Assignment 3 - DD2423 - Computer Vision and Image Processing

Alexandra Hotti

November 30, 2018

1 Introduction

In this lab a number of popular methods for image segmentation are explored. Furthermore, these methods deal with spatial coherence in different ways. Some methods do not take the spatial locations of pixels into consideration at all, whereas others do it explicitly using some neighbourhood system.

2 K-means clustering

Segmentation is often done in stages that involve different algorithms. In many cases it can be too time-consuming to work on individual pixels for a final segmentation. The solution is to create an over-segmentation and divide the image into many so called superpixels, i.e. groups of connected pixels that are so similar that they must originate from the same physical object in the scene. Each superpixel might contain from a handful of similar coloured pixels to tens of thousand. It all depends on the particular image. The purpose of using superpixels is to reduce the number of pixels in an image needed to segment the image.

In this first exercise the K-means clustering algorithm is used to cluster all pixel colours to a smaller set of colours, each cluster represented by its center colour, which is the mean of all pixels assigned to it. Often this procedure is used to change colour representation from 24 bits (8 bits per colour channel) to a palette of colours, where the colour of each pixel is given by an index that requires a fewer number of bits. However, it can also be used to create superpixels, which is its purpose here. For a more detailed description of the assignment see Assignment3_Description.pdf.

Question 1: How did you initialize the clustering process and why do you believe this was a good method of doing it?
Try modifying the K and L parameters, the scale factor and the amount of preblurring, to see how the changes affect the results.

We initialize the clustering by randomly selecting center values from the image pixels color values as our center means.

We believe that this is a good initialization method since it is likely that we will obtain centers that approximately represent the proportion of different colors in the image that are

represented in the objects.

Another method could have been to manually set appropriate starting means to exactly represent the proportion of colors in the image. We could for instance have manually set the starting pixels for the orange as lets say 70% orange pixel variations and 30 % white pixel variations.

2.1 Modifying K

When the number of clusters K is set to low the superpixels cannot distinguish between the similarly colored orange halves in Figure 1. Instead K means segments the two halves as a single superpixel.

When we increase K we get more clusters and the number of superpixels increase. However, we can see in Figure 2 that increasing K to 8 still does not completely seperate the two orange halves. However, as this algorithm only takes color into consideration and the two orange halves are very similarly colored, getting two perfectly separated halves would be difficult.

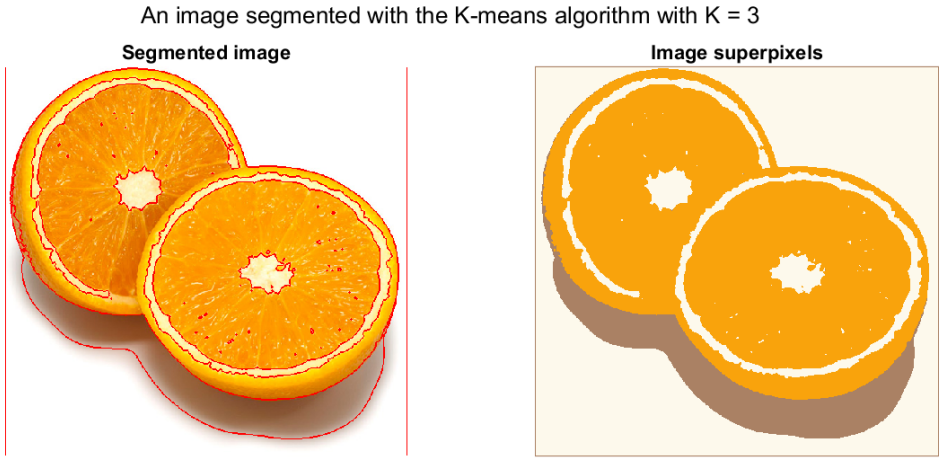


Figure 1: An image segmented with 3 clusters.

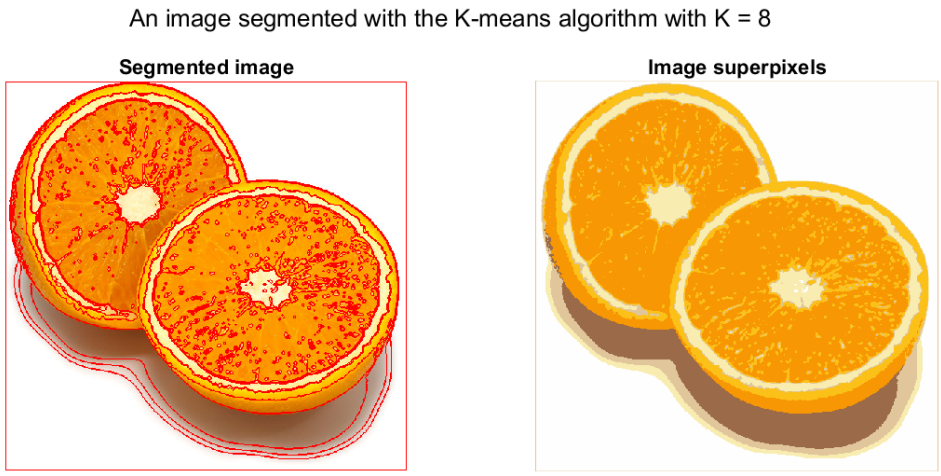


Figure 2: An image segmented with 8 clusters.

When we increase the number of iterations we refine the mean of the color clusters. In the beginning after only 2 iterations in Figure 3 we randomly ended up with 3 colors in the image: yellow, blue and black. The black is already close to the mean of the colors in that region of the image. However looking at the other 2 clusters we randomly generated a blue and a yellow cluster. Thus the blue area that contains both blue and purple is expected to become more purple, while the green/yellowish area is expected to become greener as the number of iterations increases. This can be seen in Figure 4.

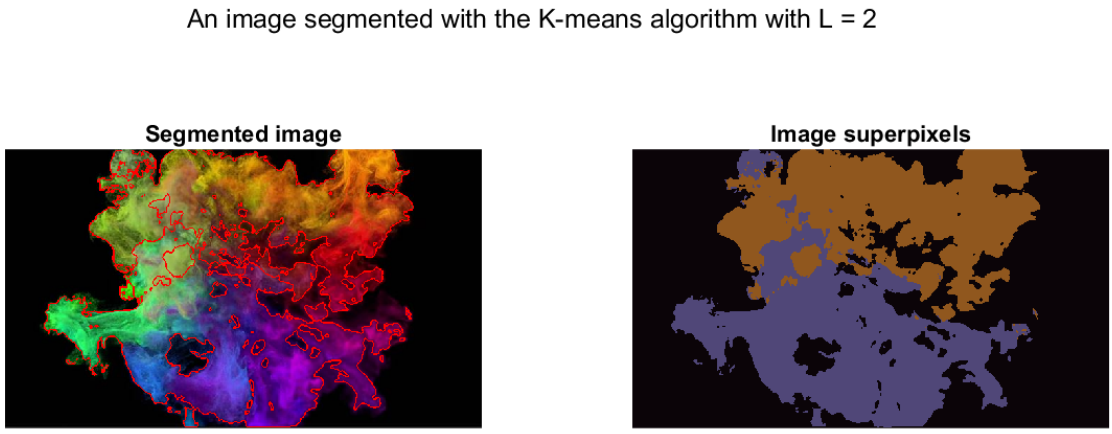


Figure 3: An image segmented with 2 center updates.

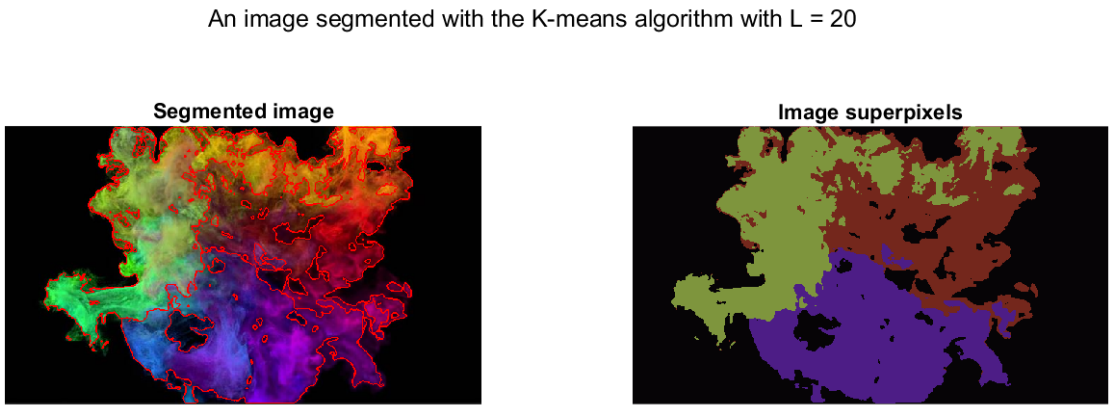


Figure 4: An image segmented with 20 center update.

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?

To ensure convergence it was checked that each new center after an iteration that had moved less than a threshold of an Euclidean distance of 0.01.

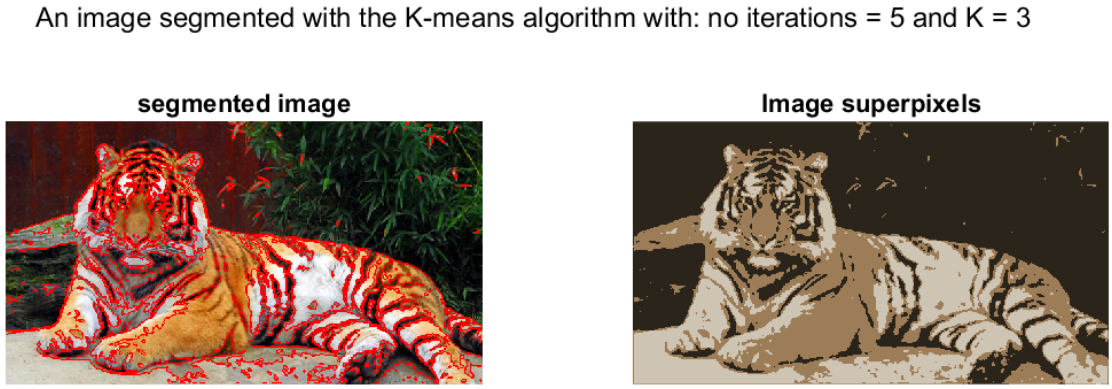
Table 1: Sizes of differentiation filters.

K	orange.jpg	tiger 1.jpg	tiger 2.jpg	tiger 3.jpg
3	8	13	7	4
7	18	31	102	109
9	39	78	140	144

Varying K: The number of iterations depends on the value of K, when we have many cluster centers the pixels tend to change cluster more often and therefore convergence takes longer.

Varying the Scale factor: Decreasing the scale factor is good in an image with large objects with few small segments such as in the third tiger image below. But if we have larger objects in an image with less details it should work better for instance like the oranges.

Super pixel wise blurring: This removes coloring distinctions between parts of the image. Thus, for instance noise and sharp shadows are suppressed. However, blurring to much removes the distinction between true segments. For instance, in the image of the tiger the paw becomes part of the background when there is to much blurring.



An image segmented with the K-means algorithm with: no iterations = 12 and K = 3

segmented image

Image superpixels

An image segmented with the K-means algorithm with: no iterations = 6 and K = 3

segmented image

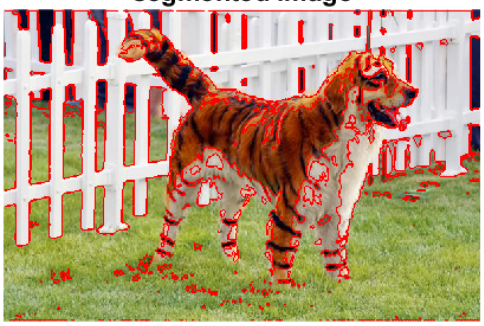


Image superpixels

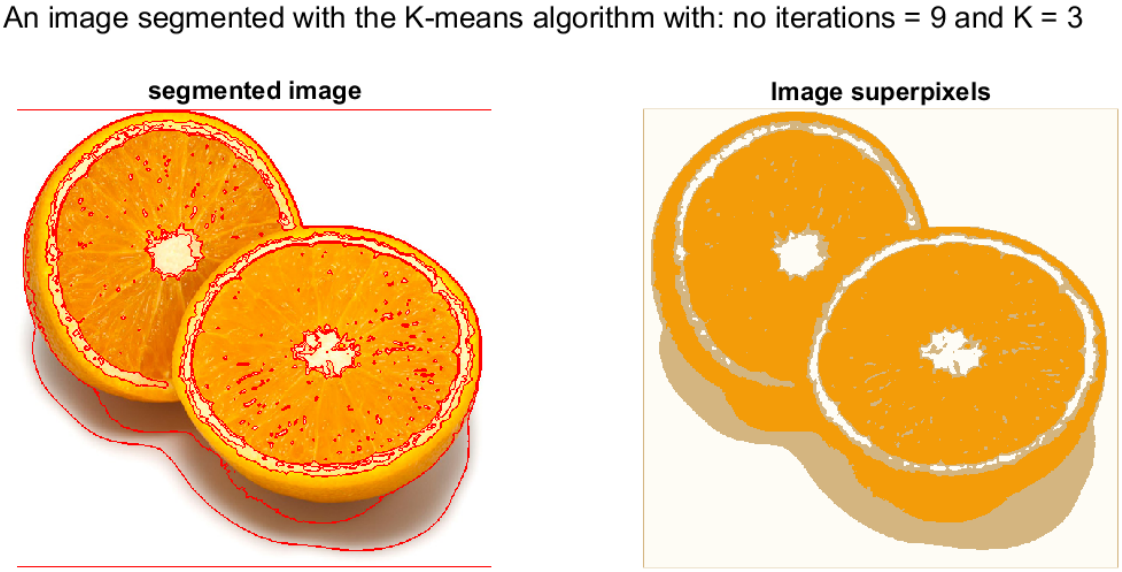


Figure 5: Image segmentation's using a threshold for the euclidean distances between the cluster centers between 2 iterations.

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.

Increasing K means that we can represent more colors. So when we have 2 objects with really similar colors, we need many clusters to try to model the small dissimilarities between colors between them.

If we look at the upper corners where the oranges intersect in the upper image in Figure 6 there is hole which means that the algorithm cannot distinguish between the 2 orange halves when K = 24. If we inspect the same point when K = 25, one can notice that the same hole is closed and the algorithm can thus distinguish between the halves.

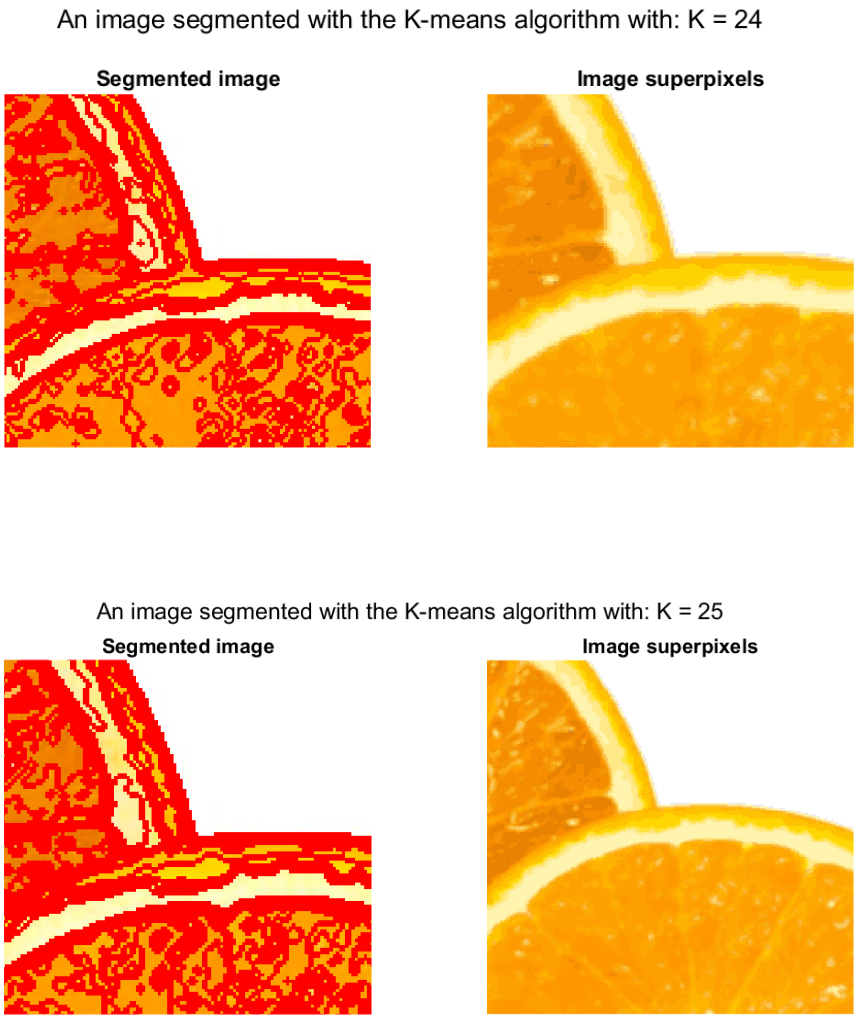


Figure 6: Orange halves - Increasing K to separate the orange halves with different super pixels.

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

The first tiger image in Figure 7 needs lower values for both K and L compared to the orange to get pleasing results for the human eye. Since it consists of objects of colors that are well separated from each other compared to the orange where the two halves consists of similar colors.

However, we can see that some areas of this tiger still blend in to the surface, such as the top of the front paw.

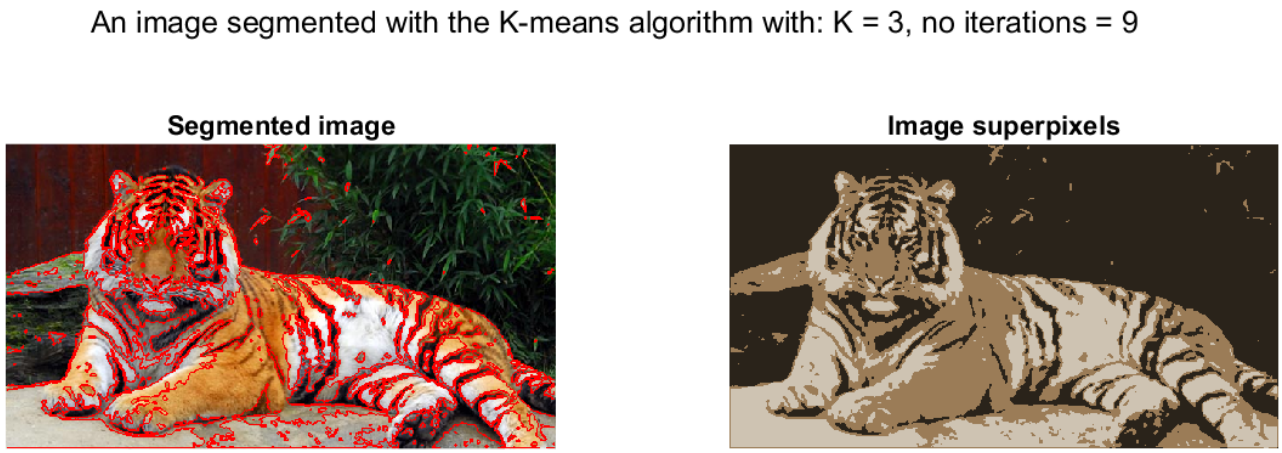


Figure 7: Best result for segmentation with K-means of tiger image no 1.

The image in Figure 8 consists of a tiger that blends in to the background more compared to the tiger in Figure 7 and therefore its colors are not as easily distinguishable from the background compared to the first tiger. Therefore we need more clusters and more iterations to account for less clear difference between the colors of the tiger and the background.

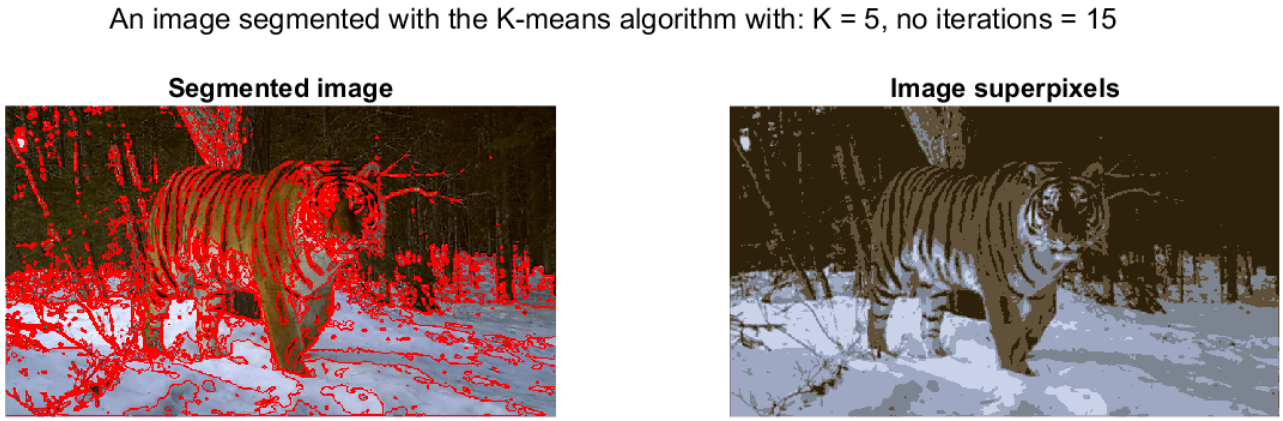


Figure 8: Best result for segmentation with K-means of tiger image no 2.

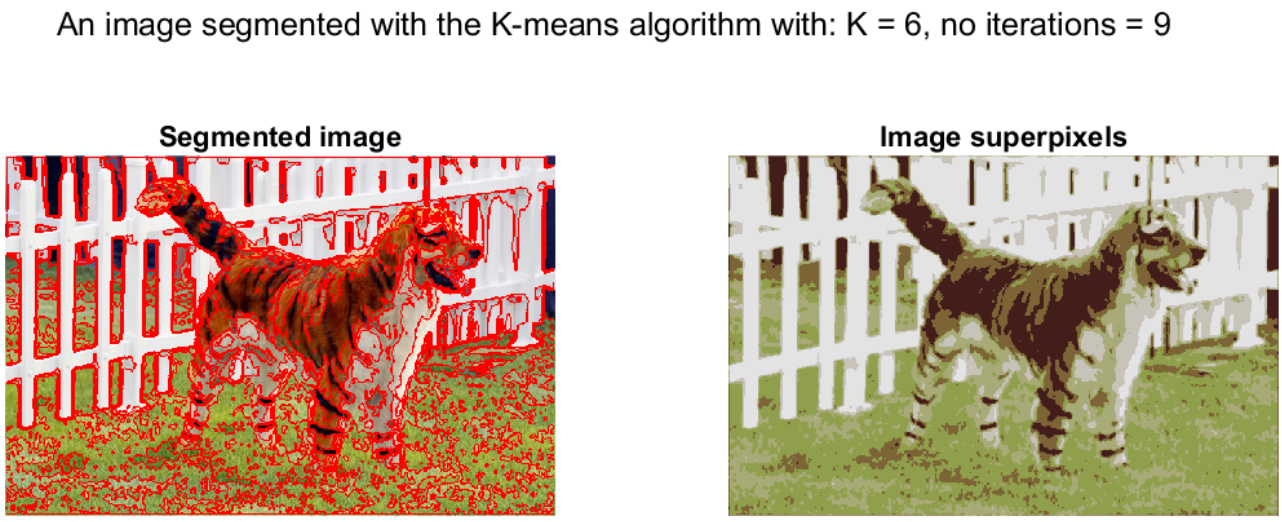


Figure 9: Best result for segmentation with K-means of tiger image no 3.

3 Mean-shift segmentation

Typically, K-means clustering does not consider image positions for clustering. Spatially similar pixels can still be grouped together in an post-processing step, in which connected components are found, each component having a particular cluster colour. The segments could, however, stretch over large parts of the image, if colours are similar, leading to an unnecessary fragmentation of the result. An alternative to K-means clustering, that usually does clustering both spatially and in colours, is mean-shift segmentation. For a more comprehensive explanation see the assignment description.

Question 5: How do the results change depending on the bandwidths of the Mean-shift algorithm? 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.

Relation spatial vs color variance: How much importance do we give to color vs location.

Weights: So, when we increase the bandwidths we spread out the kernels more. Therefore, we will give less weights to the center pixels and more weight to pixels further away from the center. See Figure 10.

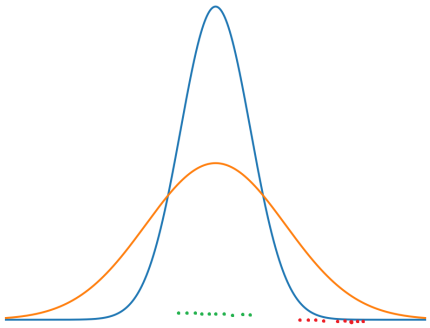


Figure 10: Relationship between variance and weights.

Number of modes: Also, when we increase the bandwidth we should find less number of modes. Since a smaller bandwidth allows us to find more local maximas/modes. Therefore, using a larger variance becomes more robust against outliers and we get more global regions of interest. See Figure 11.

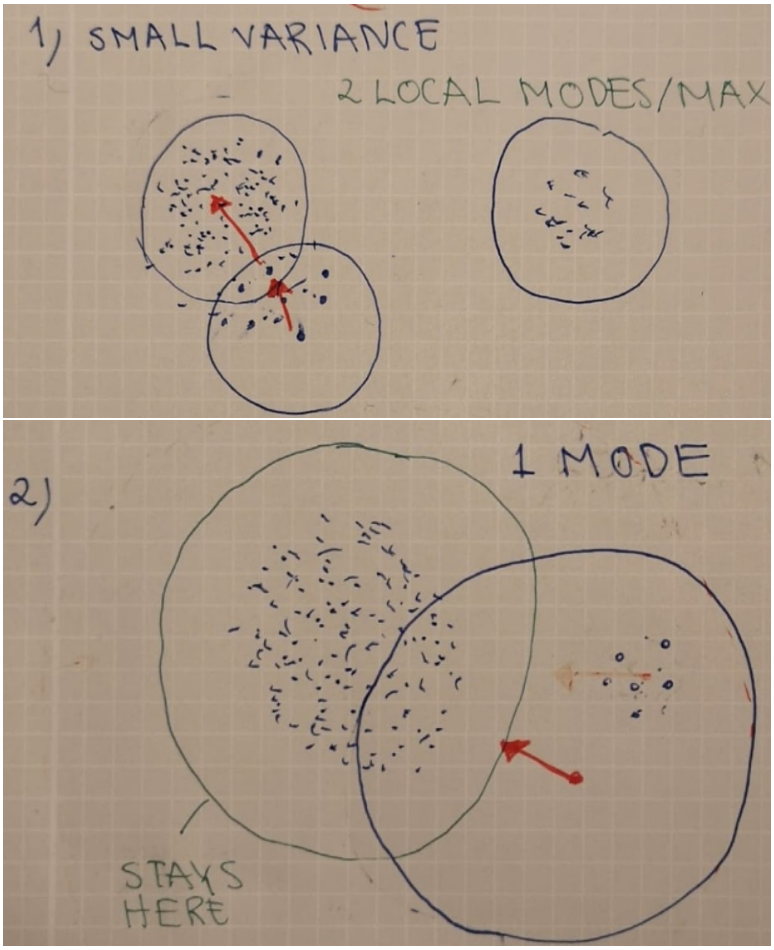


Figure 11: Relationship between variance and the number of modes found.

Color variance: We increase the influence of neighboring pixels color intensities. Therefore, in accordance with the images in Figure 11 we will get less modes and less color variations in the image and more colors will be segmented to less colors and therefore we will get larger segments in our image. This is shown in Figure 13

When we have small variations between colors within the image we risk losing really fine differences between colors. If we use large color bandwidths all colors could go to the same mode and therefore, we could not distinguish between fine color differences such as in the image of the 2 orange halves.

In the tiger image the object and the background colors are well distinguished. Therefore, there is less risk of removing fine orange variations on the tigers back since the background is already well separated from the tiger. Thereby we get better segments within the orange cluster.

Spatial Variance: The same reasoning as for the color bandwidth holds also for spatial bandwidth. The segments become larger and we get less modes when we increase the spatial bandwidth. See Figure 12

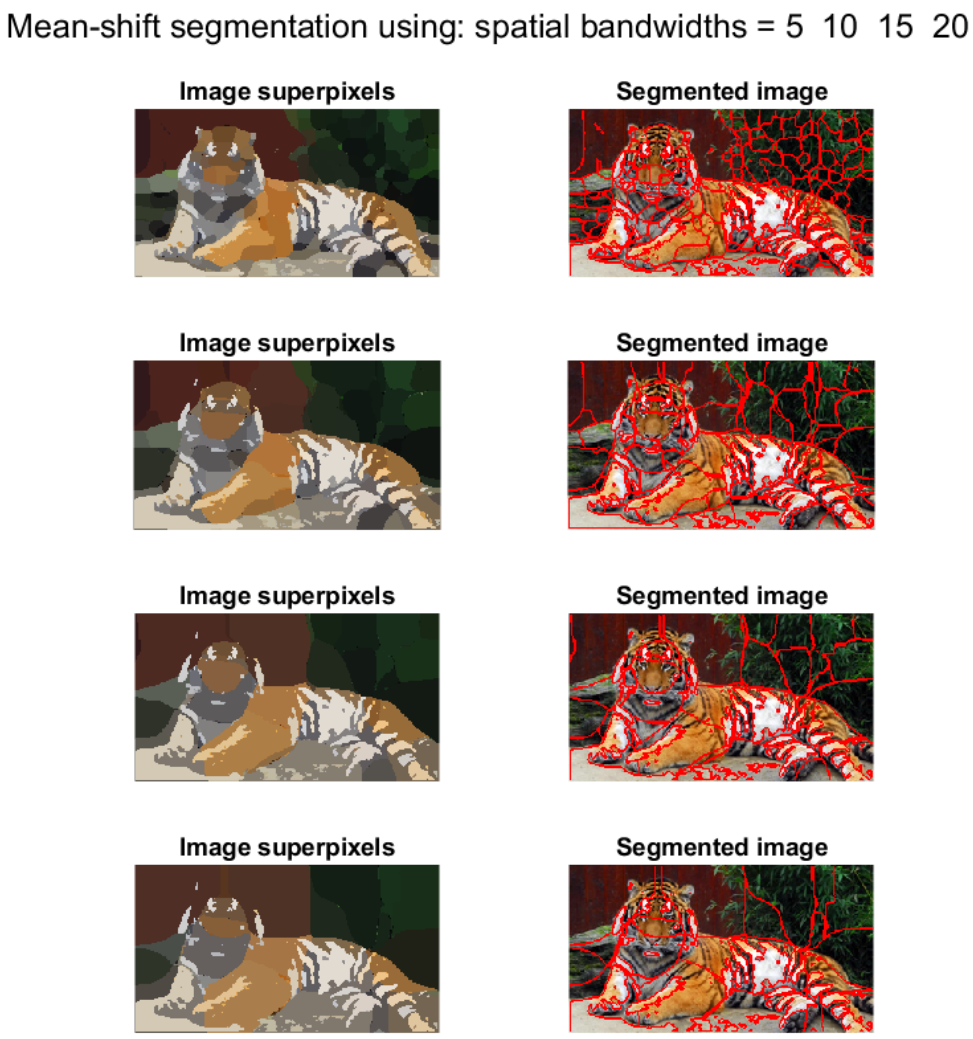


Figure 12: Image mean-shift segmentation with varying spatial bandwidth.

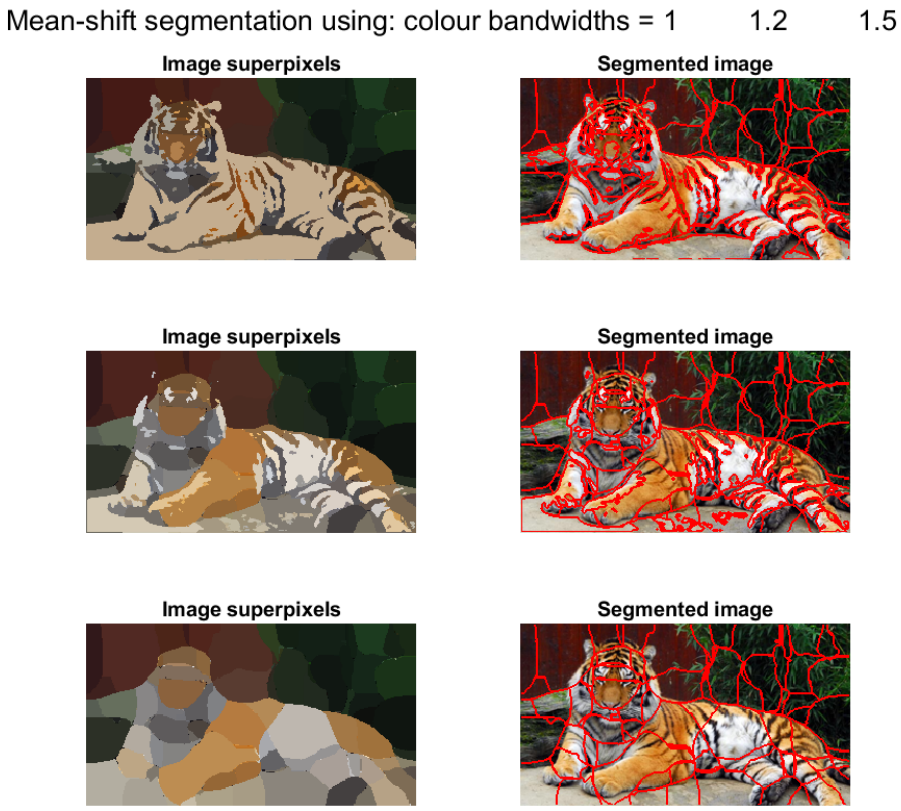


Figure 13: Image mean-shift segmentation with varying color bandwidth.

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

- **Finding K:** In K-means we manually set the value for K which is difficult to get right. But in the Mean-Shift algorithm this is done automatically as every pixel/superpixel finds a mode on its own by finding regions of high density.
- **Spatial Location:** K-means only takes an image's color value into account during segmentation, while mean shift also takes location into consideration. Often there are dependencies between neighboring pixels.

  Thus, for instance when we want to distinguish two objects with very similar colors, it is advantageous to also take location into account. Such as in the case of the image of the orange halves.
- **Initialization:** K-means is sensitive to where we place the initial centers, since this could result in different solutions. This could also result in clusters where no pixels are assigned to the cluster which could result in zero division.

- **Robustness against outliers:** Mean-Shift is more robust to outliers compared to K-means. Since Mean-Shift creates modes using kernels to weight every pixels contribution to the modes, a far away outlier will not affect the result as much compared to K-means, which takes every point equally into account.
- **Complexity:** in Mean-Shift you find a mode for each pixel. But in K-means you assign several pixels at once to a cluster. Therefore K-means converges faster.

4 Segmentation using Normalized Cut

With mean-shift segmentation you cannot control in how many segments an image will be divided. It depends on how many modes (cluster centers) the method finds, which in turn depends on the bandwidths used and the particular image. In figure ground segmentation, however, you often like to limit the number of segments, preferably having one foreground segment and one background segment. A method that allows this is Normalized Cut. Normalized Cut uses a neighbourhood system, here defined by a parameter radius, to connect pixels into a large graph G = (V; E) with pixels on the vertices V and edges E between neighbouring pixels. Each edge is given a weight corresponding to the similarity between the two pixels that is connects. The method then tries to divide V into two subsets of vertices (pixels), A and B, such that

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{1}$$

is minimized. Here cut(A;B) is the sum of all edges that connect two vertices in A and B respectively, and assoc(A; V) is the sum of all edges connected to any vertex in A. That is, Normalized Cut tries to minimize the similarities between pixels on the cut, while maximizing the similarities within each respective part of the cut.

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.

Yes, the ideal parameter setting varies a lot depending on the image.

Setting the Radius parameter:
First the orange image contains large objects and therefore it needs a large radius parameter to account for similarities between opposite edges of the halves. When we decrease the radius, we take less neighbors into account for each pixel when we compute the affinity matrix, therefore we cannot find similarities between pixels that are far away from each other. For instance in Figure 14 we can see that at first the radius is to small to account for similarities between opposite edges, but as we increase the radius we can do this.

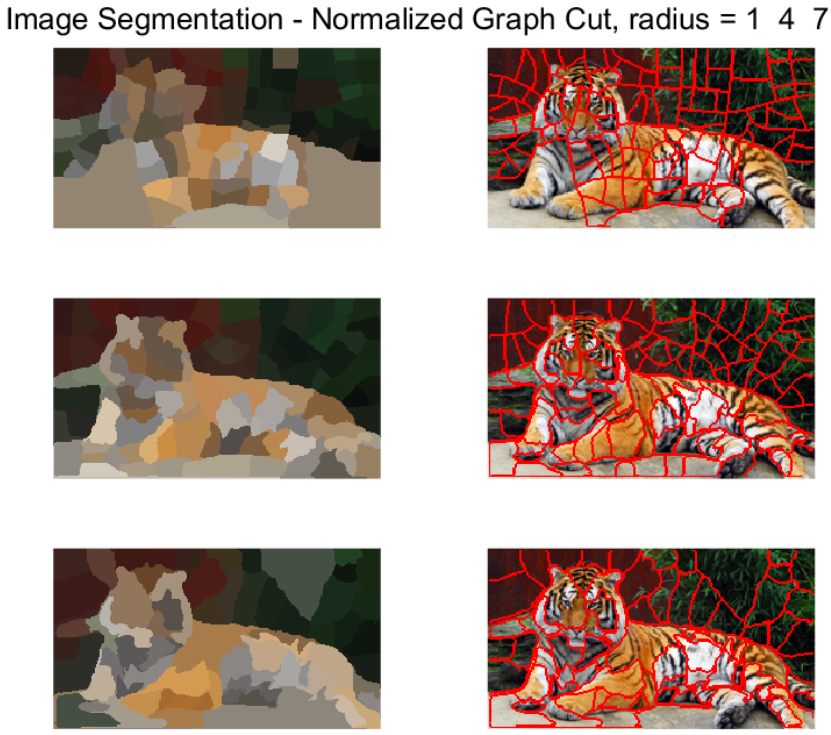


Figure 14: Normalized Graph Cut using different radiuses.

Setting the Min Area parameter:
Min area sets the minimum size of a segment. If we want to represent small objects we would like a small min area. However, think about for instance the image of the orange. If we have a really small min area the white part in the middle could perhaps become segmented into a single object. Increasing the min area incorporates it into the orange half instead. See Figure 15.

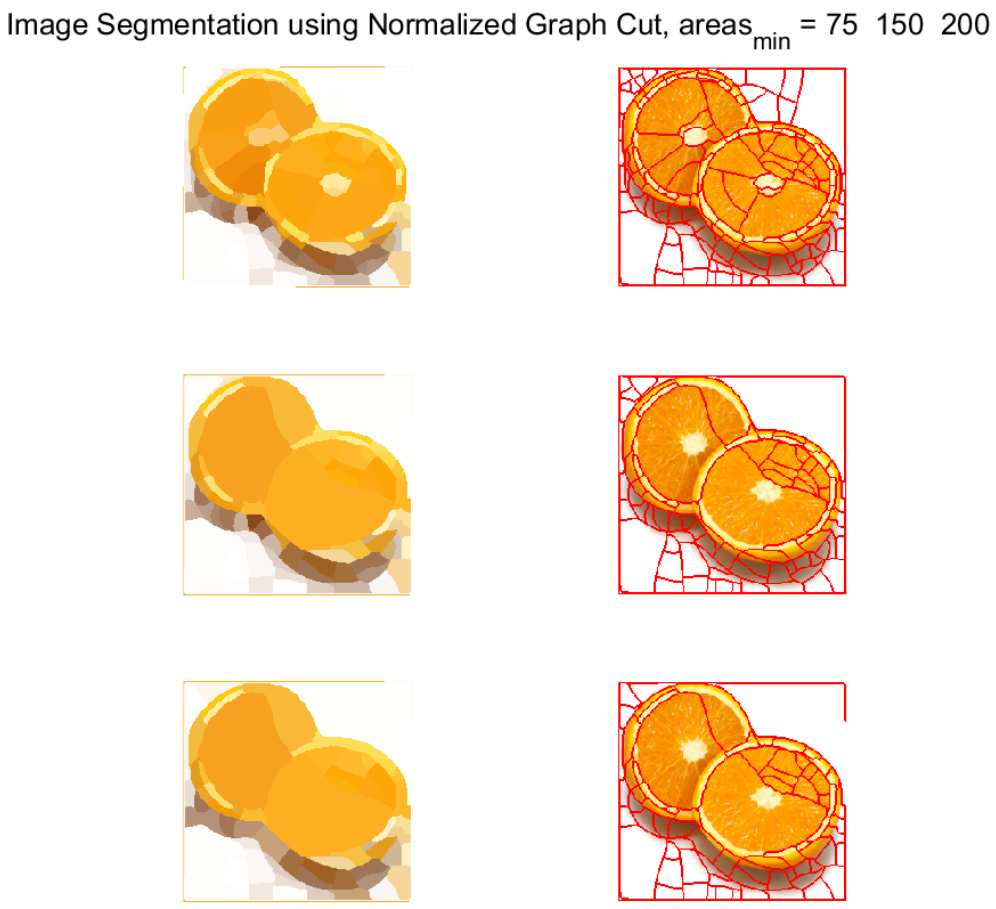


Figure 15: Normalized Graph Cut using different minimum areas.

The Color Bandwidth parameter:
In the code we compute the elements in the affinity matrix by looking at color similarities as well as distances between pixels.

Increasing the colour bandwidth should thus increase the spread of the Gaussian around the pixel in the color domain. Therefore, as we decrease the peak of the Gaussian we give less weight to the center pixel and more to surrounding pixels in the colour domain.

When we decrease the variance we get a Gaussian with a really high peak where most of the probability mass is centered right above the pixels own colour. Thus, it is almost like we are using the own pixels color. All in all, increasing variance means that we give surrounding pixel colors more importance, decreasing variance means that we care more about the center pixels own color.

Setting the Max Depth parameter:
Max Depth tells us the maximum number of cuts that we can do in the graph. The orange image consists of two objects that stand out from the background. Therefore, we do not need a deep depth to segment the image. Meaning that the very first cuts already have small values.

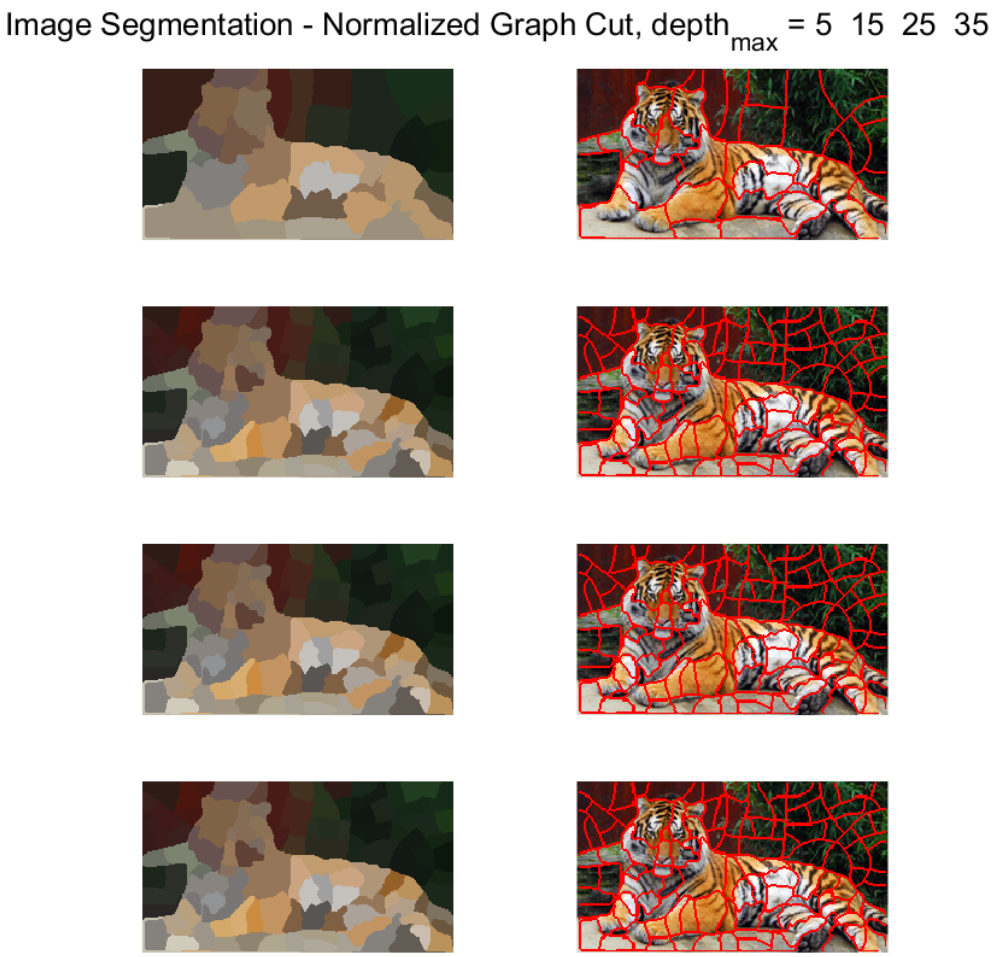


Figure 16: Normalized Graph Cut using different values for depth_{max}.

By analyzing the max depth parameter in the tiger image in Figure 17 we find that:

Depth 1: Here first with depth 1 the most obvious cut should be the one between the tiger and the upper background part. The tigers head is darker than its body and spatially it is closely located to the red background, which could explain why it is chopped off.
Depth 2: Next above the tigers head we see that the cut is made such that the wall becomes separated from the stone and the green plant which have more similarities in color compared

to the red wall.
Depth 3: Here we can for instance see that we separate the tigers head from the read wall.

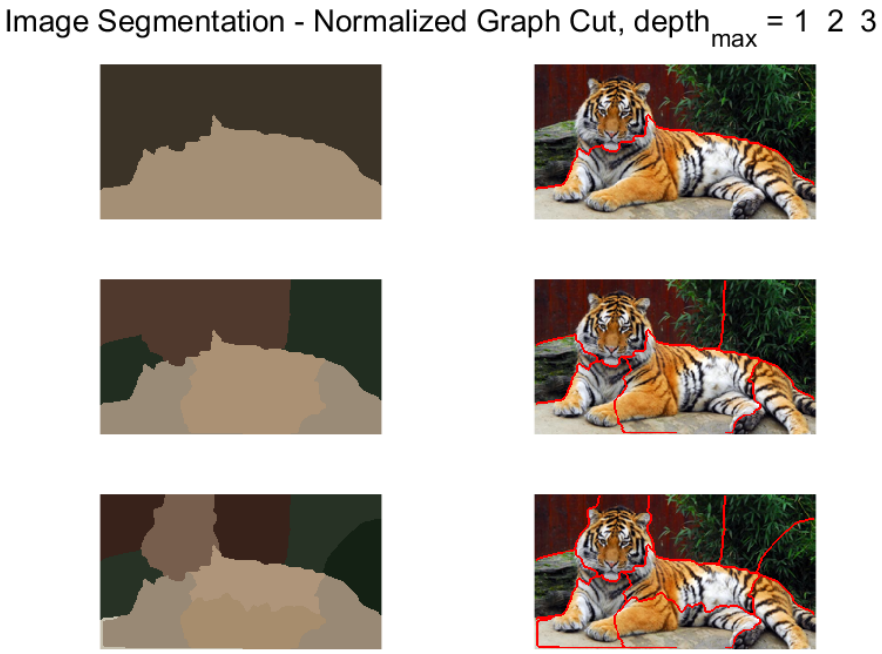


Figure 17: Normalized Graph Cut using different values for depth$_{max}$.

Setting a threshold for the number of cuts:
if $Ncuts(A, B) > threshold$ we do not make a cut. Therefore we stop when we do not find small enough cuts anymore.

The cutting threshold should be pretty low if we look at the edges of the orange compared to the white background. However, we need a higher threshold to make the cut in between the two oranges which have similar colors.

In the tiger image in Figure 18 we only make 1 cut when ncuts is equal to 0.01. Which seems like the most obvious boundary to the human eye and therefore this ncut gets a really low value. As we increase the threshold we make less obvious cuts, like for instance several cuts through the red wall in the tiger image.

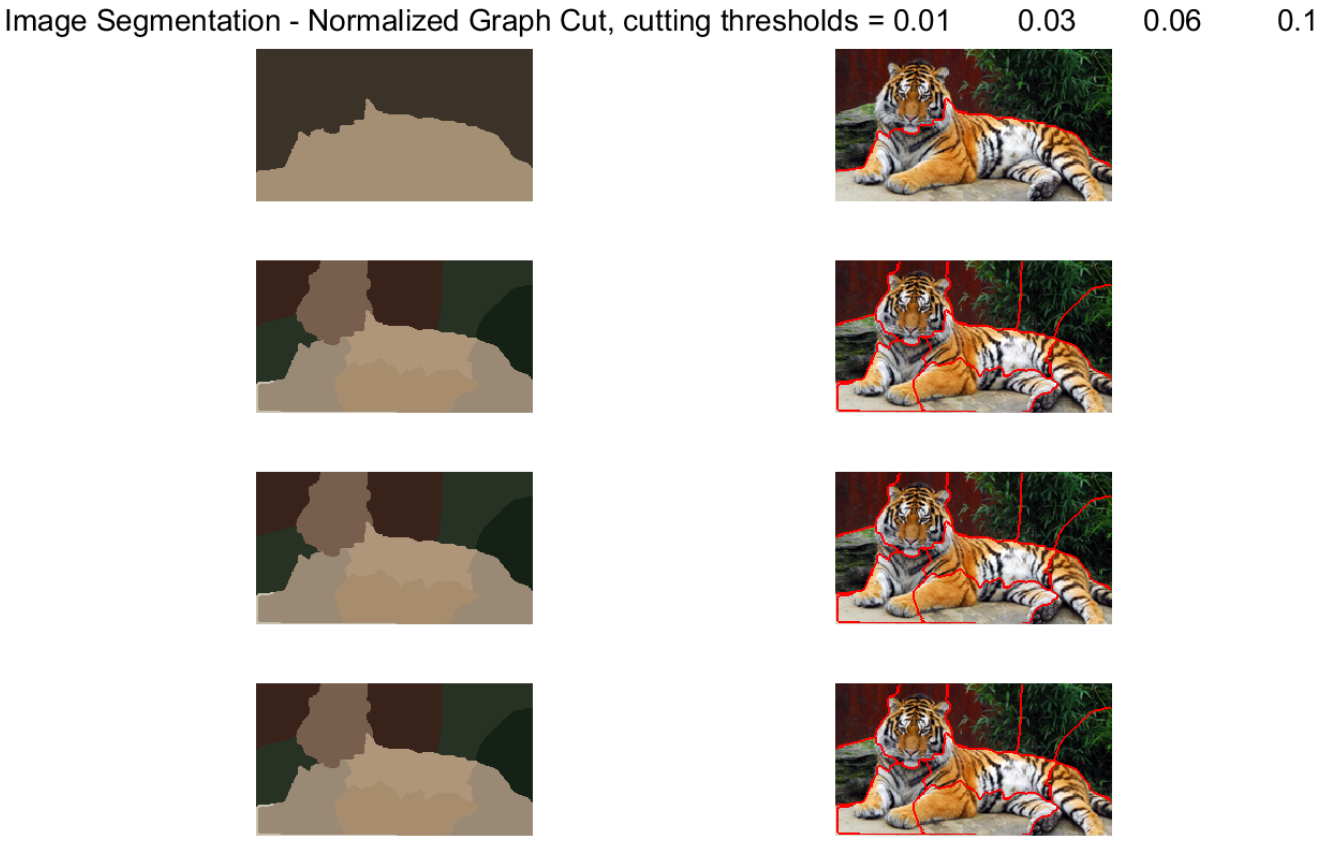


Figure 18: Normalized Graph Cut using different thresholds.

An image with preferred parameters:

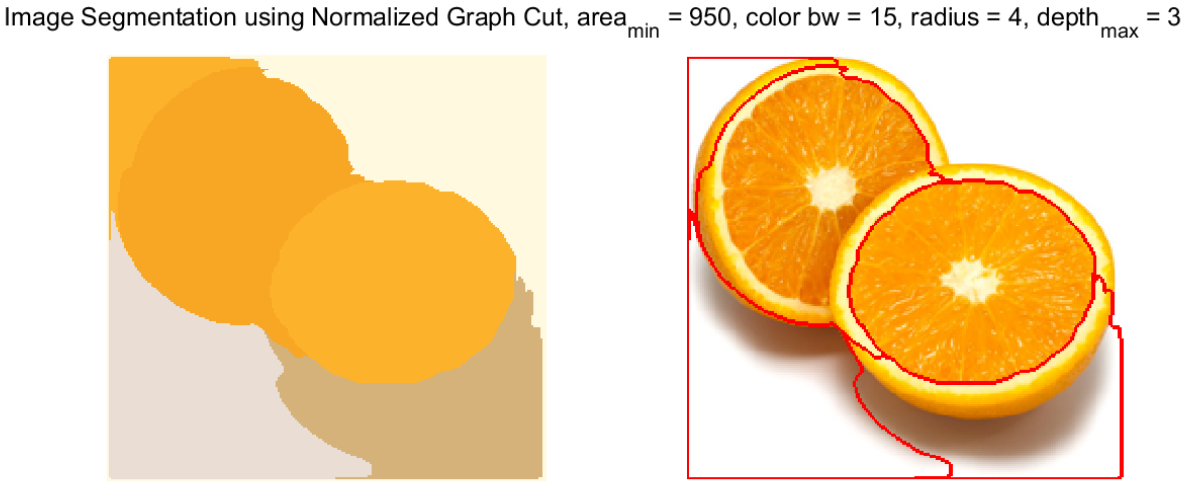


Figure 19: Normalized Graph Cut - best parameter setting for the orange image.

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

The number of segments are effected by min_area, n_cuts_thresh and max_depth.
Moreover, avoiding to much segmentation depend allot on the image. Overall, setting a good min_area for the image at hand and not setting n_cuts to high is important.

Furthermore, how much we can reduce min area and still get satisfactory segmentation depend on how large the objects in our image is. Larger objects requires lager min area. This is the case for the orange image.

However, even if the tiger is quite large we cannot get one large min area segment to fit it. Instead, as it has multiple colors, with complex shapes, it seems better to segment smaller areas that we later merge based on neighboring region similarities.

Moreover, a low n_cuts_thresh is good if we only want to make really low cost cuts in the graph. Therefore, if we decrease n_cuts_thresh we can be more certain that we are making good cuts. However, like in the case of the two orange halves, when we have two objects that are close both spatially and color wise we want to be able to make higher cost cuts to. Which could results in unwanted cuts at for instance the middle of one orange half.

Lastly, in the case of the orange using a high max depth could result in over segmentation in the image below. Instead it is better to use a lower max depth here. This depth is clearly enough to separate the two halves. However if we look at the next image of the tiger, if we use only a max depth of three the object has not become well separated from the background.

Question 9: Why does Normalized Cut prefer cuts of approximately equal size? Does this happen in practice? Try to increase the radius to include neighbouring pixels that are a bit further away from each other. This usually leads to a better segmentation, but at the cost of slower computations.

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

$$assoc(V) = assoc(A,V) + assoc(V,B) - cut(A,B) \tag{3}$$

By combining 3 and 2 we get:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(V) - assoc(A,V) + cut(A,B))} \tag{4}$$

We differentiate the expression and set it equal to 0 to minimize Ncut. We obtain the following:

$$\frac{dNcut(A,B)}{dassoc(A,V)} = \frac{cut(A,B)\cdot(assoc(V)+cut(A,B))\cdot(-2\cdot assoc(A,V)+assoc(V)+cut(A,B))}{assoc(A,V)^2\cdot(assoc(V)-assoc(A,V)+cut(A,B))^2} = 0 \rightarrow$$

$$cut(A,B)\cdot(assoc(V)+cut(A,B))\cdot(-2\cdot assoc(A,V)+assoc(V)+cut(A,B)) = 0 \rightarrow$$

$$-2\cdot assoc(A,V)+assoc(V)+cut(A,B) = 0 \rightarrow$$

$$2\cdot assoc(A,V) = assoc(V)+cut(A,B) \rightarrow$$

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

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

So in Figure 17 we can see the effect of getting approximately two equally sized cuts.

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

It depends on the size of the objects in the image. Large objects require a larger radius to find similarities between far away pixels.

When we decrease the radius, we take less neighbors into account for each pixel when we compute the affinity matrix, therefore we cannot find similarities between pixels that are far away from each other. For instance, in the orange image we have large objects and therefore a small radius cannot account for similarities between pixels at opposite ends of the orange half. See Figure 20.

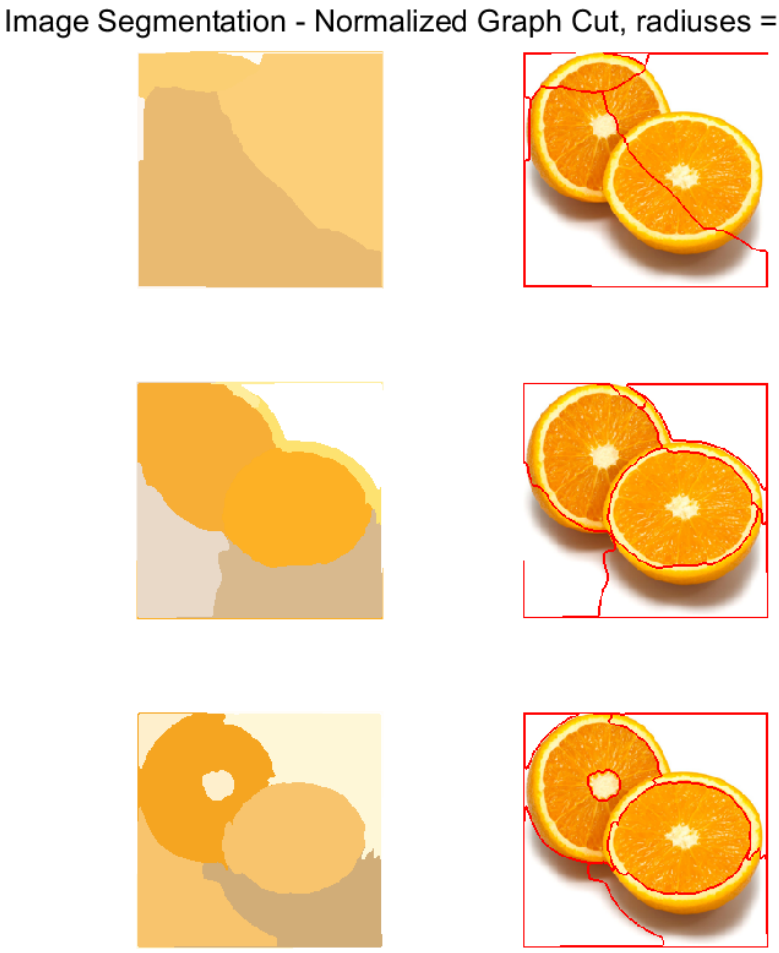


Figure 20: Normalized Graph Cut using different radiuses.

5 Segmentation using graph cuts using Mixture of Gaussians and Expectation Maximization

Graph theoretic methods are frequently used for figure ground segmentation. A segmentation is typically found by setting up an energy based formulation, where the energy depends on how pixels are divided into foreground and background pixels. Each such combination results in a particular energy and the segmentation is sought that minimized the energy. This gives us a tool for defining what can be expected from a good segmentation.

The theory for this section is quite extensive and can thus be find in the assignment description.

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.

$$e_{i,j} = \frac{\alpha \cdot \sigma}{|c_i - c_j| + \sigma} \tag{5}$$

First, α gives us the maximum edge cost for neighboring pixels. Since when $c_i = c_j e_{I,j} = \alpha$. Therefore, when we increase α, we say that we want to give more weight to neighbor pixels with similar colors. To make less cuts between these.

Below, in for instance the tiger image, when we increase α more of the grey area around the tiger's foot become part of the image foreground. Since we increase α, we are giving neighboring coloring more importance in the graph and these costs becomes costly.

Moreover, how edge cost decays for decreasing similarity between pixels is affected by σ. Meaning that when we increase σ the relative impact of similar colors becomes higher compared to the relative impact of dissimilar colors. Thus, we get a wider scale. Therefore, a low σ is good when we have clear edges like at the top of the tiger.

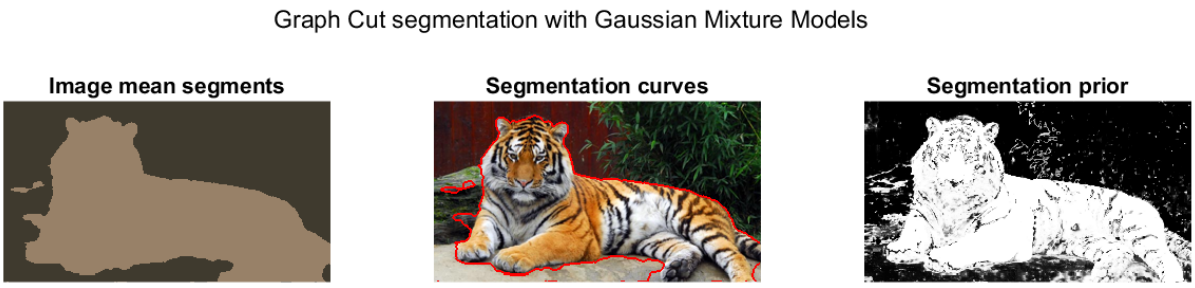


Figure 21: Graph cut using Gaussian Mixture models for a tiger image.

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

When an image has more colors more clusters K are needed to represent the colors in the image. Thus, when we have more colors in an image we need more Gaussians.

Example: First we use K = 6 and then K = 5 for segmentation of the "Tiger" in Figures 22 and 23. When we lower K below 6, we get considerably worse results. Which likely is due to the fact that we can no longer model all of the colors within the background and the foreground with so few clusters. Therefore, for instance the white leg of the "Tiger" probably becomes more similar to the fence, i.e. a background Gaussian model, then the models used to model the dogs colors.

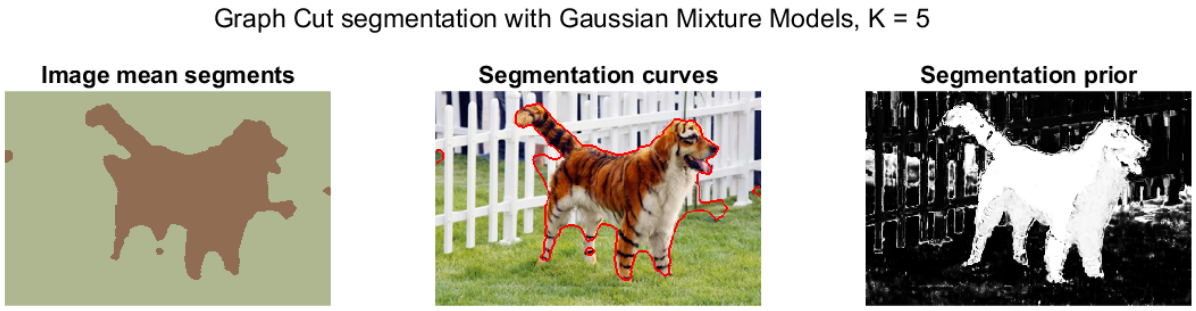


Figure 22: Graph cut using Gaussian Mixture models for a tiger image with K = 5.

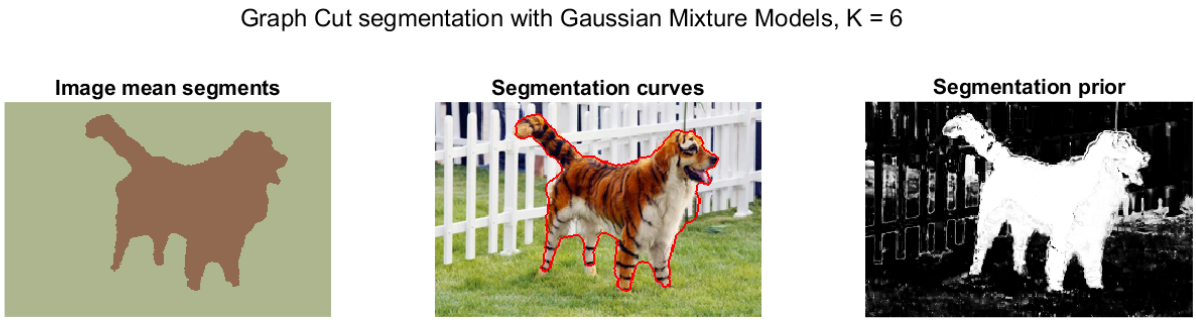


Figure 23: Graph cut using Gaussian Mixture models for a tiger image with K = 6.

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!

It depends.
Accuracy: This method clearly gives us significantly better accuracy then all of the previous models. But, annotating images by creating these rectangles would take a lot of work if it was done manually. Therefore, it depends on the accuracy that we would like in our segmentation.
Object shape: Lets say that we want to segment a picture of an object which results in a rectangle where most of the inside of the rectangle contains background pixels, but these background pixels are dissimilar to the rest of the background. Then most of the Gaussian's modeling the foreground will not model the object that we are looking for.

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!!

- All methods look at similarities between color and group pixels that are similar in color.
- In K-means the user specifics the number of clusters to be used in advance, thus it is parametric. Also we do not take location into account and therefore its segmentation can result in areas that are not spatially connected.
- In Mean-shift we do not need to specify the number of modes to use. We also take the location into account as opposed to K-means.
- In our implementation of Graph Cut we also specify the number of Gaussian's K to be used to model the colors in the background and the foreground of the image.
- Both Normalized Cut and Graph Cut build graphs based on pixel similarities.
- Both Mean-shift and K-means cluster pixels to mean centers and centers of maximum density.
- Mean-shift becomes slower than K-means since K-means is parametric and Means shift is not. Which means that we try to fit all the pixels to the model at once I K-means but in mean shift we instead cluster one pixel at the time.
- Graph Cut requires some prior information regarding which pixels belongs to the foreground and which belongs to the background.
- Normalized Cut also tries to segment clusters into equally large segments, which the other algorithms do no.