Computer Vision

Assignment 5: Image segmentation

Student: Ivan Alberico

1. Image preprocessing

Image preprocessing is an important step to do before any segmentation algorithm. In this case, we first smooth the image with a Gaussian filter with a window size of 5x5 and $\sigma = 5$. Then, we convert the smoothed RGB image into the L*a*b color space, through the Matlab functions `makecform('srgb2lab')` and `applycform`. For image segmentation tasks, shifting from the RGB to the L*a*b domain ican lead to better performances of the algorithm. This is so because while in the RGB domain each pixel is expressed by 3 color values, in the L*a*b domain each pixel is expressed according to a lightness component (the L value) and two color values, namely a and b. In this way, the image is more robust against changes in the lightning of the same color, since these changes affect only the L value, while in the RGB domain all the three components. Hence, areas with the same color but different shading will be still very likely to be segmented together, since the a and b values would still have comparable values.

Figure 1 — Image segmentation using the mean-shift algorithm. 258 clusters have been detected with a radius of 5 and a convergence threshold of 0.05.

2. Mean-Shift segmentation

In order to implement the mean-shift algorithm we first need to compute the density function X, which is a L x 3 matrix containing the L*a*b values for each pixel of the image, where L is the number of pixels.

Next step concerns the implementation of the function `find_peak.m` which returns the mode of the density function X for each pixel. To do so, we need to iteratively compute the relative distance of the specific point from all the other L-1 points (Euclidean distance) and check whether these distances are smaller than a certain specified radius. We then pick only those points satisfying this condition, we average them and compute the distance of the new mean from the one obtained at the previous iteration. This whole procedure is repeated until convergence, such as until the mean shift becomes less then a specified threshold.

That being done, we can now implement the mean-shift algorithm that calls the `find_peak.m` function for each pixel in X. This is done in the function `meanshiftSeg.m`, which returns the matrix `map`, containing the membership of each pixel to its cluster.

At each iteration, we have to check the distance of the current peak from all the ones previously computed. We distinguish the following cases:

- For the first pixel we do not have to perform any check on the distance from other peaks (since there are not any yet). Hence, we just add the first cluster ID in the map matrix.

- From the second to the last pixel, once we get the peak, we have to check whether its distance from the other peaks is bigger or smaller than a certain threshold r/2, to see whether they belong to the same cluster or not. If the peak related to the specific pixel has a distance larger than r/2 with respect to all the previous peaks, it means that it belongs to a new cluster, so we add a new cluster ID in the map matrix.

- Instead, if the peak has a distance smaller than r/2 with respect to only another peak, it means that they belong to the same cluster and so they have the same cluster ID.

- Finally, if the peak has a distance smaller than r/2 with more than one of the previous peaks, we merge it to the closest peak, meaning that the pixel will be assigned to the same cluster of that with the smallest distance.

Different results were observed by setting different values for the window size (the radius):

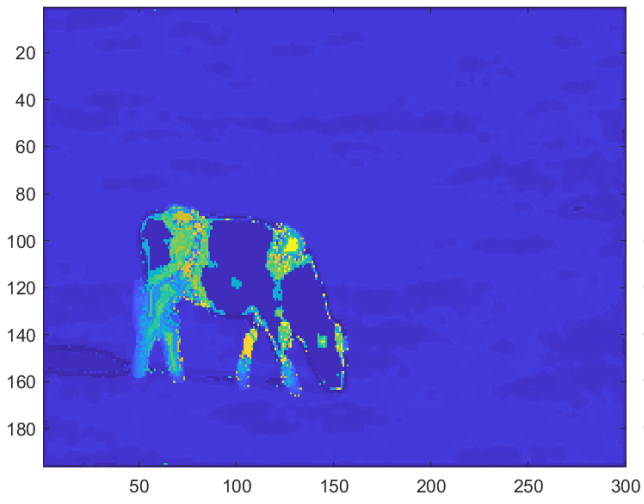

Figure 2 — Image segmentation using the mean-shift algorithm. 258 clusters have been detected with a radius of 5 and a convergence threshold of 0.05.

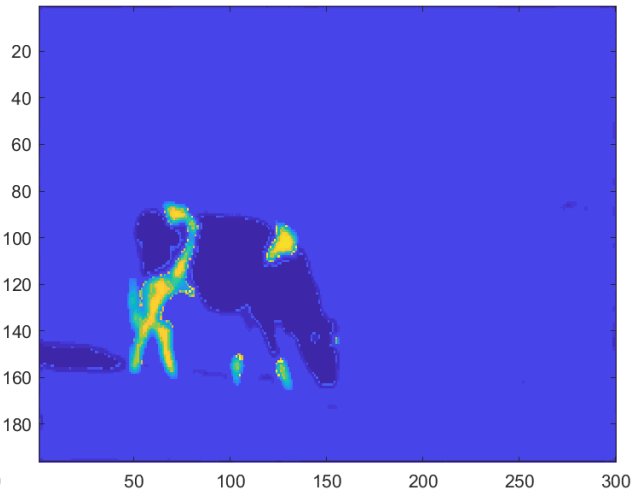

Figure 3 — Image segmentation using the mean-shift algorithm. 29 clusters have been detected with a radius of 10 and a convergence threshold of 0.05.

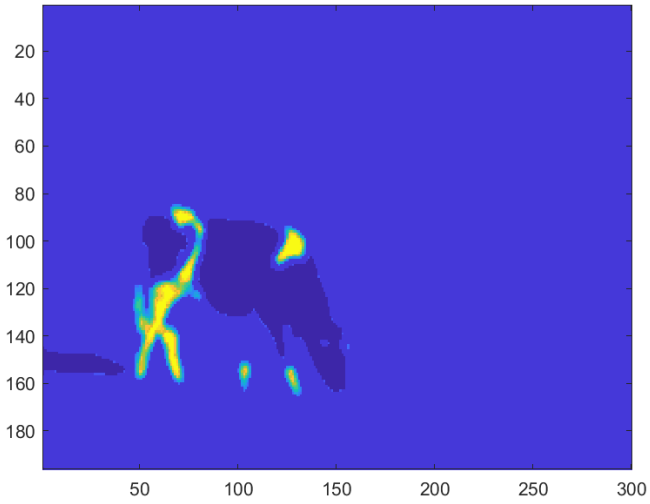

Figure 4 — Image segmentation using the mean-shift algorithm. 15 clusters have been detected with a radius of 15 and a convergence threshold of 0.05.

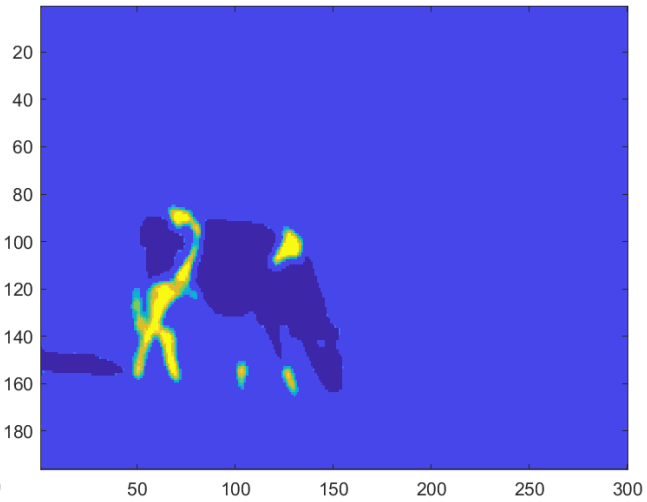

Figure 5 — Image segmentation using the mean-shift algorithm. 10 clusters have been detected with a radius of 20 and a convergence threshold of 0.05.

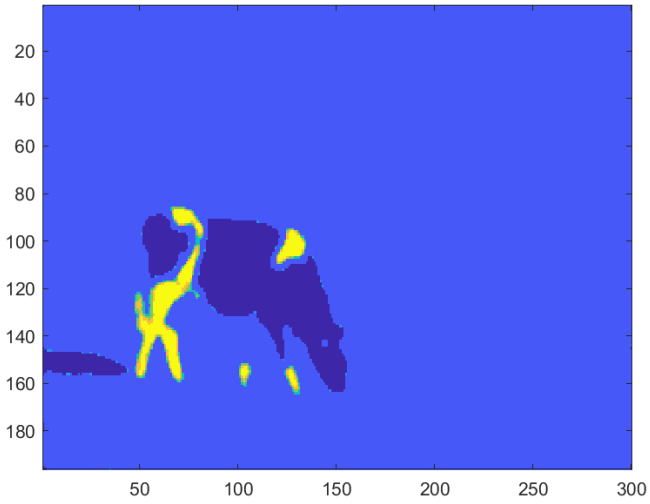

Figure 6 — Image segmentation using the mean-shift algorithm. 7 clusters have been detected with a radius of 25 and a convergence threshold of 0.05.

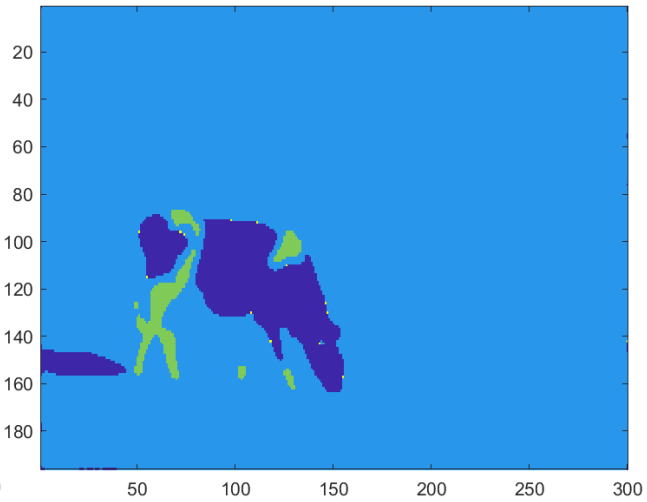

Figure 7 — Image segmentation using the mean-shift algorithm. 4 clusters have been detected with a radius of 30 and a convergence threshold of 0.05.

From the pictures above, it can be seen that the choice of these hyperparameters plays an important role in the performance of the algorithm. The window size, in particular, is very important. In fact, the smaller is the radius and the more clusters we are taking into account in the segmentation process. On the other hand, the choice of the window size can be seen also as a limitation of the algorithm itself, since in this way we have the dependency of the output on a parameter, which is also not trivial to determine.

The convergence threshold is important as well. In fact, the lower is this threshold and the more accurate the result will be, since we are getting closer to the real mode of the function and the approximation is more accurate.

From the test I performed, I was also able to see that the mean-shift algorithm is computationally rather expensive, as it takes a lot of time to run if we consider the entire image, without resizing it.

3. EM segmentation

Image segmentation could be also implemented in a probabilistic manner through the Expectation Maximization algorithm. The general idea is that the image is set to be a mixture of K Gaussians (generative model), where K is the number of components we assume to have in the image. Each mixture model has a mean and a covariance matrix that is unknown to us and need to be learned.

The first step is to initialize the parameters. As it was suggested in the assignment slides, α was initialized to $1/K$, while the mean μ was initialized such that each μ_k is a 3×1 vector representing a point in the L*a*b domain, and such that they are equally spread in the space.

The covariance matrices $\Sigma_{1:k}$ are set to be 3×3 diagonal matrices with elements corresponding to the range of the L, a and b values.

After that the parameters have been initialized, we can start running the algorithm by alternating between Expectation and Maximization steps.

In the Expectation step, we compute the probability for each pixel to belong to the segments, according to the previous values of α, μ and Σ. This is done though the following expressions:

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

$$I_{lk} = p(\alpha_k|\,x_l, \theta_k) = \frac{\alpha_k\, p(x_l|\,\theta_k)}{\sum_{k=1}^{K} \alpha_k\, p(x_l|\,\theta_k)}$$

In the Maximization step, instead, we update the parameters α, μ and Σ for all K segments such that they maximize the expectation of the complete log likelihood over the parameters θ, as in the following way:

$${\alpha_k}^{(t+1)} = \frac{1}{L}\sum_{l=1}^{L} I_{lk} \qquad {\mu_k}^{(t+1)} = \frac{\sum_{l=1}^{L} x_l\, I_{lk}}{\sum_{l=1}^{L} I_{lk}}$$

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

The alternation between the E and M steps is done repeatedly until the mean shift becomes lower than a certain threshold. Finally, the cluster membership for each pixel is assigned by picking the index of the maximum value among the probabilities of the K segments (meaning that the pixel is more likely to belong to that specific cluster). Furthermore, it was observed that normalizing the images before running the EM algorithm, would improve the performance of the segmentation and reduce the running time.

Results for different values of K are shown below:

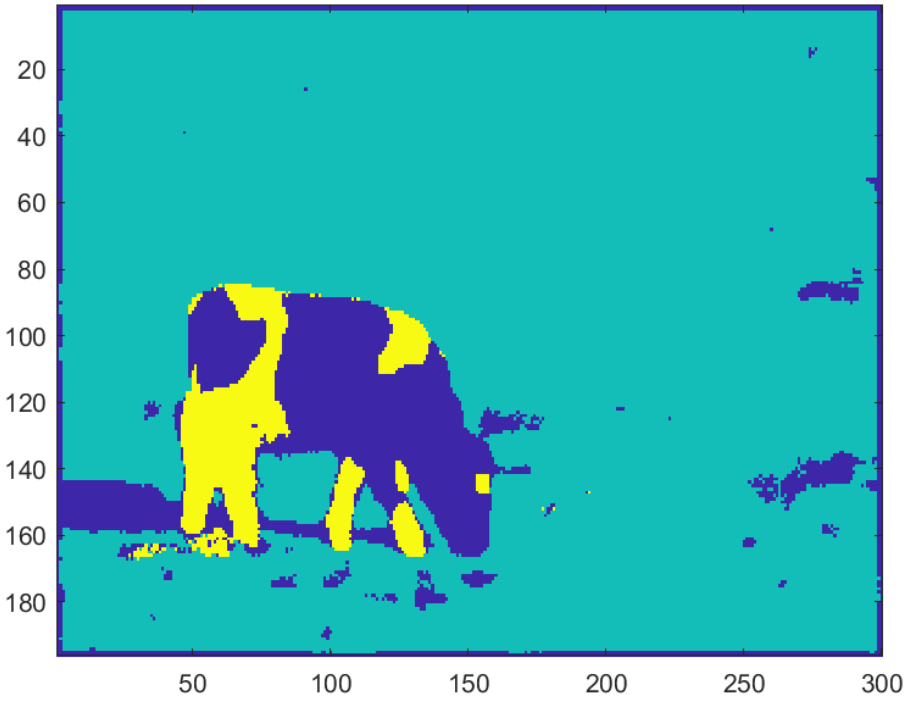

Figure 8 — Image segmentation using the EM algorithm, with normalized images, K = 3 and convergence threshold = 0.05

$$\Sigma_1 = \begin{pmatrix} 3.2954 & -0.6510 & 0.9796 \\ -0.6510 & 0.1717 & -0.2252 \\ 0.9796 & -0.2252 & 0.3308 \end{pmatrix} \qquad \mu_1 = (47.4625 \quad 121.5266 \quad 139.0500) \qquad \alpha_1 = 0.1450$$

$$\Sigma_2 = \begin{pmatrix} 0.2505 & 0.0000 & 0.0023 \\ 0.0000 & 0.0025 & -0.0007 \\ 0.0023 & -0.0007 & 0.0072 \end{pmatrix} \qquad \mu_2 = (89.3763 \quad 114.4499 \quad 149.0995) \qquad \alpha_2 = 0.8115$$

$$\Sigma_3 = \begin{pmatrix} 11.2186 & 0.4666 & -0.0517 \\ 0.4666 & 0.0723 & -0.0604 \\ -0.0517 & -0.0604 & 0.1443 \end{pmatrix} \qquad \mu_3 = (130.4543 \quad 124.3842 \quad 141.6248) \qquad \alpha_3 = 0.0435$$

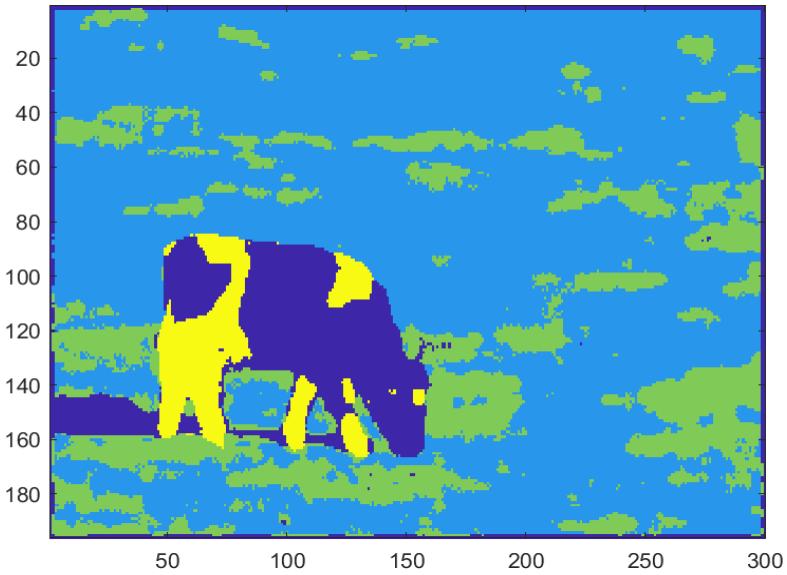

Figure 9 — Image segmentation using the EM algorithm, with normalized images, K = 4 and convergence threshold = 0.05

$$\Sigma_1 = \begin{pmatrix} 3.0673 & -0.5799 & 0.866 \\ -0.5799 & 0.1483 & -0.1943 \\ 0.866 & -0.1943 & 0.2832 \end{pmatrix} \qquad \mu_1 = (42.6386 \quad 122.7395 \quad 137.2501) \qquad \alpha_1 = 0.1231$$

$$\Sigma_2 = \begin{pmatrix} 0.1453 & -0.0019 & 0.0035 \\ -0.0019 & 0.0025 & -0.0008 \\ 0.0035 & -0.0008 & 0.0063 \end{pmatrix} \qquad \mu_2 = (91.6650 \quad 114.4911 \quad 148.9573) \qquad \alpha_2 = 0.6143$$

$$\Sigma_3 = \begin{pmatrix} 0.3047 & -0.0015 & 0.0239 \\ -0.0015 & 0.0100 & -0.0004 \\ 0.0239 & -0.0004 & 0.0097 \end{pmatrix} \qquad \mu_3 = (81.5858 \quad 114.3781 \quad 149.5503) \qquad \alpha_3 = 0.2220$$

$$\Sigma_4 = \begin{pmatrix} 11.4994 & 0.3456 & 0.1432 \\ 0.3456 & 0.0584 & -0.0404 \\ 0.1432 & -0.0404 & 0.1198 \end{pmatrix} \qquad \mu_4 = (133.3341 \quad 125.0050 \quad 140.8083) \qquad \alpha_4 = 0.0406$$

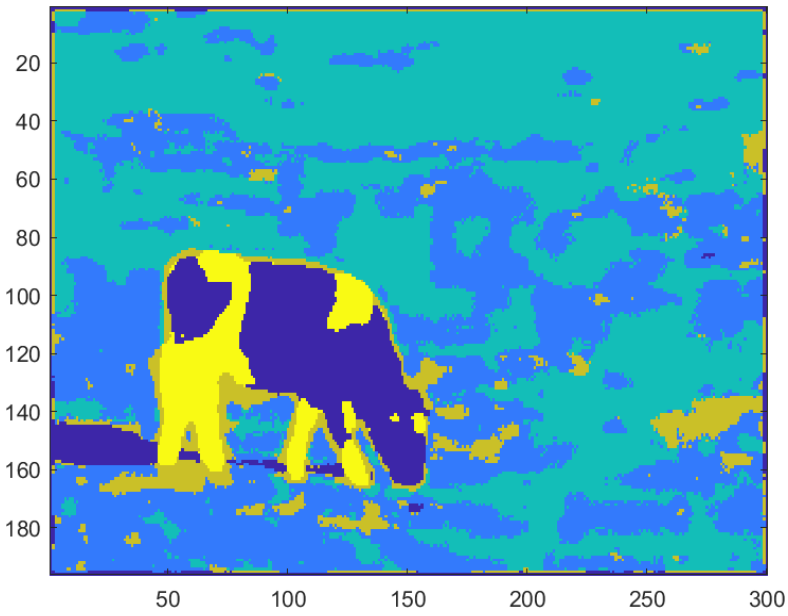

Figure 10 — Image segmentation using the EM algorithm, with normalized images, K = 5 and convergence threshold = 0.05

$$\Sigma_1 = \begin{pmatrix} 1.9619 & -0.4310 & 0.6268 \\ -0.4310 & 0.1311 & -0.1651 \\ 0.6268 & -0.1651 & 0.2341 \end{pmatrix} \qquad \mu_1 = (33.7372 \quad 124.1145 \quad 135.0885) \qquad \alpha_1 = 0.0972$$

$$\Sigma_2 = \begin{pmatrix} 0.3665 & -0.0080 & 0.0288 \\ -0.0080 & 0.0046 & -0.0016 \\ 0.0028 & -0.0016 & 0.0078 \end{pmatrix} \qquad \mu_2 = (86.7071 \quad 114.4781 \quad 149.9070) \qquad \alpha_2 = 0.3202$$

$$\Sigma_3 = \begin{pmatrix} 0.1209 & 0.0035 & -0.0005 \\ 0.0035 & 0.0025 & -0.0003 \\ -0.0005 & -0.0003 & 0.0026 \end{pmatrix} \qquad \mu_3 = (91.7692 \quad 114.4265 \quad 148.5193) \qquad \alpha_3 = 0.4554$$

$$\Sigma_4 = \begin{pmatrix} 0.3169 & 0.0183 & 0.0287 \\ 0.0183 & 0.0250 & -0.0106 \\ 0.0287 & -0.0106 & 0.0313 \end{pmatrix} \qquad \mu_4 = (79.9713 \quad 115.4061 \quad 148.4116) \qquad \alpha_4 = 0.0886$$

$$\Sigma_5 = \begin{pmatrix} 12.6462 & 0.1804 & 0.3856 \\ 0.1804 & 0.0442 & -0.0295 \\ 0.3856 & -0.0295 & 0.1151 \end{pmatrix} \qquad \mu_5 = (133.8232 \quad 125.7238 \quad 139.9514) \qquad \alpha_5 = 0.0386$$

Overall, the results obtained with the EM algorithm are pretty satisfying. First of all, I could see that the algorithm was much less computationally expensive compared to the mean-shift algorithm, since it took much less time to run. Of course, the final result depends on the value of K we choose, and the higher is K the more clusters and segmented areas we will have in our image.

However, in all the 3 cases tested, I could see that there is always a class which predominates over the others (it can been seen from the values of α), which is the one related to the background. The values of the parameters α_k, μ_k and Σ_k which have been written above are referred to the case with normalized images.

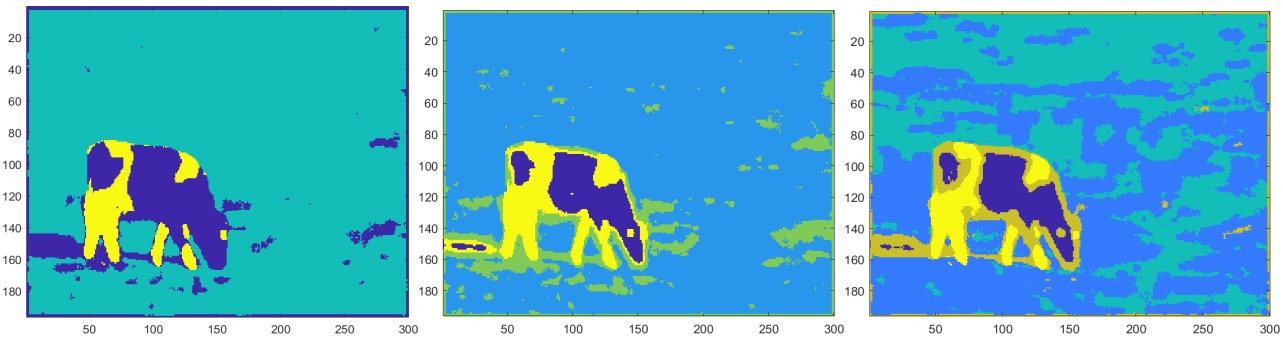

Figure 11 — Results obtained from the EM algorithm, without normalizing the images, respectively for the cases with K=3, K=4 and K=5 and convergence threshold = 0.05. It is evident that the results are slightly less accurate with respect to that previously shown with the normalization.

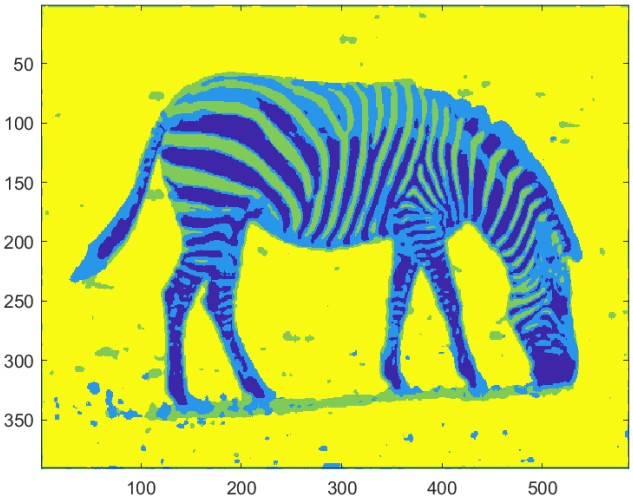

Figure 12 — Image segmentation using the EM algorithm with K = 4 and convergence threshold = 0.05

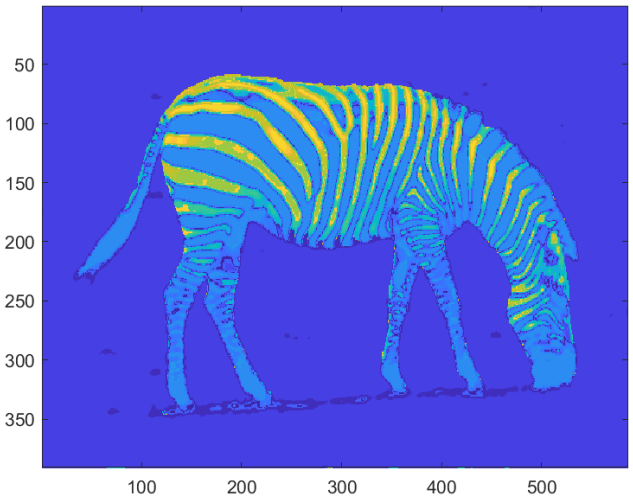

Figure 13 — Image segmentation using the mean-shift algorithm with radius = 15 and convergence threshold = 0.05