Digital Image Processing: 3rd assignment

Tiago Moreira Trocoli da Cunha (226078)[1]*

Abstract
Morphological operations in image processing is a set of operations that process images based on shapes. They can be used in applications such as extracting connected components, extracting boundaries on objects, search for patterns and etc [1]. They apply a structuring element as input to images, producing other images of the same size. The most basic operations are dilatation and erosion. The first adds pixels to the boundaries of objects in a images, while the other removes pixels on the object boundaries. The number of pixels added or removed from the objects depends on the size and shape of the structuring element used to process the image. However, more advanced techniques such as closing and opening were applied in the 3rd assignment. Closing can be applied to join close objects and remove small holes, while opening can be used to eliminate narrow connections between objects and remove salient regions.

Keywords
Morphological Operations — Connected Components — Image Classification

[1] *Institute of Computing, State University of Campinas*
*__Corresponding author__: t226078@dac.unicamp.br

Contents

Introduction

The 3rd assignment involves morphology operations in image processing, it explains ten exercises with the goal of applying them in image segmentation in order to classify text and non-text components. Erosion, dilatation, closing as well as some calculations are applied as a means to achieve it. The code is written using Python 3.7 in Ubuntu 18.04. I used the following libraries:

- **scipy**: used to save images, apply morphological operators, that are, erosion [2], dilatation [3], closing [4] and auxiliary functions.
- **opencv**: just used it to open an image.
- **matplotlib**: used to plot images.
- **numpy**: used for matrix operations.
- **skimage**: used to draw a rectangle around text components.

The .zip file contains assingment3.pdf, homework3.ipynb with bitmap.pbm as an input image. The program generates image_r2.pbm in steps 1 and 2, image_r4.pbm in steps 3 and 4, image_and.pbm after applying *and* operation (step 5), image_r6.pbm in step 6 and image_r10.pbm in step 10.

1. Implementation

Tool functions were developed to support the algorithms and are in the first cell of the program. The *display_images* function takes a list of images and the desired dimension to be displayed as input to show them in the jupyter notebook. Opencv loads .pbm images as grayscale (0 to 255), however binary ones are needed for morphological operations, so a function *grayscale_to_binary* that converts image from grayscale to binary composed of 0 (white) and 1 (black) is necessary. Opencv saves .pbm images but for unknown reasons Image Viewer recognizes .pbm as a grayscale, so the function *binary_to_grayscale* that converts 0 (white) and 255 to 1 (black) and 0 is needed.

Steps 1 and 2 apply dilatation and erosion (closing) to library to image bitmap.pbm using a structuring element of $1x100$ resulting in image 1.

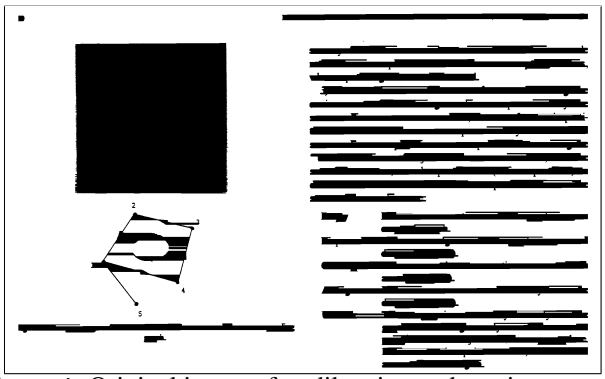

Figure 1. Original image after dilatation and erosion operations.

Steps 3 and 4 apply dilatation and erosion (closing) to image bitmap.pbm using a structuring element of $200x1$ resulting in image 2.

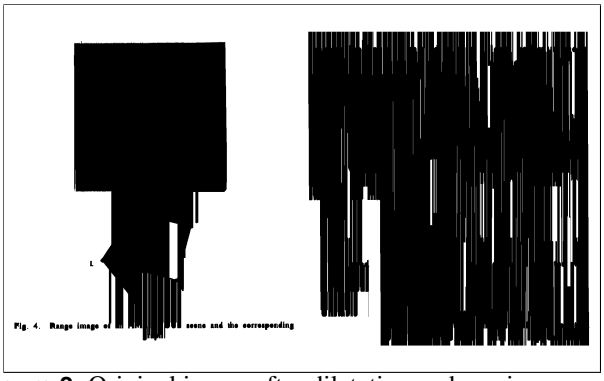

Figure 2. Original image after dilatation and erosion operations.

Step 5 applies *and* operation resulting in figure 3. From Set Theory, if $true = 1$ (black) and $false = 0$ (white), it has the following result: $0\,and\,0 = 0$, $1\,and\,0 = 0$, $0\,and\,1 = 0$ and $1\,and\,1 = 1$. So, by analysing images 1 and 2, *and* operation selects the text in the right side, since 2 almost encompasses all this text from 1, the vertical white lines are caused by 2. The squared object remains the same, since it composed of all *true* form both images.

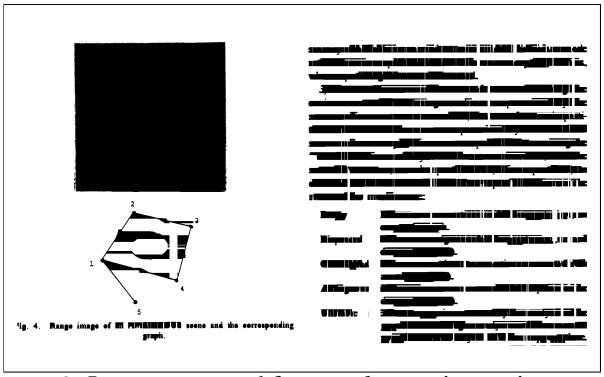

Figure 3. Image generated from *and* operation on images 1 and 2.

Next, *closing* operation is applied in image 3 resulting in 4. This operation is equal to $A \bullet B = (A \bigoplus B) \ominus B$, so it is an erosion of a dilatation of A in which the A is image 3 and B is the $1x30$ structuring element. Closing is similar in some ways to dilation in that it tends to enlarge the boundaries and fill gaps of images, but it tries to better preserve the original shape of them.

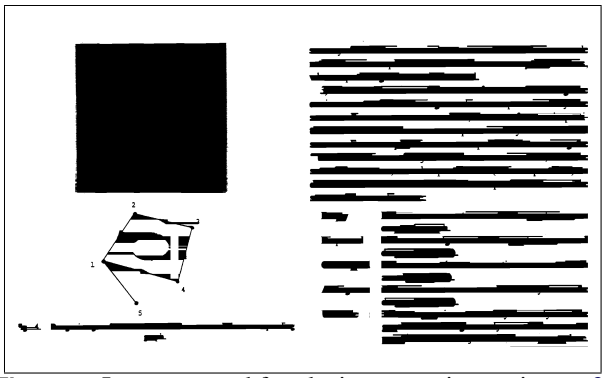

Figure 4. Image created for closing operation on image 3.

In step 7, instead of applying the algorithm provided by the professor, it was used the *label* [5] function implemented in *scipy.ndimage*. According to the website, it labels all the connected components of an image, that in this case is **53**, and also returns the number of components as follow:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \xrightarrow{label()} \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 2 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \end{bmatrix}$$

In step 8, the function *ndimage.find_objects* [6] was applied to extract each component from the image, this function returns a list of two slices for each component. Putting them in the image array, shows the component itself. In the example above, the components are $[2 \quad 2]$, $[[1 \quad 1], [0 \quad 1]]$, $[3]$ and $[4]$. After extracting each component, the algorithm finds number of black pixels, its percentage and the number of vertical and horizontal transition from white to black. These statistics are all print with its respective image in the Jupyter Notebook.

In step 9, I needed to analyse the statistics to find a rule that classifies text and non-text components. So, in step 8, each statistics with the component were shown in Jupiter Notebook. I could find that the the number of black percentage in text components were between 0.35 and 0.85. This rule englobes all text components but 2 or 4 non-text ones, to remove them, I found that the number of horizontal transitions that separates all text from non-text was 14. So the rule to select only text is:

$$\text{rule} = (0.35 < \frac{b}{b*w} < 0.85)\,and\,(h \geq 14),$$

in which b is the number of black pixel, w number of white ones and h the number of horizontal transitions from white to black. There are **34** text components, and **19** $(53 - 14)$ non-text ones.

In step 10, it is required to find all words in the original image. In order to do that, it was applied dilation of $1x15$ followed by erosion of $1x8$. However this procedure wrongly selects a group of the topmost-right words as one word. Thus, the text "IEEE Transactions on Robotics and" is one word

according to my algorithm. The wrong segmentation can be seen in the image 5. I didn't find time to develop a rule to classify text from non-text, thus, rectangular boxes are around all objects. The resulting image is shown below.

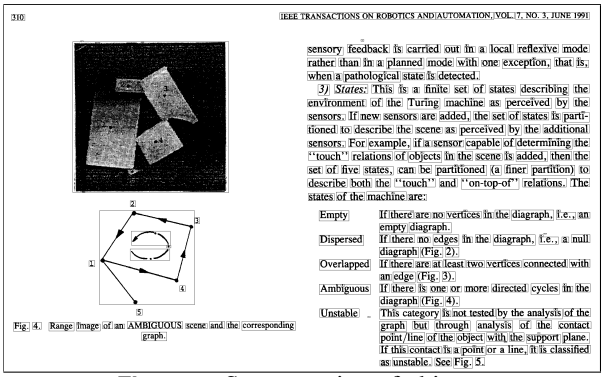

Figure 5. Segmentation of objects.

2. Results and Discussion

The 3rd assignment was more challenged, needed more work and time to finish than the others, specially because of exercises 8, 9 and 10.

References

[1] Mathematical morphology. https://en.wikipedia.org/wiki/Mathematical_morphology. Accessed: 2019-05-22.

[2] scipy.ndimage.morphology.binary_erosion description. https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.ndimage.morphology.binary_erosion.html. Accessed: 2019-05-22.

[3] scipy.ndimage.morphology.binary_dilation description. https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.ndimage.morphology.binary_dilation.html. Accessed: 2019-05-22.

[4] scipy.ndimage.morphology.binary_closing description. https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.ndimage.morphology.binary_closing.html. Accessed: 2019-05-22.

[5] scipy.ndimage.label description. https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.label.html. Accessed: 2019-05-22.

[6] scipy.ndimage.measurements.find_objects description. https://docs.scipy.org/doc/scipy-0.16.1/reference/generated/scipy.ndimage.measurements.find_objects.html. Accessed: 2019-05-22.