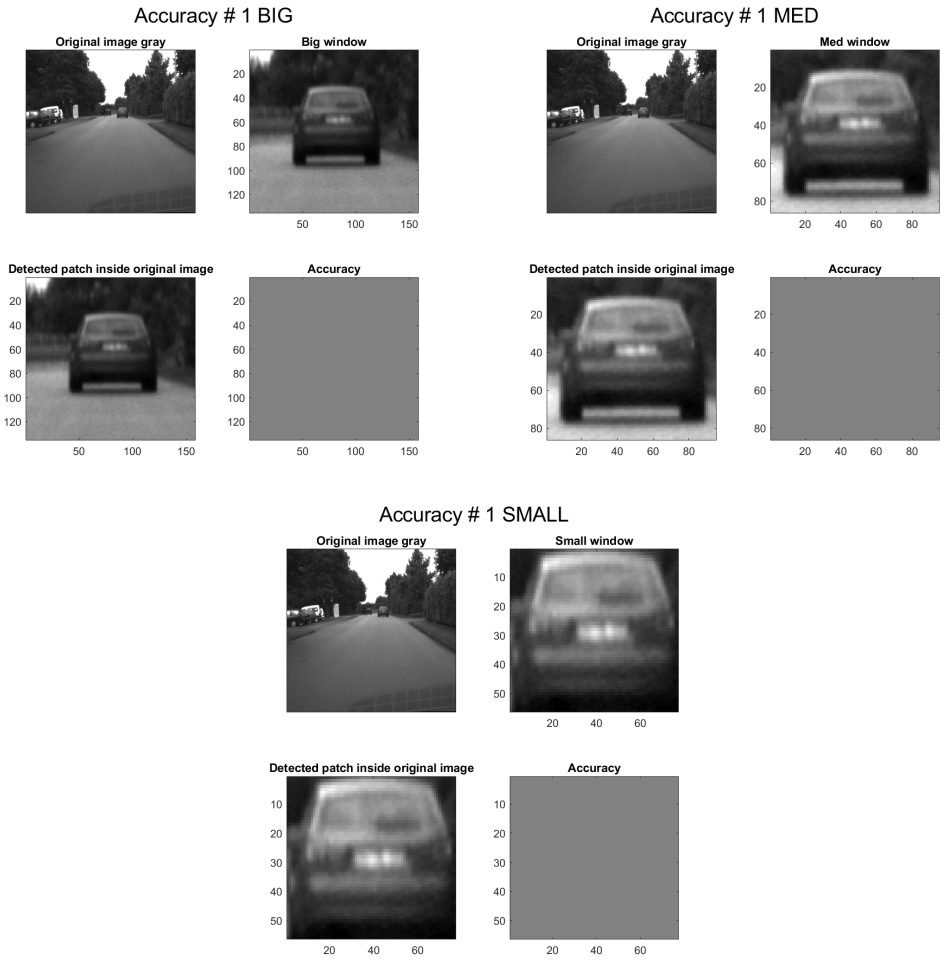
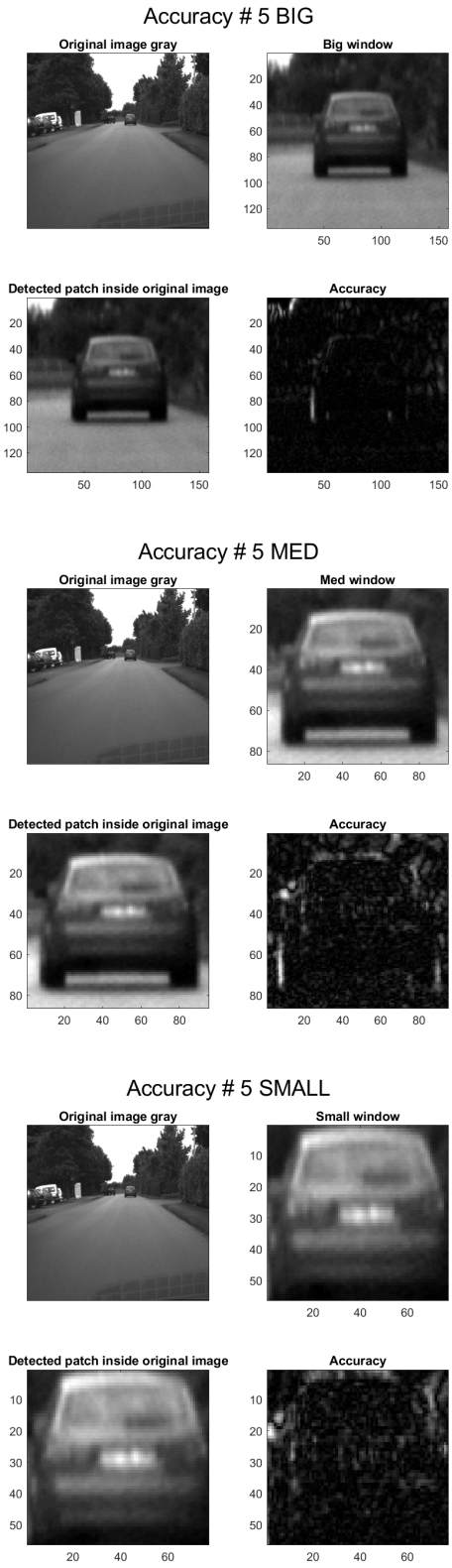


Figure 4.5: Accuracy matrices computed considering image 1

[1]Just shown in **Figure 4.4** .
[2]This result is represented through the third image of each subplot.



The results obtained using all other images are example of *similarity image match*. We decided to show only the outcomes obtained by considering the fifth image because in our opinion are the most significant ones.

Observing the aside subplots it is possible to say that the template containing other elements beyond the subject may bring to an incorrect detection due to the fact that the environment that surrounds the subject may be completely different from the template image to the considered one. On the other hand a template image too zoomed on the subject may give unpredictable results: it could happen that the template characterizes very well only a detail and not the entire subject (i.e. taillight and car plate).

A further comment on this algorithm can be done making a comparison with the NCC-based segmentation result and the color-based segmentation one, implemented in the previous lab session.
Since the blob segmentation algorithm relaying on color pattern template to detect the desired subject, the results obtained are acceptable assuming that the subject has a precise color and the environment does not disturb the detection (i.e inside the image there are other red objects comparable with the desired one). A positive aspect of this method is that the subject, if is well characterized by a specific pattern, can be detected even if it change its orientation whit respect to the camera (e.g. the red car steer). This is no more true for the NCC-based detection.

The following figure makes a comparison between the result obtained through the color-based segmentation and through the NCC- based segmentation considering the same input image[3].

Figure 4.6: Color-based segmentation (left) and NCC-based segmentation (right)

[3]The chosen image is the last one.