Computer Vision

Alessia Paccagnella

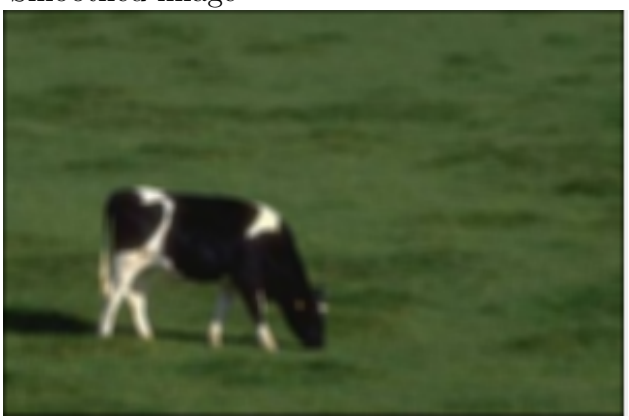
November 6, 2019

1 Image Preprocessing

First of all I smoothed the image with a gaussian filter ($\sigma = 5$) by using the functions `fspecial` and `imfilter`.
Secondly, I converted the image from RGB to `L*a*b` colorspace.
The `L*a*b` is better to use because the distance from colors in `L*a*b` color space maps better to the perceptual difference of them. Two very different colors will be have very different values in this space. Also, in the `L*a*b` color space the perceptual difference of colors does correspond to the Euclidean distance within the vector space, while this does not happen with the RGB vector space.

Original image

Smoothed image

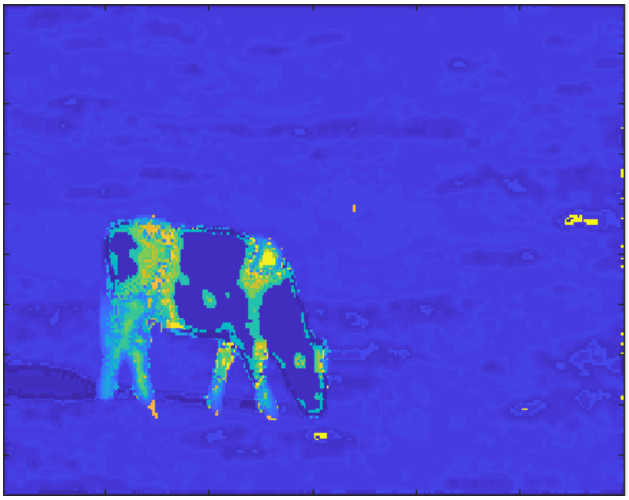


Converted image to L*a*b color space

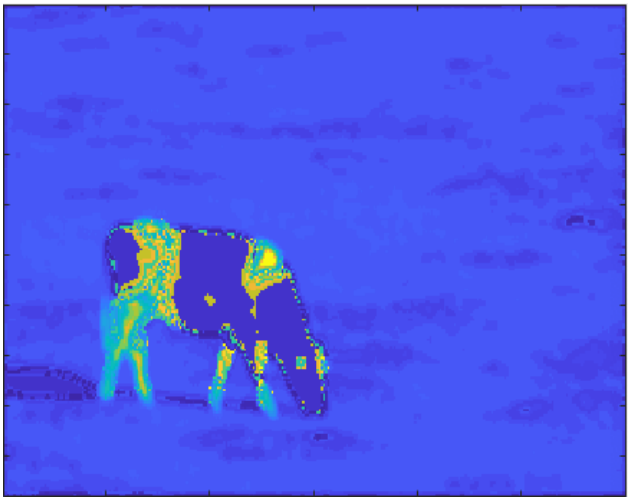
2 Mean-Shift Segmentation

The first step was to implement the function `findpeak`, that receives in input the matrix `X` (Lx3), `xl` (the single pixel) and a radius `r`, which is the radius of the spherical window in which we are looking for the peak. I tried with different radius.
In the function, I find the distance between every pixel and xl, and find the ones that are inside the circle of radius r (with which the distance is minor or equals to r). After that I find the centroid. I shift the window to the mean and I keep repeating the cycle until convergence.
After that I implemented the mean-shift algorithm in the meanShiftSeg.m. I reshape the img to get the X matrix (Lx3), initialize the map and the first peak and then start looping on all the pixels. This is the approach that I chose: everytime I find out that the minor distance between two peaks is major than r/2, I add the new peak to the map, otherwise I merge the two peaks together and I don't add the new one to the map.
I tried to run it with different radius, getting the best result for r = 0.04 or r = 0.035.

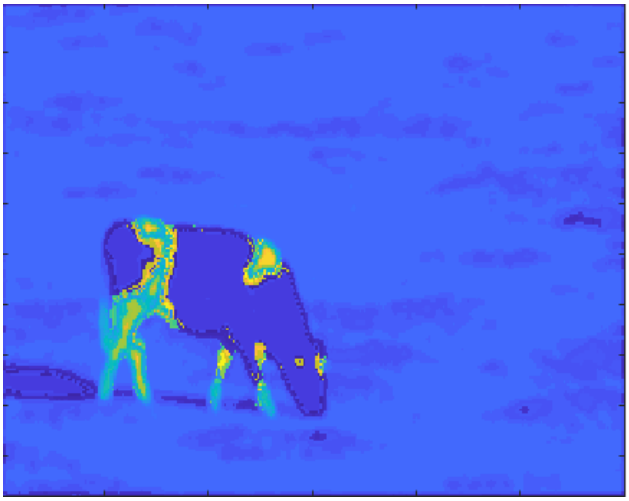
r = 0,03



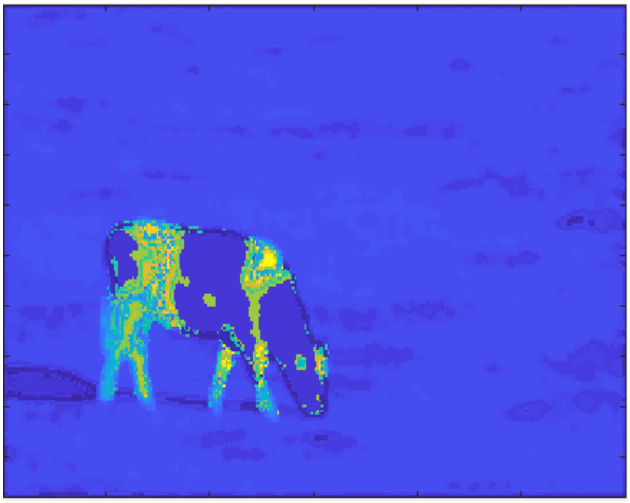
r = 0.04



r = 0,05



r = 0.035



3 EM Segmentation

I had to implement the two functions of `expectation` and `maximization`.

3.1 Expectation

To implement this function, first I had to initialize the matrixed of Mean, Covariance and Alpha. To do so, I initialized a matrix of ranges, that held the max and min value of all the pixels for L, a and b.
Mu is the mean vector, has a size of 3xK and is generated with the function linspace, that spreads them equally in the L*a*b space as requested.
Alpha is initialized with the formula on the slides.
The covariance matrix instead is initialized as a 3x(3xK) matrix: it's a matrix of matrixes of covariance, one for each K. Every covariance submatrix is a diagonal matrix with elements corresponding to the range of the L*a*b values.
I calculate the expectation matrix with the following probability formula and return it.

$$p(x_l|\theta_k^{(s)}) = \frac{1}{(2\pi)^{n/2}\left|\Sigma_k^{(s)}\right|^{1/2}} \exp -\frac{1}{2}(x_l - \mu_k^{(s)})^T \Sigma_k^{(s)-1}(x_l - \mu_k^{(s)})$$

3.2 Maximization

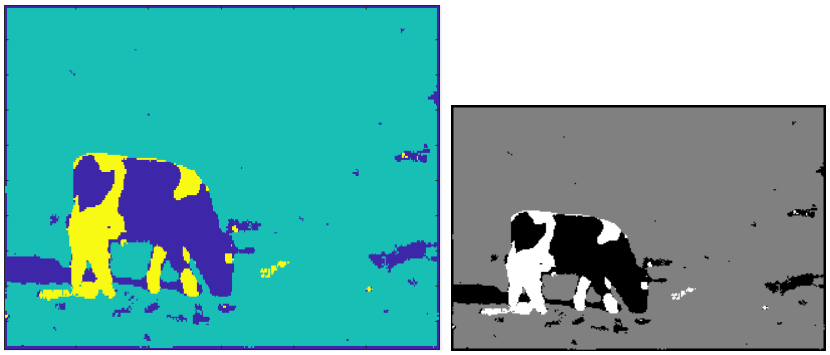
I update the parameters calculated in the expectation step with the following formulas.

$$\alpha_k^{(s+1)} = \frac{1}{L}\sum_{l=1}^{L} I_{lk} \qquad \mu_k^{(s+1)} = \frac{\sum_{l=1}^{L} x_l I_{lk}}{\sum_{l=1}^{L} I_{lk}} \qquad \Sigma_k^{(s+1)} = \frac{\sum_{l=1}^{L} I_{lk}\left\{(x_l - \mu_k^{(s+1)})(x_l - \mu_k^{(s+1)})^T\right\}}{\sum_{l=1}^{L} I_{lk}}$$

3.3 Iteration

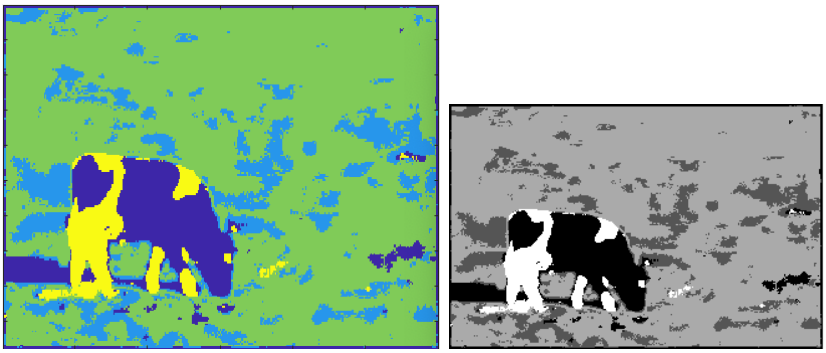
I wrote the code that iterated between the expectation step and the maximization one until it converges. I tried to compute it for different K as requested (such as K = 4, K = 5).
K = 3 seems to obtain better results than the others.

K = 3



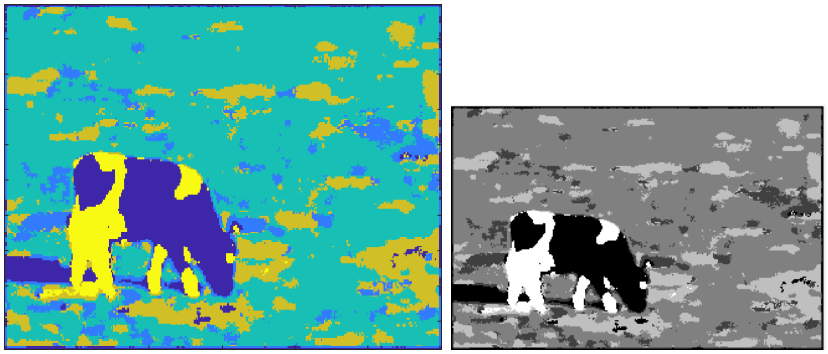
```
alpha =

    0.1473    0.8000    0.0526


covariance =

    0.0113   -0.0023    0.0033    0.0008   -0.0000    0.0000    0.0368    0.0023   -0.0011
   -0.0023    0.0006   -0.0008   -0.0000    0.0000   -0.0000    0.0023    0.0004   -0.0003
    0.0033   -0.0008    0.0011    0.0000   -0.0000    0.0000   -0.0011   -0.0003    0.0005


mu =

    0.1897    0.3508    0.4765
    0.4759    0.4488    0.4818
    0.5458    0.5847    0.5610
```

K = 4



```
alpha =

    0.1334    0.3351    0.4816    0.0499


covariance =

    0.0107   -0.0021    0.0031    0.0009   -0.0000    0.0000    0.0008    0.0000   -0.0000    0.0373    0.0021   -0.0008
   -0.0021    0.0006   -0.0007   -0.0000    0.0000   -0.0000    0.0000    0.0000   -0.0000    0.0021    0.0003   -0.0003
    0.0031   -0.0007    0.0010    0.0000   -0.0000    0.0000   -0.0000   -0.0000    0.0000   -0.0008   -0.0003    0.0005


mu =

    0.1775    0.3510    0.3492    0.4850
    0.4786    0.4494    0.4485    0.4836
    0.5420    0.5853    0.5843    0.5594
```

K = 5



```
alpha =

    0.1175    0.1593    0.3851    0.2902    0.0478


covariance =

    0.0099   -0.0019    0.0029    0.0014   -0.0001    0.0001    0.0006   -0.0000    0.0000    0.0010    0.0000   -0.0000    0.0375    0.0019   -0.0007
   -0.0019    0.0005   -0.0007   -0.0001    0.0000   -0.0000   -0.0000    0.0000   -0.0000    0.0000    0.0000   -0.0000    0.0019    0.0003   -0.0002
    0.0029   -0.0007    0.0010    0.0001   -0.0000    0.0000    0.0000   -0.0000    0.0000   -0.0000   -0.0000    0.0000   -0.0007   -0.0002    0.0005


mu =

    0.1618    0.3410    0.3561    0.3432    0.4926
    0.4822    0.4502    0.4490    0.4481    0.4850
    0.5371    0.5846    0.5844    0.5847    0.5582
```

Everything that I tested with the cow image was tested also with the zebra image. I do not attach the pictures for those tests as I would fill more than 4 pages.