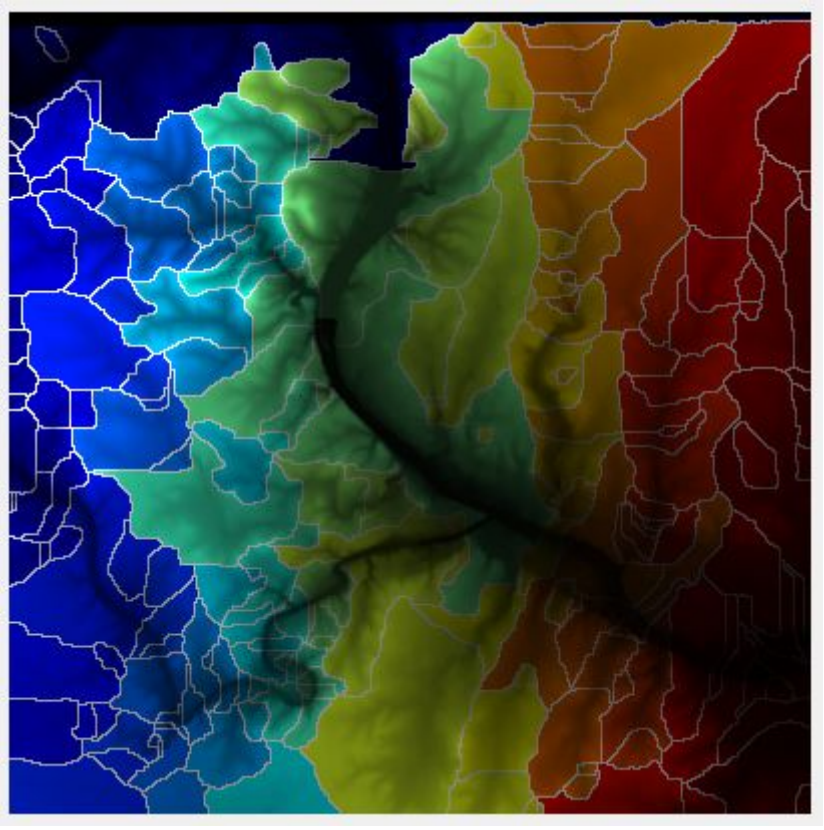
Paula Castillo, 184646
Helena Cots, 183951

Lab5: Morphological Segmentation: Watershed Algorithm

1. Extraction of watersheds

In this exercise what we had to do was to extract watersheds from the image dem.gif using the *watershed* function. The aim of this exercise is to segmentate the image into labels from the pixels' intensity, which gives us the information of how high above the sea level is each pixel.
In order to achive that, we have implemented a code that read the image and normalize it (transform its values from uint8 to double). Then, after using the *watershed* function with a value of 8 connected pixels, we normalize again and assign a different color value to the labels obtained in watershed, using *label2rgb*. To conclude the code, we multiply each component RGB by the original image. The result we have obtained is the following:

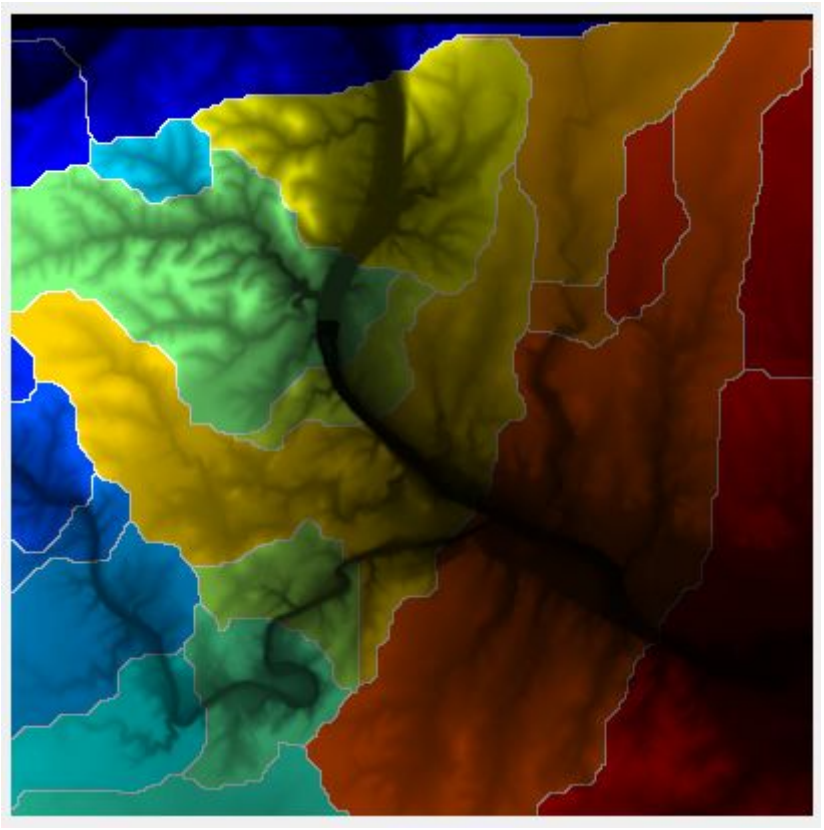


If we had used a value of 4 connected pixels, much more labels would appear, because of the connection between pixels is more limited.

2. Apply morphological filters to remove maximas and minimas

Use the same code as before in the exerxise 1, but apply a morphological filter with a structuring element 10x10 to remove all maximas and minimas before segmenting the image. See any difference? Do the same with a structuring element of 20x20. From the point of view of the watershed, how to interpret the results?

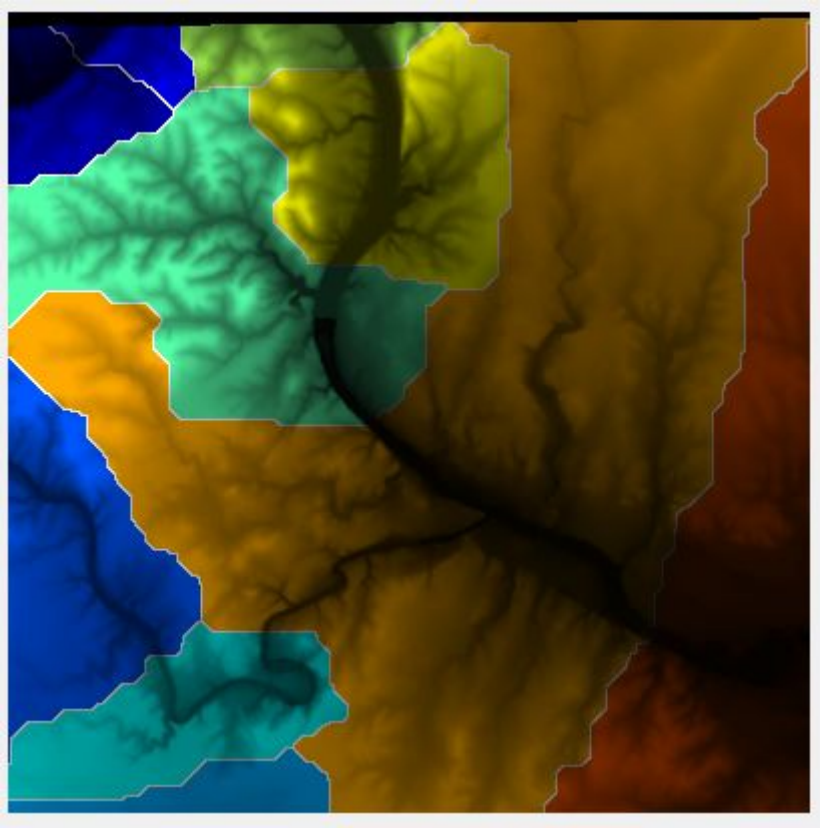
As the statement says, what we want is to remove those regions that have smaller areas than a particular size we choose, since they don't give us important information. This way, when we apply a closing filter with a structuring element of 10x10, we remove the local minimums of that image *(minima killer)*. Then we apply an opening filter *(maxima killer)* to the image we have obtained after applying the closing filter, with the aim of achieving an image with the local maximums removed too *(extrema killer)*. The following step is to do the watershed to segmentate the image with the local minimums and maximas removed into regions according to its label. We are going to obtain as many regions as colors the image will have. Finally, we multiply by the original image in order to visualize the way of the water on that original image, and the result is:



As we can see, the difference between this image and the one obtained in the exercise 1 is the number of regions in which we have segmented the image. That is because we have removed those regions with a smaller area than 10x10.

If we do the same but with an structuring element of 20x20, as the area we work with is bigger, the amount of regions that have a bigger area than 20x20, and as a consequence,

the amount of regions in which the image will be segmented, are less. Here we have the resulting image working with a structuring element 20x20:



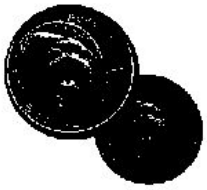
3. Extraction of objects interest or segmenting a single image

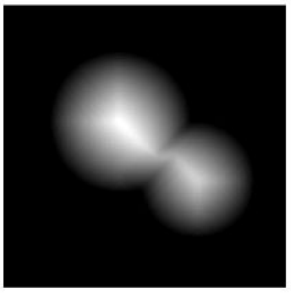
In this third exercise it is asked to implement a function that, given an image of a certain number of coins, does the segmentation of each coin and count the amount of coins.

The first step it to read and normalize the image and have it in grayscale using *rgb2gray()*. Then, we binarize the image at 0.9 in order to separate the shape of the coins of the background. We apply an opening filter to remove the holes that the binarization left in the coin and we make the complementary image, where the background becomes black and the shape of the coins white. To achieve the following step, that is to extract the edges of the coins, we apply a morphological filter, advanced morphological gradient, with a value of 1 to compute the outer borders of the image. From this we get the edges of the coins. Once we have the edges, we calculate distance function, which returns the distance between each pixel and its nearest edge. After that, we apply a small dilatation of 5 to the distance function to join the join the maximas that are very close together. Before doing the segmentation, we do the complementary image of the previous dilation of the distance function and, then, we are ready to segmentate the image with the function *whatershed()* with a 8-connected component.

At the end we compute the number of connected components of the image we have with the function *bwconncomp().*

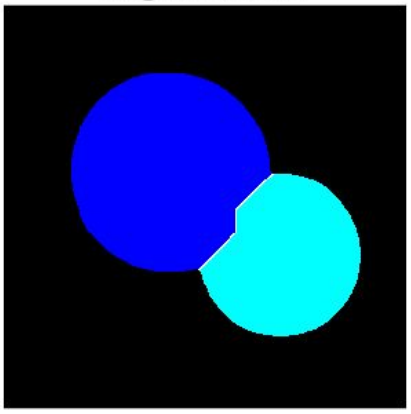
Process before segmentation





Original Image

Segmentation



```
NumberOfCoins =

     2
```