UNIVERSIDAD DE LOS ANDES
Facultad de Ingeniería y Ciencias Aplicadas
Digital Image Processing 2019-20

Homework N°1: Correlation and Convolution

Tarik Sáez Riadi

08/22/2019

1 Introduction

One of the most common tasks in computer vision is to find an image, or its features, within another larger image. For these tasks, correlation and convolution come up as the main tools, and some of the most sophisticated functions are based on these two basic yet useful definitions. However, it is important to notice that computers can only perform so many calculations per second, therefore the problems that are addressed in machines, must be limited in size; in other words, it cannot be infinite as a continuous function. It is for this reason, that the definitions presented in this report are limited to discrete and finite operations, although the only difference between continuous and discrete definitions are the operators used (integral for continuous ones and summation for discrete ones).

In this report, the difference between convolution and cross-correlation will be explained, along with the problem that each one of these attempts to solve. Additionally, two examples of how correlation and convolution work will be presented along with its python code[1].

[1] Along with Python 3, the reader must have the libraries *opencv*, *matplotlib* and *numpy* installed.

2 Discrete 2D Convolution vs. Cross-Correlation

Let f be an image and h a kernel (an image no larger than f) and g the image obtained from the operations performed between f and h. Convolution is a mathematical function that combines two separate functions [1], f and h, into a third function g. Its discrete 1D mathematical definition is the following.

$$g[n] = (f * h)[n] = \sum_{k=-\infty}^{\infty} f[m]h[n-m] \tag{1}$$

An image is 2 dimensional and finite in size, therefore, the previous definition might be extended by performing two convolutions in two separate directions, and by limiting the number of iterations from infinity to just the dimensions of the image (width and height) which is translated to the following definition:

$$g[i,j] = (f * h)[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} f[u,v]h[i-u, j-v] \tag{2}$$

On the other hand, cross-correlation is the measure of the similarity between both f and h, and it is represented as a function of the displacement relative of one of the functions, relative to the other [2].

$$g[n] = (f \star h)[n] = \sum_{m=-\infty}^{\infty} f[m]h[m+n] \tag{3}$$

Extending this definition to two dimensions in a similar way to that of the convolution, we obtain the following.

$$g[i,j] = (f \star h)[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} f[u,v]h[i+u, j+v] \tag{4}$$

The difference between (2) and (4) lies in the operation performed in h. Notice how convolution subtracts the summation's variable while the cross-correlation adds it. This will translate into how the image is rasterized. During convolution, the kernel will traverse the image from the bottom-right pixel to the top-left one, first from right to left and then from the bottom to the top. Meanwhile, cross-correlation will make the kernel traverse the image from the top-left pixel to the bottom-right one, from left to right and then from top to bottom.

Because convolution is associative and cross-correlation is not, convolution is preferred to modify images (f) with kernels (h) with predefined values, also known as filters. On the other hand, cross-correlation is used to compare two images (f and h) and determine how similar they are to each other.

3 Finding Wally

The game "Where's Wally?" can be more challenging than it seems for a machine, because Wally is a character that does not look exactly the same, from the point of view of a machine, in every pictures. This means that in some cases, his persona will be more covered by the crowd than other puzzles, his face could be rotated, and he might or might not be wearing a camera around his neck. This differences make it virtually impossible to implement a general algorithm to find him without the use of a convolutional neural network.

In order to find Wally using cross-correlation, implicates to manually find Wally within the Puzzle, crop his face and use it as a template with the *opencv* function *matchTemplate*. This function has various "matching" methods. The one used was the normalized cross-correlation coefficient ($R[i,j]$) [4], which escentially computes a number between 0 and 1, where the closer the number is to one, the more similar the template and the image or puzzle are.

$$R[i.j] = \frac{\sum_{u=-k}^{k}\sum_{v=-k}^{k} f[u,v]h[i+u,j+v]}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k} f[u,v]^2 \sum_{v=-k}^{k} h[i+u,j+v]^2}} \tag{5}$$

A threshold for the detection was set, this means, that $R[i,j]$ had to be equal or major than the threshold value to consider a match in the template and the puzzle region. The method was 100% successful for all Wally puzzles.

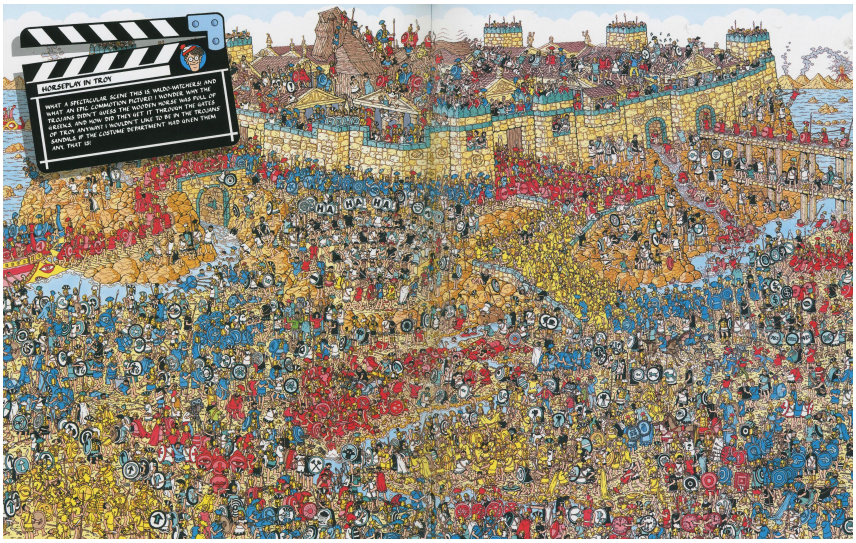

Figure 1: One of Wally's Puzzles.

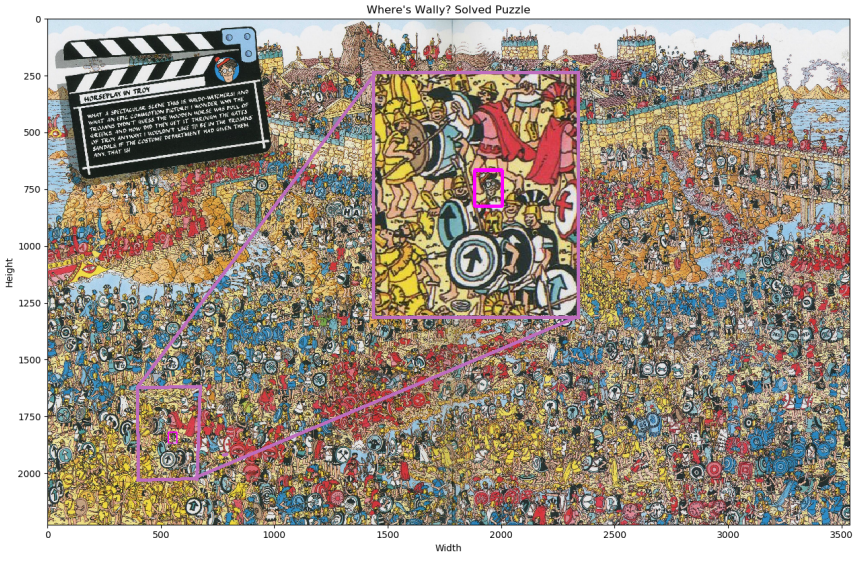

Figure 2: Solved Wally Puzzle.

Figure 3: Template used.

4 Artificial Image

The same code was used to detect an artificial image. Here, the threshold plays a more important role for detecting more than one match, as it helps to filter all non-matches from the results. Then, a purple rectangle is placed over each match. Notice that Wally, implemented with the same code, will always give a single match.

The image used was a crowd with several characters from the show "The Simpsons". Here, one of the characters was cropped, copy and pasted in several parts of the crowd, as the images below show.

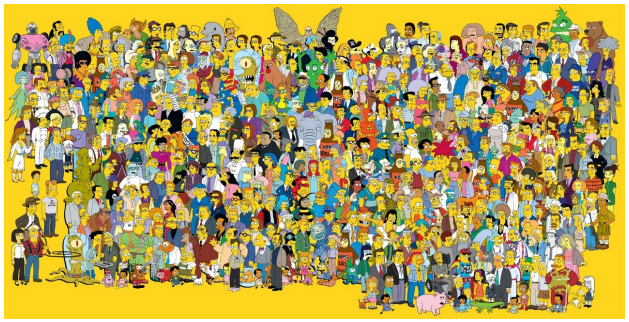

Figure 4: Original Image.

Figure 5: Professor Frink. A character from "The Simpsons".

Here, it can also be noticed that the accuracy of the algorithm is 100%. However, it is worthy to notice again that, just like in the "Where's Wally?" puzzle, Professor Frink had to be manually found, cropped, copy and pasted in order to run the algorithm.

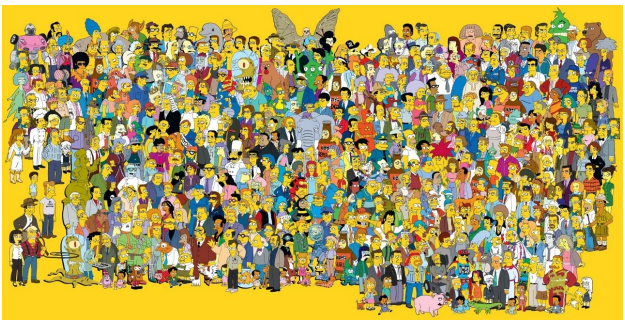

Figure 6: Artificial image, with professor Frink repeated 7 times within the puzzle.

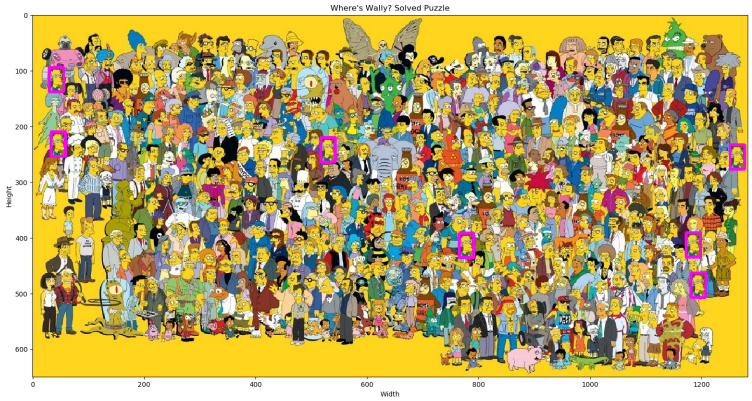

Figure 7: Artificial image solved for every Profesor Frink in the crowd.

5 Conclusion

In conclusion, the algorithm is very effective when the user knows that the template will be found in the image, mostly because the template is nothing more than a cropped sample from the image or puzzle. However, in order to implement a more general solution, a CNN should be used, as it can learn to identify the target despite being slightly different in the puzzle's image set. Nevertheless, this method is worthy to be highlighted as it can be enhanced to detect similarities or features instead of perfect matches, although the error rate will certainly fall to much less acceptable standards.

References

[1] "Convolution" *Wikipedia*, [Online]. Available: `https://en.wikipedia.org/wiki/Convolution` [Retrieved: 08-21-2019].

[2] "Cross-Correlation" *Wikipedia*, [Online]. Available: `https://en.wikipedia.org/wiki/Cross-correlation` [Retrieved: 08-21-2019].

[3] "Convolution vs Cross-Correlation" *YouTube*, [Online]. Available: `https://www.youtube.com/watch?v=C3EEy8adxvc` [Retrieved: 08-21-2019].

[4] "Template Matching Documentation" *OpenCV*, [Online]. Available: `https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html` [Retrieved: 08-21-2019].