Computer Vision Assignment 5

Luca Ebner (ebnerl@student.ethz.ch)

December 12, 2020

The goal of this exercise is to program an image segmentation algorithm based on the image pixel colors in L*a*b* color space. We will do a segmentation based on the *Mean-Shift* algorithm and later with the so-called *Expectation-Maximization* algorithm.

1. Image Preprocessing

Before we perform the image segmentation itself, we have to process the input image by smoothening and color space conversion.

To smoothen the image and eliminate noise, we apply a 5×5 Gaussian filter with $\sigma = 5$. For this purpose, the matlab functions `fspecial` and `imfilter` can be used.
Afterwards we convert the image's color space to L*a*b* color by using `makecform` and `applycform`.
In Figure 1 we can see the different stages of the image preprocessing.

Why is it better to do segmentation in the L*a*b* color space as compared to RGB color space?
The L, a, b values stand for lightness, green-red and blue-yellow color axis. As stated in the beginning, the goal of this exercise is to segment the image based on the pixel's color. Two pixels can in fact display the same real-world color but with a different lightness. In L*a*b*, they would then have the same a, b value, but different lightness L. Compared to RGB color space, there all three values R, G, B would differ between the two pixels. As a result, two pixels with same color but different lightness are less distant in L*a*b* color space than they are in RGB. Since we are interested in the color space distances between pixels, we use L*a*b* color space for this exercise.

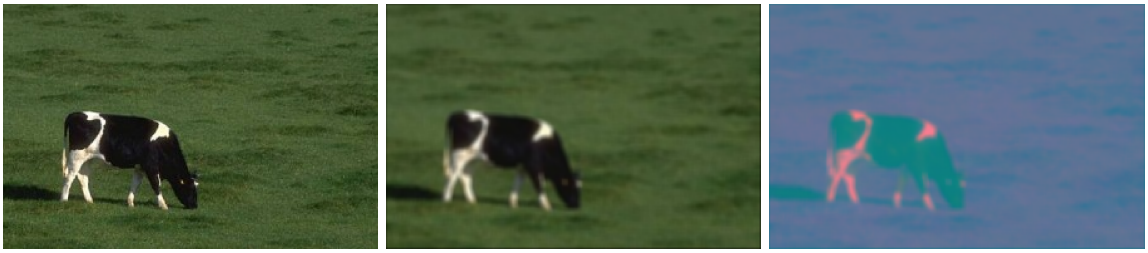

Figure 1: Stages of image preprocessing. Raw image (left), smoothed image (middle), L*a*b* color space image (right).

2. Mean-Shift Segmentation

The input image can be seen as a color density function, where the L*a*b* values of each pixel form the discrete set of samples from the density function. The goal now is to assign a mode from this density function to each pixel, which represents the color segment the pixel belongs to.

For this purpose, the Mean-Shift algorithm repeatedly computes the mean L, a, b values of all the pixels that lie within a spherical L*a*b* color window of radius r around the pixel we consider. The window is shifted to

the mean until convergence. The L, a, b values found after convergence are the mode that the pixel belongs to. Note that in the case of Mean-Shift, the number of modes is not fixed prior to execution.

In the exercise, we have to implement the functions `find_peak(X,x,r)` and `mean_shift(X,r)`.

- `find_peak` takes the data set pixels X, the pixel to consider x and the radius for the spherical window r as inputs and returns the L, a, b values it has found for the L*a*b* peak that the pixel belongs to.
- `mean_shift` iterates over all pixels, calls `find_peak` and returns the list of found peaks as well as a map with the same size as the image, containing the corresponding peak index for each pixel.

In Figure 2 we can see a visualization of the resulting segmentation map from the Mean-Shift algorithm.

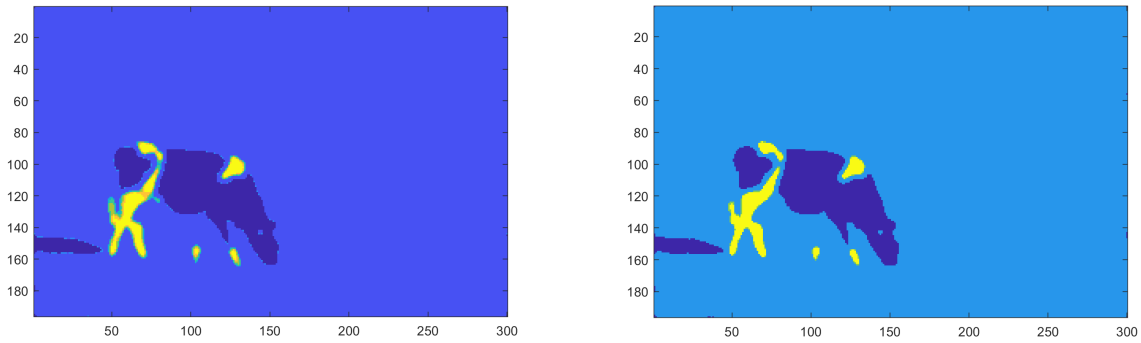

Figure 2: Segmentation map resulting from the Mean-Shift algorithm with parameter $r_1 = 25$ (left), $r_2 = 35$ (right). Number of peaks found: $peaks(r_1) = 8$, $peaks(r_2) = 4$.

Note that the number of peaks found by the Mean-Shift algorithm greatly depends on the choice of the parameter r. This is due to the fact that two peaks get merged if their L*a*b* distance is smaller than $\frac{r}{2}$. Also note that some parts of the cow and especially its legs were counted to the same segment as the background.

3. Expectation Maximization (EM) Segmentation

Instead of the deterministic Mean-Shift algorithm, we can also go with the EM algorithm which uses a probability approach. Here, we determine the number of segments K in the image prior to execution. Each segment k is modelled as a Gaussian, which in combination lead to a Gaussian mixture model.
The overall goal is to compute the probabilities for each pixel to lie in segment $k \in K$.
In the end, we label each pixel with the segment where the probability is the highest.

To calculate these probabilities, we complete the functions `expectation(theta,X)` and `maximization(P,X)`, where $\mathbf{\Theta} = (\alpha_\mathbf{1}, ..., \alpha_\mathbf{k}, \theta_\mathbf{1}, ..., \theta_\mathbf{k})$ are the parameters for the Gaussian mixture model, $\mathbf{X}$ are the data set pixels and $\mathbf{P}$ is the probability matrix that stores the probability of each pixel to lie in each segment.

- `expectation` takes the current Gaussian mixture model parameters $\mathbf{\Theta}$ and the data set pixels $\mathbf{X}$ as inputs and computes as output the probability matrix $\mathbf{P}$ that stores the probability of each pixel to lie in each segment with the given formulas (1) and (2) from the exercise sheet.
- `maximization` takes the found probability matrix $\mathbf{P}$ and the data set pixels $\mathbf{X}$ as input and updates the Gaussian parameters Θ given the current probability matrix $\mathbf{P}$ by maximizing the log likelihood.

We then iterate over `expectation` and `maximization` until convergence. Convergence is found if the μ parameters between two iterations differ less than a defined threshold. The results for $\mathbf{\Theta} = (\alpha_\mathbf{1}, ..., \alpha_\mathbf{k}, \theta_\mathbf{1}, ..., \theta_\mathbf{k})$, $\theta_i = (\Sigma_i, \mu_i)$ of the EM Segmentation algorithm with $K = 3, 4, 5$ can be seen below:

K=3

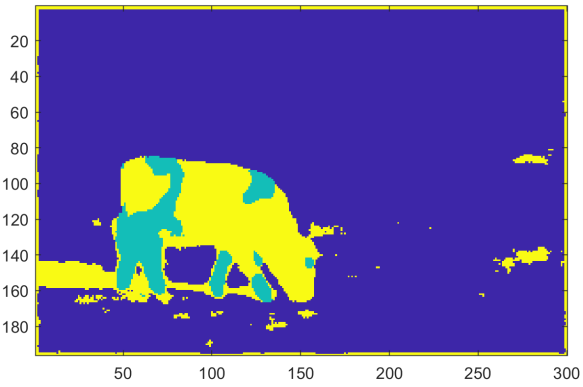

Figure 3: EM Segmentation map with 3 segments.

$$\Sigma_1 = \begin{bmatrix} 55.5780 & 0.0548 & 0.5629 \\ 0.0548 & 0.8254 & -0.1643 \\ 0.5629 & -0.1643 & 1.5511 \end{bmatrix}$$

$$\Sigma_2 = 10^3 * \begin{bmatrix} 2.2905 & 0.0711 & 0.0237 \\ 0.0711 & 0.0114 & -0.0085 \\ 0.0237 & -0.0085 & 0.0270 \end{bmatrix}$$

$$\Sigma_3 = \begin{bmatrix} 718.1680 & -131.9926 & 207.1896 \\ -131.9926 & 34.1849 & -45.4922 \\ 207.1896 & -45.4922 & 69.1235 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 89.2343 & 114.4452 & 149.0985 \end{bmatrix}$$
$$\mu_2 = \begin{bmatrix} 138.5224 & 125.2065 & 140.8752 \end{bmatrix}$$
$$\mu_3 = \begin{bmatrix} 47.0586 & 121.9063 & 138.7283 \end{bmatrix}$$
$$\alpha_1 = 0.8205$$
$$\alpha_2 = 0.0374$$
$$\alpha_3 = 0.1421$$

If we look at Figure 3 and compare it to the Mean-Shift algorithm, we can see that the EM Segmentation delivers much more realistic results. For example the cow's legs are not missing anymore. Apart from that, the EM Segmentation seems to be much faster than Mean-Shift. On the machine used to do this exercise, the Mean-Shift algorithm took 2-3 minutes to finish, while the EM algorithm took only around 15 seconds. (Of course the computation times can still be improved by tuning parameters such as the radius r for Mean-Shift or threshold values for convergence in both algorithms.) Below listed are the results for setting the number of segments to $K = 4$ and $K = 5$.

K=4

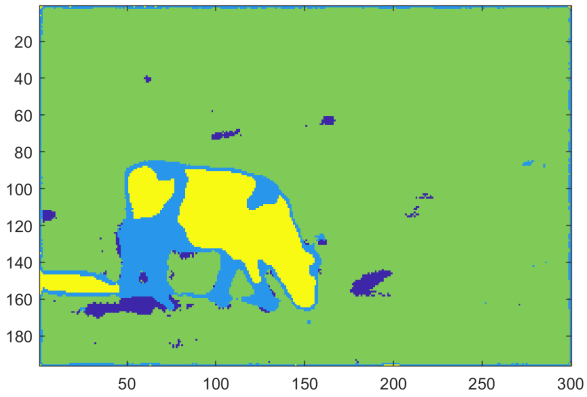

Figure 4: EM Segmentation map with 4 segments.

$$\Sigma_1 = \begin{bmatrix} 28.2019 & -2.9680 & -2.8449 \\ -2.9680 & 1.6486 & 0.5563 \\ -2.8449 & 0.5563 & 2.5989 \end{bmatrix}$$

$$\Sigma_2 = 10^3 * \begin{bmatrix} 2.6838 & 0.1616 & -0.0252 \\ 0.1616 & 0.0186 & -0.0098 \\ -0.0252 & -0.0098 & 0.0190 \end{bmatrix}$$

$$\Sigma_3 = \begin{bmatrix} 65.3485 & -0.2353 & 1.5375 \\ -0.2353 & 0.9148 & -0.3220 \\ 1.5375 & -0.3220 & 1.6426 \end{bmatrix}$$

$$\Sigma_4 = \begin{bmatrix} 228.2155 & -33.7642 & 60.8782 \\ -33.7642 & 13.9053 & -15.4351 \\ 60.8782 & -15.4351 & 24.0012 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 90.8762 & 115.1805 & 149.8799 \end{bmatrix}$$
$$\mu_2 = \begin{bmatrix} 101.3712 & 121.9065 & 142.1329 \end{bmatrix}$$
$$\mu_3 = \begin{bmatrix} 88.5810 & 114.4286 & 149.0188 \end{bmatrix}$$
$$\mu_4 = \begin{bmatrix} 24.8622 & 126.7519 & 131.7056 \end{bmatrix}$$

$$\alpha_1 = 0.0572$$
$$\alpha_2 = 0.0738$$
$$\alpha_3 = 0.7973$$
$$\alpha_4 = 0.0718$$

K=5

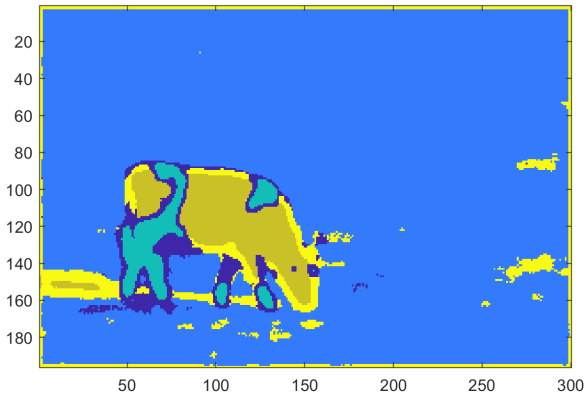

Figure 5: EM Segmentation map with 5 segments.

$$\Sigma_1 = \begin{bmatrix} 357.1671 & -26.3617 & 50.2227 \\ -26.3617 & 19.4392 & -24.1867 \\ 50.2227 & -24.1867 & 42.9975 \end{bmatrix}$$
$$\Sigma_2 = \begin{bmatrix} 53.5878 & -0.0291 & 0.5094 \\ -0.0291 & 0.8125 & -0.1582 \\ 0.5094 & -0.1582 & 1.5444 \end{bmatrix}$$
$$\Sigma_3 = 10^3 * \begin{bmatrix} 1.3881 & 0.0205 & 0.0214 \\ 0.0205 & 0.0051 & -0.0031 \\ 0.0214 & -0.0031 & 0.0209 \end{bmatrix}$$
$$\Sigma_4 = \begin{bmatrix} 35.8346 & 3.8502 & 4.0172 \\ 3.8502 & 2.7327 & -0.9013 \\ 4.0172 & -0.9013 & 4.4311 \end{bmatrix}$$
$$\Sigma_5 = \begin{bmatrix} 295.7289 & -46.1677 & 71.4814 \\ -46.1677 & 12.4705 & -15.4870 \\ 71.4814 & -15.4870 & 23.1961 \end{bmatrix}$$

$$\mu_1 = \begin{bmatrix} 82.2508 & 121.2239 & 143.3394 \end{bmatrix}$$
$$\mu_2 = \begin{bmatrix} 89.3563 & 114.4504 & 149.1052 \end{bmatrix}$$
$$\mu_3 = \begin{bmatrix} 166.7459 & 126.5773 & 140.7664 \end{bmatrix}$$
$$\mu_4 = \begin{bmatrix} 16.6074 & 128.6706 & 128.9521 \end{bmatrix}$$
$$\mu_5 = \begin{bmatrix} 59.7178 & 117.6128 & 143.7762 \end{bmatrix}$$
$$\alpha_1 = 0.0360$$
$$\alpha_2 = 0.8150$$
$$\alpha_3 = 0.0232$$
$$\alpha_4 = 0.0483$$
$$\alpha_5 = 0.0775$$

If we compare the results for $K = 3$ with those of $K = 4$ and $K = 5$, we can see that the results for $K = 3$ seem to be the most realistic. If we look at the original image this indeed makes sense, because the most present colors in our test image are white, black and green ($K = 3$), using more segments does only lead to splitting up these main colors into subgroups (light and dark green, for example).
As a conclusion I would say that the benefit of using Mean-Shift is that it is very easy to implement. On its downside, the computation time is very long and the algorithm needs fine tuning of the parameters, especially for radius r.
The EM algorithm on the other hand is more complex to implement, but very efficient. It delivers pretty good results but requires the user to specify the number of segments prior to execution.