# Image processing project : Adaptive document image binarization

Essayegh Nour

May 2020

## 1 Introduction :

Image binarization is the process of taking a grayscale image and converting it to black-and-white, essentially reducing the information contained within the image from 256 shades of gray to 2. This method is widely used on document image where the problems caused by noise, illumination and many source type-related degradations are addressed.

As seen in class we were able to compare some Histogram-based image segmentation methods beginning with Manual thresholding to a more sophisticated method with a more adaptive threshold and with this project will be programming a new method, Sauvola. This method is also a histogram-based image segmentation method and is strongly inspired by the Niblack but has a more sophisticated threshold calculation to deal with illumination in textual images.

## 2 A new method for document image binarization : SAUVOLA

### 2.1 The algorithm

The idea of the Sauvola method is to vary the threshold over the image, based on a local mean and local standard deviation computed in a small neighborhood for each pixel. The width of this window will be one of the parameters of the function. The local threshold is defined as

$$T(x,y) = mean(x,y).[1 + k.(\frac{s(x,y}{R}) - 1)] \tag{1}$$

.

This threshold is more adapted to the change of shades of the background. R is the dynamic range of standard deviation and takes the value $R = 128$ with an 8-bit gray-level image we are working with and k is a fixed value generally taken between 0.2 and 0.5.

### 2.2 The implementation with python

First of all, I loaded an image that already exists in the data library that shows a big variation of illumination and that in my opinion is the perfect kind of image to test the algorithm.

I chose to plot the histogram to have a quick idea of the distribution of the grey values of the image and as expected with this kind of image, the manual thresholding is nearly impossible because of the very distributed values of the greyscale.

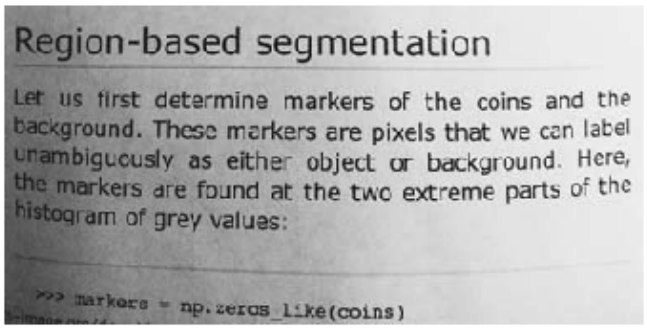



Figure 1: The image from skimage.data

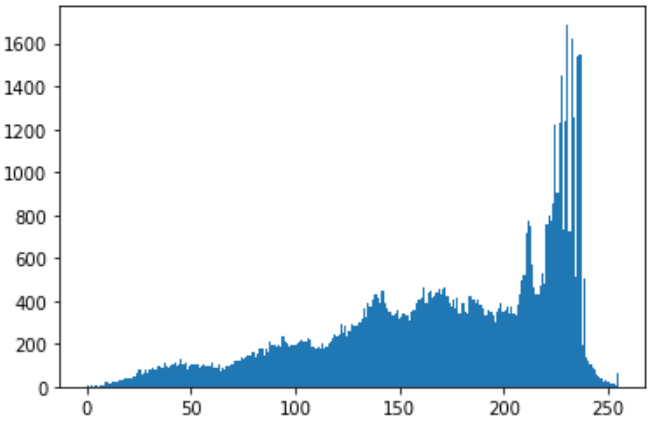



Figure 2: The histogram

I started defining a summing function that will be used in the main program. For the implementation of my program, I chose to code a function that gets the loaded image and the width of the window as a parameter and returns the binarized image. For that, I defined the main constants and the intermediary matrixes that would be used in the calculation of the threshold. To build the threshold for the pixel $(x, y)$ we need to calculate the mean value around this pixel and that with the expression:

$$mean(x, y) = \frac{1}{w^2} \sum_{i=x-l}^{x+l} \sum_{j=y-l}^{y+l} Image(i, j) \tag{2}$$

.

Then comes the local standard deviation for the pixel $(x, y)$ that we can find with the expression :

$$s^2(x, y) = \frac{1}{w^2} \sum_{i=x-l}^{x+l} \sum_{j=y-l}^{y+l} (Image(i, j) - mean(x, y))^2 \tag{3}$$

.

The last step is to put all the elements into the threshold equation and comparing the value of every pixel with the threshold. If it's bigger than the threshold the value if this pixel becomes white ($image(x, y) = 225$) and if it's not it becomes black ($image(x, y) = 0$)

To avoid some information loss at the edges, I used the function np.pad with the mode edge. This function expands the edges with the edge values of the image.

The result of my computer program shows a good binarisation as the negative effect of illumination is no longer a problem. I chose a window of 18x18 and $k = 0.2$.

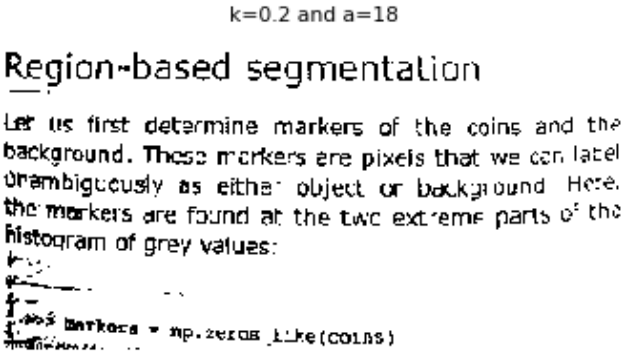



Figure 3: The finale result of the program with a window of 18x18 and k=0.2

We can try the computer program with other values of k and width of the window.

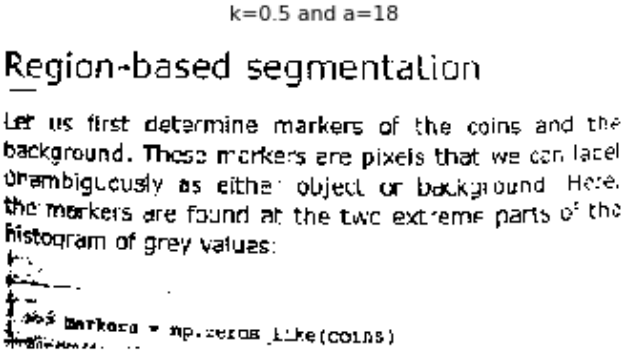
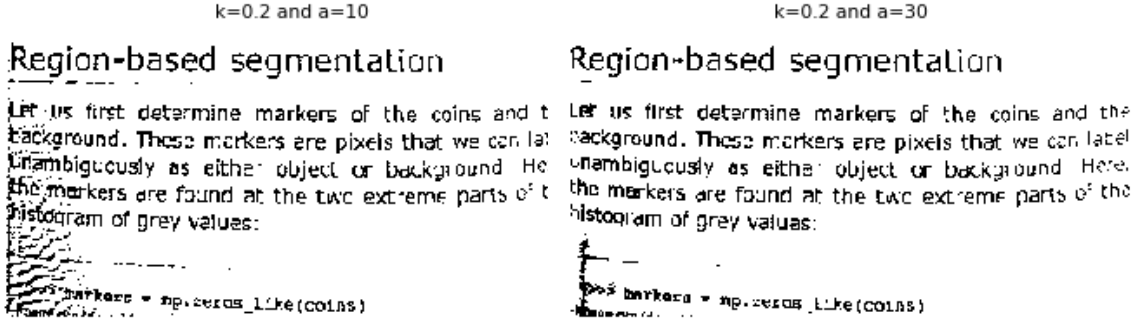
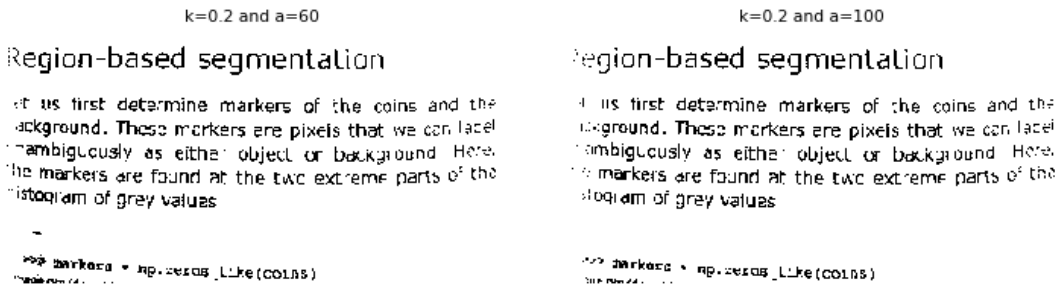



Figure 4: Sauvola's method with a window of 18x18 and k=0.5

**Note :** Even with the two extreme values of k there is nearly no difference between the two pictures. We can say that the variation of k doesn't affect the threshold.






**Note :** The width of the windows plays a significant role in the segmentation as we can see in images above. Indeed, a smaller window can leave some noise as the calculation of the mean is not that precise in some darker areas. With a very wide window the algorithm takes a lot of time to plot the image and we can see some information loss on the edges.

In conclusion, the Sauvola method is very efficient for binarisation of textual documents but the quality of the image depended on the width of the window chosen.

# 3 A comparison of some well-known methods

I thought it would be interesting to compare this new thresholding method with some other methods we have seen in class as Manual thresholding, Automatic thresholding and Otsu thresholding on the same image.

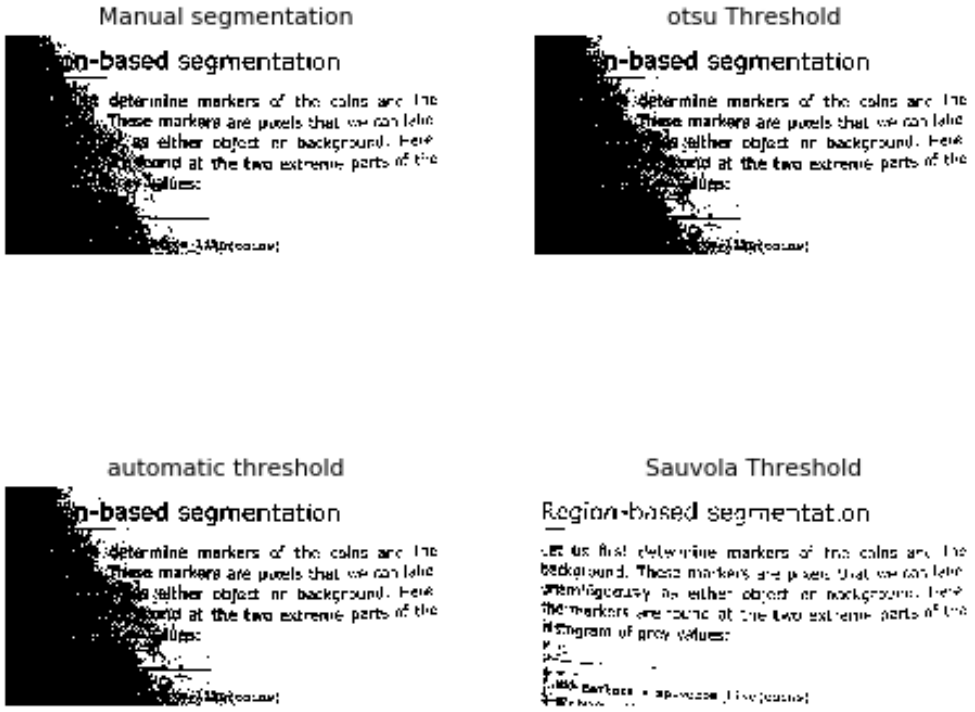



Figure 5: A comparison between some segmentation methods

Here we can see that the Sauvola method is the one that has the most precise and adaptive threshold to treat the darkest areas of the image.

# 4 conclusion :

Document image binarization is an important basic task needed in most document analysis systems.The quality of the binarization is due to a good definition of the threshold by specifying the calculus of its values to the size of its pixels. And the result of that is that we can obtain a respectable segmentation of the image regarding different types of defects as illumination, noise, and resolution changes. Therefore, the question that we can ask ourselves is, is noise treatment before the segmentation process necessary even with a good adaptive threshold?