Image Segmentation
Computer Vision Lab 5

Álvaro Belmonte Baeza
Student ID 19-940-386

November 28, 2020

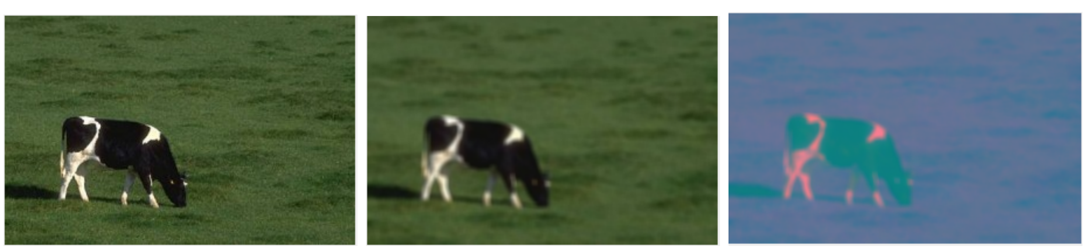
Abstract

In this fifth assignment of the Computer Vision course, we addressed the task of image segmentation, implementing and comparing two different algorithms. First of all, a necessary preprocessing of the image is performed, in order to represent the information in a color space which is more suitable for segmentation tasks than RGB. Then, the implementation of the Mean-Shift and the Expectation Maximization algorithms will be presented, as well as their results and the advantages and disadvantages observed for each method. Finally, some conclusions after the completion of the whole assignment will be discussed.

1 Image Preprocessing

In this first section, the preprocessing performed for the image used in this assignment will be presented. We begin by smoothing the image with a gaussian kernel of window size 5×5 and $\sigma = 5.0$, in order to get more uniform pixel values in each region. Then, we change the color space of the image from RGB to L*a*b* space. This is done because the RGB color space is not well suited to differentiate between colors. On the other hand, the L*a*b* space explicitly differentiates color (a*,b*) and Lightness (L*), which makes it ideal to perform color-based segmentation without being much affected by lightning conditions in the image.

The results of applying the described image processing are shown in Figure 1.



(a) Original Image. (b) Smoothed Image. (c) Image in L*a*b* space.

Figure 1: Preprocessing performed to the image before segmentation.

Other color spaces could have been used as well, such as the HSV color space, that also separates color and lightning. In the end, the key concept here is that we should choose a color space that fits with the task that we want to perform, in our case image segmentation.

Now that we have our image ready, we can get to implementing the algorithms proposed for this assignment.

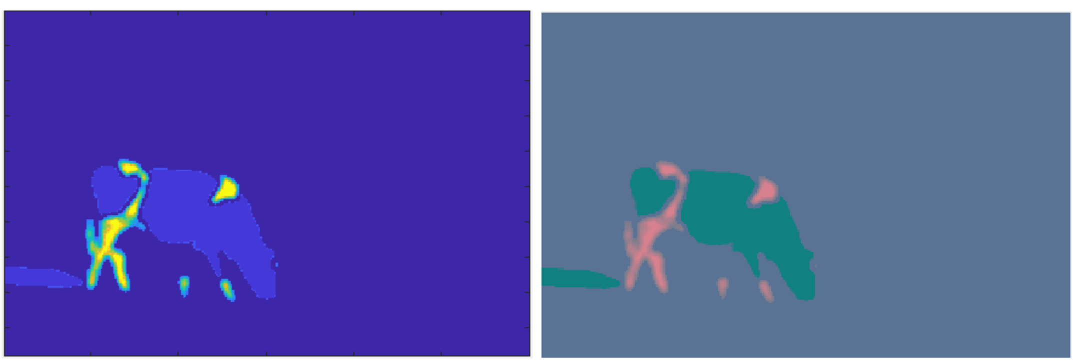
2 Mean-Shift Segmentation

In this second section, the implementation and results of the Mean-Shift algorithm for segmentation are presented. Mean-Shift consists in iteratively computing the mean of all pixels inside a spherical window of a defined radius, and then shift that window until getting to a point where it doesn't move anymore (up to a specified threshold).

The first step is creating the density function X in L*a*b* space. This function is a $3 \times N$ matrix, where N is the number of pixels in the image, and the three rows correspond to each channel in the L*a*b* image represented as a 1D vector.

Next, we implement the function *find_peak* as instructed in the assignment. This function computes the mode of X for each pixel. We compute the distance of this pixel to the rest and keep those with a SSD distance smaller than a specified radius. Finally, we complete the algorithm by shifting the window to the mean of this kept points, and repeat the whole process until the distance is smaller than a specified threshold.

In Figure 2, we can see the results of the algorithm, being image 2a the obtained clusters in different colors, and image 2b the reconstruction in L*a*b* space.



(a) Resulting Clusters. R=15.

(b) L*a*b* reconstructed image. R=15.

Figure 2: Results of applying Mean-Shift segmentation.

As it can be seen, the white and black parts of the cow form a cluster each, together with the shadow, and the last cluster is formed by the grass in the background. These results are not what we would have done manually, because we as humans know that a cow has that kind of pattern, but the algorithm hasn't this kind of knowledge. In addition, Mean-Shift is extremely slow, as it has to iterate among every pixel, and for each one iteratively compute its mean and shifting it until convergence.

In the next section, we will present the second implemented algorithm, and discuss its performance.

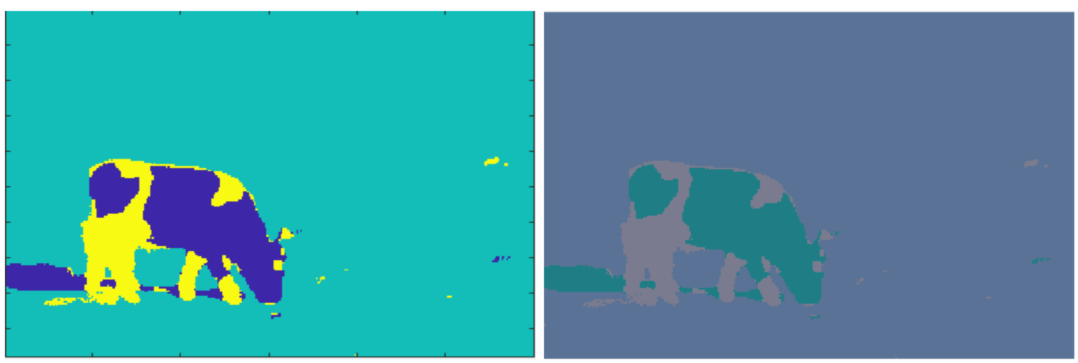
3 EM Segmentation

In this third section, the Expectation Maximization (EM) algorithm implementation and results are shown. This approach addresses the segmentation task in a probabilistic way. First, we have to assume that the number of different clusters, K, is known. Then, we model each of those clusters as a univariate Gaussian, with mean μ and covariance Σ, and together they form a mixture of Gaussians weighted by a mixing weight α. Then, this algorithms tries to find the set of Gaussians that fit in the density function X, which was defined previously for the Mean-Shift algorithm.

EM addresses this fitting problem by alternating between Expectation and Maximization steps. In the expectation step, we compute the probability of each pixel to be in segment k given the current mixture of Gaussians. Then, in the Maximization step, we maximize the expectation of having a the X distribution given the current estimates, and obtain the parameters for the mixture of gaussians that maximize this likelihood. This is repeated until convergence is achieved.

As it can be easily guessed, the final results are highly dependent of the knowledge about the number of clusters K beforehand. Thus, in the next figures we will see the results of this algorithm with different number of clusters. Also, as requested, we will include the resulting values of μ, Σ, and α for each K

K = 3 clusters



(a) Resulting Clusters. K=3. (b) L*a*b* reconstructed image. K=3.

Figure 3: Expectation Maximization segmentation with K=3.

With values for the Gaussian mixtures as follows:

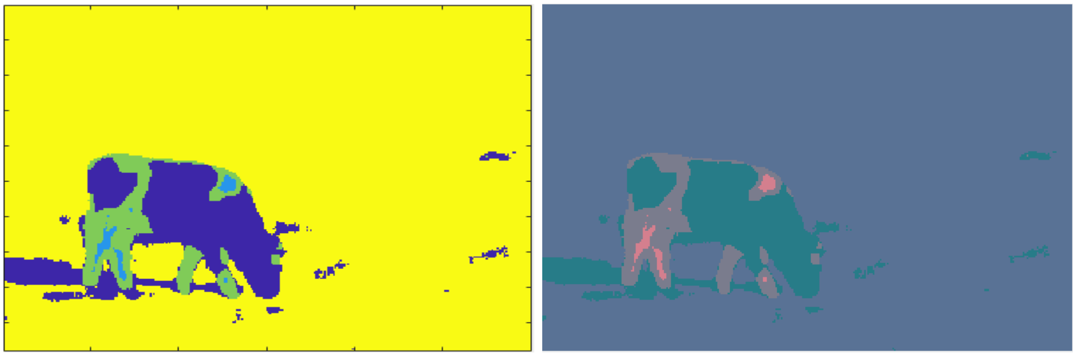
$$\Sigma_1 = \begin{bmatrix} 817.7307 & -137.2490 & 238.9018 \\ -137.2490 & 33.0421 & -47.4319 \\ 238.9018 & -47.4319 & 79.2173 \end{bmatrix} \mu_1 = \begin{bmatrix} 42.1939 & 123.5471 & 137.0383 \end{bmatrix}$$

$$\Sigma_2 = \begin{bmatrix} 56.6925 & 0.3524 & 0.4153 \\ 0.3524 & 0.8359 & -0.1713 \\ 0.4153 & -0.1713 & 1.5365 \end{bmatrix} \mu_2 = \begin{bmatrix} 89.1010 & 114.4193 & 149.0963 \end{bmatrix}$$

$$\Sigma_3 = \begin{bmatrix} 2358.5 & 72.3 & 40.0 \\ 72.3 & 11.6 & -7.0 \\ 40.0 & -7.0 & 24.3 \end{bmatrix} \mu_3 = \begin{bmatrix} 137.3819 & 125.1853 & 140.5781 \end{bmatrix}$$

$$\alpha = \begin{bmatrix} 0.1059 & 0.8562 & 0.0379 \end{bmatrix}$$

K = 4 clusters



(a) Resulting Clusters. K=4. (b) L*a*b* reconstructed image. K=4.

Figure 4: Expectation Maximization segmentation with K=4.

With values for the Gaussian mixtures as follows:

$$\Sigma_1 = \begin{bmatrix} 12.1649 & 2.6217 & -1.0733 \\ 2.6217 & 2.2133 & -0.7861 \\ -1.0733 & -0.7861 & 2.6823 \end{bmatrix} \mu_1 = \begin{bmatrix} 14.7460 & 128.6379 & 128.4870 \end{bmatrix}$$

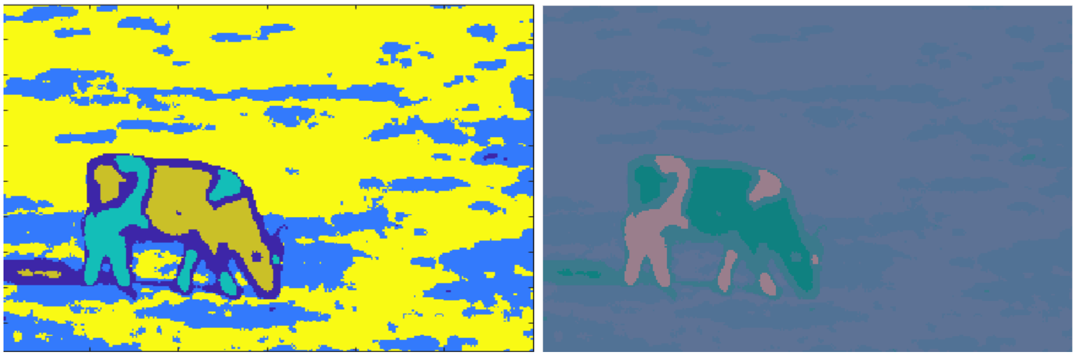
$$\Sigma_2 = \begin{bmatrix} 2128.0 & 42.3 & 63.1 \\ 42.3 & 7.6 & 4.0 \\ 63.1 & 4.0 & 23.1 \end{bmatrix} \mu_2 = \begin{bmatrix} 148.5388 & 125.9875 & 140.1645 \end{bmatrix}$$

$$\Sigma_3 = \begin{bmatrix} 55.6166 & 0.2550 & 0.3902 \\ 0.2550 & 0.8070 & -0.1607 \\ 0.3902 & -0.1607 & 1.5197 \end{bmatrix} \mu_3 = \begin{bmatrix} 89.1625 & 114.4249 & 149.0970 \end{bmatrix}$$

$$\Sigma_4 = \begin{bmatrix} 578.7113 & -77.0557 & 139.6450 \\ -77.0557 & 26.6815 & -31.5136 \\ 139.6450 & -31.5136 & 49.7302 \end{bmatrix} \mu_4 = \begin{bmatrix} 63.3717 & 119.9356 & 142.9102 \end{bmatrix}$$

$$\alpha = \begin{bmatrix} 0.0407 & 0.0311 & 0.8500 & 0.0782 \end{bmatrix}$$

K = 5 clusters



(a) Resulting Clusters. K=5. (b) L*a*b* reconstructed image. K=5.

Figure 5: Expectation Maximization segmentation with K=5.

With values for the Gaussian mixtures as follows:

$$\Sigma_1 = \begin{bmatrix} 10.7837 & 2.3780 & -1.5428 \\ 2.3780 & 1.9857 & -0.7587 \\ -1.5428 & -0.7587 & 2.4905 \end{bmatrix} \mu_1 = \begin{bmatrix} 14.5146 & 128.6191 & 128.3877 \end{bmatrix}$$

$$\Sigma_2 = \begin{bmatrix} 53.9675 & 0.6000 & 5.3666 \\ 0.6000 & 2.1210 & 0.0716 \\ 5.3666 & 0.0716 & 2.0712 \end{bmatrix} \mu_2 = \begin{bmatrix} 80.3024 & 114.3898 & 149.5999 \end{bmatrix}$$

$$\Sigma_3 = \begin{bmatrix} 705.7917 & -73.1943 & 131.0146 \\ -73.1943 & 22.6300 & -25.2600 \\ 131.0146 & -25.2600 & 38.1279 \end{bmatrix} \mu_3 = \begin{bmatrix} 56.5676 & 122.3648 & 139.3233 \end{bmatrix}$$

$$\Sigma_4 = \begin{bmatrix} 1973.2 & 40.4 & 50.4 \\ 40.4 & 6.9 & -4.0 \\ 50.4 & -4.0 & 24.1 \end{bmatrix} \mu_4 = \begin{bmatrix} 152.3501 & 126.0971 & 140.3552 \end{bmatrix}$$

$$\Sigma_5 = \begin{bmatrix} 28.0724 & -0.2883 & 0.9094 \\ -0.2883 & 0.5482 & -0.1793 \\ 0.9094 & -0.1793 & 1.3424 \end{bmatrix} \mu_5 = \begin{bmatrix} 91.8799 & 114.4685 & 148.9502 \end{bmatrix}$$

$$\alpha = \begin{bmatrix} 0.0388 & 0.2295 & 0.0575 & 0.0293 & 0.6449 \end{bmatrix}$$

As it can be seen from figures 3, 4, and 5, the resulting segmentation changes depending on the value of K as expected, getting a good result with $K = 3$, and creating extra clusters for higher values. Thus, if we are going to use this method, it is important to try to figure out beforehand the number of clusters, otherwise it will be very likely to perform badly.

This algorithm is much faster than Mean-Shift, but has the counterpart of the need of choosing K, so in the end the use of one of other algorithm would depend on the application. However, nowadays there are much better-performing learning-based algorithms.

With this, the assignment comes to an end, and some final conclusions will be discussed in the last section of this report.

4 Conclusion

In this fifth assignment of the Computer Vision course, we've successfully implemented two basic image segmentation algorithms. First, a preprocessing step has been performed on the image, in order to get a smoother image in a color space which is more suitable for the purposes of image segmentation like L*a*b* space.

Then, the Mean-Shift algorithm for segmentation has been implemented. This algorithm iteratively finds the mean of a set of pixels in a determined spherical window, and shifts that window towards the mean until convergence. We've found that, although the performance can be good enough, it is a very slow algorithm, so a reduction in the image size would be needed for faster performance, thus losing accuracy in the segmentation as we would have less resolution in the image.

Finally, we implement the Expected Maximization algorithm for segmentation. This algorithm addresses the segmentation task probabilistically by modelling each different cluster as a Gaussian, and computing its mean and variance by iterating between expectation and maximization steps. Although it achieves good performance, we need to define beforehand the number of clusters, which might be not known, so we highly depend on a good value of this parameter to have accurate results.

All in all, it has been a complete assignment where we have been able to try two different image segmentation algorithms, comparing their results and analyzing their pros and cons. That way we have been able to get a perspective of the challenges that exist in this field, so it has been a great practical experience that has helped to build a great intuition about the methods discussed.