Answers to questions in Lab 3: Image segmentation

Name: _____PIETRO ALOVISI_____ Program: _____CDATE_____

Instructions: Complete the lab according to the instructions in the notes and respond to the questions stated below. Keep the answers short and focus on what is essential. Illustrate with figures only when explicitly requested.

Good luck!

Question 1: How did you initialize the clustering process and why do you believe this was a good method of doing it?

Answers:
I initialized the centers using a normal distribution having mean the average RGB values of the whole image. Then I chose an arbitrary variance, but not too small to give allow for some variations in the center. I think this is a sensible choice because the mean value should represent the center of the cloud of pixel intensities, therefore the centers after initialization will spread more easily around the points, from the center outwards.

Question 2: How many iterations L do you typically need to reach convergence, that is the point where no additional iterations will affect the end results?

Answers:
By looking at the numerical value of the distance of the centers we see that we get a reasonable result quite fast, in particular the results are displayed in the table below. The number of iterations represent the average number of iteration for which the centers components change less that 1 unit.

Number of cluster	iterations needed (orange)	iterations needed (tiger2)
5	7	9
8	7	9
10	8	12

If we would do a qualitative effect on the segmented image then the number of iterations to have a stationary result must be a little bit higher, around 12 or so.
As conclusion we can see that even if the let the number of clusters vary, the number of iteration needed is almost the same.
If we look at the data for complete convergence we get the following table:

Number of cluster	iterations needed (orange)	iterations needed (tiger2)
5	19	35

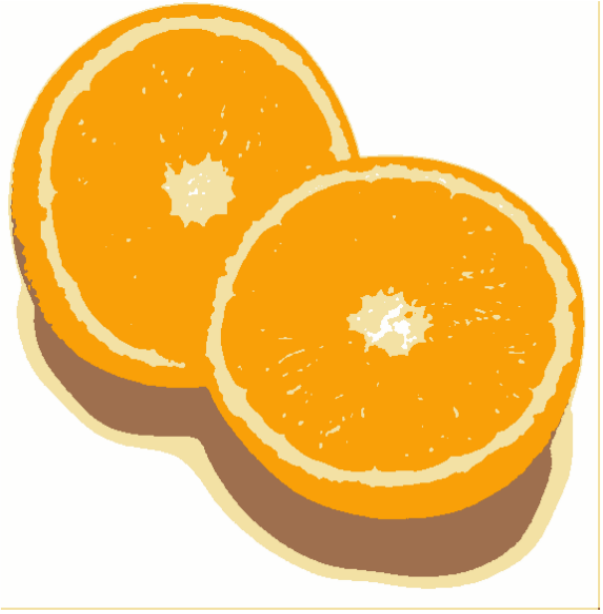
8	60	74
10	106	70

Where we can see that depending on the distribution of colors in the image, the number of iterations can grow quite fast or stabilize after some number of clusters.

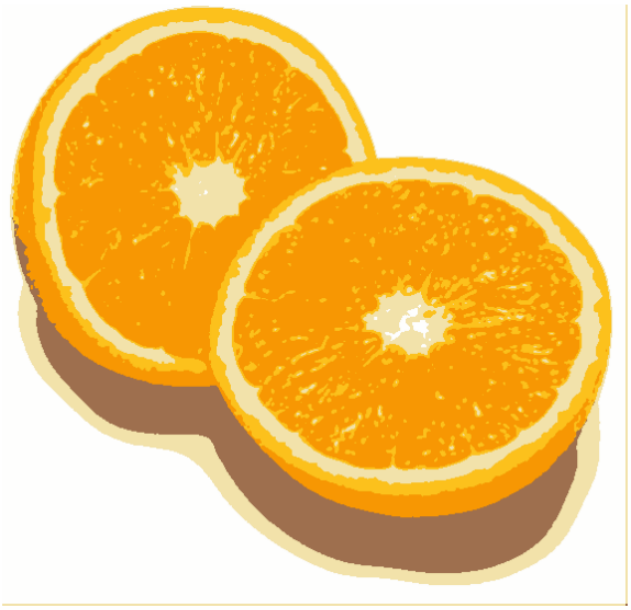
Question 3: What is the minimum value for K that you can use and still get no superpixel that covers parts from both halves of the orange? Illustrate with a figure.

Answers:
Using the initial parameters, the lowest value in my experiments is 5. Four is sometimes a good choice but still has the superpixel that blends the two halves in the bottom part.

K = 4

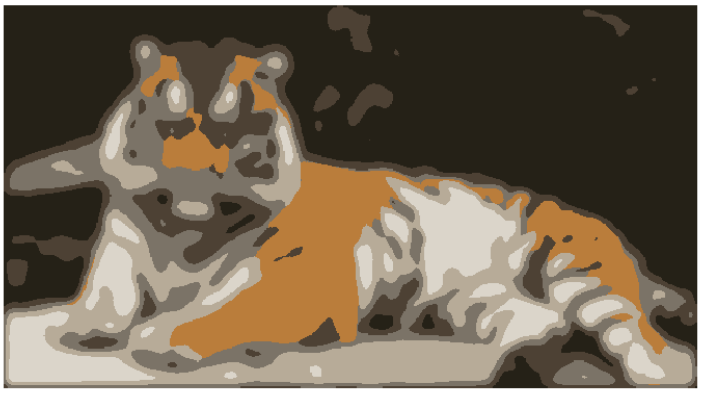
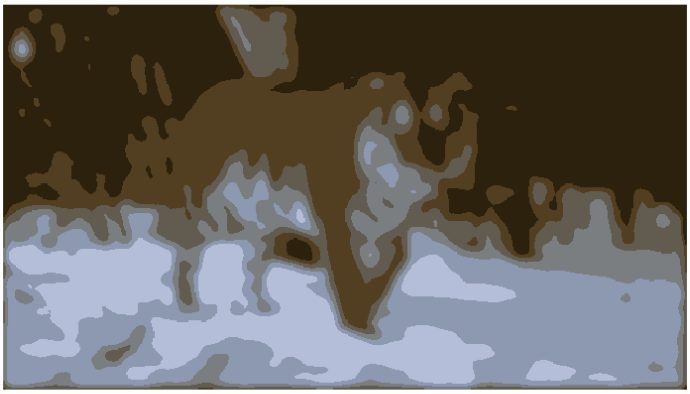


K = 5



Question 4: What needs to be changed in the parameters to get suitable superpixels for the tiger images as well?

Answers:
The tiger images are more rich in details and colors, which leads to bad results if we use the same parameters used in the orange image. Here are some results:

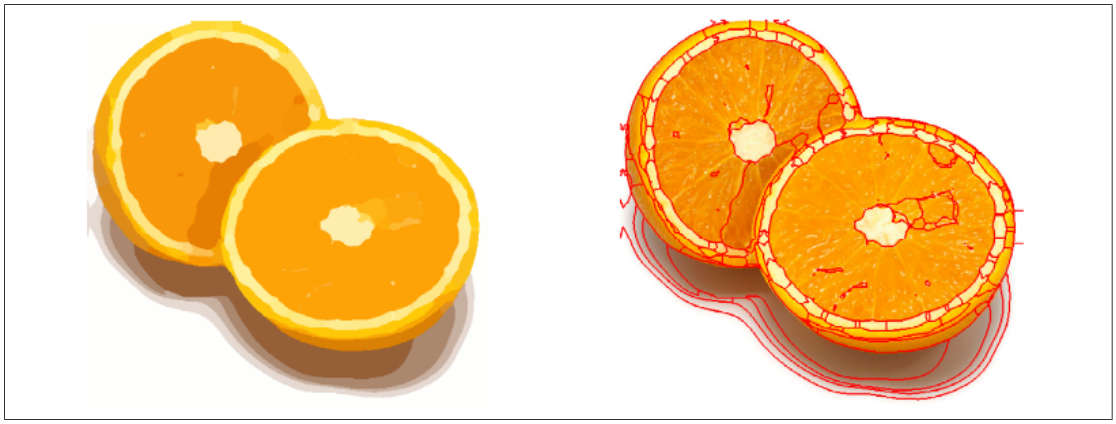



As we can see the clusters are too few to represent the whole image. So we need the change the number of cluster K for sure, but that might not be the only measure we must adopt. In fact there are many small details we want to remove, and that requires a balancing of the blurring and subsampling.

Question 5: How do the results change depending on the bandwidths? What settings did you prefer for the different images? Illustrate with an example image with the parameter that you think are suitable for that image.

Answers:
Changing the bandwidth changes the landscape of the density function, the bandwidth weights the distance between two points: a high bandwidth means that two points far away still give a meaning contribution to the density function, while a low bandwidth means that two points to give an almost non-zero contribution must be very close. Also the number of regions depends on the choice of these parameters.

spatial_bandwidth = 12.0; colour_bandwidth = 5.0; image_sigma = 5.0;
 
spatial_bandwidth = 7.0; colour_bandwidth = 7.0; image_sigma = 8.0;
 
spatial_bandwidth = 7.0; colour_bandwidth = 3.5; image_sigma = 3.0;

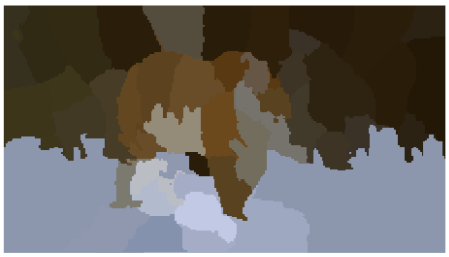


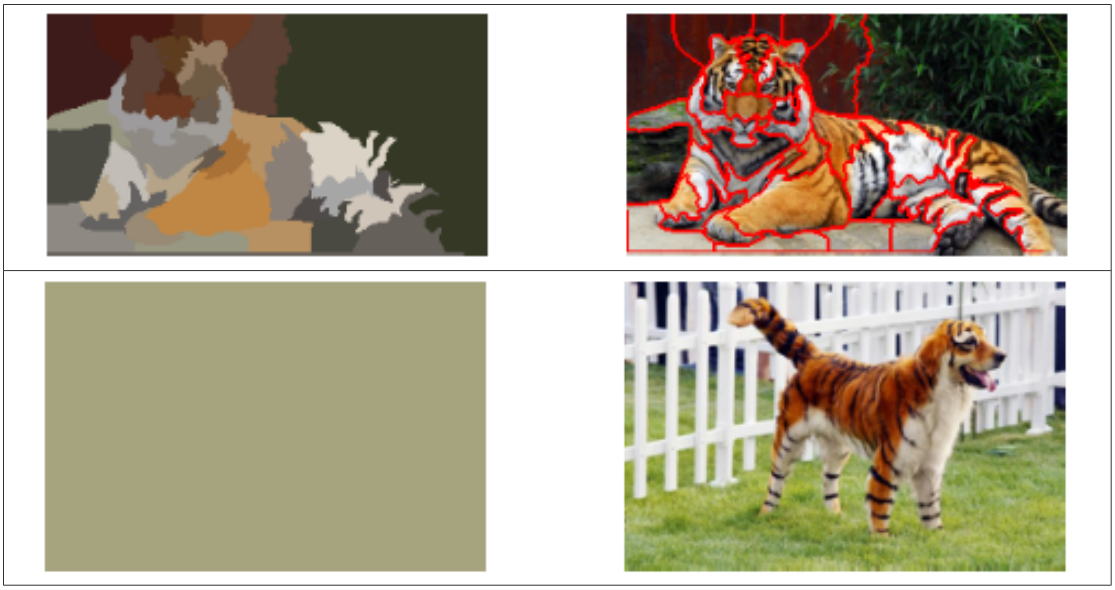
Question 6: What kind of similarities and differences do you see between K-means and mean-shift segmentation?

Answers:
K-means and mean-shift have in common the fact that they are based on the grouping of pixels based on their closeness in some space, and both do not use information about gradient in the image or known edges.
As differences we can see how the fact that we also take into consideration the spatial dimension of the image leads to some better segmentation, more robust to noisy pixel segmentation that happens sometimes in K-means clustering.

Question 7: Does the ideal parameter setting vary depending on the images? If you look at the images, can you see a reason why the ideal settings might differ? Illustrate with an example image with the parameters you prefer for that image.

Answers:
Here in the table below are listed the optimal parameters used for the first image, and then applied to the other two. The parameters are different because of the color intensities and range, in fact the parameters like the colour_bandwidth or the ncuts_thresh change and filter the similarity function that is dependent on the type and contrast of colors used in each image.

colour_bandwidth = 17.0; ncuts_thresh = 0.1; min_area = 250; max_depth = 8; scale_factor = 0.4; image_sigma = 1.0; radius = 4;	
	



Question 8: Which parameter(s) was most effective for reducing the subdivision and still result in a satisfactory segmentation?

Answers:
Probably the best way to reduce subdivision is to make the `ncuts_thresh` smaller or the `min_area` bigger: the final segmentation is still good, and we do not increase the execution time.
Another fast way to reduce subdivisions is by choosing a smaller `max_depth` which makes the segmentations bigger but if too low then the segmentation is bad, this is because the parameter does affect only the "carefulness" of the algorithm, the higher the more it investigates into the structure of the image.
Then the slower way to do it is to increase the `radius`.

Question 9: Why does Normalized Cut prefer cuts of approximately equal size? Does this happen in practice?

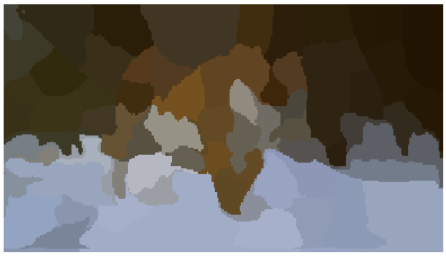
Answers:
Yes, in practice this happens more than often. This is due of two pulling and contrasting forces: the will to minimize the cut of the graph, and the maximization of the weights inside the partition. The first one wants a small region, while the second one wants a bigger region. The algorithm then finds a solution that trades-off these two quantities with an average size region. The size is determined by the parameters chosen for the algorithm.

Question 10: Did you manage to increase *radius* and how did it affect the results?

Answers:
Changing the radius creates bigger segmentation, and the end result is better. Though the computation slowed down quite significantly.

Radius = 2


Radius = 8

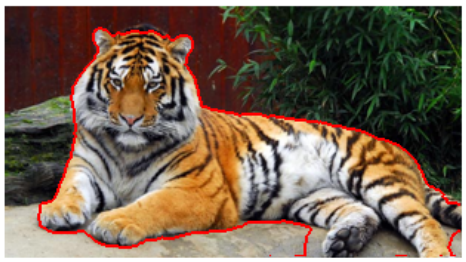



Radius = 20


Question 11: Does the ideal choice of *alpha* and *sigma* vary a lot between different images? Illustrate with an example image with the parameters you prefer.

Answers:

K = 15; alpha = 15.0; sigma = 10.0;

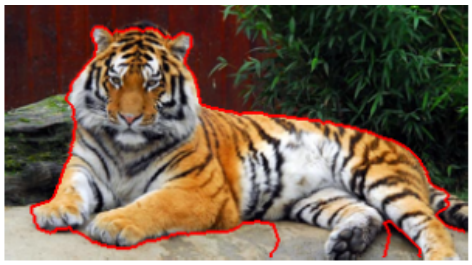




K = 15; alpha = 30.0; sigma = 1.5;




Question 12: How much can you lower K until the results get considerably worse?

Answers:
I can lower K as low a 2 and still get a reasonable result.

Question 13: Unlike the earlier method Graph Cut segmentation relies on some input from a user for defining a rectangle. Is the benefit you get of this worth the effort? Motivate!

Answers:

In our case is highly beneficial because the result is good. If we move the rectangle to focus mostly on the background then the result gets bad. Anyhow this procedure is good because let's me build a correct model for the foreground and the background. If we had not used the mask, we would have had to find automatically what is the foreground and which is the background, maybe leading to the recognition of others object, like the tree for example.

Question 14: What are the key differences and similarities between the segmentation methods (K-means, Mean-shift, Normalized Cut and energy-based segmentation with Graph Cuts) in this lab? Think carefully!!

Answers:
All the procedure are iterative and use information about the neighboring points, but in a different way: the graph methods include the neighboring pixel inside the model, they use them in a direct way, while K-means and Mean Shift use them by considering only the density in the space with some metric.
Another aspect is that K-means and Mean Shift focus on *grouping similar pixels*, while the other try to s*eparate different ones*.
Also there are some differences in how the informations about the color and the position are used, for example K-means uses only the RGB values for computing the similarities, Mean Shift uses both RGB and spatial coordinate, and the last two approaches use RGB to compute similarity and use the spatial not for computing some values but to find the neighboring pixels.
