Global and Local Threshold*

Leonardo Rezende Costa - RA262953**

I. Introduction

This report describes the work done related to the second project proposed on class. The aim of this project is to implement different threshold algorithms and compare the results obtained, especially on the local based ones. The input images have different variation in luminance and most of them doesn't lead to good results with standard global threshold.

II. Script execution

This section describes how to execute the script and it's dependencies.

The ***t2.py*** file has the script used in this work to generate the results. It's made statically to process the images in the project's folder. It can be executed by ***python t2.py*** in an environment with all the dependencies listed on subsection II-A

A. Dependencies

The dependencies to run the python3 script are:

- Numpy
- Opencv-python
- math (standard)

III. Implementation

This section describes how the algorithm works and what was the approach to develop the script.

A. Global threshold

This approach receives an input value that defines the threshold and compares every pixel to it. If a pixel is below the threshold, it receives the minimum intensity value; otherwise it receives the maximum.

B. Local threshold

Every method described in this section calculates the threshold value for the pixel neighborhood and then compares it to the pixel intensity.

1) Bernsen's method: This method simply takes the maximum and minimum values in the neighborhood and calculate the average between them to estimate the threshold. Its important to note that if the pixel needs to be grater than the threshold to receive the maximum value, homogeneous regions will end up with black values. If it needs to be grater or equal, those homogeneous regions will end up white.

2) Niblack's method: This method combines the mean value and the standard deviation of the neighborhood multiplied by some constant to define the threshold. The mathematical form is

$$T(x,y) = \mu(x,y) + k\sigma(x,y) \tag{1}$$

where *(x,y)* is the pixel region, *k* is the constant, μ is the mean value and σ is the standard deviation. The values tested for *k* were: 0.25, 0.5 and 0.75.

3) Sauvola's and Pietaksinen's method: This method is a variation of III-B.2. The goal is to perform better in documents (text) context. They propose the calculation as

$$T(x,y) = \mu(x,y)[1 + k(\frac{\sigma(x,y)}{R} - 1)] \tag{2}$$

and the values tested for k were: 0.25, 0.5 and 0.75. The R was fixed on 128 as the method suggests.

4) Phansalskar, More and Sabale: This method is a variation of III-B.3 that aims to perform better in images with low contrast. The trashold T is calculated as

$$T(x,y) = \mu(x,y)[1 + pexp(-q\mu(x,y)) + k(\frac{\sigma(x,y)}{R} - 1)] \tag{3}$$

and the values tested for k, p, q are listed in Table I, and the R was fixed on 0.5 as suggested by the method.

k	0.25	0.75	
p	1	2	3
q	5	10	15

TABLE I: k and R values tested for Phansalskar, More e Sabale's method

5) Contrast method: This method compares the distance to pixel value to the maximum pixel of the region and the minimum pixel of the region. The pixel is 'rounded' to 0 when it's closer to the minimum and to 255 when it's closer to the maximum value of the neighborhood. The neighborhood window sizes tested were: 3,5,7,11 and 25.

6) Mean method: This method uses the mean value of the neighborhood as the threshold. It's defined as

$$T(x,y) = \mu(x,y) \tag{4}$$

where μ is the mean value. The neighborhood window sizes tested were: 3,5,7,11 and 25.

7) Median method: This method uses the median value of the neighborhood as the threshold. It's defined as

$$T(x,y) = Mdn(x,y) \tag{5}$$

where Mdn is the mean value of the region (x,y). The neighborhood window sizes tested were: 3,5,7,11 and 25.

*MO443 report - Introdução ao processamento de imagens digitais

**Instituto de Computação - Mestrado, Universidade Estadual de Campinas

C. Algorithm

All methods described at III were implemented. Each one of them can bee seen on the *t2.py* script and generated an histogram with same name as the output and a log with the proportion of black points of each result.

IV. Results

For each of the approaches, the images represented on Fig. 1 was used as input[1].

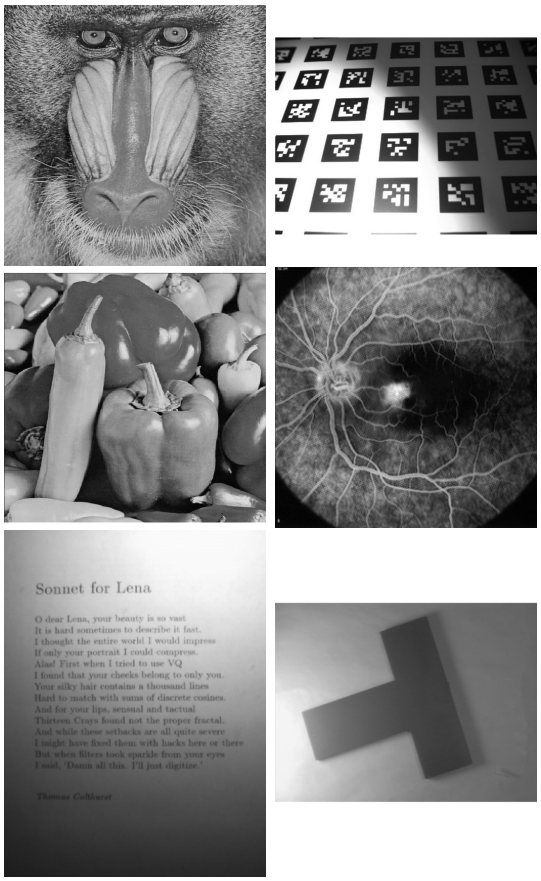

Fig. 1: Original images used as input

A. Bernsen's method

For this method, different neighborhood window sizes were tested and one additional flag was passed as parameter to establish the comparison as **greater then** or **greater or equal then**. Fig. 2 shows the tests on different window sizes. When the pixel is equal to its neighbours in a homogeneous region, max and min values are equal and the pixel is pushed up or down besides it's real value. The more the window size grows, the chance to take a more dark pixel and a more bright one increases and this problem is suppressed. Fig 3 shows the difference between the same window size, but with the comparison rule changed. The wall (background) becomes black in the bottom left of the right image, because the homogeneous amount of points result in a threshold equals to 0. It doesn't happen too much with the black parts in the left image, because there is not too much regions with an dark homogeneous window with size equals to 25x25.

[1]All those images was get at <https://www.ic.unicamp.br/~helio/imagens_pgm/>, accessed in 11/09/2019

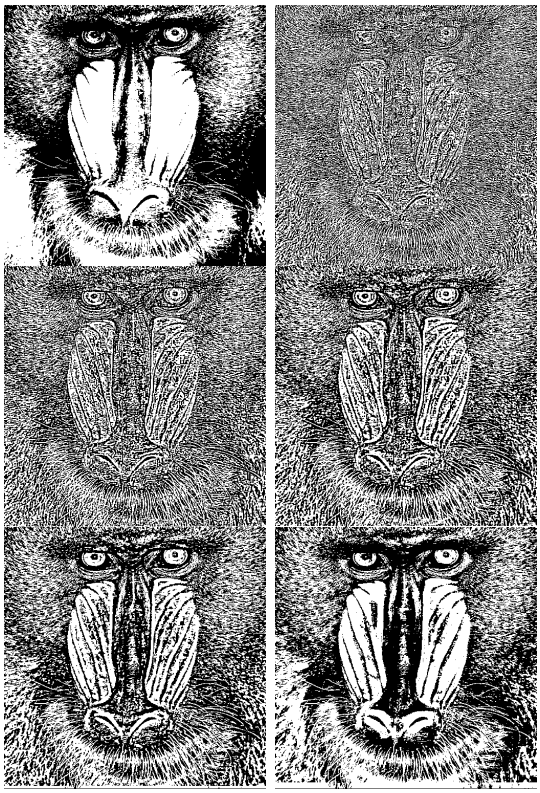

Fig. 2: Different window sizes tested on Bernsen's method. From top left to bottom right: global threshold result (with t = 127), Window size as 3,5,7,11 and 25, respectively

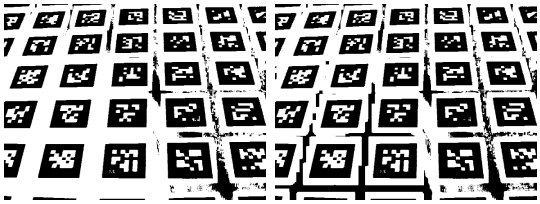

Fig. 3: Result of changing the comparison rule. The left image puts the maximum value if the original value is greater or equal to the threshold. The Right image puts the maximum only if the original pixel is greater than te threshold.

B. Niblack's method

For this method, the **k** value was changed within different window sizes. Fig. 4 shows different results for the variation of **k** in the retina figure with window of size 25.

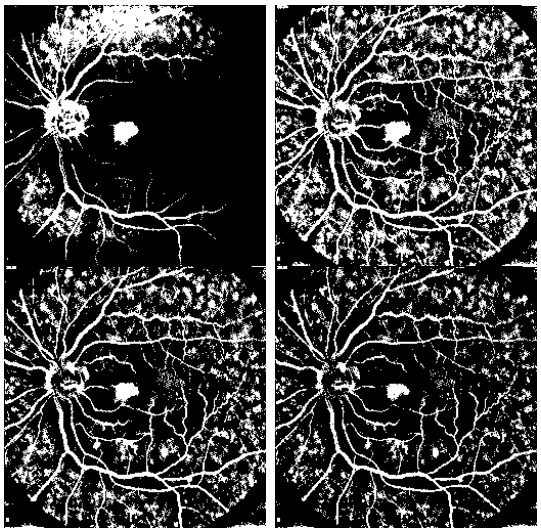

Fig. 4: Result for Niblack's method with different K's. From top left to bottom right: global with threshold equals to 128 and **k** values assuming 0.25 , 0.50 and 0.27, respectively

C. Sauvola's and Pietaksinen's method

For this method, the **k** value was changed within different window sizes. Fig. 5 shows different results for the variation of **k** in the retina figure with window of size 25.

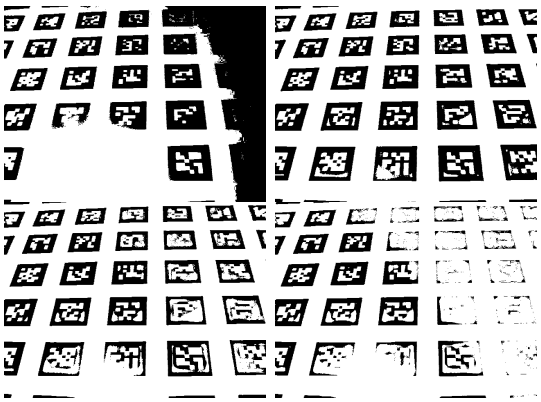

Fig. 5: Result for Sauvola's and Pietaksinen's method with different K's and R set as recommended on 128. From top left to bottom right: global with threshold equals to 63 and **k** values assuming 0.25 , 0.50 and 0.27, respectively

D. Phansalskar, More and Sabale's method

For this method, the **k**, **p** and **q** values was changed within different window sizes. Fig. 6 shows different results for the variation of **k** in the retina figure with window of size 25.

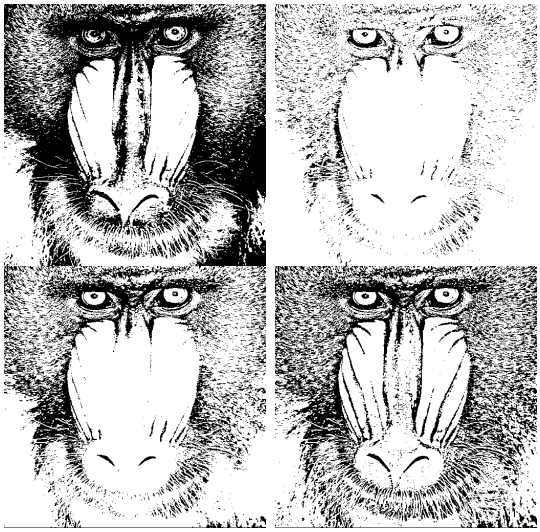

Fig. 6: Result for Phansalskar, More and Sabale's method with different parameters. Top-left: global gradient with t=127. Top-right: **k** = 0.5, **p** = 1, **q** = 10. Bottom-left: **k** = 0.5, **p** = 2, **q** = 5. Bottom-right: **k** = 0.125, **p** = e, **q** = 10.

E. Contrast method

This method doesn't receive any parameters. It does work well on regions with low contrast, but it has the same problem as Bernsen in homogeneous regions. Fig. 7 shows the results for retina input. It can be seen in the figure that the edges are well defined, but the background isnt removed because of the low contrast texture on it.

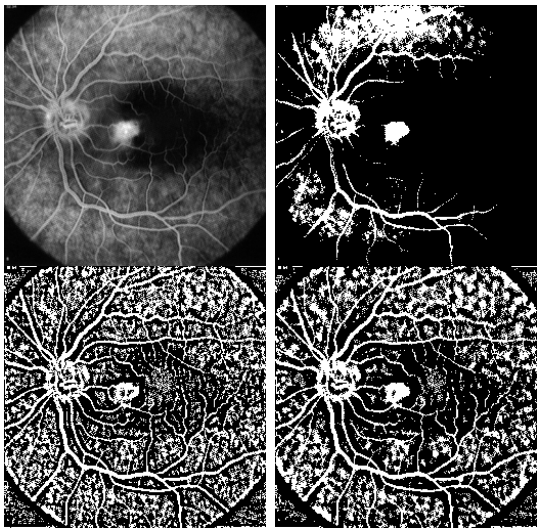

Fig. 7: Result for Contrast method. Top-left: original image. Top-right: global gradient with t=127. Bottom-left: Contrast method with window size = 7. Bottom-right: Contrast method with window size = 11.

F. Mean method

This method doesn't receive any parameters. It does work well on high contrast areas, but it doesn't achieve good results on homogeneous regions and tend to paint with a black and whit pattern regions with low contrast. Fig. 9

shows the results for the *peppers* and *wedge* inputs. Both of them doesn't paint the background in a way too different to the object, but the edges are quite well defined.

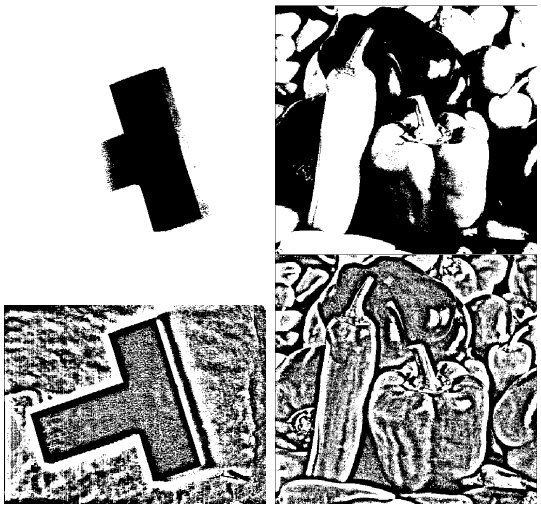

Fig. 8: Result for Mean method. Top-left: global threshold with t = 84. Top-right: global threshold with t = 127. Bottom-left: Mean method with window size = 25. Bottom-right: Mean method with window size = 25

G. Median method

This method doesn't receive any parameters. It does not work well on high contrast areas, but it creates a more soft pattern on homogeneous regions. Fig. **??** shows the results for *sonnet* input. The poem is readable and the result is way better than with the global threshold.

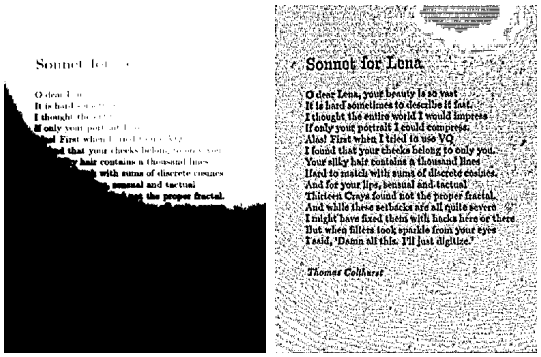

Fig. 9: Result for Median method. Left image: global threshold with t = 127. Right image: Median method with widow size = 7

V. Conclusion

Each one of the methods performed better than global threshold in almost every test with window size grater than 7. It was noted that a small neighborhood leads to a noisy output. Bigger neighborhoods performed better as the range of values reached higher contrast values. The *sonnet* input was the most difficult to binarize. Most of the problem is the brightness variation, but the image is also blurred and the method that would reach the best result (Sauvola's and Pietaksinen's) end up with a white background, but no legible letters. Also, the Contrast, Mean and Median methods reached good edge detection, but the homogeneous regions (the inside of the objects or the background) got a gray-like pattern because of low contrast leading to a high frequency signal.