Lab report 5 Computer Vision - Matteo De Gobbi

I tried two different approaches K-means clustering and Otsu's thresholding (I also tried watershed but I didn't manage to segment the image correctly).

K-means

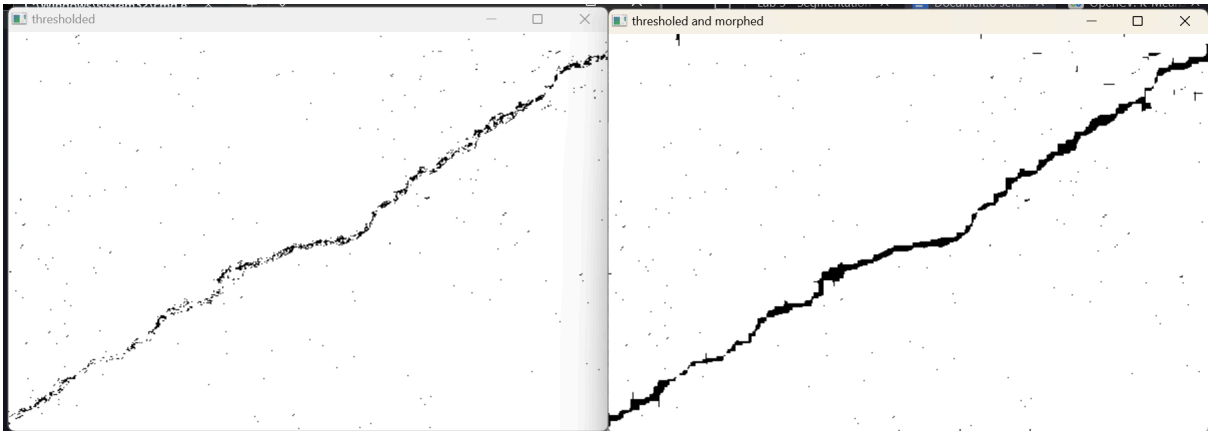
We can use the kmeans function provided by opencv which takes as input:

- the image to segment.
- the number of clusters, which I fixed as 2 because we want to segment the image into two categories {"crack","not crack"}.
- A Mat object that will contain for each pixel which is the closest centroid.
- A TermCriteria object that specifies the termination conditions of the algorithm, I used both a maximum number of iterations (10) and a minimum distance in the update of the centroid (0.01), the algorithm terminates if either of the two conditions is not met.
- Initialization method for the centroids, I used the kmeans++ version, this is more robust than random initial centroids, because centroids further apart are chosen with bigger probability (to avoid having two very close centroids).
- Mat to store the centroids.

I didn't use any preprocessing before calling kmeans (only converted the image to CV_32F) but after kmeans I threshold the image to make it a binary image, I use a threshold of 15 which works well experimentally with all 3 images.

Then I use the opening morphological operator to connect separated components of the cracks, usually the closing operator would be used in this case but the image has a white background and the crack is black so the effect is the opposite of usual.
For the morphological operator I used a 9x9 rect kernel.

Here we can see the image before and after the morphological operator:

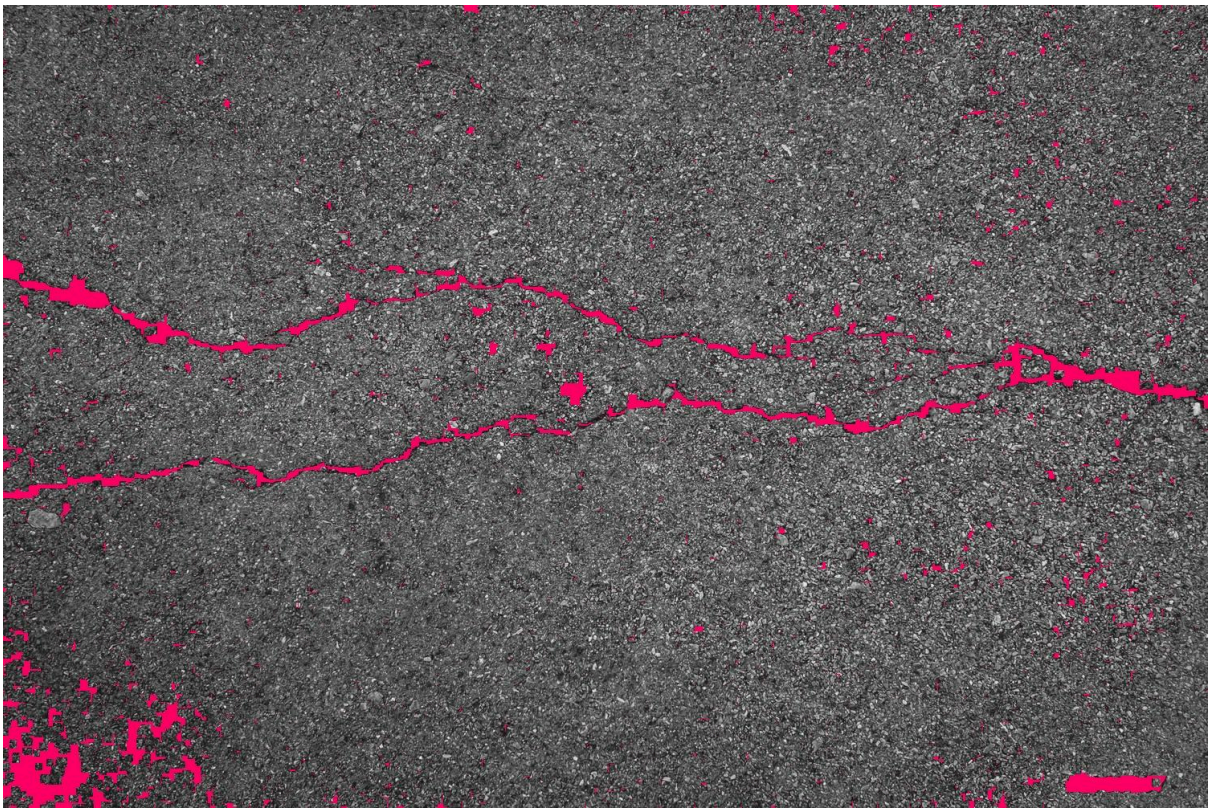


After the opening the crack in the asphalt is thicker and more connected, we can also see some noise in the asphalt due to other imperfections in the asphalt that get clustered together with the crack.

After I overlay the highlighted crack on the original image in red in order to evaluate the segmentation:

We can see how in the first image the segmentation works very well.
It works a little bit worse in the second image because it misses some smaller cracks.

In the third image it manages to identify the cracks but it also clusters some other parts of the asphalt with the cracks, this is probably due to the fact that in those parts the asphalt is darker so it has a color more similar to the cracks than to the rest of the asphalt.

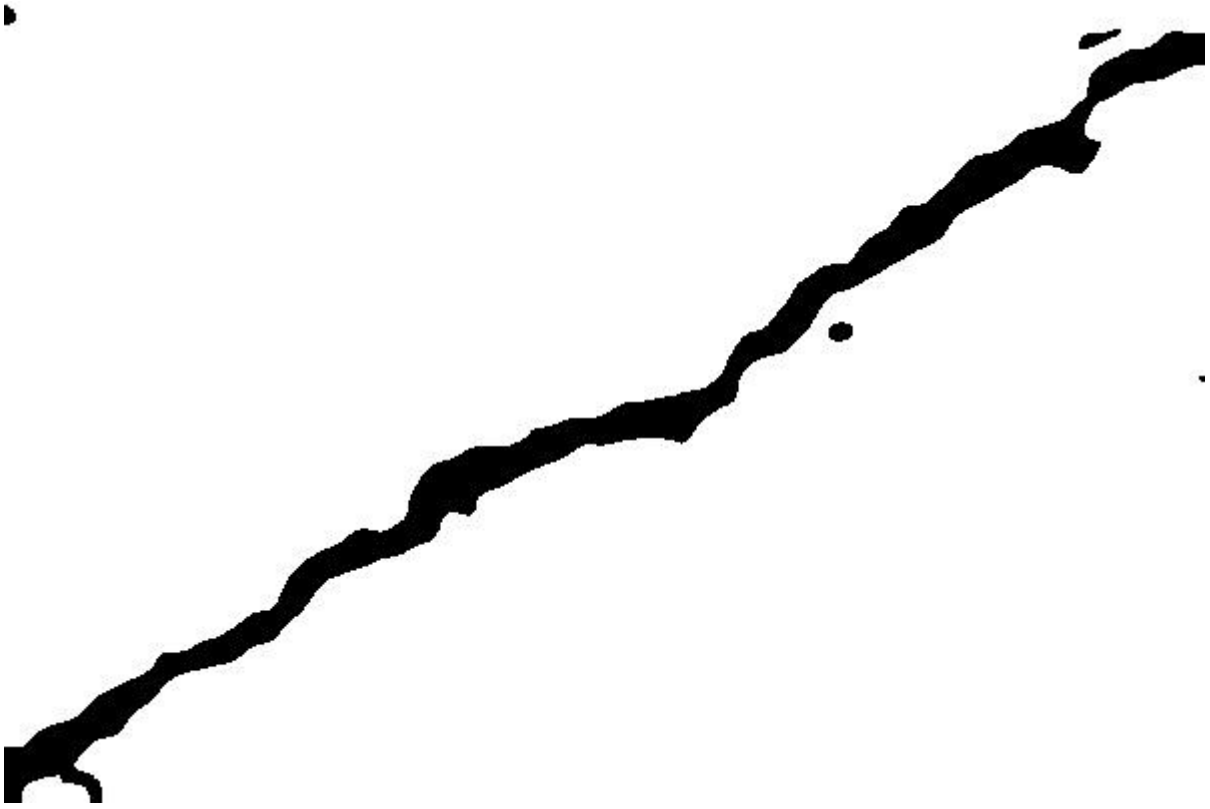


By changing the threshold when converting the image to a binary one we can obtain better results in the second image but this is to the detriment of the other two, so i settled on this threshold of 15 as a good middle ground.

Otsu's thresholding

For completeness I also show the results I obtained with Otsu even though they are worse than the previous method.
I found that this method is very sensitive to noise so before thresholding I use a big (21x21) Gaussian kernel to blur the image to reduce noise:

After this blurring I can apply Otsu's thresholding and also make the image binary at the same time by using the flag THRESH_OTSU + THRESH_BINARY.
With the first image it works quite well, we can see the crack and there isn't almost any noise:



For the second image the thresholding doesn’t work because the yellow road line makes otsu algorithm choose a threshold that separated the road line from all the rest:

For the third image we can see the crack but also there’s a lot of noise due to the darker parts of the asphalt being included together with the cracks by the thresholding: