Answers to questions in Lab 3: Image segmentation

Name: Alice Karnsund Program: TMAIM-17

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?

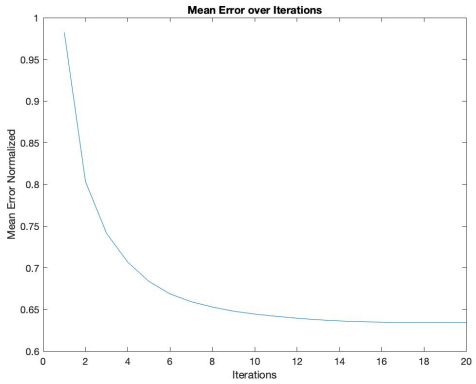
Answers:
If one is given prior information about the image that is going to be segmented, it is easier to select initial centers more close to the optimal values. For example, if we know that we want to segment the image, `'orange.jpg'`, we can handpick initial centers to be close to the dominant colors in the image, such as white, orange, yellow and so on. In general, one might not have as much information about the image. Therefor randomly choosing K centers among the pixels in the image, possibly taking the diversity of the colors into account, is a good approach, which was the approach we used. This is a good way since it makes use of pixel values that for sure are in the picture. A bad idea would be to randomly sample a pixel value from the full color range, this could lead to initialization of centers that are extremely far from the actual values in the image and thus less amount of cluster will be used.
One of the greatest challenges in k-means clustering is positioning the initial cluster centers as close to optimal as possible in a recent amount of time. In 2007 the k-mean++ algorithm was introduced and is doing a good job in achieving this. This is also the way the built in MATLAB function kmeans(X,k) initializes the cluster centers.

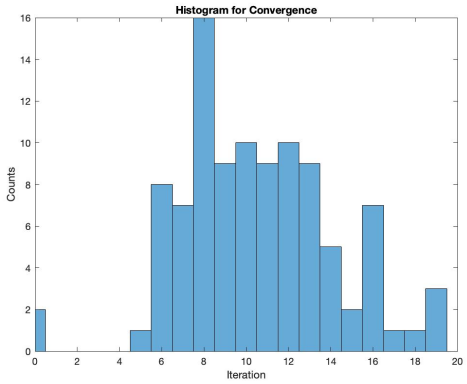
Question 2: How many iterations L do you typically need to reach convergence, that is the point where no additional iterations will affect the end results?

Answers:
Convergence depends on the parameter K, the quality of the image and the size.
A low K value means fewer clusters and often, but not necessarily, faster convergence. The scale_factor and the image_sigma affects the blur of the image. A high value on the scale_factor and a low value on the the image_sigma makes the image more detailed and the running time increases and the convergence is slower. Below are 4 plots for the convergences of the `'orange.jpg'` and the `'tiger1.jpg'` image. For both cases we made use of the convergence criteria that the average error should not change over the last 4 iterations with a precision of 2 decimals. The first two figures represents the mean error over 100 runs and a corresponding histogram over the runs for the orange, with the parameter settings,

$$\left[K, L, scale_{factor}, image_{sigma}, runs\right] = [5, 20, 1.0, 1.0, 100]$$

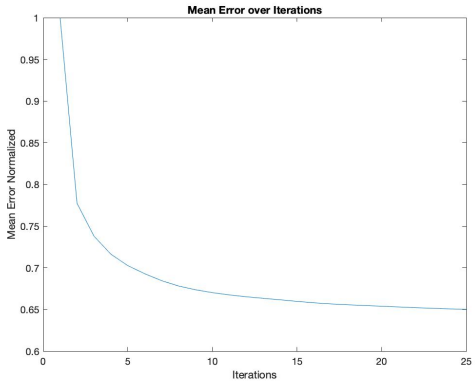
In this case, with K=5, around 8-10 iterations are needed to reach convergence.

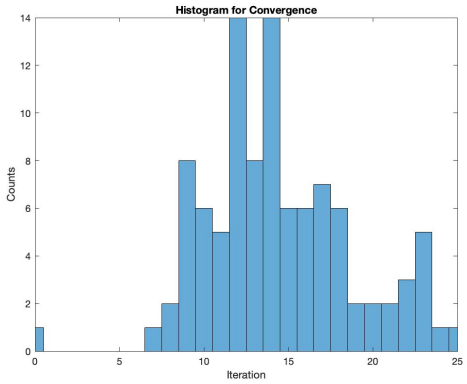




The next two images represents the mean error over 100 runs and a corresponding histogram for the tiger, with the parameter settings,

$$[K, L, scale_{factor}, image_{sigma}, runs = [10,25,1.0,1.0,100]$$



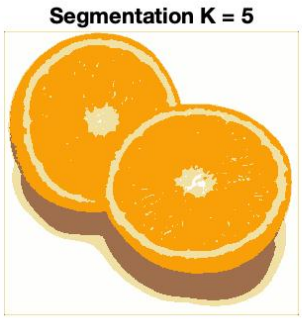


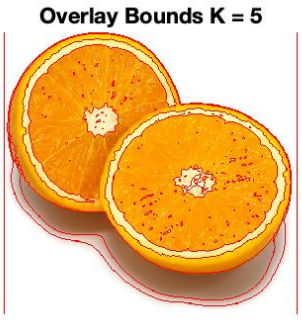
In this case when K=10, the tiger image needs around 14 iterations for convergence.

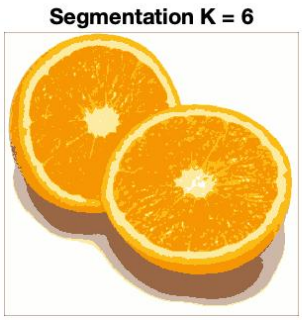
–

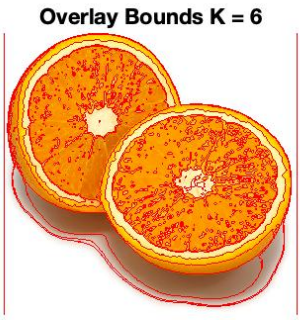
Question 3: What is the minimum value for K that you can use and still get no superpixel that covers parts from both halves of the orange? Illustrate with a figure.

Answers:
To make sure that no super pixels covers both parts of the orange halves we must, on average, increase K a bit. We found that on average this happens when K=6. Due to the random initialization, sometimes this happens at K=5 or K=7. Below is a figure showing the case when K=5 and K=6. This clearly illustrates that when going from K=6 to K=5 in this case, the criteria is no longer fulfilled.





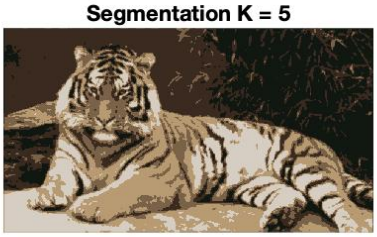




Question 4: What needs to be changed in the parameters to get suitable superpixels for the tiger images as well?

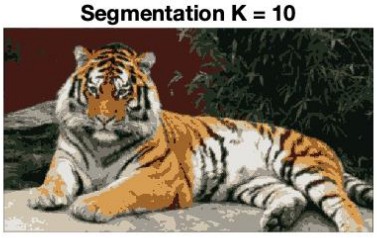
Answers:
First of all the images of the tigers has a much greater diversity in the pixel values, they hold a greater variety of colors. This means that we must us a greater amount of clusters to be able to convey this diversity. Using K=7, which was appropriate in the case for the orange, will not achieve this. A K-value around 10 is a much better choice for example. Also, as can be seen from Question 1, the algorithm needs more iterations to converge. Therefore, increasing L will also be necessary. Below is a figure showing this case for the `'tiger1.jpg'` image. Since the algorithm only focuses on clustering pixels of the same color, we will have clusters that holds pixels both in the background and on the tiger. Also, the tiger images holds small areas of great variation in color and this will give rise to small super pixels. To decrease this effect, one can blur the image more before applying the segmentation procedure.

Segmentation K = 5



Overlay Bounds K = 5

Segmentation K = 10

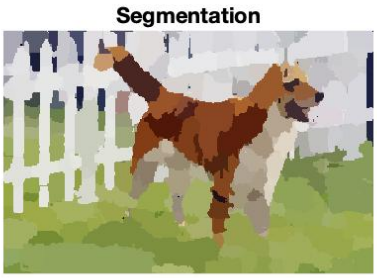


Overlay Bounds K = 10

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:
The greater the value of the bandwidths, the wider the gaussian kernel. This means that more pixels are taken into account when the new cluster centers are being computed. This will result in fewer segments. But if the bandwidths are small, the gaussian kernel is more like a peak in 5G space, and not as much information is taken into account when computing the new cluster centers. This will thus give rise to more segments and the segmented picture can more easily be interpreted.
A high value on the color bandwidth will put almost all focus in segmenting the picture after colors. And a high value on the spatial bandwidth will make the algorithm focus on segmenting into spatial areas. To get as large segments as possible, but segments that each do not cover more than one object in the scene, we must find a balance between the two bandwidths. For almost all pictures, a slightly higher value on the spatial bandwidth compared to the color bandwidth seems to be profitable. In the figure below, an example is shown for the image Tiger3, with $(\sigma_s^2, \sigma_c^2) = (6,4)$.

Image	σ_s^2	σ_c^2
Orange	7	4
Tiger1	10	4
Tiger2	10	3
Tiger3	6	4



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

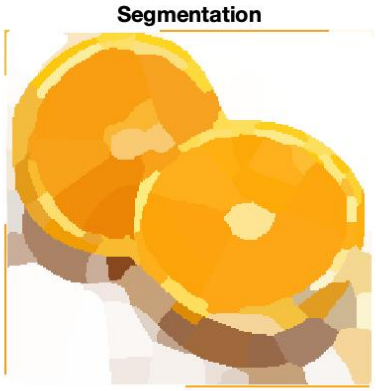
Answers:
Both K-means and mean-shift are two unsupervised learning algorithms that both aim at finding optimal cluster centers. K-means are given the number of clusters K, thus it can be seen as a parameterized algorithm, and the outcome is strongly dependent on K. Mean-shift on the other hand finds an unknown number of modes and then determine which pixel corresponds to which mode. Thus mean-shift is not parametrized by the number of clusters K. K-means only looks at the color values of the pixels, not the spatial position, thus a segment in K-means can be spread out over different regions in the image. This is not the case with mean-shift, since mean-shift also takes the spatial position into account and represents each pixel as a 5 dimensional array. But this also means that mean-shift is strongly dependent on the parameters σ_s and σ_c which controls the size of the kernel and so the segments.

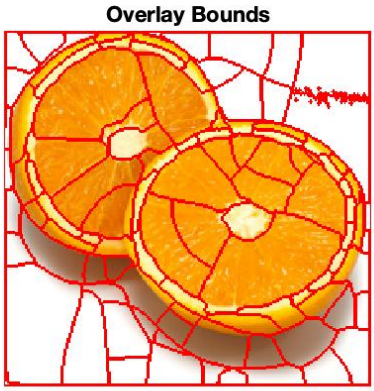
Question 7: Does the ideal parameter setting vary depending on the images? If you look at the images, can you see a reason why the ideal settings might differ? Illustrate with an example image with the parameters you prefer for that image.

Answers:
Each image can be represented in a different amount of segments, since the number of objects, colors, details and size vary from image to image. This implies a parameter setting designed dependent on the image. The min_area, max_depth and the ncuts_threshold parameters can on its own control the number of segments very well, but depend on different things. The min_area parameter depends on the size of the image. Ncuts_threshold depends on the variation in the image and max_depth is simply the depth of recursion, and is not as

affected by the content of the image. The parameter settings is mostly a trial and error activity. But by looking at the image, we can get some clues about how to tune these. A clear fore- and background with not so much variation implies that fewer segments are needed to be able to represent the image. Below are settings for orange image that we found suitable.

Image	Threshold	Min area	Max depth	C-bandwidth
Orange	0.1	80	8	20





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

Answers:
A high value on min_area will decrease the number of segments, the same goes for a low value on the max_depth. A low value on Ncut(A,B) can be achieved by a great similarity among pixels in the segments and very little similarity among pixels on the cut. But it can also be achieved when each segment contains a lot of pixels that in comparison to the cut is very big. Therefor a low value on the ncuts_threshold parameter may decrease the number of segments. We found that the max_depth and the ncuts_threshold influenced the number of segments the most, while at the same time gave a satisfactory segmentation.

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

Answers:
Normalized Cut aims at minimizing the similarities between pixels on the cut, while maximizing the similarities within each respective part of the cut. This corresponds to minimizing,

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

This will be minimized when the cuts are of approximately equal size. If $assoc(V)$ represents the sum of all the edges connected in V, this means that $Ncut(A,B)$ is minimized when $assoc(A,V) \approx assoc(B,V) \approx assoc(V)/2$. To prove this we will rewrite $Ncut(A,B)$ in the form of $assoc(B,V)$ and look at where the derivative is 0,

$$assoc(V) = assoc(A,V) + assoc(B,V) - cut(A,B)$$

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(V) - assoc(A,V) + cut(A,B)}$$

$$\frac{d}{d(assoc(A,V))} Ncut(A,B) = -\frac{cut(A,B)}{(assoc(A,V))^2} + \frac{cut(A,B)}{(assoc(V) - assoc(A,V) + cut(A,B))^2}$$

$$= cut(A,B)\frac{(assoc(A,V))^2 - (assoc(V) - assoc(A,V) + cut(A,B))^2}{(assoc(A,V))^2(assoc(V) - assoc(A,V) + cut(A,B))^2}$$

$$= cut(A,B)\frac{(assoc(V) + cut(A,B))(2assoc(A,V) - (assoc(V) + cut(A,B))}{(assoc(A,V))^2(assoc(V) - assoc(A,V) + cut(A,B))^2}$$

This expression is equal to 0 when

$$assoc(A,V) = \frac{assoc(V) + cut(A,B)}{2}$$

And

$$assoc(V) = assoc(A,V) + assoc(B,V) - cut(A,B)$$

Gives a corresponding expression for $assoc(B,V)$. Thus $assoc(B,V)$ and $assoc(A,V)$ is each approximately half $assoc(V)$. This is not exactly the case in practice. If we wish to divide the parts into exactly equal size, then the median value should be considered as the choice of splitting point. But this usually yields bad result, since it normally results in some points being divided into the wrong group. Since our aim is to segment the image according to similarities in the image, the mean value is a much better choice as splitting point, since it allows for groups that are not of equal size.

Ref: (Haining Wang, Image Segmentation by Normalized Cut, University of California, Davis)

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

Answers:
A higher value on the radius will include more pixels that are further away, and this will lead to larger segments in the segmented image, i.e. less segments in total. Using a low value of the radius, one may create segments that does not really reflect the true content of the image. A greater radius may be a help for this. Therefor a greater radius does not imply a less accurate segmentation. For example, consider the figure in question 7. From that figure , it can easily be seen that, in the white area under the two halves, the process has introduced segments that are not really representative for the image itself. These types of “errors” can be reduced with a higher value on the radius. In this case we will get fewer, more distinct, segments representing the shadows and the white background.

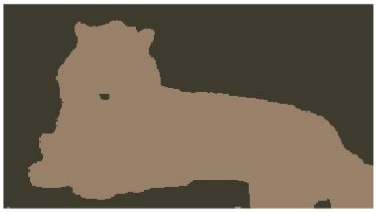
Question 11: Does the ideal choice of *alpha* and *sigma* vary a lot between different images? Illustrate with an example image with the parameters you prefer.

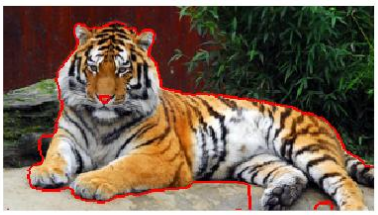
Answers:

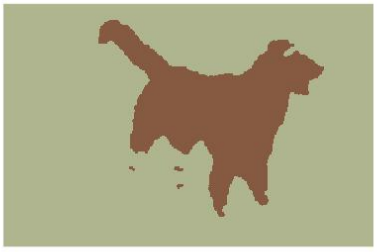
The pair-wise edge costs is written as $e_{i,j} = \frac{\alpha\sigma}{|c_i - c_j| + \sigma}$, here α is the maximum cost of an edge and is taken on when the pixels i and j have the same color. σ is controlling the speed of decay when the similarity between the pixels decreases. The proper settings of these parameters vary from image to image. A high α value is useful in pictures where the colors of the fore- and background are not super different, as in the case of the image tiger3. A low value on σ will also help when one wants to separate colors that are not too far from each other in the color spectra. This means smaller dissimilarities between pixels will be noticed also. Note, that K do also have a great impact on the segmentation, and the segmentation gets more sensitive when fore- and background colors lay more close in color space. Below are some settings that we found suitable for the images tiger1 and tiger3 and the corresponding segmentations. In this case the fore- and background for tiger1 are more distinct than in the case of tiger3, thus tiger3 settles with a higher α and lower σ as expected.

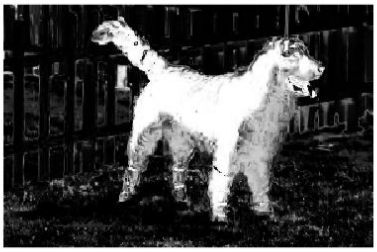
30 4.

Image	**K**	**α**	**σ**
Tiger1	8	30	4
Tiger3	8	40	3









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

Answers:
Considering the same images as in the previous question, tiger1 and tiger3. With the same settings for σ and α, we see that the tiger3 images is more sensitive to the changes in K, compared to tiger1. This has to do with the fact that the background pixels are more close to the foreground pixels in tiger3, than they are in tiger1, as brought up in question 11. Also tiger3 has more different colors over all and should therefor favor a higher number of Gaussian components. If K is too low, tiger3 will assume that for example the grass part belongs to the foreground pixels. Tiger3 gives reasonable results when K=7, but when K=6 or lower, the grass and other nonrelative areas are considered to be part of the foreground pixels. Tiger1 does a better job in this case, since the contrast between the tiger and the background is much greater. For this image the segmentation is still reasonable when K=3, but when K=2 some parts, such as the bush and stone, are assumed to be foreground pixels.

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:
To be able to use Graph Cut in an efficient way, one must be given the image one wants to segment, prior to the operation. And from that, specify a rectangle that can represent the foreground. This approach is giving our algorithm some prior information, and thus has knowledge about in what direction the weights should be updated. This implies a that Graph Cut is a type of supervised learning, whereas the earlier methods are unsupervised. More information usually results in better results, which is the case here. Comparing the methods

used in this lab, Graph Cut is by far doing the best job. All methods need parametric tuning, and specifying a rectangle should not be too much to ask, as the segmentation result increases a lot. As noted in this lab, the initialization of the rectangle does not have to be very precise, as long as the major part are foreground pixels. Furthermore this method guarantees a global solution, which is the best solution.

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:
First of all, as mentioned above, the Graph Cut is a supervised method while K-means, Mean-shift and Normalized Cut are unsupervised methods. K-means in this case, only considers the colors when performing the clustering. The other three methods also takes the spatial positions of the pixels into account, which will yield more coherent clusters. K-means and Graph Cut also need a predefined number of clusters/components, K. In K-means these will simply give rise to finding K optimal cluster centers. In Graph Cut, K is the number of Gaussian components that represents different color distributions. In Mean-shift and Normalized Cut we cannot control the number of segments in the same way.
Normalized Cut and Graph cut, both make use of a graph model, where the weights on the edges depends on the similarities between the pixels. All algorithms works to minimize a type of cost function, Normalized Cut and Graph cut aims at minimizing a function of the edges on the cut. K-means aims at minimizing the distances between points and the cluster means. And Mean shifts uses gradient ascent to climb to modes in density space that has a derivative of zero. In our case, where we wish to segment the image into fore- and background, the Graph cut was by far the best method and will find a globally optimal solution. Since the running time for this method is the worst, one may consider other methods depending on the problem. Lastly, some drawbacks were the time it took for trial and error in the case of finding good parameters for the different algorithms. Mean shift and Normalized Cut was most time consuming for this task.
