


VIP :: Assignment 5

Segmentation

Olga Iarygina (hwk263), Ioannis Manasidis (jdv382)

In this assignment, our task was to implement two segmentation algorithms (k-means and Otsu's method), as well as apply a denoising algorithm afterwards. We used the Absalon page images (cameraman, coins, rock sample slice and page images) and also included one additional picture of Ioannis's cat .

1. K-means

The first algorithm was k-means, which works in the following way: K-means is a region-based segmentation algorithm, where we extract features of the image and then group them according to a similarity measure. The resulting segments are groups that correspond to similar features. It tries to divide space into a given number of segments, where every point of a particular segment is closer to its cell centroid than to centroids of any other segments.

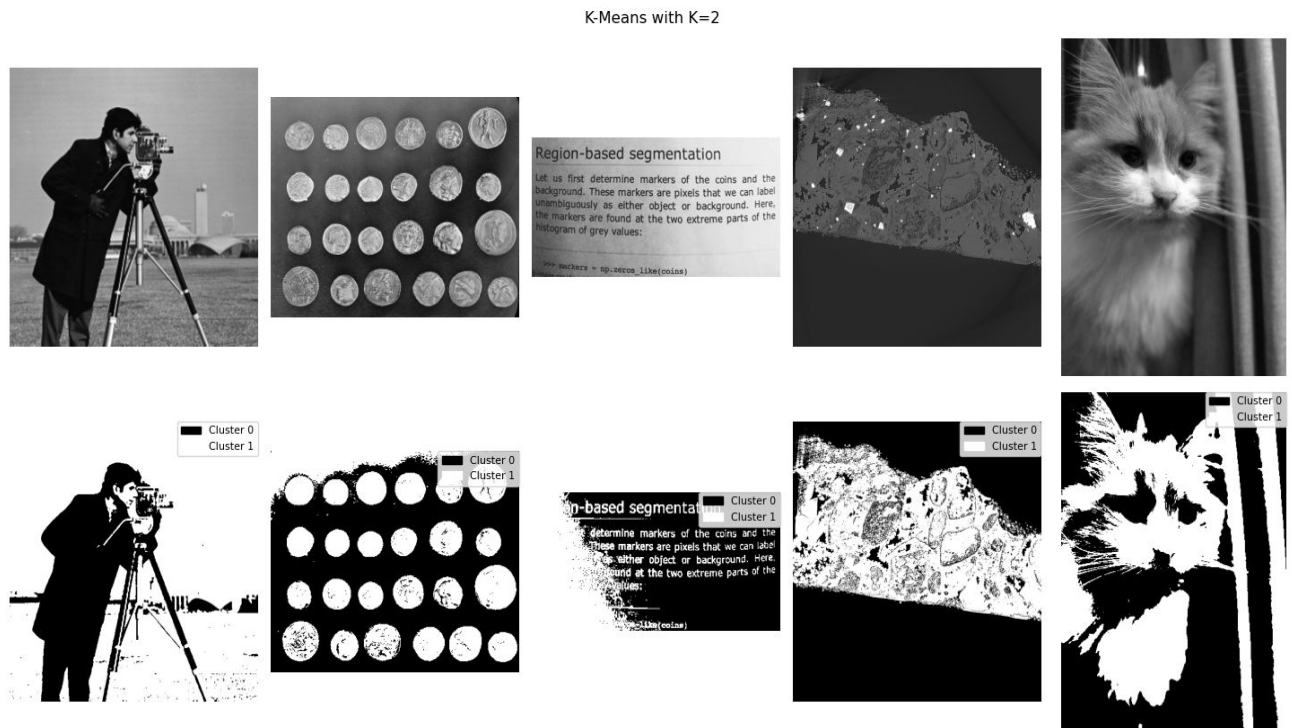
With k-means Lloyd's algorithm, we first decide about the number of clusters (k) and then try to find the optimal values of those clusters. The initial k cluster centroids are either picked randomly from k provided data points, or generated randomly based on the standard deviation and mean of the whole dataset. The cluster update is done in two constantly-repeating steps: first we assign each data point to its closest cluster by calculating its L-2 distance from every cluster and assigning the one with the least distance. Finally, we get the data points assigned to each cluster and, for each cluster/centroid calculate their mean value. These mean values, one for each cluster, become the new clusters and the two steps start over.

The loop ends when every cluster was assigned the same value it previously had (model has converged) or the maximum number of iterations has been reached (set by user/programmer)

We started with $k = 2$. Here is what we got:

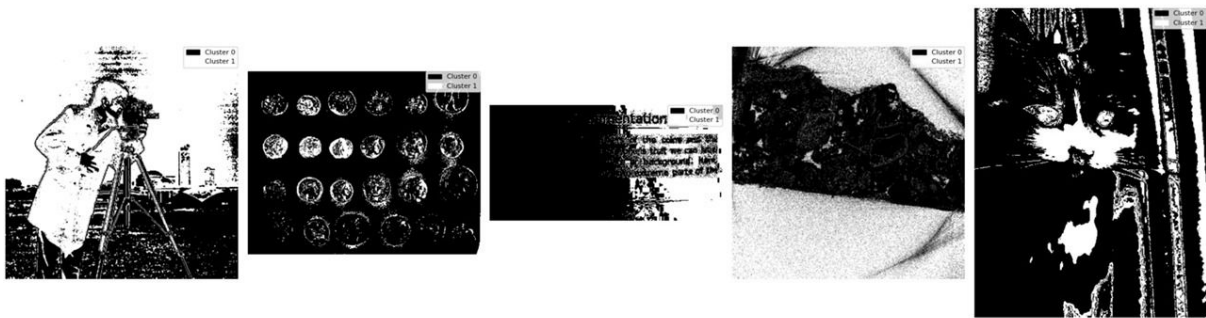
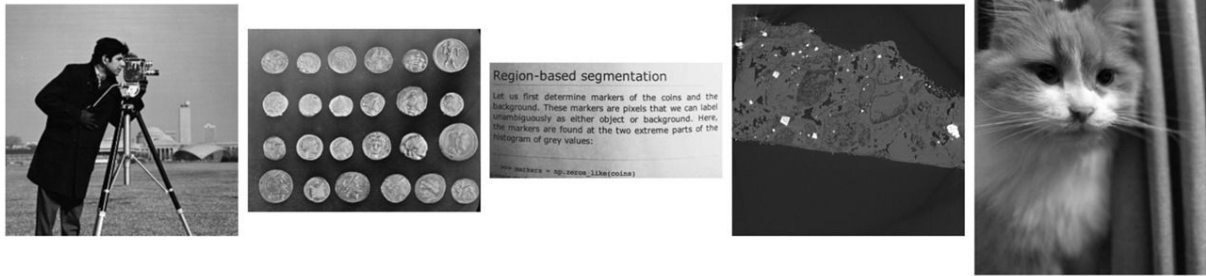
In the case of 2 centers, we can see that the segmentation seems to, on one hand, separate the object from its background, because the pixels are pretty contrasting (pictures of the cameraman, coins and the rock sample), while, on the other hand, to not make such a distinction, with the algorithm apparently separating dark and light areas from each other, where each segment is simply making up parts of the

foreground object (spots on the cat and shaded and light areas on the page). In the pictures, some noise throughout the segments can be seen, like the rock texture, buildings behind the cameraman and some roughness on coins. This happens since we make the separation based on the intensity histograms, which do not take into account the position of the pixel in the picture. We can get rid of this noise passing the segmented image to a denoising algorithm.



We ran the k-means 3 times (tries) for every image, each time starting with random initialisations, and obtained the clustering where the summed squared distance from every point to its corresponding cluster (within-cluster sum of squares) was minimal.

We implemented k-means in two modes: a histogram and a pixel mode. The first one gets the counts of every pixel intensity and performs clustering on those counts. This version is much faster, as the initial k-means input amounts to an array of 256 values, but is also very noisy, because the relative position of each pixel to every other pixel is lost when computing the intensity counts. On the other hand, the second method is way more accurate, but also takes more time to compute, because we treat every pixel as a data point for the algorithm. This means that the initial k-means input is equal to the amount of pixels in the image. We left this mode as the default, because it is way better at segmenting the image.



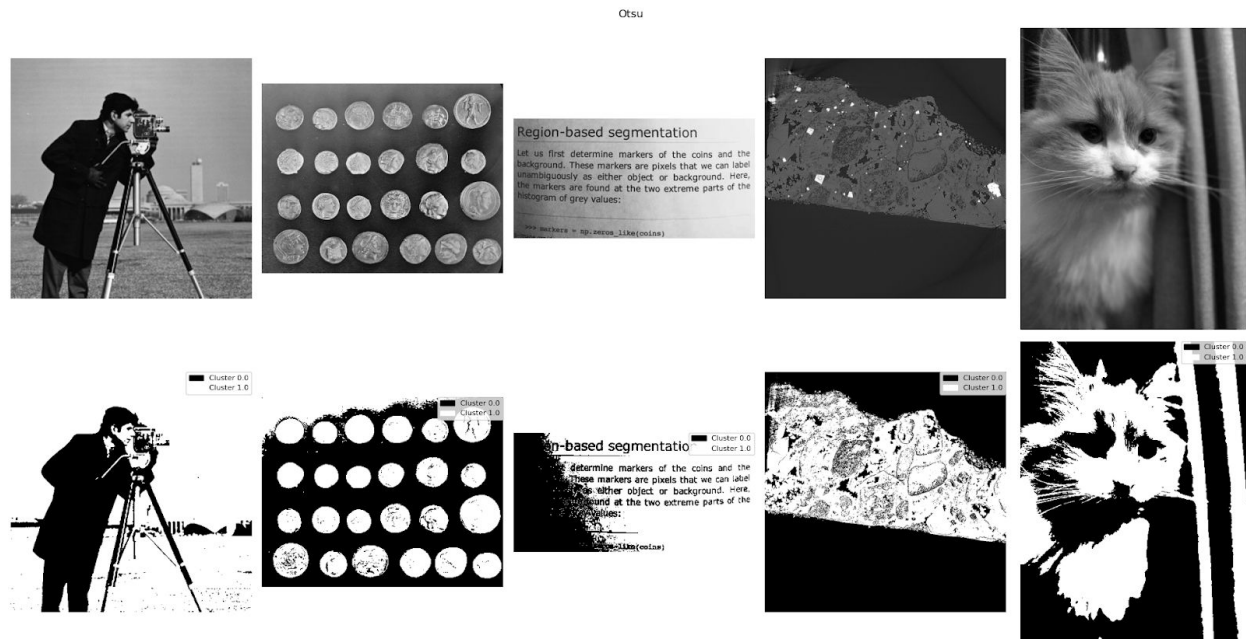
2. Otsu's thresholding algorithm

Otsu clustering method is trying to find the threshold where the intensity histogram of the image can be cut into two parts so that the left part is the first class, the right one - the second class, and the variances within the classes is minimum. At the same time, the difference between them is as large as possible. So classes are as homogeneous as possible and as distinct as possible between each other.

So, it calculates the variance within first-class, weighted by the probability that the pixel value is above the threshold and the probability that it is above. Then it searches for the perfect threshold.

As the separation in this case is also in two parts and both are based on the intensities, we can see the results very similar with what we obtained in k-means with $k = 2$. For the rock sample, cameraman and coins we see the separation of the object from its background. And for the cat and page we can see the distinction of just light and dark spots with the contrasting pixel intensities. However, we can see that, for example, in

a picture with a rock on the rock itself, we can see its relief, specks and other structural elements. A similar thing is observed with buildings in the background of a man with a camera, rubbed coins and so on. In order to get rid of these details and make the segmentation cleaner, we can do denoising.



3. Cleaning algorithm

When we perform segmentation based on the intensity histograms, usually there are some holes, especially if the original picture is noisy. This happens because we do not consider the position of the exact pixel on the image when dividing the histogram into two parts, so the result is some noise in segmentation. To deal with it, we perform denoising. Here, for each pixel we look at its neighbourhood (up-down-left-right pixels in the case for 4 pixels, and the same, but with the diagonals for 8 pixels), and look at which labels they have. Then we calculate the ratios of the neighbouring labels, each ranging from 0 to 1, get the maximum ratio, and assess whether it is higher or lower than our threshold. If it is above the threshold, we reassign the label of our initial pixel to the label that had the maximum ratio. If, on the other hand, the percentage is below our threshold, we leave the label as-is.

If we want to generalize the cleaning into three or more clusters, we can just compare the maximum percentage of the surrounding cluster labels with the threshold. We reassign the label of initial pixel only if differing labels around have reached the specified threshold.

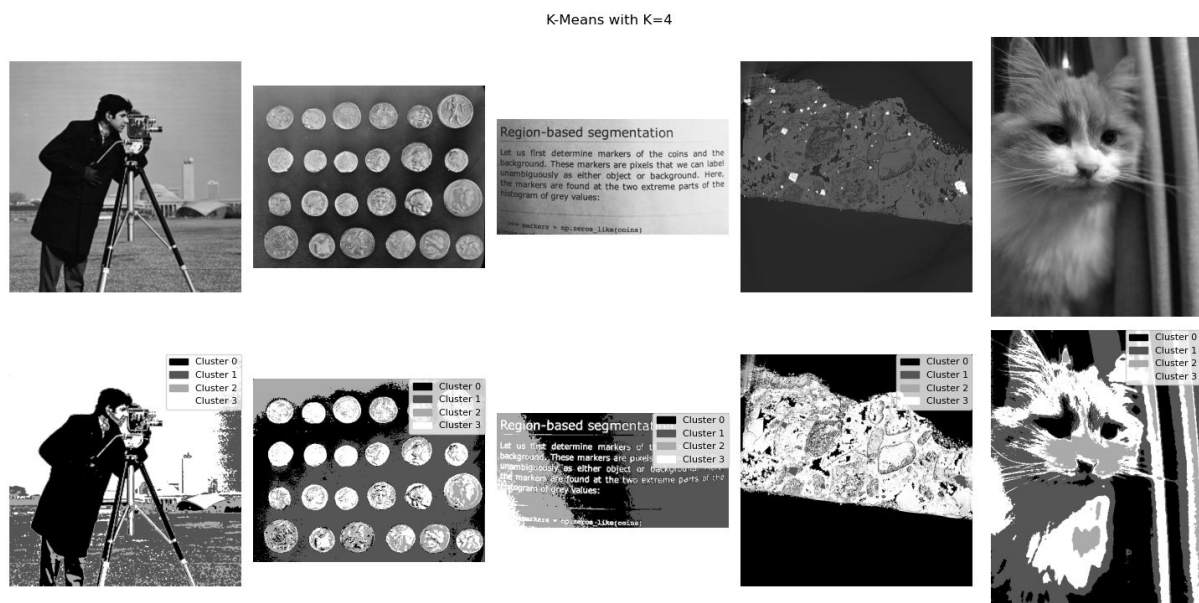
Below is an example of how the threshold affects the smoothing effect. We perform 2 passes on each of the threshold values (0.1, 0.6, 1.0) to intensify the visual effect of the smoothing.



It should be noted that, logically, the more denoising passes we set our algorithm to do, the smoother it makes the images.



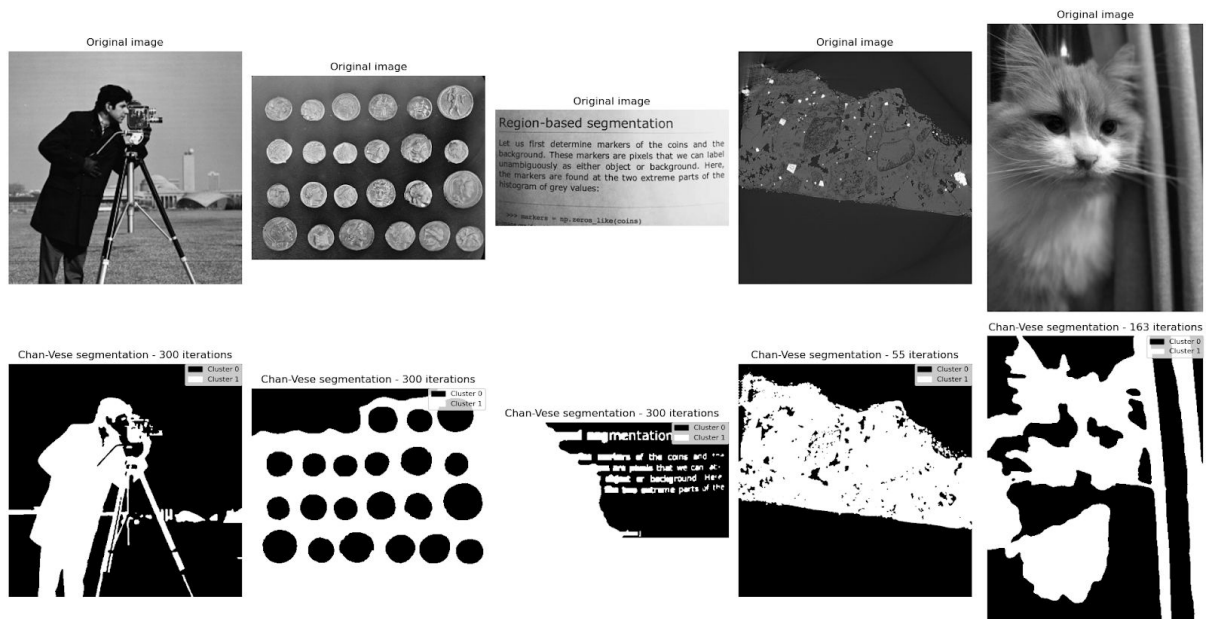
We tried increasing the number of clusters to 4, which definitely improved the segmentation with the page and cat images. This probably hints to the fact that when lighting conditions are not stable throughout the image, more clusters seem to encode that information.



We also tried out the Chan-Vese segmentation method from skimage and got the following results. The lower its μ parameter, the smoother it tries to make the image, and the more iterations it takes. If its λ parameters are less than one, a “raster”

print pattern emerges. The higher the lambda values, the more “smooth” the image becomes and the less visual “holes” are in the segmented image overall.

Below is an segmentation with $\mu=0.2$ and both λ_1 and λ_2 equal to 1.5



And here is the same segmentation, but with λ_1 and λ_2 equal to 1, which is a little bit more detailed.

