Answers to questions in

Lab 3: Image segmentation

Name: Per Emil Hammarlund Program: TMAIM

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!

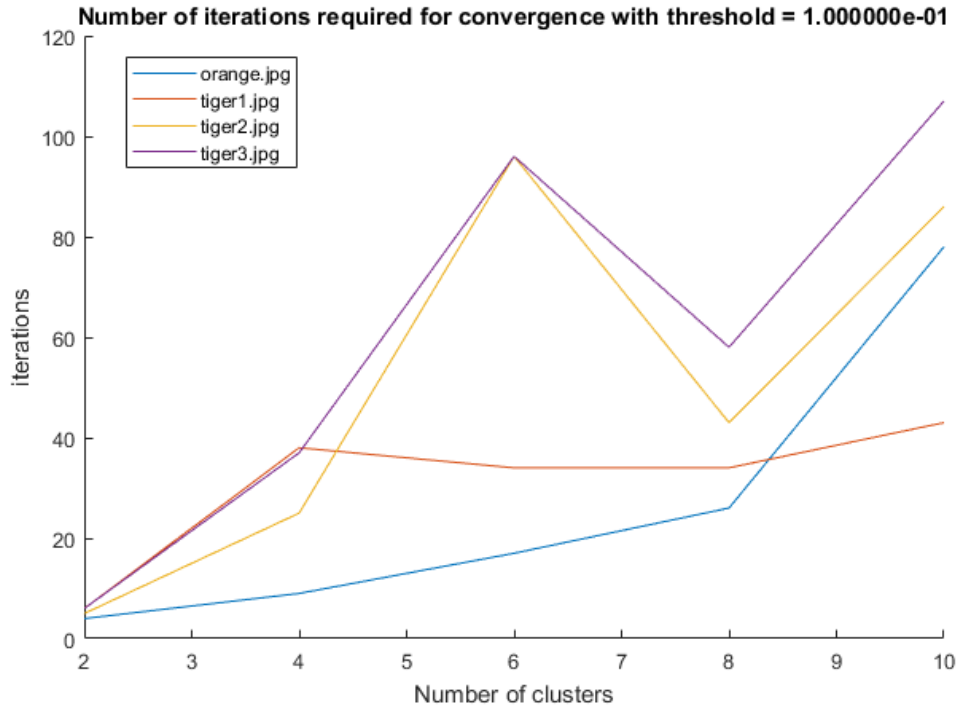
Question 1: How did you initialize the clustering process and why do you believe this was a good method of doing it?

Answers:
I initialize the clusters using random partition initialization, I believe that this is a reasonable choice since this gives me pixels that are within the range of the images pixel values and some degree of variability between initial clusters.

Using uniform discrete numbers 0-255 leads to initial clusters that could be too far from the actual image content, and Forgy initialization risks giving initial clusters with the same value when there are larger portions of the image that are the same pixel value.

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:
When defining convergence as when the largest individual difference between the previous iterations cluster values and the current iterations cluster values being less then 0.1, the following results where observed:

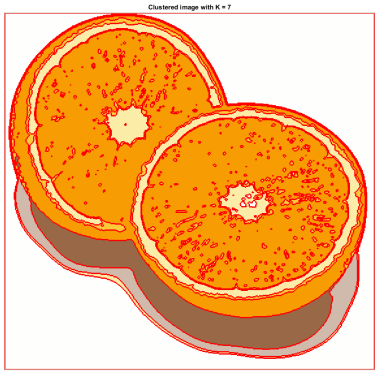


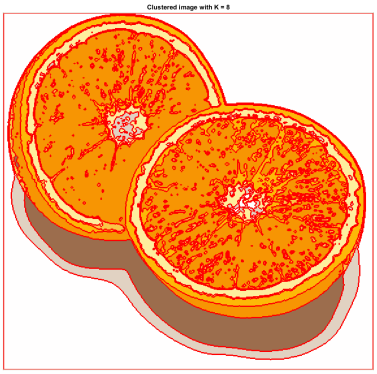
As can be seen in the figure above, there is a peak in the number of iterations required for achieving convergence for tiger2 and tiger3 before they continue to grow again. Tiger1 peaks shortly at 4 clusters and then continues to grow and the orange image does not stop growing, and seems to grow exponentially.

The continuing growth of the orange image could be explained from its shadow that gives us an almost gradient-like behavior in the pixel values.

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:
The minimum value for K that does not produce super pixels that cover parts from both halves of the oranges is eight:





Looking at the figure above, we see that the super pixel that covers the lower half of the two orange halves merges when we decrease K to 7. Increasing K back up to 8 separates the two orange halves again.

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

Answers:
Due to the tiger images being more complex, we will have to increase the number of clusters to produce satisfactory results. As we increase the number of clusters, we also have to increase the number of iterations required for convergence is proportional to the number of clusters. Testing various different larger values for K and using the figure in question 2 as guidance as to how many iterations we need for convergence gave the K and L values:

```
K = [9, 9, 10]; L = [50, 80, 110]; (Where each index corresponds to the tiger number),
```

This gave the following results:

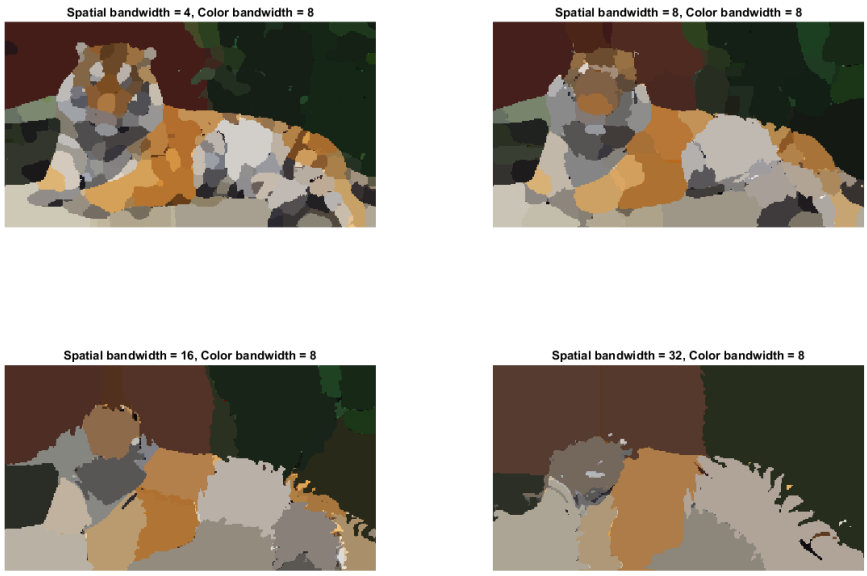
tiger1.jpg

tiger2.jpg

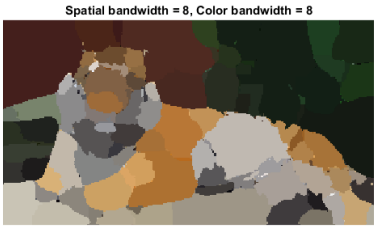
tiger3.jpg

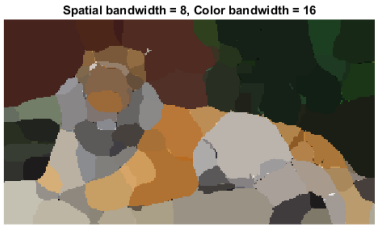
Question 5: How do the results change depending on the bandwidths? What settings did you prefer for the different images? Illustrate with an example image with the parameter that you think are suitable for that image.

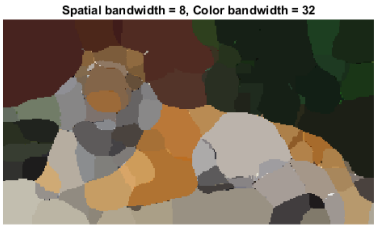
Answers:



As can be seen in the figure above, the smaller the spatial bandwidth, the smaller the region of interest and the more modes. The larger spatial bandwidth, the larger the region of interest and the fewer the modes.







From the figure above, we can see that the smaller the color bandwidth the more small blobs are within larger regions. The larger the color bandwidth, the more these small blobs disappear. In a way, increasing the color bandwidth could be seen as a method for "denoising" larger image segments.

Judging from the previous two images, I would like to argue that a suitable set of parameters for segmenting the image with mean-shift segmentation would be:

```
spatial_bandwidth = 8;
colour_bandwidth = 32;.
```

Using these parameter settings gave the following result:

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

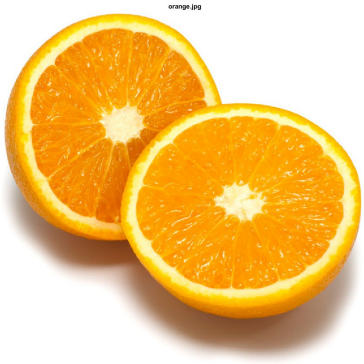
Answers:
K-means uses a predetermined number of clusters and then computes the most likely cluster for each pixel while mean-shift segmentation uses a similarity measure between pixels in a region of interest and maximizes a density function.

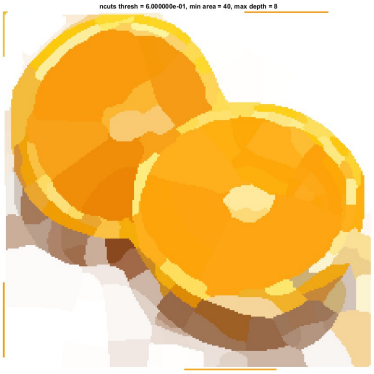
K-means (at least in the version we have implemented here) does not take the position of pixels into consideration, while mean-shift does. So when using K-means, a cluster can be in disjoint areas of the image.

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.

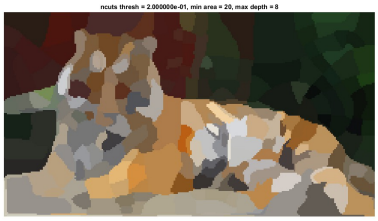
Answers:
Testing different parameter settings for ncuts_thresh, min_area, and max_depth gave the following results:

```
ncuts_thresh = 0.6; min_area = 40; max_depth = 8;
```

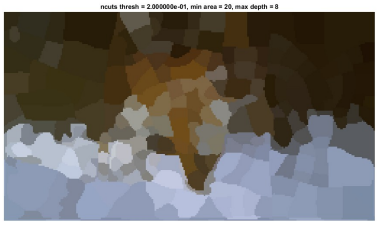




ncuts_thresh = 0.2; min_area = 20; max_depth = 8;



ncuts_tresh = 0.4; min_area = 20; max_depth = 8;

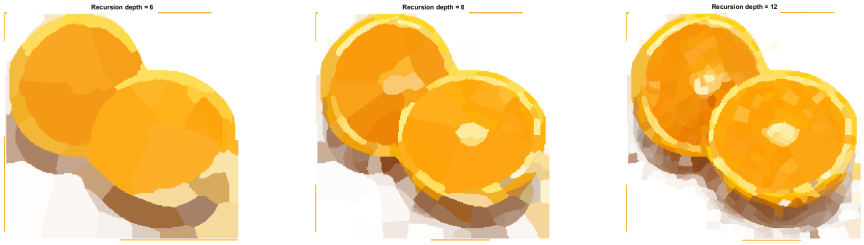


ncuts_thresh = 0.4; min_area = 20; max_depth = 8;

As can be seen in the images above, the minimum area of the segments that can be used increases as the size of the features in the image grows and the more fine details there are in the image, the stricter the threshold must be to create a satisfactory segmentation.

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

Answers:



Increasing the recursion depth led to greater subdivision, and decreasing the recursion depth led to segments covering larger portions of the image. There is no doubt that the recursion depth does influence the subdivision, though the depth 8 was used in all images.

Unsurprisingly, the minimum size of the segments controls how small the image segments are allowed to be. Which is undoubtedly effective at ensuring that segments do not become too small, but needs to be tested on a given image to see if the segmentations that are produced are satisfactory.

The ncuts_thresh paramenter influences the accuracy of or cuts, and the more complex the image is the stricter threshold has to be to produce a satisfactory segmentation.

Weighing these conclusions together, all three do have noticeable effects on the segmentation. But I would like to argue that the combination of the correct minimum size is most important parameter for producing a satisfactory segmentation. Since if we are unable to find a matching minimum segment size, we will not be able to produce segments that match the features of the image.

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

Answers:
Looking at the objective function that we are minimizing in normalized cuts:

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

where cut(A, B) is the sum of the weight of the edges between A and B, and assoc(A, V) as well as assoc(B, V) is the sum of the weight of connections between the nodes in the given partition and the rest of the graph.

If the cut was not divided by the factor assoc(A, B), then the algorithm would have created favored cutting off small sets of isolated nodes in the graph. But since we have the "normalization" factor, performing a cut that will create small isolated nodes would lead to the value approaching 1. This is further illustrated by the expression:

$$Ncut(A, B) = \frac{\sum_{a \in A, b \in B} w(a, b)}{\sum_{a \in A, v \in V} w(a, v)} + \frac{\sum_{a \in A, b \in B} w(a, b)}{\sum_{u \in B, v \in V} w(b, v)},$$

where w is a function that returns the weight of the edge between two nodes.

The expression is minimized when the total weight of the cut edges is at its lowest and the total weight of the edges in each segment are at its highest. If one segment grows its corresponding denominator grows and the other segments denominator will decrease. So if there are too many nodes in one segment the other segments ratio will grow. Thus the minimum is achieved when there is a balance between the two.

In practice the globally optimal minimum is not found with almost all certainty since solving for a minimal graph cut is an NP-complete problem.

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

Answers:

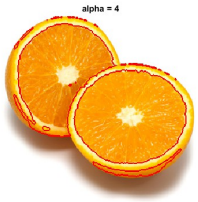
As can be seen in the figure above, increasing the radius increases the accuracy of the segmentation.

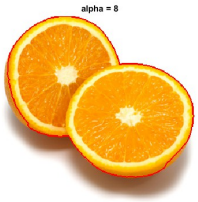
Question 11: Does the ideal choice of *alpha* and *sigma* vary a lot between different images? Illustrate with an example image with the parameters you prefer.

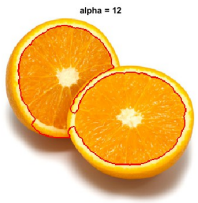
Answers:
Testing for different alpha values with tiger1, and the default bbox gave the following result:

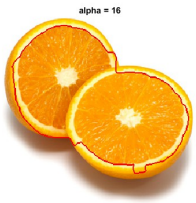
And doing the same for the orange with the bbox:

```
area = [ 80, 110, 570, 300 ];
```



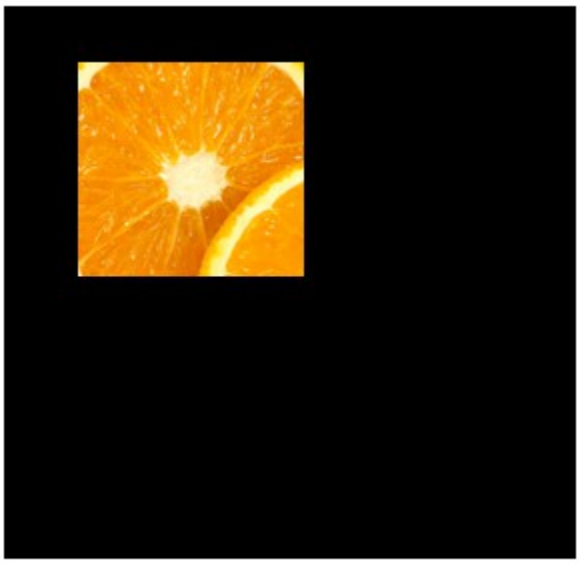






As can be seen the the two images above, the optimal alpha does infact depend on the image. For the tiger, lower alpha values led to the segmentation getting smaller islands and larger alpha values decreased the number of smaller segments and giving an overall better segmentation. A trend that could be explained from the increased cost of cutting between pixels of that have similar values.

As for the orange halves, the optimal alpha is found at a much lower value. (alpha = 8). This is likely due to the large difference in color between the meat of the orange the white-beige color of its inner skin. This is probably not the only reason why though, the cut that was chosen has very little inner orange skin in it. So the GMM that is trained on the selected region is highly biased towards the meat of the orange:



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

Answers:

As can be seen in the figure above, there is a tipping point once K = 4 where the entire bottom portion of the image gets included in the foreground segment. All other K before produce similar results though.

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:
I would like to argue that the benefit in the results is most definitely worth the effort. When we receive the region of interest, it is much easier to classify whether pixels are in the background or the foreground of the image. In a sense, using the user input as an indicator of what the foreground is takes the problem from an unsupervised classification, to a supervised classification.

This is however, entirely contingent on the assumption that the user is able to select a region that represents the foreground. If that is not the case, the method falls apart. Though there are various methods in which we could ensure that the user is able to define the foreground sufficiently. Such as allowing the user to trace out the region of interest e.t.c.

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:
The first and most obvious difference between the four methods is that k-means and mean-shift segmentation are both not graph based methods whereas Normalized cut and Graph cuts are.

There is a similarity between mean-shift segmentation and the graph-based methods in that the segmentation is performed using a similarity measure between the current pixel and neighboring pixels in a region of interest. But mean-shift segmentation differs in that it uses gradient accent to modes.

K-means has the least in common with the two graph based methods, when restricting our analysis to the version of k-means that has been implemented in this lab. There is no consideration as to where the pixels are located, nor any distance measure between other pixels. Only a distance measure between each pixel and each cluster.

K-means and mean-shift segmentation are differ in that k-means seeks a predetermined number of modes whereas mean-shift seeks an unknown number of modes. There is a similarity between k-means and mean-shift segmentation though. Both of them seek a certain number of modes to converge towards. Something that neither of the graph-based methods do.

Normalized-cuts is obviously related to graph-cut in that they both are graph-based methods and they both define their edges between pixels in the image using a similarity measure. But they differ in how the perform their segmentation. Normalized-cuts divides the image into two regions and then continues to subdivide recursively using an objective function to determine where to divide the graph. Graph-cut uses both a similarity measure between pixels as well a a probability of each pixel being a part of either the background or the foreground and the final graph is then fed into the min-cut algorithm to find the minimal cut that separates the background and foreground. Finally, graph-cut is assisted by the user in and receives a region that is known to be the foreground, whereas normalized-cuts is entirely unsupervised.
