Report: Fifth Assignment.

Giovanni Battista Borrè, Tommaso Gruppi, Federico Tomat

May 14, 2019

1 Introduction

In order to evaluate similarity between patches various approach exist. In this assignment is requested to do a NCC segmentation.
Normalized cross correlation is a rather simple formula that describes the similarity of two signals. As such, it serves well for searching a known pattern in an image. You can use it when looking for a specific face in a photograph or for a letter in a scanned document. However in its pure generality, the formula can be used to countless tasks completely unrelated to computer vision. A downside is that cross-correlation can hardly account for rotation or other deformations. If you need to find a way around that, it will usually not be by far as pretty. Each possible location of the searched pattern is evaluated, and we can even get results more precise than the pixel level. Cross correlation might well be the simplest thing to calculate for the purpose and the normalization we use is the most profound normalization all around. That formula to evaluate is one single step and outputs clear values between 1 and 1. No parameters involved. Apart from simple arithmetic, the only computation performed is Fast Fourier Transform or Time domain Convolution where the latter one method is the one used in this assignment.

The function describing the cross correlation is defined as following:

$$\Phi_{NCC}(N_1, N_2) = \sum_{k,l=-\frac{W}{1}}^{\frac{W}{2}} \frac{(N_1(k,l)-\mu_1)(N_2(k,l)-\mu_2)}{W^2\sigma_1\sigma_2}$$

where:

$$\mu_i = \frac{1}{W}\sum_{k,l=1}^{W} N_i(k,l)$$

$$\sigma_i = \sqrt{\frac{1}{W}\sum_{k,l=1}^{W}(N_i(k,l)-\mu_i)^2} \qquad \text{with } i = 1, 2$$

1.1 Problem Description

This assignment consists of applying the NCC (normalized cross correlation) through the MATLAB® function *normxcorr2* [9] to a window (in our case selected by the user) around the red car in the gray scale input image in order to find the template in all the 5 input images. After that it is requested to show the position of the maximum of the score map and a box corresponding to the size of the template. Then we have to consider some different sizes of the window (centered in the red car) and to discuss the results in terms of computational time and accuracy of detection and finally to compare the results of this Lab (NCC-based segmentation) with the ones of $Lab4$ (color-based segmentation).

2 Solution Implementation

For doing these steps two functions were implemented: one to select the region of the red car on the street in the input image and one to apply the NCC (normalized cross correlation) between a selected template image and the target one. In this case the operations needed for doing the assignment' s requests are done mostly inside the main function called $Lab05$ so it will be described independently inside one of the following sections.

2.1 Area Selection

The function

$$[x1, x2, y1, y2] = selectArea(imgRGB) \tag{1}$$

takes as input the first RGB image and inside its body it uses the MATLAB® function *getrect* [4] that lets the user select a rectangle using the mouse and once this is selected, the function returns and stores information about the position and the size of the rectangle inside the *rect* variable; then these information are stored inside some other variables that will be the outputs of the *selectArea* function and that will represent the extreme points of the rectangle selected.

2.2 Normalized Cross Correlation

Inside the

$$[xmin, ymin, width, height, timeSpent] = ncc(targetImage, template) \tag{2}$$

function' s body initially the function *normxcorr2* [9] computes the normalized cross-correlation of the matrices $targetImage$ and $template$ that it receives as parameters, the resulting matrix is stored in a variable and it contains the correlation coefficients. After that the MATLAB® functions *find* [10] and *max* [11], where the last function plays the role of parameter for the first one, extract the maximum of the score map and store its coordinates in some variables that will be used to construct the box as requested in the $Lab05$ function. In the meanwhile the computational time of the execution of this function, that will be an output of the function, is computed through simple temporal operation and through the MATLAB® function *cputime* [12].

2.3 Time analysis

The function
$$[npixels, t] = timeEval(x1, x2, y1, y2, imgGray) \tag{3}$$
is implemented in order to evaluate the computational time of the execution of the function by varying the size of the region considered through the vector variable called $areaFactor$, these factors will be used to compute the size of the box inside a for cycle. Then it applies the *ncc* function between the input images and the varying selected area, it takes the computational time for each iteration of the cycle and then it stores them inside a vector variable that will be an output of the *timeEval* function; the other output is a variable containing each area considered.

2.4 Lab05 function

Inside this function in a for cycle the input images are loaded and converted to the gray scale. After the uploading of the first input image, the function requests to the user to select the region of the image desired through the *selectArea* function described previously, and then it computes the position and the sizes of the bounding box using that outputs, after that it extracts that region from the image. At this point it applies the *ncc* function using all the input images and the region selected by the user before as parameters. The last operation inside the for cycle is to show the image considered and the box constructed from the last function' s outputs and through the MATLAB® function *drawrectangle* [13]. Finally through the MATLAB® function *getframe* [7] captures the axes at the same size that it appears on the screen as a movie frame, its output is a structure containing the image data. After that it uses the *timeEval* function in order to plot and evaluate the computational times of the code' s execution while the size of the region selected is varying. The $Lab05$ function uses the MATLAB® functions *scatter* [14], *polyfit* [15] and *polyval* [16]: the first one creates a scatter plot with circles at the locations specified by the vectors received as input; the second function returns the coefficients for a polynomial p of degree n, in our case n is equal to 1, while the last one evaluates the polynomial p at each point. After these operations the function shows a picture containing all the information that let us to do some more detailed conclusions about the computational times.

3 Result

Our visual results consists on a gif, stored here and some images, stored here.
As we can note in the following the recognition is performed smoothly. Comparing the results obtained from the color-based segmentation method and the ones deriving from the NCC segmentation, we can deduce that the last one' s result is clearly better since only the selected car is tracked and not both the red cars. Moreover this method allows to track region having color not necessary heavily different from the background since it is based only on similarity.

Figure 1: Recognized car rectangle using normal cross correlation

Figure 2: Recognized car rectangle using color detection based segmentation

The time taken for the computation is also computed while varying the template's number of pixels considered. We can note how an almost linear relation elapses between those two quantities.

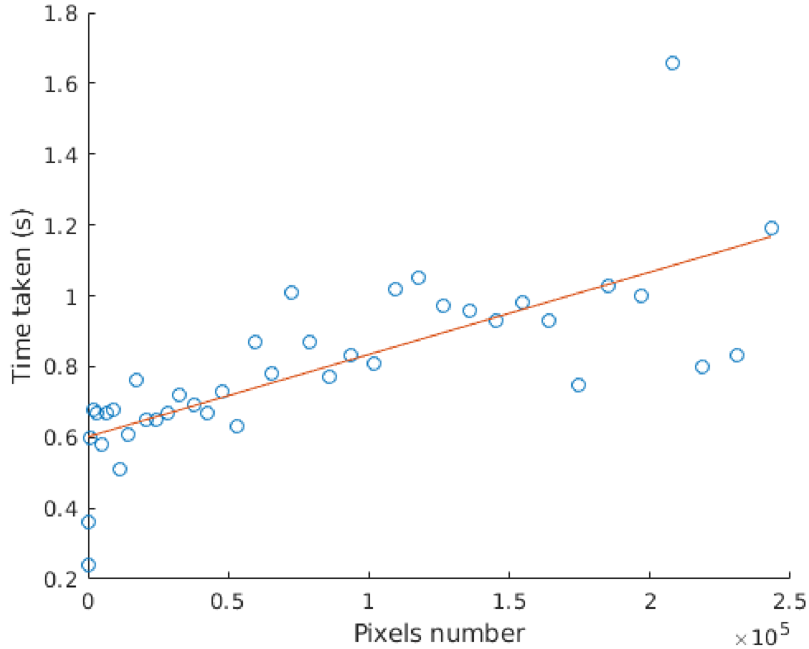

Figure 3: Time distribution

Tests were performed on a machine with the following configuration:

- CPU: Quad core Intel Core i7-8550U (-MT-MCP-) arch: Kaby Lake rev.10 cache: 8192 KB
- Memory: 8 GB

4 External links

Images: https://drive.google.com/open?id=1GUMk3hqi2RXl8b4NVWs6PzAv5d2uOhQm

Gifs: https://drive.google.com/open?id=12LgjfVACyNfjNa_x4SzqOP4SOS52rw4f

5 GitHub Repository

https://github.com/federicotomat/Computer-Vision

References

[1] Richard Szeliski, *Computer Vision: Algorithms and Applications*, (University of Washington, 2010).

[2] Francesca Odone & Fabio Solari, *Computer Vision Course: Image Foundamental*, (University of Study of Genoa, 2019).

[3] Mathworks documentation on *rgb2gray*

[4] Mathworks documentation on *getrect*

[5] Mathworks documentation on *std2*

[6] Mathworks documentation on *mean2*

[7] Mathworks documentation on *getframe*

[8] Mathworks documentation on *frame2im*

[9] Mathworks documentation on *normxcorr2*

[10] Mathworks documentation on *find*

[11] Mathworks documentation on *max*

[12] Mathworks documentation on *cputime*

[13] Mathworks documentation on *drawrectangle*

[14] Mathworks documentation on *scatter*

[15] Mathworks documentation on *polyfit*

[16] Mathworks documentation on *polyval*