Tiny Image in JavaScript : Mathematical Morphology

Adrien MENDES SANTOS

Link to Github : https://github.com/ReiNoIkari/Morphological-Image-Processing

Introduction

Nowadays, scientists have access to huge image databases which can make studying organisms easier. One problem still persist : those collected datas are worthless if they are not interpreted and the concern is that a lifetime won't be enough for a scientist to follow the flow. That's were informatics come in order to help to analyse those huge amount of raw data, and in our case more precisely Image Processing. Indeed, the developement of some Image processing tool made that easier to computerize and analyse.

Mathematical morphology has been invented in 1964 by Georges Matheron and Jean Serra in the MINES ParisTech's laboratories. Its development was always motivated by industrial application. At the begining, the main purpose was to answer issues in the mining exploitation field. Then this purpose diversified itself to biology, medical imagery, material science, industrial vision, multimedia, teledetection, geophisic, etc. It consist in a mathematical and informatical theory and technique which is linked with algebra, the lattice theory, topology and probabilities.

Currently, one of the mathematical morphology's main field is Image Processing. It particularly allows to use filtering, segmentation and quantification tools. Since it's emergence in 1964, it knows a growing success and constitutes a part of many Image Processing softwares yet.

For the purposes of object identification required in industrial vision applications, the operations of mathematical morphology are more useful than the convolution operations employed in signal processing because the morphological operators relate directly to shape.

For this third step corresponding to the Javascript implementation, we had to make our own implementation of the skeletonize operation from the available code from ImageJ and coded in Java to a Javascript version.

This report will contain the translation of the skeletonize operation found in ImageJ from Java to Javascript. We will focus on describe in detail the skeletonize operation in the Material & Methods part. We will then present our results with the use of the same input image sample, followed by a comparison about our implementation, the ImageJ processes and the WebGL version. Finally, we will discuss about our work and some possible improvements.

It has to be noted that, the results, the discussion and the conclusion parts will be similar between the two person's report working on the skeleton operations, since those parts were made by the two person and we tough that since thoses parts are linked,it would be more pleasant to share the common parts.

Material & Methods

As we have seen during the step 2 and the implementation of different multiple morphology operators in JavaScript, our implemented operators are slower than the one in ImageJ.

While the difference can be acceptable for small images, for big images, the difference is way too big to be considered acceptable. This is mainly due to the language itself, JavaScript and the use of a browser.

In order to make those implementations faster, we decided to use webgl technology, to gain as much as performance as possible to be as close as possible to the ImageJ benchmarks.

Results & Discussion

We were not able to provide any results for the webgl part. Sadly,we have not been able to succeed to convert the JavaScript code into a WebGl version.

The main reason for this failure are mainly due to two encountered problems.

The first one is that we were not able to get the pixel values of the neighbors from the copy and not the original one.

The second one, and probably the more problematic is that we had no idea how to repeat the *thin()* function until all the pixels have been modified.

After analysis, the key point of our failure was that we had more difficulties than we thought to understand how webgl works and show how to adapt our JavaScript code in a webgl version.

Conclusion

Our task was to adapt the skeletonize morphological operator in webgl.

As explained, we were not able to succeed in our task to convert the JavaScript part into a functional webgl version mainly due to the fact that we have not understood completely how webgl works.

We can however say that if we had a webgl skeletonize, the performance would have been better than the JavaScript one, and so closer to the original benchmark from ImageJ.

References