Image processing with javascript and ImageJ software : geometric shape detection

Arthur Thouvenin*

*arthur.thouvenin@etu.u-bordeaux.fr

ABSTRACT

Image processing is nowadays essential to biology analyzes and research. In fact, in research, image processing allow to study the smallest molecules of the living and their functioning, nowadays we need to see interaction between molecules and the appearance of these molecules. It can help to understand how they work. Image processing is useful during medical tests or during environmental tests. Image processing takes place in many areas of science but here we will focus mainly on biology. Here we focus on image processing method, in which we will try to get the best results possible. We are going to use cobblestone's pictures taken in different places.These are pictures of cubic shaped cobblestones in Cherbourg-en-Cotentin.

Here we will use several tools, first : javascript language, a programming language commonly used in the programming of websites. But also the ImageJ software, which is an open source software specialized in the processing of scientific images. ImageJ provides the user with many features, however here a combination of functions will be needed for the image processing. We will use for this the recorder of this software, in order to obtain the commands necessary to our script.

For this project we also have to resort to statistical tests to compare methods and efficiency of our program. We need here the concept of sensitivity, it corresponds to the real positives, that's mean that the detected elements actually correspond to the targeted elements. But we will also need the notion of specificity, this corresponds to the rate of true negatives, that is, elements that are not and must not be detected.So the higher the sensitivity, the more we will detect elements. And the higher the specificity, the more specifically we will detect the elements we are looking for, so probably fewer elements in total, but the method used will be more precise. These notions will help us better understand our results and guide us towards a better script, with more adapted tools.

Introduction

The purpose of this project is to create a javascript script that will include tiles on photos. We will have to use various tools:

- First of all, the javascript language
- But also the ImageJ software, it is an open source software that is regularly used in the scientific field.

Javascript is a language commonly used for HTML programming but also here for ImageJ software, so you can create plugins from javascript script. We can therefore use all features of the software in a script or implement new ones and thus perform more complete and automated analyzes of scientific images.

ImageJ is a very precise software in its features and allows accurate scientific studies. From the various functionalities proposed, ImageJ proposes to record in a console with a javascript syntax all the commands used during the recording. Thanks to that we will be able to establish our script with the commands used by the software to use the different tools proposed.

However our project must also takes into account the efficiency of our script and that's why we will have to resort to statistics, allowing us to improve the script according to results of statistical tests. During this project we had to resort to notions like sensitivity or specificity previously explained. Indeed to improve the quality of our results some parameters will have to be adjusted or implemented features. This should be done on the basis of elements to detect or not detect but also the accuracy of our plugin and thus adjust the number of detected elements of this not to be detected.

- The specificity in statistics corresponds to the elements that must not be detected, that's mean the rate of true negatives compared to the number of true and false negatives.

$$specificity = \frac{True\ negatives}{True\ negatives + False\ negatives}$$

- Sensitivity corresponds to the rate of true positives, the set of true positives compared to true positives and false negatives, that's mean all the elements targeted by the analysis, all those that must be detected.

$$sensitivity = \frac{True\ positives}{True\ positives + False\ negatives}$$

So thanks to all elements presented previously our project was realized and our photos analyzed by our script, we will discuss next about results.

Material & Methods

Material

Figure 1. Photo 1

Figure 2. Photo 2

Figure 3. Photo 3

Figure 4. Photo 4

Here we have the four images of our project. These photos were taken on a sunny morning in the city of Cherbourg-en-Cotentin, with the camera of an Iphone 5S.

They all correspond to different pavers. It can be noted that the interviews of these pavers are very different, those in photo 1 are significantly less maintained than those in photo 3.

We can also notice that the lighting conditions are different, indeed for photos 1 and 3 lighting is good and flawless. However for the other two, we notice that a strong morning light influences the quality of the photos. We can equally notice that for photo 4 we see a shadow at the bottom right of the photo.

Methods

For this project we had to use the ImageJ software first and especially the "Record ..." function (*Plugins - Macros - Record ...*). This function allows us to save javascript commands used by the program to respond to user requests.

Thanks to this, we have established our plugin, but also thanks to the API of the software.

Implementation of a counting method

To count blocks of our photos we had to set up an algorithm able to recognize paving stones on any photos.

Firstly we preferred to work on images of same size and that's why the program resize the canvas(*Image-Adjust-Canvas size...*). After that the program will increase contrast of image according to automatic parameters (here saturated = 0.35;*Image-Adjust-Brightness/Contrast...*). Then in order to remove the background noise present in photos, the program will apply a mean filter of radius = 0.5(*Process-Filters-Mean...*). And subsequently the contrast is increased again with the method described above. Subsequently the image is converted to 8-bit (*Image-Type*)to use the threshold function which returns a binary image(*Image-Adjust-Threshold...*). So we are going to trim the edges of our objects twice to make sure they are well separated(*Process-Binary-Dilate*). Then the image will be reversed to fill the possible holes in the elements(*Process-Binary-Fill Holes*). Finally we analyze the elements with a size greater than 90pixels, then we return an image of the contours of the found objects(*Analyze-Analyze particles...*). This allows us to not take into account too small items that may not be cobblestones. After that we have automated the script so that it uses all the images present in a folder and that it returns us a table with the number of elements detected according to the image in a table.

Statistics

To realize the effectiveness of our script we need to resort to statistics.

First of all some notions :

- True positive : It is a detected element that is targeted by the analysis. So these are the elements sought by the analysis.Here are the detected cobblestones.
- False positive : This corresponds to a random detection of an object that should not be targeted by the analysis. Here, for example, false positives correspond to detected elements that are not cobblestones. This is an error of the program.
- False negative : This is all that should not be detected and is not detected. The scan does not detect its elements because it is not supposed to detect them. Here all that is not cobblestones and is not detected by analysis.

We will also need notions of specificity and sensitivity.

Specificity : This corresponds to a measure of the ability of our analysis to give a negative result when the condition is not verified. Here it corresponds to the extent of the ability of our program to give a negative result if the scanned element is not a cobblestone.

$$specificity = \frac{True\ negatives}{True\ negatives + False\ negatives}$$

Sensitivity : It is a measure of the ability of a test to give a positive result when a condition is verified. Here this is the measure of our script's ability to detect an item when the scanned item is a cobblestone.

$$sensitivity = \frac{True\ positives}{True\ positives + False\ negatives}$$

Results

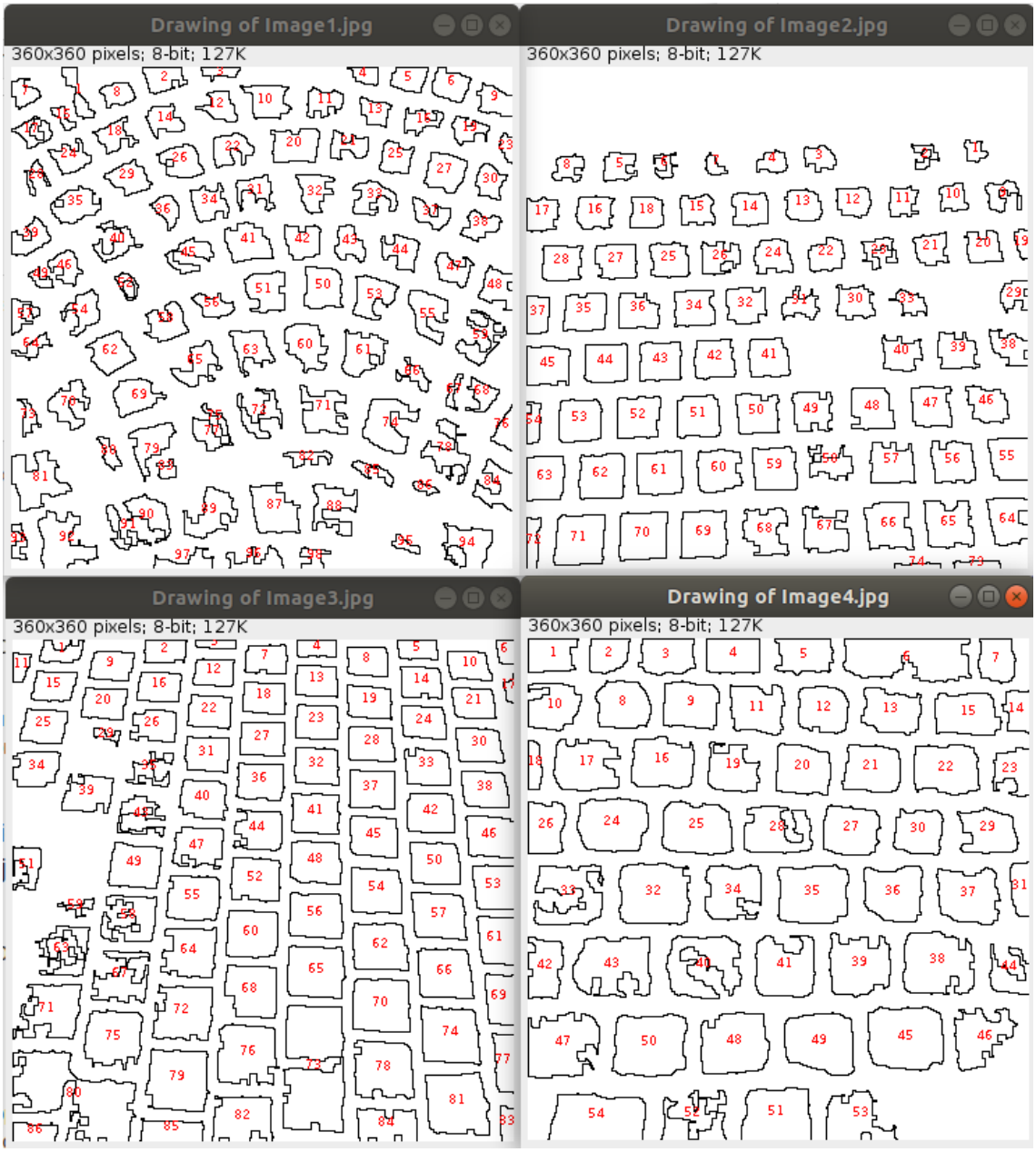

Figure 5. Script results "Outlines"

	Number of cobblestones	Picture's names
1	54	Image4.jpg
2	74	Image2.jpg
3	98	Image1.jpg
4	86	Image3.jpg

Table 1. Script results

After running the script, we obtain these results but also a bigger table, not shown here :

The four images correspond to the outlines of the elements detected by the script as tiles. We can observe that the cobblestones are well detected, however we notice some errors (to see the following remarks see Figure 5) :

- It is observed that the entire upper part of the image 2, a small part of the image 3 on the left but also on the bottom right of the image 4 are blank of any detections. This corresponds to our false negatives.
- It is also observed at the bottom of the image 3 and at the top right of the image 4 that some blocks are not separated from each other and are therefore counted as a single element.
- In Table 1 we get the results of the cobblestones counter

For the analysis of the efficiency of our script we will calculate the sensitivity and specificity of our method.

	True positives	Detected	Detected %
Image1.jpg	101	98	97.03%
Image2.jpg	106	74	69.81%
Image3.jpg	98	86	87.76%
Image4.jpg	59	54	91.53%

Table 1. True positives and detected cobblestones

Here the calculation of the specificity will not be feasible, indeed there are no real negative elements in our images. This prevents us from calculating this rate.

On the other hand the calculation of the sensitivity is realizable and thus thanks to the formulas quoted previously but also to the values returned by the software and those recovered by hand on the photos, we arrive at the following result :

sensitivity=0.857 That is a rate of 85.7% detection of true positives.

Discussion and Conclusion

First of all we can return to some anomalies of the photos and our script. Indeed during our analysis some parts of the photos do not seem to be scanned, the blocks are not detected. For image 4 this can be explained by the shadow in the picture. For image 2 it is the joints that change color slightly and tend towards green, we notice that at the top of the photo the colors are more softer and less contrasting preventing a good analysis.

We should therefore pay attention to each image and treat them individually to solve their specific problems. We also ran into a problem, when running the script the images should be closed one by one if this is not the case many errors occur.

Moreover the configuration of the script here makes it possible to detect with a great specificity the paving stones, however reducing the specificity and increasing the sensitivity we could approach the expected results. For that it would be necessary to establish a script more developed and adapted to different types of image. Indeed here the analysis is about photos of fairly good quality, which is not always the case for scientific images. A better use of the various parameters and functions of the ImageJ software could result in an optimization and improvement of the script and the algorithm.

In addition the script was used on a total of 12 images (not shown here), providing us with similar results of sensitivity and specificity.

We can conclude here that our program works on most of the photos, depending for better results we should move towards a more specific approach to the different images of cobblestones.

References

- Hartig, S. M. 2013. Basic Image Analysis and Manipulation in ImageJ. Current Protocols in Molecular Biology. 102:14.15:14.15.1–14.15.12.
- Geissmann Q (2013) OpenCFU, a New Free and Open-Source Software to Count Cell Colonies and Other Circular Objects. PLoS ONE 8(2): e54072. https://doi.org/10.1371/journal.pone.0054072
- Schmitz, Christoph et al. "Current Automated 3D Cell Detection Methods Are Not a Suitable Replacement for Manual Stereologic Cell Counting." Frontiers in Neuroanatomy 8 (2014): 27. PMC. Web. 11 Dec. 2017.

Source

- https://imagej.net/Welcome ImageJ Software Site
- https://imagej.nih.gov/ij/developer/api/index.html ImageJ Software API