Computer Vision
Exercise 6

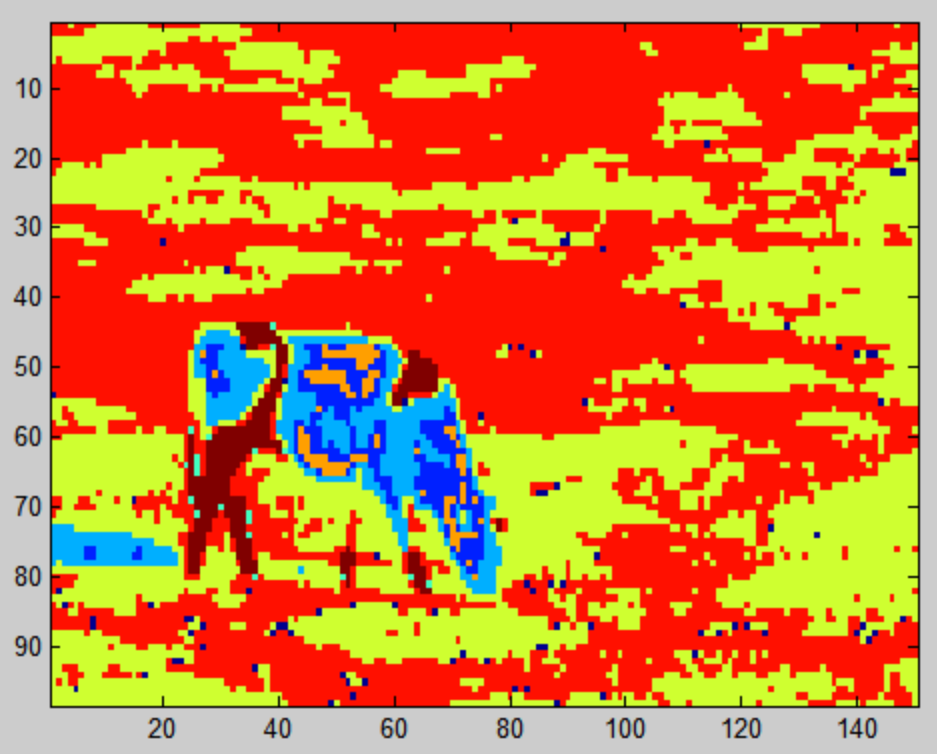
Amrollah Seifoddini

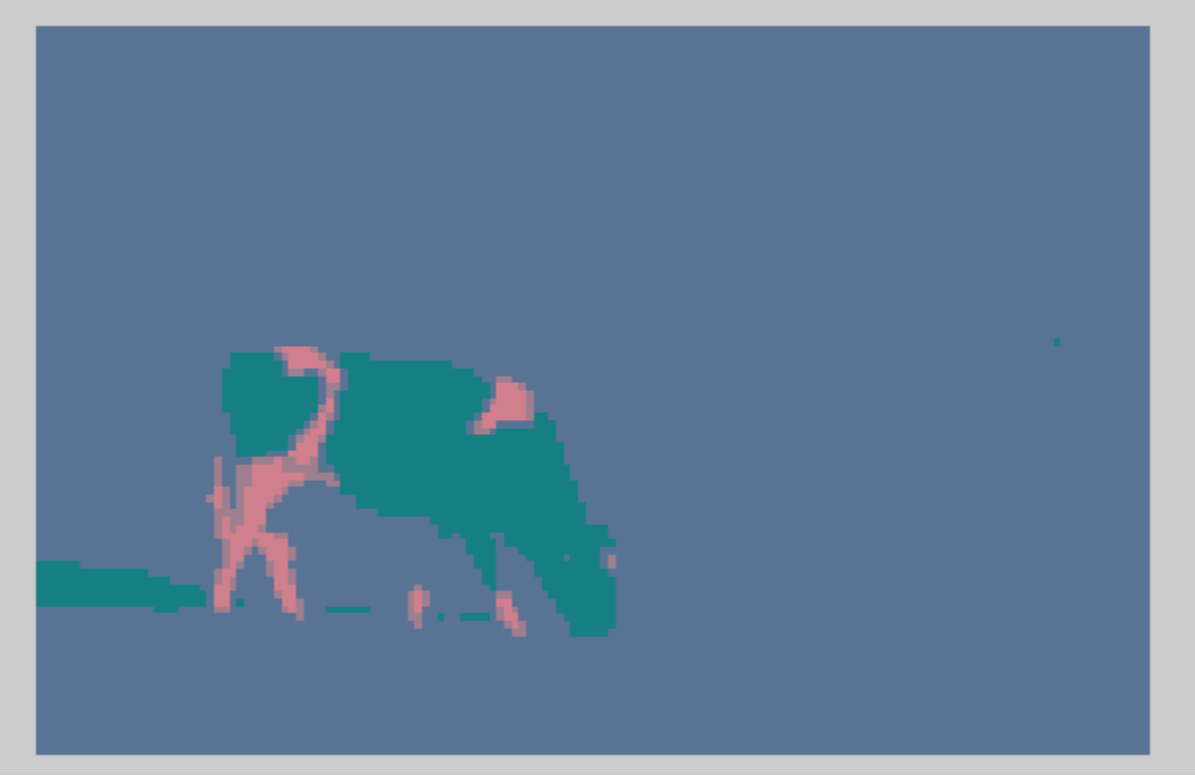
Part 1)

This part was done easily with the instruction provided by TA session slides. So, I resize the image, then apply a Gaussian filter on it for smoothness. Finally I convert the image from RGB to L*a*b color space. The reason for this conversion is that, in L*a*b space, intensity (Luminance) and color components get separated. And, it is better for segmentation because we want to segment based on color difference, not intensity of each other. Also, there are other color space which do the same separation, but we only used L*a*b. The following images show, smoothed and convert images in l*a*b space respectively.

Part 2)

In this part, I used general algorithm from the reference paper with small enhancement. Firstly, I vectorized the image. Then for each pixel, I call the find_peak for every pixel. In that function, I form a spherical window and calculate the mean of points inside it. At the first I was using normalize distance for finding points inside this window, but at the end, I decided to use the real values. So, I had to adjust the radius accordingly. My radius is around 35 for l*a*b images and works good. Then I move the window center to the shift and repeat this procedure until the difference between two means goes below a threshold (0.01*r). In each iteration, I add the visited points to an array and also add a vote for them in another array which represents how many times a point has been seen in a cluster. Outside of the find_peak function, I will search over all cluster to find ones that are close enough to the new cluster. If there was none, I will save the current cluster as a new one. The whole procedure repeats only for not_visited points until all points are associated to a cluster. Then I will find the maximum of cluster votes per pixel and the corresponding cluster will be save as the cluster_id for this pixel. The results are shown below.

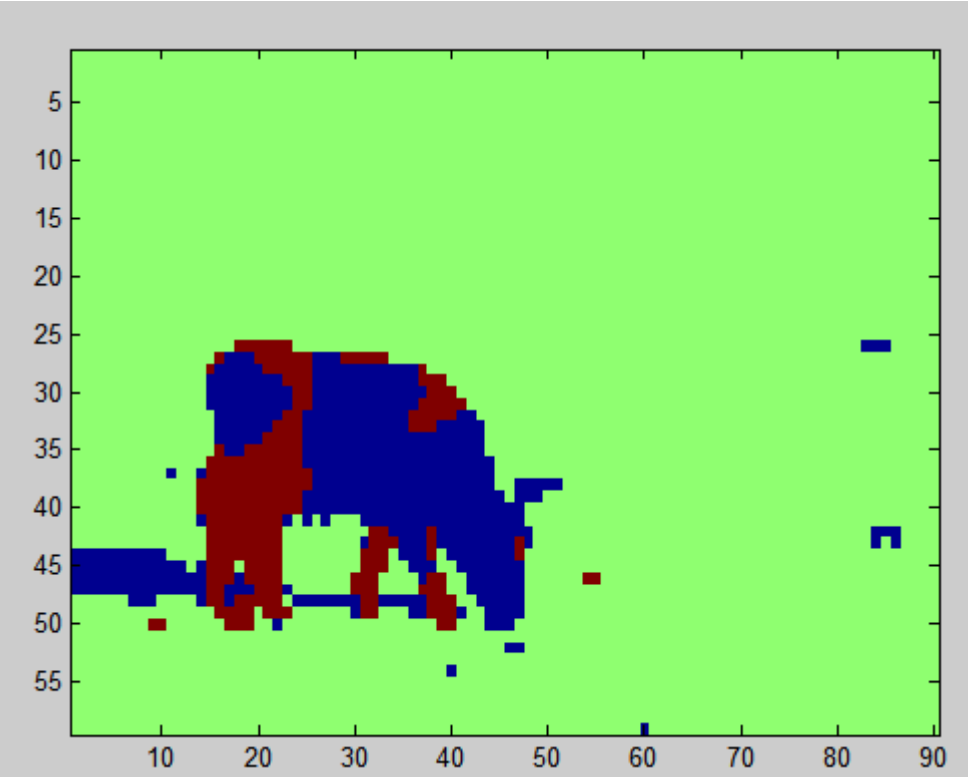

10
20
30
40
50
60
70
80
90
20
40
60
80
100
120
140



Part 3)

For this part, I implemented expectation and maximization functions exactly from the formula provided in exercise slides. Then I set the initial parameters. I set t “mu” with randomly selecting points in the image space. And for “cov” I used a diagonal matrix with range of image coordinates as the diag elements. “alpha” was set uniformly. Then I will iterate over expectation and maximization functions to calculate new values for parameters and probability. This procedure will repeat until an exit criteria satisfy. At first, I used a exit criteria based on “mu” and “sigma”, but at the end, I used difference between two consecutive probability for exit criteria. The value for these criteria are set experimentally. I resized images more in this section for faster computation. That is why pictures are low resolution.

The results for K=3 are shown below:



And for K=3 the parameters are:

Alpha:

0.0915845710632074

0.861014746042541

0.0474006828942536

Mu:

36.7812950570380	89.0119155938987	125.620182867148
124.485680328396	114.429671178186	123.603644399933
135.181545152796	149.099809361442	142.328430592739

Cov:

val(:,:,1) =

1.0e+003 *

0.6540 -0.1114 0.1836

0.0546 0.0002 0.0004

2.2015 0.1203 -0.0411

val(:,:,2) =

-111.4059 29.6290 -39.3418

0.2094 0.8613 -0.1609

120.3037 18.2834 -14.9167

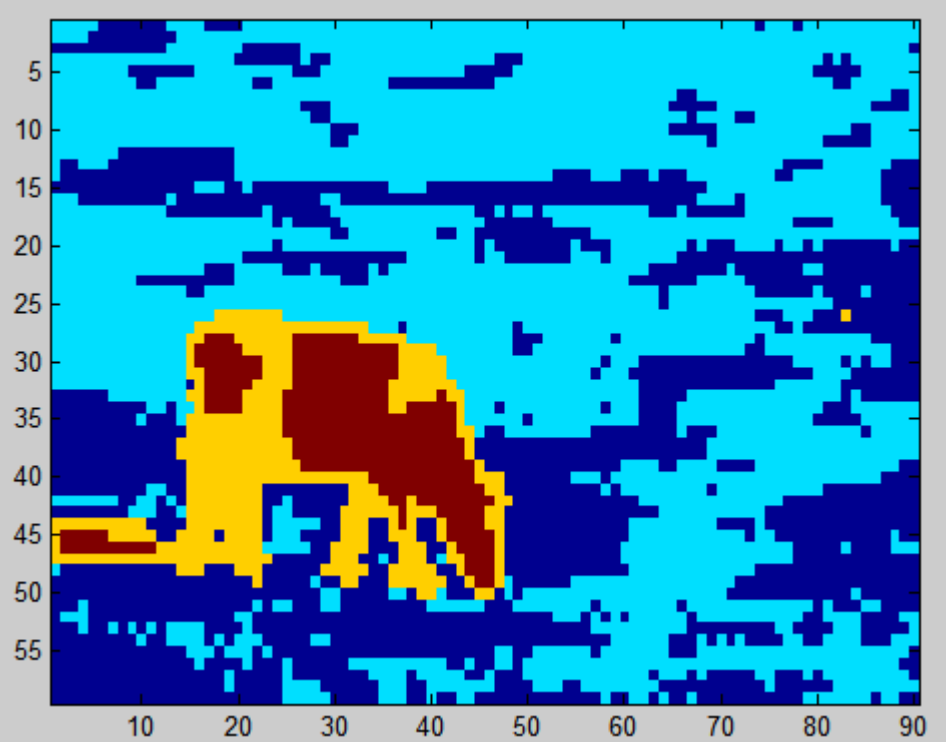
val(:,:,3) =

183.5594 -39.3418 61.0814

0.4147 -0.1609 1.5319

-41.1204 -14.9167 30.2117

For the K=4 the parameters obtained are as follows:



Alpha:

0.0464433046686935

0.431435143283251

0.0913945894648411

0.430726962583216

Mu:

126.197222144386	88.9594344040037	36.9989593223242	89.0144099437756
123.974611007714	114.166145808517	124.365654885840	114.703924465395
141.777221261084	148.460844913134	135.421039854586	149.727251384097

Cov:

val(:,:,1) =

1.0e+003 *

2.2419 0.1041 -0.0141

0.0533 0.0026 -0.0010

0.6751 -0.1173 0.1960

0.0564 -0.0023 0.0019

val(:,:,2) =

104.0698 16.8958 -13.1555

2.6062 0.7553 -0.2178

-117.2560 30.4186 -41.4486

-2.3289 0.8483 -0.4587

val(:,:,3) =

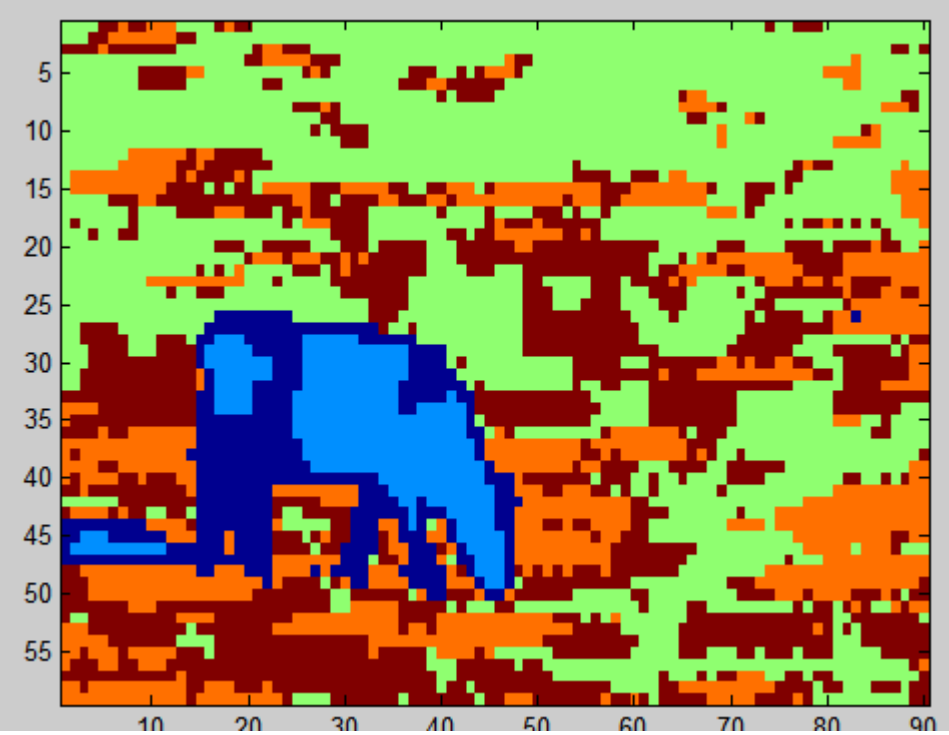
-14.0536 -13.1555 28.5661

-0.9555 -0.2178 0.5283

196.0137 -41.4486 65.8605

1.8663 -0.4587 1.7401

For the K=5 the pictures and parameters obtained are as follows:



Alpha:

0.0432016047416935

0.883820091425650

0.0480733703882410

3.24808421232737e-06

0.0249016853601994

Mu:

106.642234309815 88.5223495398705 16.5191986179923 94.0845091720256 93.3377634211930

124.991356763435 114.709062368965 127.957062153156 114.505812692275 114.522317209378

139.135645551172 148.856579425677 129.395116244038 148.963596069604 148.981126164245

Cov:

val(:,:,1) =

1.0e+003 *

4.6065 4.6065 4.6065

0.0651 0.0651 0.0651

0.0234 0.0234 0.0234

0.0045 0.0045 0.0045

0.0083 0.0083 0.0083

val(:,:,2) =

1.0e+004 *

0.4606	0.4943	0.5203
0.0065	0.0751	0.1645
0.0023	1.2442	1.2602
0.0004	0.0422	0.1125
0.0008	0.0457	0.1187

val(:,:,3) =

1.0e+004 *

0.4606	0.5203	0.5662
0.0065	0.1645	0.3705
0.0023	1.2602	1.2764
0.0004	0.1125	0.3016
0.0008	0.1187	0.3104