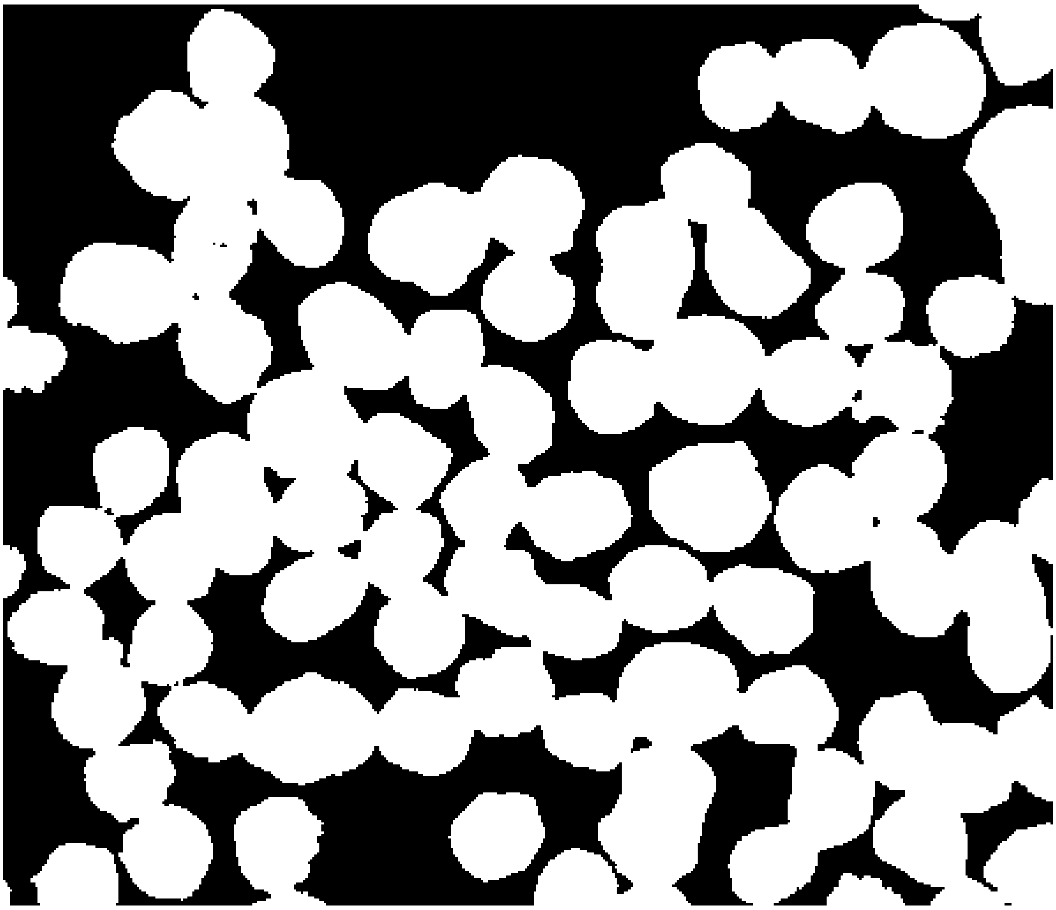
Audio processing, Video processing and Computer vision

LAB 3 - REPORT



By Alberto Castellano Macías (100414242)
and Mohamed Afif Chifaou (100452024)

Audio processing, Video processing and Computer vision

INTRODUCTION

In this laboratory, we are given a segmentation problem case to study. Basically, we are required to work on this project with the goal of segmenting several images containing cells by performing image segmentation with its various procedures, such as ***preprocessing*** (noise reduction using filters, image enhancement), ***segmentation*** (Otsu's method & watershed)***, morphological operations*** (dilation, erosion) ***and post-processing*** (remove holes). Let us see the obtained results using each technique and explain the work we developed.

Technique	Performance (IoU)
#0: Baseline code	0.685
#1: #0 + Median Filter	0.692
#2: #1 + Dilation + Erosion + Remove Small Holes + Watershed Segmentation	0.833
#3: #2 + Changing disk of Dilation, Background and Cell Threshold and Threshold of Remove Small Holes	0.865

DESCRIPTION OF TECHNIQUES

1. ***Baseline + Median Filter* : 0.692**

   In this technique, we just applied a **median filter** to the baseline code to remove the noise of the image, and we chose a disk of radius 3 because overall there is not much noise. We can see that by using the median filter instead of the gaussian one, we obtained a slightly better performance. We also kept the **otsu thresholding**, an algorithm to separate pixels of an image and split it into regions, as it was in the baseline code. We also tried a clustering algorithm using k-means but we failed to obtain a good performance, so we decided to just keep Otsu's method.

2. ***#1+Dilation+Erosion+Remove Small Holes + Watershed Segmentation*: 0.833**

   For the second technique, we used **dilation** with a disk of 5**,** which adds pixels to the boundaries of the cells or bright regions, and **erosion,** which removes pixels from boundaries of bright regions, making cells smaller. This makes the cells look more similar to the ground truth. We also used **remove_small_holes()** (with an area_threshold value of 500), to fill the holes inside the cells to contribute to a higher average IoU.

Furthermore, we tried **watershed segmentation** in order to separate the cells from each other. However, for some reason, we failed to improve our performance by doing this. Therefore, after some research, we decided to use watershed twice. The first one was used to segment the background pixels from the cell pixels by setting a manual threshold after printing the minimum and maximum pixel values of the smoothed image. We are aware that this is also kind of the job of Otsu's method, but doing watershed by flooding actually improved our performance, so we kept both methods.

In addition, for this first watershed segmentation, we started by defining the **basins/starting points** (by filling our image with pixels of value 1 if they were smaller than the background threshold set to 0.25, and 2 if they were bigger than the cell threshold set to 0.3), we then used the watershed function to "flood" the basins and segment into regions and we ended up obtaining a mask where we had "True" values for those pixels bigger than 1 and "False" values for the rest of them.

We did this to later use the **ndi.distance_transform_edt()** function to assign a value to each pixel by computing its euclidean distance to the nearest background pixel. We need this to perform watershed afterwards.

Moreover, we computed the local maxima, or points where the intensity is greater than the surrounding 8 pixels, by using the **peak_local_max()** function. We then used **label()** in order to assign a unique label to each of the local maxima and be ready for the second watershed segmentation.

Finally, the second watershed segmentation consisted in distinguishing and segmenting each cell from the rest of cells. We applied the **watershed()** method once again and used negative distances so that basins are created around local maxima.

As we can see, this technique performed quite well, but after looking at the final segmented masks in the plots, we can see that not all of the cells in all of the images were perfectly segmented and not all holes in the cells were covered, so there was still room for improvement.

3. ***#2 + Changing size of disk Dilation, Background and Cell threshold and area_threshold of remove_small_holes(): 0.865***

In this attempt, we reduced the size of the disk when doing dilation to a **radius of 3** pixels as we saw that the cells were mostly relatively small.

Also, we changed the **background and foreground** (cells) to **thresholds** of 0.3 and 0.5 respectively, so that they are better differentiated than before as we had seen some poorly segmented areas.

To end with, we changed the **area_threshold** value of remove_small_holes() from 500 to 1500 as we believed it was not fully covering all of the holes in the cells. By doing this, we succeeded and holes were fully covered.

DISCUSSION OF RESULTS

After many hours of research to better understand the problem and possible solutions, and after trying different methods and libraries, such as OpenCV and skimage, we believe that we have obtained a decent performance in terms of average IoU. We have tried different filtering methods, segmentation methods, parameter values, and even different ways to apply watershed until we were finally able to code a fine algorithm that obtained a score of 0.865.

However, we believe that there was still room for improvement as we have seen that in some of the final segmented masks we predicted, some cells are not perfectly segmented, so there is probably some parameter or some adjustment to it that we did not accomplish to set as well as we could, even though we tried many times. Also, we probably could have improved the efficiency of the code by altering the watershed method in some way but we believe it is still quite good.

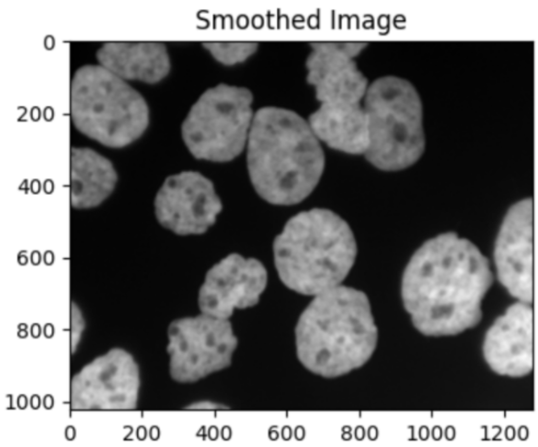
Personally, we found the assignment very useful as we got deeper into the different techniques that we can apply to segment an image and understood very well the algorithms behind each method. We are looking forward to working with CNNs!

Audio processing, Video processing and Computer vision

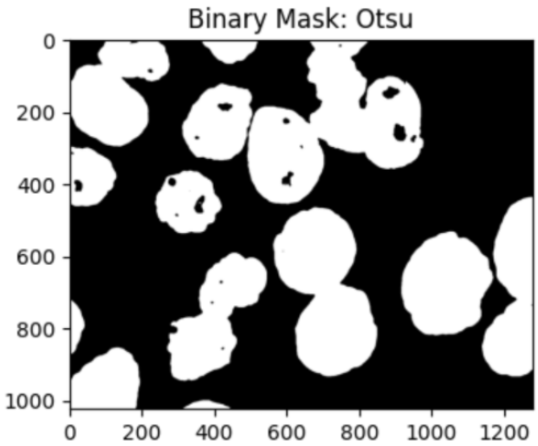
GRAPHIC SUMMARY

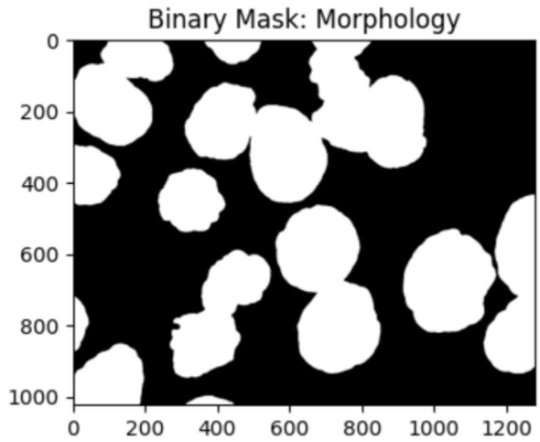
The following images are the graphic summary of our best performing technique, focusing on the first image as the output follows the same process for all of them.

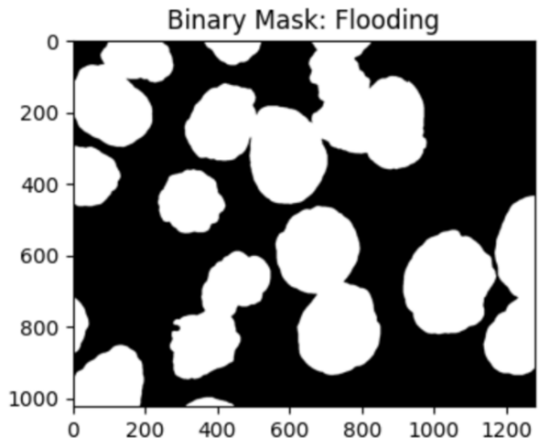
First, we can see the smoothed image after applying the median filter. Then, we printed the minimum and maximum pixel values of the image for us to have some insights. Following, we showed the binary mask after applying Otsu's method. After that, we applied all the morphological operations, including dilation, erosion, and removing small holes, and showed the mask after the first watershed segmentation. We also checked that the flood_seg contained True and False values as we wanted.




```
Minimum pixel value: 21
Maximum pixel value: 197
```



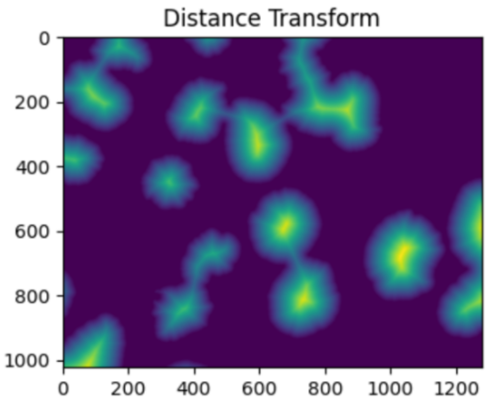




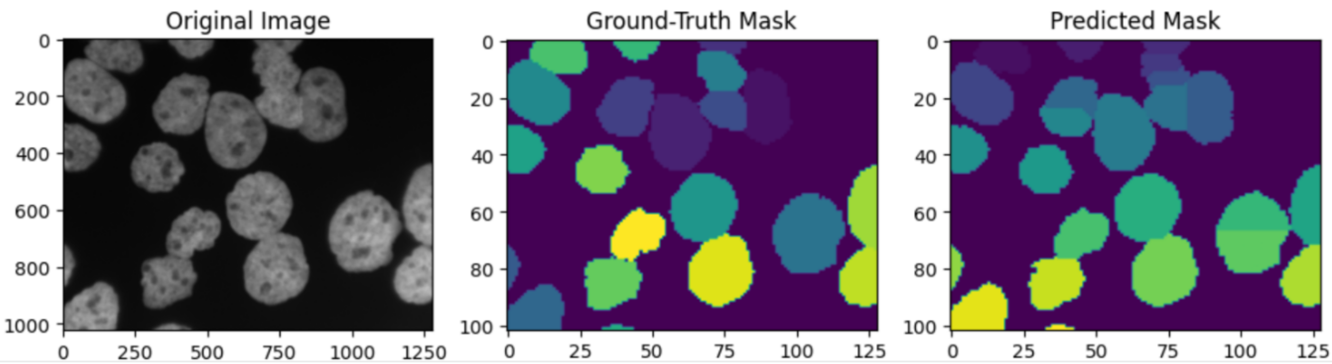

```
Minimum pixel value in flood_seg: False
Maximum pixel value in flood_seg: True
```

Audio processing, Video processing and Computer vision

Furthermore, we plotted the distances figure and printed the minimum and maximum value in the distances matrix to check it made sense. We also checked that there were no NaN values nor infinite values there. Finally, we printed the IoU score for the specific image and showed the original image next to its ground truth mask and our final predicted mask to visualize how it differs.




```
Minimum pixel value in distances: 0.0
Maximum pixel value in distances: 115.35163631262454
Contains NaN values in distances: False
Contains infinite values in distances: False
<ipython-input-23-c8dbb7f7b26c>:123: FutureWarning: indices argument is deprecated and will be removed in version 0.20. To avoid thi
  local_max = peak_local_max(
Image 0, IoU=0.880008132573832
Image 0, IoU=0.880008132573832
```



Audio processing, Video processing and Computer vision

REFERENCES

1. **Scikit-image**: https://scikit-image.org/
   This is the official documentation of scikit-image, and we used it to understand the roles of each of the methods we use.

2. **PyImageSearch**: https://pyimagesearch.com/2015/11/02/watershed-opencv/
   This page was useful to try out the OpenCV library to see if we could improve our performance.

3. **Medium**:
   https://medium.com/@jaskaranbhatia/exploring-image-segmentation-techniques-watershed-algorithm-using-opencv-9f73d2bc7c5a
   We used this page to learn more about watershed flooding.

4. **Medium**:
   https://dhairya-vayada.medium.com/intuitive-image-processing-watershed-segmentation-50a66ed2352e
   This page also gave us helpful insights in how watershed works on every step of the algorithm.