Find local maxima

Author : Théo Falgarone

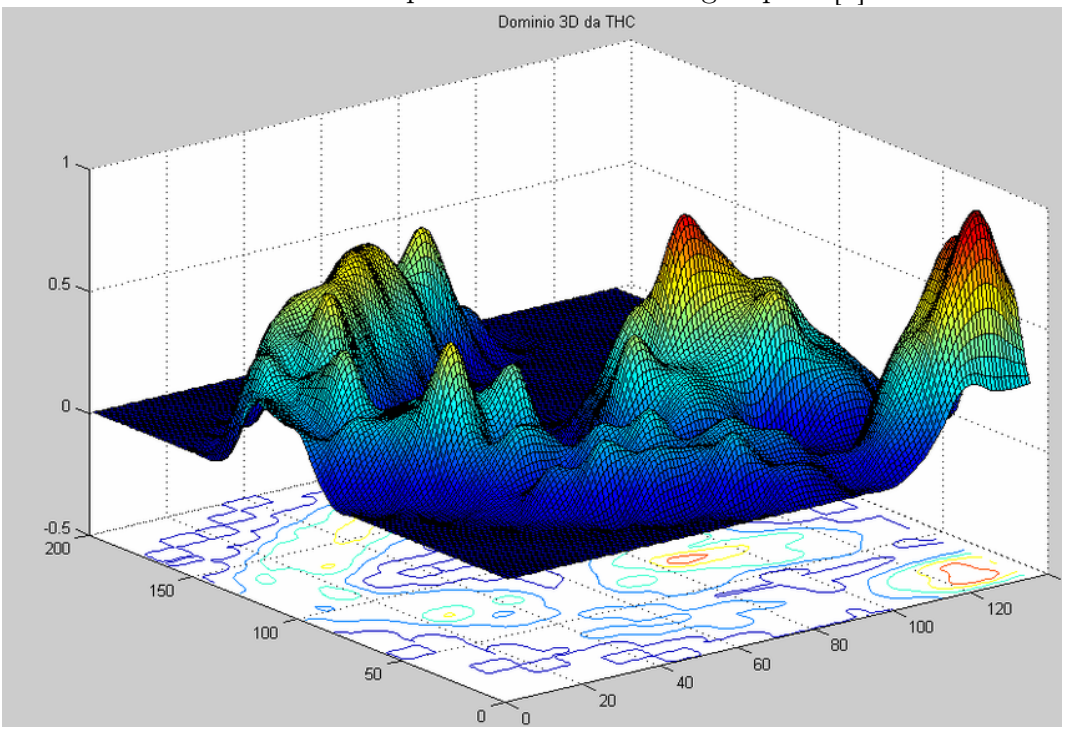
1 Introduction

Image processing is a very extensive domain composed of many features, which represents a significant amount of work for a beginner. The project entitled Tiny Image in JavaScript (TIMES) consists of grouping image processing main functions in an online toolbox in order to facilitate initiation to students. Each students group has to develop one or more features in the form of an ImageJ plugin written in Javascript with functional programming and using graphics acceleration in WebGL.

The work presented here concern the find local maxima, a tool that allow to identify every peaks in an image. It is part of the group working on the line and circle detection using hough transformation, which one of the steps requires an accumulator called the hough space. The find local maxima algorithm have to locate every peaks of the accumulator corresponding to lines or circles from the input image.

In this report, will be presented two algorithms for the find local maxima, then the results from the available tool in ImageJ and the implementation in TIMES will be presented and their efficiency will be compared, finally a conclusion about this project and the improvement that can be bring to it.

FIGURE 1 – Representation of a Hough Space [6]



2 Material & Methods

2.1 Naïve algorithm

A simple way to process an image with the local maxima algorithm is to test for every pixel if its indeed the highest value compared to its neighbour. More exactly, a pixel will be considered as a local maximum if the difference between its own value and its neighbours individually is higher than a threshold value. We defined a neighbour by its distance with the pixel currently tested, the distance is usually settle to four. As an output, we obtain the position of all the local maximum pixels from the input image.

Algorithm 1 find local maxima naïve

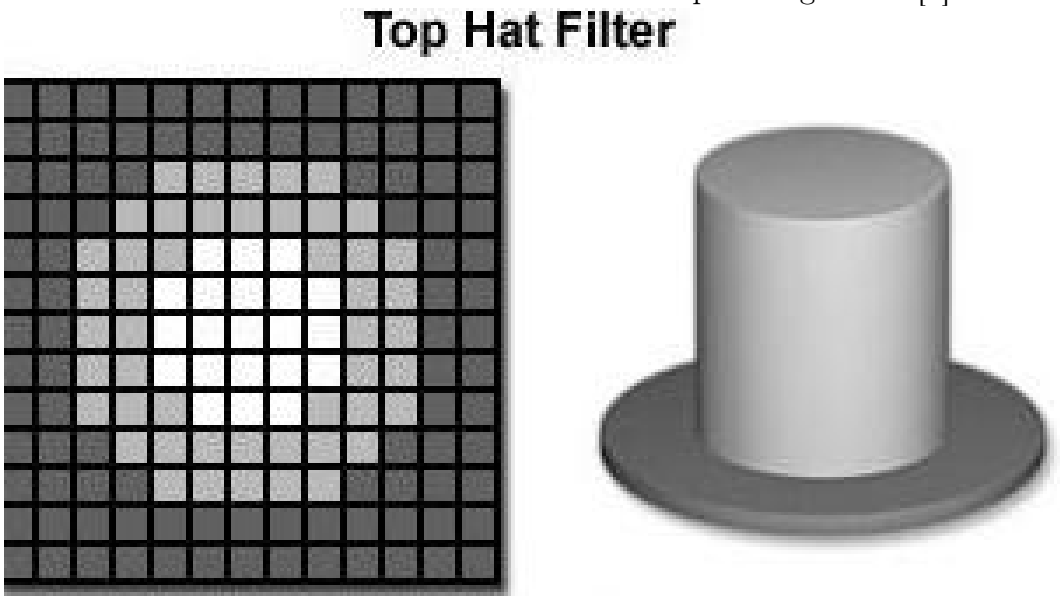
Input: Matrix $M(W * H)$
Output: $R = \{\{X1, Y1\}, \cdots, \{Xi, Yi\}\}$

```
Distance = 4
Threshold = 10
for all m ∈ {M} do
    Bool = True
    for all p ∈ (m + / − Distance) do
        if (valueof(p) + Threshold) > valueof(m) then
            Bool = False
            break
        end if
    end for
    if Bool == True then
        add m to R
    end if
end for
```

2.2 Top hat algorithm

A second method to apply a local maxima algorithm is called top hat level due to its hat looking form (see figure 2). The algorithm work as follow, for every pixel we define two matrix with the pixel in question as the center. The first matricd referred as the top of the hat and the second matrice called the brim is also the largest. For each matrix we are looking for the maximum value, then we compute the difference between these two values and compare it to a predefined threshold value. If this difference is greater than the threshold value then the pixel with the highest value in the first matrice is outputed as a local maximum.

FIGURE 2 – Scheme of the matrix from top hat algorithm [7]



Algorithm 2 find local maxima top hat

Input: Matrix $M(W * H)$; $small\_kernel$; $large\_kernel$; Threshold = 10
Output: $R = \{\{X1, Y1\}, \cdots, \{Xi, Yi\}\}$
  for all $m \in \{M\}$ **do**
    $small\_kernel = get\_small\_kernel(p)$
    $large\_kernel = get\_large\_kernel(p)$
    if $maximum\_value\_of(large\_kernel) - maximum\_value\_of(small\_kernel) > Threshold$ **then**
      add m to R
    end if
  end for

2.3 Benchmark

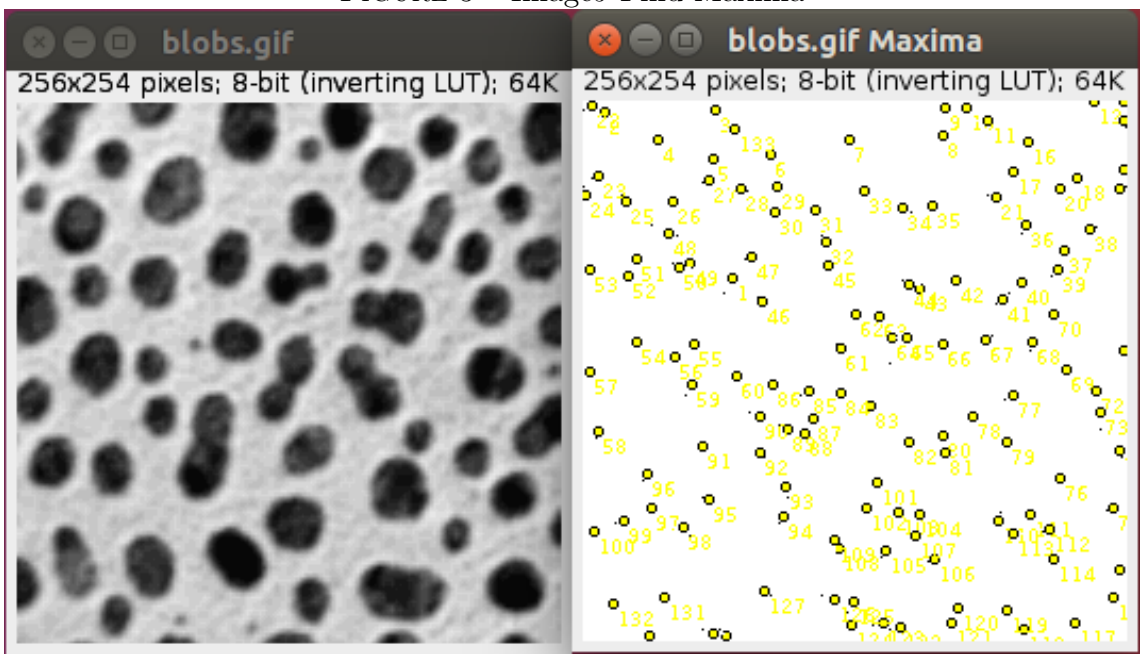
In order to compare the efficiency of the algorithm available as plugins in ImageJ or implemented in the Times project, a common evaluation of execution time has been realised. To execute a benchmark, we apply the tested algorithm to a sample image and measure the execution time. Then we duplicate the source image and add it to the sample, on which we apply the tested algorithm again and measure the execution time, this duplicate operation is repeated two hundreds times. Note that the same image will be used as sample to test every algorithm. All benchmarks were performed on a laptop CPU : intel core i5-4200H 2.80GHz x 4, GPU : Nvidia Geforce 950M.

3 Results

3.1 ImageJ Find Maxima

Find Maxima is a tool present in the tab Process in ImageJ, it allow the user to choose the threshold value and to find all the local maxima in an input image (see figure 3).

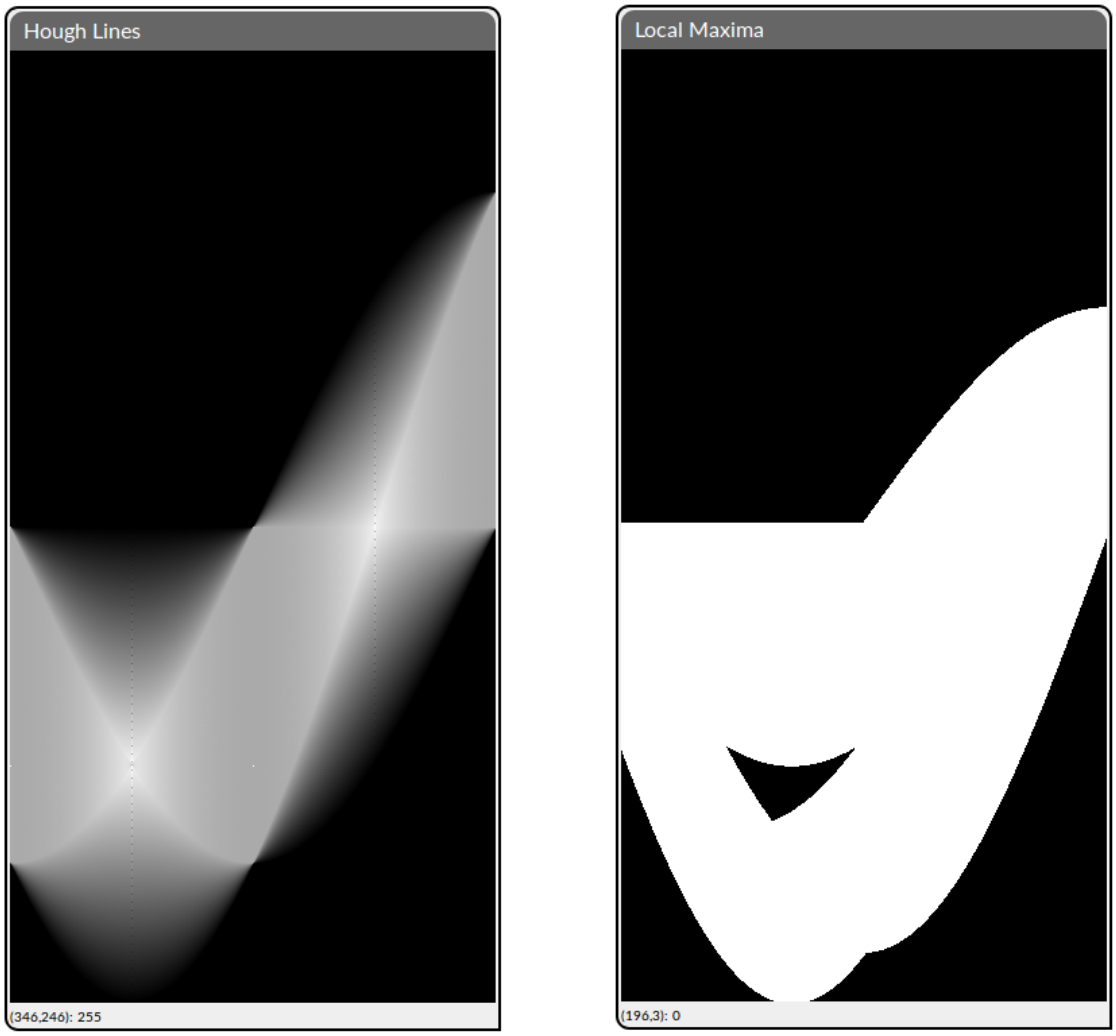
FIGURE 3 – ImageJ Find Maxima



3.2 Time implementation

Unfortunately the implementation in Javascript with the functional programming is not a success (see figure 4). On the figure below we can see a Hough Space on the left and the result of the implementation made is shown on the right. With this kind of input image we would expect only a few white spots as a result, instead here we have a white area which does not correspond to our expectations.

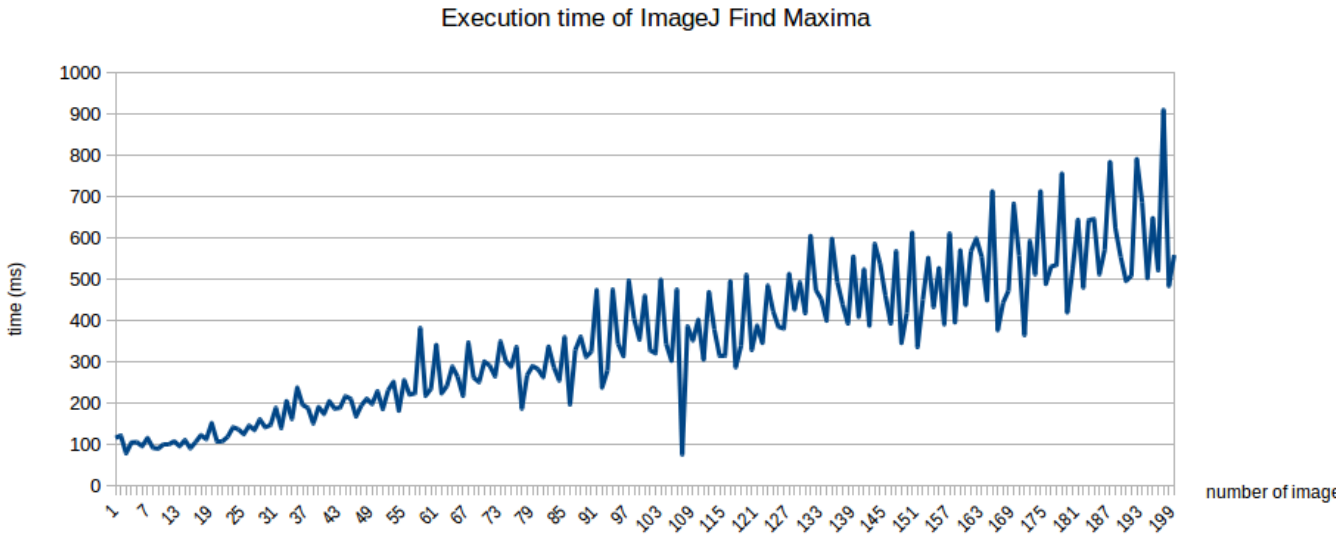
FIGURE 4 – Time implementation of Find Local Maxima



3.3 Benchmark

Despite some fluctuations, the curve is showing a regular slope with a minimum value of 80 ms and a maximum value of 911 ms.

FIGURE 5 – Benchmark ImageJ Find Maxima



4 Conclusion

This project was meant to implement the find local maxima algorithm in javascript using functional programming, in order to find local maxima values in the Hough Space acculmulateur and then detect lines and circles from input images or for other purposes. Unfortunately the javascript implementation does not work as expected and can not fulfill its objective. Here we tried to use the top hat algorithm, but there is many others algorithm who may be interesting to look closer like the Hill climbing algorithm for example.

Références

[1] [Figure 1,2,3] Dubska et al. Real-Time Detection of Lines using Parallel Coordinates and OpenGL, 2011.

[2] Paul V. C. Hough et al. Methods and means for recognizing complex patterns, 1960.

[3] Richard O. Duda and Peter E. Hart. Use of the Hough Transform to detect lines and curves in pictures. 1971.

[4] D. H. Ballard. Generalizing the Hough Transform to detect arbitrary shapes. 1980.

[5] Fatoumata Dembele. Object detection using circular Hough transform.

[6] D.L.Rodrigues. An Application of Hough Transform to Identify Breast Cancer in Images. 2008.

[7] https ://micro.magnet.fsu.edu/primer/java/digitalimaging/russ/tophatfilter/index.html