Computer Vision
ETH Zürich
Fall 2020

Lab Assignment 5 - Image Segmentation

Mokhles Bouzaien

1 Image Preprocessing

In this part, I simply used the `imgaussfilt` function to apply a 5×5 Gaussian filter to the image ($\sigma = 5$) to remove noise. The result is shown in figure 1.

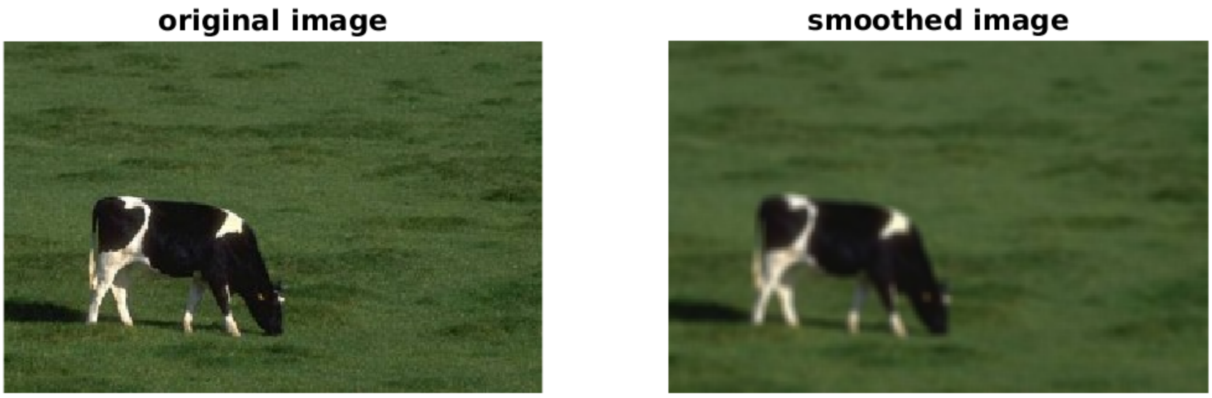


Figure 1: Original (left) and smoothed (right) images.

Then, I used `rgb2lab` to convert the image from RGB to L*a*b* color space. As we can see in figure 2, CIELAB color space resulted in better color enhancement across the channels (especially color channels, e.g., 2 and 3) which make it better for image segmentation compared to RGB color space.

2 Mean-Shift Segmentation

The first step of this section is to work on the `find_peak` function. Given the density distribution X of shape $L \times 3$ and the color values of a given pixel x_l, the distances between x_l and all other pixels are calculated and the peak is shifted to the mean of the nearest pixels, i.e., pixels which are closer than r to x_l.

The second step is to implement `meanshiftSeg` function to perform mean-shift algorithm. i.e., find the peak for each pixel and merge peaks closer than $r/2$ to each other. The `map` variable store the peak index of each pixel. The segmentation result is shown in figure 3.

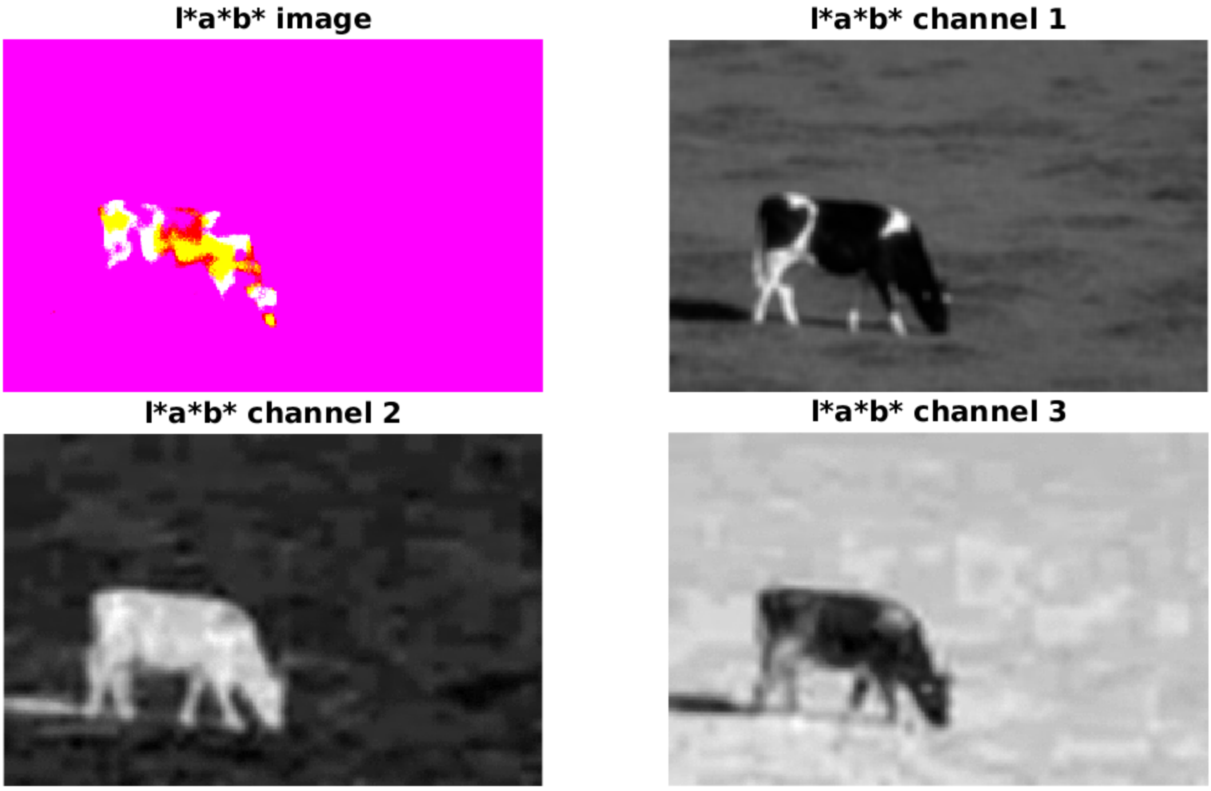
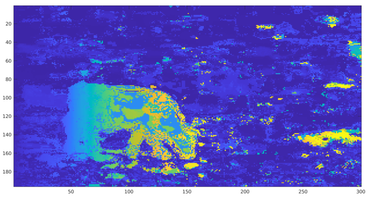
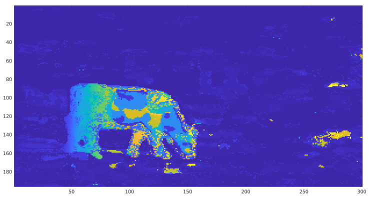


Figure 2: Different channels of the L*a*b* image.

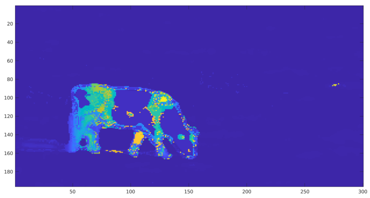
I tested different values of r ($r \in \{1, 2, 3\}$), and got different numbers of peaks. We notice that we get more peaks or segments (represented by the same color in figure 3) by decreasing the window radius r. This is due to the fact that only very close pixel values would be merged for lower values of r.



(a) $r = 1, k = 4027$



(b) $r = 2, k = 1470$



(c) $r = 3, k = 555$

Figure 3: Segmentation results for different values of r.

3 EM Segmentation

Initialization For this part, I started by implementing `generate_mu` and `generate_cov` to initialize μ and Σ. The first function creates K vectors of dimension 3 uniformly sampled from the interval $[min(X), max(X)]$. The second function creates K 3-by-3 diagonal matrices where the diagonal values are the L*a*b* component ranges. Finally, α is initialized as a uniform K-dimension vector.

Expectation For each pixel, and at each step, the Gaussian mixture model is calculated using the formula:

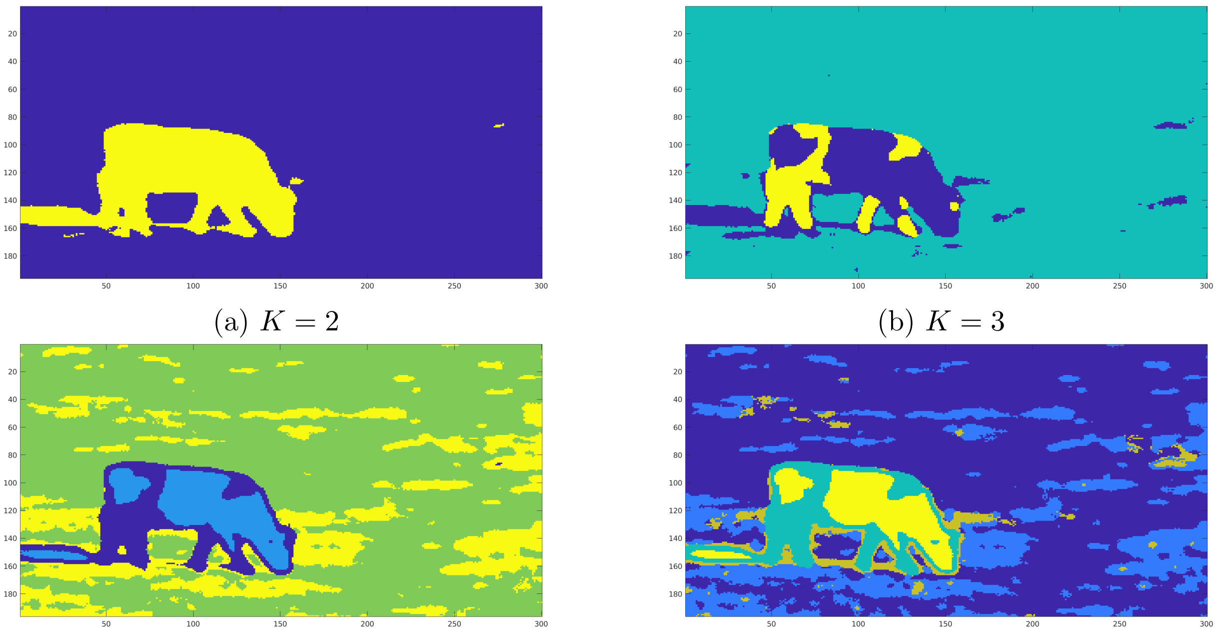
$$p(x_l|\Theta) = \sum_k \frac{\alpha_k}{(2\pi)^{n/2}|\Sigma_k|^{1/2}} \exp -\frac{1}{2}(x_l - \mu_k)^T \Sigma_k^{-1}(x_l - \mu_k)$$

where k is the segment index, x_l is the pixel values and (α, μ, Σ) are the model parameters at the current iteration. This was implemented using nested loops and stored in a $L \times K$ matrix P.

Maximization At this step, the model parameters are updated using the given formulas in order to maximize the expectations.

Expectation-Maximization Algorithm This algorithm is based on repeating the previous steps until convergence which is detected by comparing an error variable e and a threshold $t = 0.5$ where e is the maximum of $||\mu^{(s+1)} - \mu^{(s)}||$ over all K segments.

Results The segmentation results are shown in figure 4 where each pixel color represents a different segment. Depending on the objects we want to detect, we can very the value of K. For example, when $K = 2$, the cow and the background are separated; and when $K = 3$, the two texture of the cow are also separated. The model parameters of the cow image are saved to the file `values.txt` for $K \in \{3, 4, 5\}$. The segmentation results for

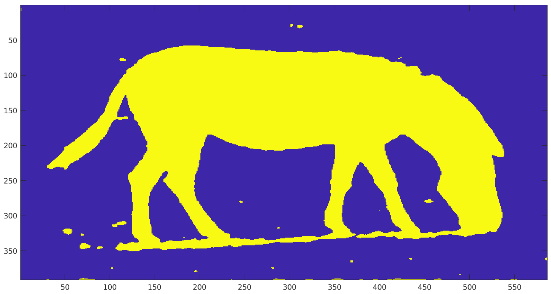


(a) $K = 2$ (b) $K = 3$

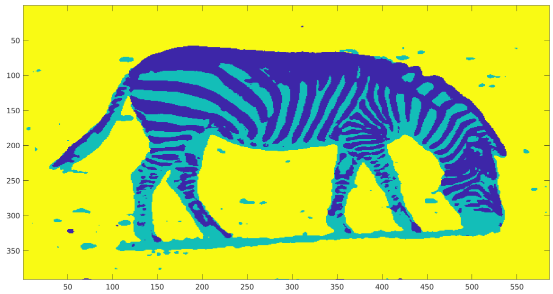
(c) $K = 4$ (d) $K = 5$

Figure 4: Segmentation results for different values of K.

the zebra images are shown in figure 5.



(a) $K = 2$



(b) $K = 3$

Figure 5: Segmentation results for different values of K.