Answers to questions in Lab 3: Image segmentation

Name: Andrea Camilloni Program: ID2222

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!

***Note:** functions for the first part up to question 10 are in the notebook lab3.ipynb, while the last part about graph cuts is in the script graphcut_example.py. All experiments are in the notebook.

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

Answers:
The clusters have been initialized randomly for each RGB value. The cluster have shape 1x3, this means we will have k arrays of shape 1x3.
The 3 respective dimensions represent RGB values in the range 0-255. The random initialization means we will have random values in that range (eg. [10,200,153]).
A better way would have been to initialize the clusters with the dominant colors.
For instance, considering the oranges image, suppose we have two clusters, we could have initialize them respectively with orange and white.

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 setting a threshold on the difference between new and old centers, calculating the difference between them, and stopping the algorithm for distance < threshold, we can observe the following:

- Number of iterations before convergence depends on the number of clusters
- Increasing the number of clusters determines an increasement of the number of iterations
- Number of iterations depends also on the complexity of the image; meaning that for instance the presence of more colours increase the number of iterations
- Gaussian blurring affects the number of iterations. Blurring the image reduce its complexity, meaning less iterations.

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:
First of all we can observe that with not enough number of iterations we do not get suitable superpixel.

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

Answers: The images are more complex than the orange image. To get suitable superpixels we have to increase the number of clusters to represent the more detailed image, and increase the number of iterations to converge.

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:

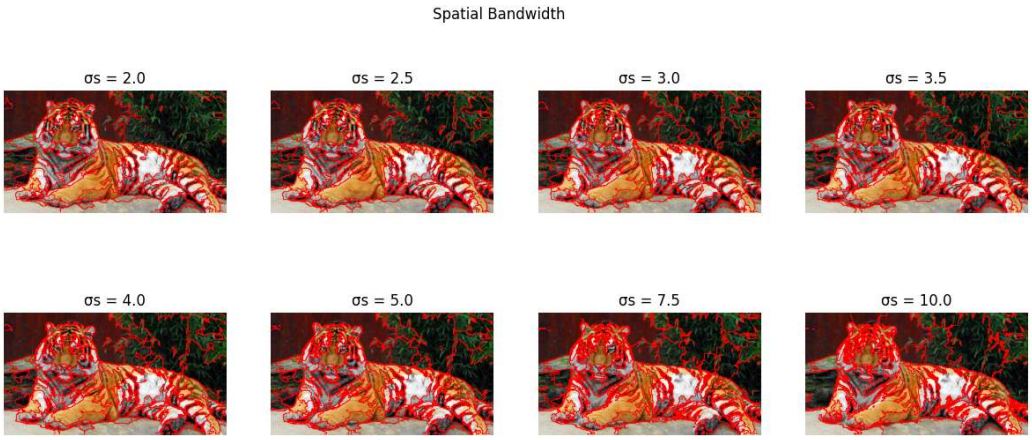

Figure 1: 'Tiger1' image for different values of spatial bandwidth and colour_bandwidth=20

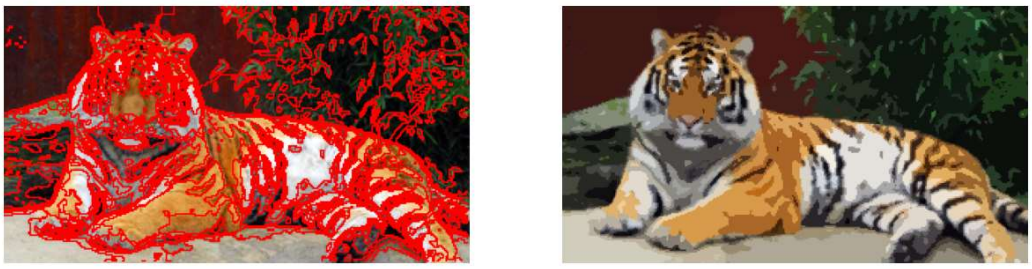

Figure 2: 'Tiger1' image for spatial_bandwidth=10 and colour_bandwidth=20

Figure 3: 'Tiger1' image for spatial_bandwidth=5 and colour_bandwidth=80

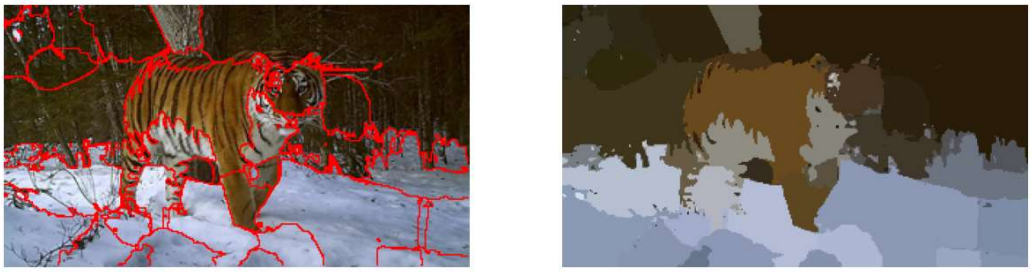

Figure 4: 'Tiger2' image for spatial_bandwidth=10 and colour_bandwidth=50

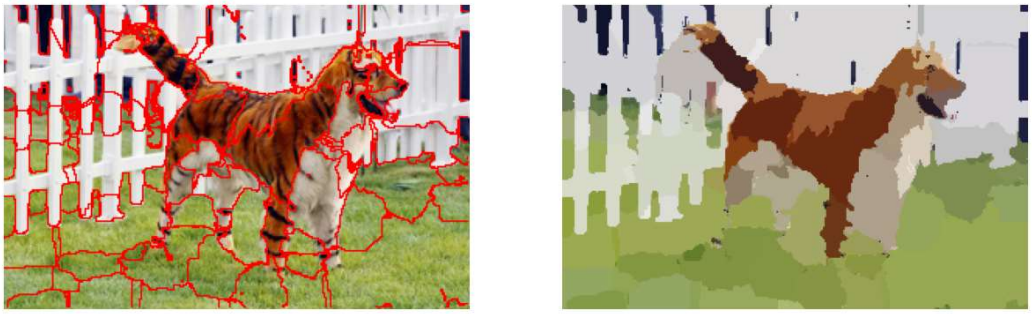

Figure 5: 'Tiger3' image for spatial_bandwidth=8 and colour_bandwidth=50

Lower values for the bandwidths imply more segments of small size, while larger values results in less segments of bigger size.
From the above figures we can observe how the results are affected by the bandwidths.

Figure 5: 'Tiger1' image for spatial_bandwidth=20 and colour_bandwidth=30

A good trade-off, if we don't want lot of segments, might be, as shown in figure 6, having spatial_bandwidth=20 and colour_bandwidth=30.

Having small values for the bandwidths might result in having lot of segments, which may not be the outcome we are looking for.

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

Answers:
Both algorithm update the centers according to the mean colours and they both samples pixels and colours as probability distribution.
K-means take into account only the colors and compute the superpixels (clusters) according to it, while mean shift is considering also the position. This means that segments in mean-shift have a specific location.
K-means requires a predetermined amount of clusters, while mean-shift will detect a variety of modes but requires a pre-determined bandwidth. K-means is more susceptible to outliers than mean-shift, which is less affected by them.

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.

Figure 7: 'Tiger1' image with original configuration

Answers:
Results with the original configuration are shown in Figure 7. The results depend on the image: quality/size, colours, details, etc..
We may want more or less segments, or take into account more colors or pixels. We may want bigger segments etc. To model these we study each parameter:

- color_bandwidth: determines the way in which weights are assigned to pixels
  - Low: large weights for similar pixels and low weights for unsimilar pixels
  - High: decreasing the weight for similar pixels and increasing weight for unsimilar pixels => affects computational cost
- The radius decides the size of the area around a pixel that will be taken into consideration when creating segments. A larger radius will include pixels that are further away, resulting in fewer segments, but it will also increase the time needed for the process.
- The ncuts_thresh parameter determines the maximum cutting cost which defines how similar regions can be separated. Increasing the ncuts_thresh value allows for more similar areas to be divided. (this affect the results with image of different complexities)

- The min_area parameter sets the minimum limit on how small the resulting segments can be.
- max_depth parameter specifies the number of recursive calls that will be made, resulting in more cuts and more segments (this affect the results with image of different complexities)

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

Answers:
The most effective parameters for reducing the subdivision while still achieving satisfactory results for segmentation were max_depth, ncut_thresh, and min_area.

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

Answers:

Normalized Cut aims to achieve a balanced cut, meaning that the size of the two parts should be approximately the same. This is because the Normalized Cut formula is based on a ratio of two values: the number of edges within the cut (intra-cluster edges) and the number of edges between the two parts (inter-cluster edges). The ratio is then normalized by the sum of the degrees of all vertices in the two parts. Since the intra-cluster edges should be maximized and the inter-cluster edges minimized, it is desirable to have two parts of approximately equal size.
This doesn't really happen in practice, as our aim is to segment the image into similar clusters which might be of different size (we consider mean value as splitting point).

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

Answers:

Figure 8: radius = 10

Figure 9: radius = 2

As expected, when the radius increases, the computation time increases significantly, since more neighbouring pixels are taken into account in the calculations. The results show larger segments (Fig. 8) with respect to a smaller radius (Fig. 9).

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.

Figure 10: K=5, alpha=15.0, sigma=10.0

Figure 11: K=20, alpha=10.0, sigma=25.0

Answers:

When we increase alpha, we raise the maximum cost of an edge, making it more difficult to cut across similar pixels or smooth surfaces, as the costs for these will be even higher. On the other hand, when we lower sigma, a decrease in similarity between neighbouring pixels results in the cost decaying even more, making it easier to do simpler cuts along strong edges. If we lower both parameters, our cuts become more sensitive, leading to decreased accuracy in segmentation, as we are more likely to separate and cut similar pixels with high edge values or costs.

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

Figure 12: k = 1

Answers:

Figures 10 and 12 shows the results for, respectively, k=5 and k=1. Even with k=1 we achieve similar results to the case in which was 5.

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:
The accuracy of an algorithm applied to a particular image depends on the context of the image. If the image contains an easily recognizable object, such as a person or a car, a bounding rectangle can be used to more accurately define a representative training set. On the other hand, if the image contains a mix of foreground and background elements, such as in a landscape image, a bounding rectangle will not be of much help.

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:

Image foreground-background segmentation is the process of classifying image pixels into two distinct groups: foreground and background. This is typically done by clustering data points together according to their similarities in colour.

K-means clusters data points according to colour similarities, without taking spatial information into account. This means that it often produces segments which cross areas that should be separated. Mean-shift, on the other hand, takes spatial information into account as well as colour, but does not need prior information about the expected ratio of foreground/background. Normalized Cut and Graph Cut are both graph-based methods, meaning that they use a graph to represent the image, with vertices and edges connecting them, based on some similarity measure. The only difference is that Graph Cut needs prior information about the ratio of foreground/background in order to produce the most accurate results, whereas Normalized Cut does not.
