Answers to questions in

Lab 3: Image segmentation

Name: Arturs Kurzemnieks Program: Computer Science

**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:

For the initial cluster centers the best method seems to be selecting K random pixel values from the image itself. The alternative would be generating K random colors with no direct connection to the image in use. If the colors used in the image are biased and some colors are not really represented in the image, this has the drawback of potentially generating dead segments, where no pixels are assigned to the segment. For example, in the oranges image, if we do absolutely random initialization and one of the centers is initialized to blue color, it’s likely it will get no pixels assigned, it won’t be shifted anywhere after the means are calculated, so it will remain an empty class.

Ideally, some method could be devised to first look for K most distinct values in the color space the images uses.

In general it seems, the best approach seems to be sampling randomly from the image, so we avoid empty classes while not needing complex initialization.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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:

Depends on the complexity of each individual image and the chosen number of clusters K. Blurring the image also seems to make the convergence faster, as it starts to merge neighboring color values together, resulting in less color variety throughout the image.

It seems that as the means are recalculated, which are floating point values, there are pixels that just keep moving back on forth between the clusters practically forever (at least for the duration tests were run for this purpose), but those are minimal changes and don’t really affect the practically usable result, so we can look at some error criterion and say that convergence has been reached when no significant change in the results is no longer produced.

For the converge criterion the maximum absolute difference between two subsequent sets of center points was chosen, i.e. if no color channel value in any of the cluster centers hasn’t changed by more than 0.5 (colors are integers 0-255, so the decimal change doesn’t change the resulting integer anymore), we say that there’s no significant change anymore and it has converged.

Using *default* seed and the **orange.jpg** image, this yielded the following results

K = 2 converged at L = 2

K = 3 converged at L = 4

K = 4 converged at L = 4

K = 5 converged at L = 7

K = 6 converged at L = 7

K = 7 converged at L = 15

K = 8 converged at L = 17

K = 9 converged at L = 20

K = 10 converged at L = 20

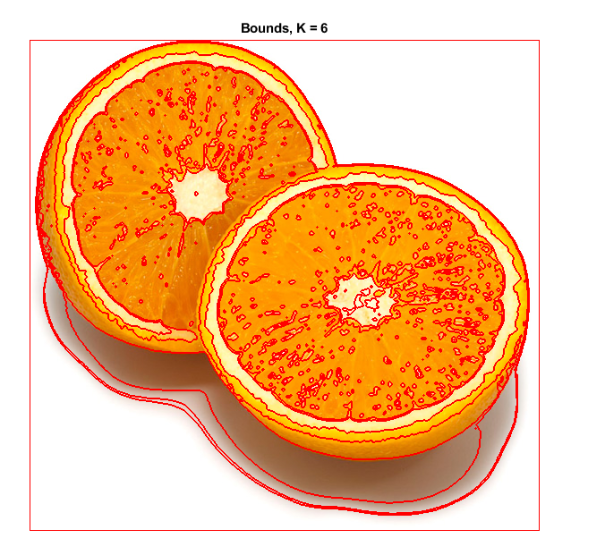
On more complex images with higher color variety the convergence would obviously be slower.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

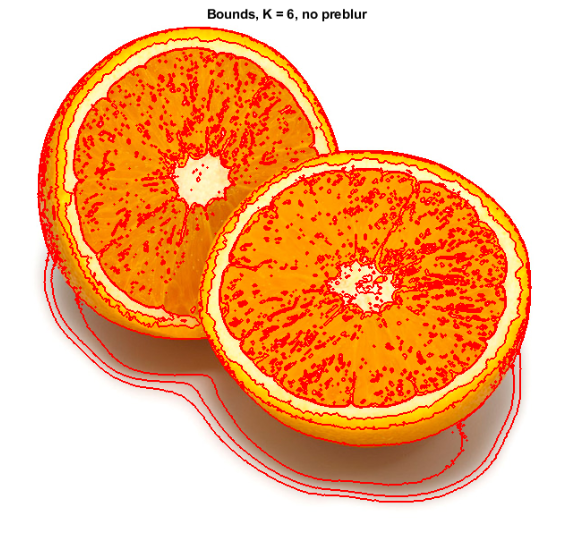
**Question 3**: What is the minimum value for K that you can use and still get no superpixel that covers parts from both halves of the orange? Illustrate with a figure.

Answers:

With lucky initialization, it comes quite close at even K = 6. For the orange image the problematic part seems to be the uppermost overlap point, where to color difference is very small.



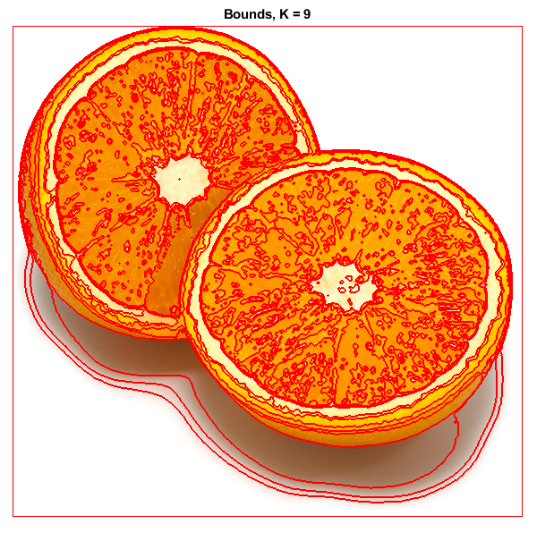
If no pre-blurring is done, no superpixel overlap can be achieved with this K.



And even with K = 5



With the pre-blur sigma = 1.0, it generally seems to be achievable at about K = 9 or K = 10.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Answers:

The **tiger1** image is easier in terms that the tiger (if the goal is to segment the tiger) is quite contrasted against the background, so similar parameters work well enough. Although there are still some areas where parts of the tiger do merge with the background.

A picture containing indoor, sofa

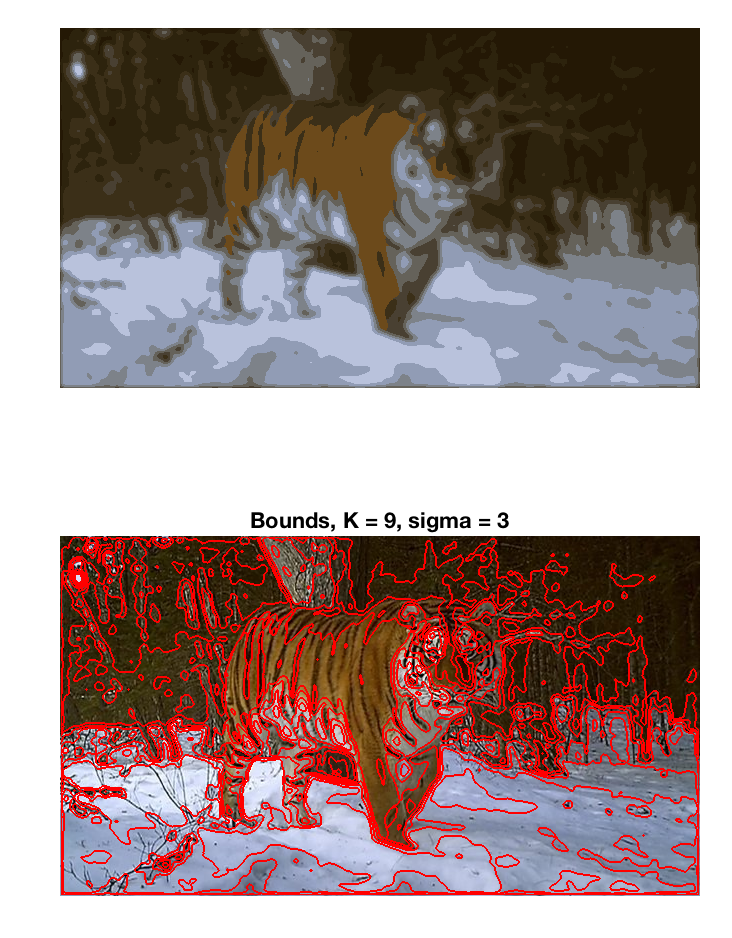
Description automatically generated

For **tiger2** is not as easy. A large K value can be used to avoid the tiger merging with background elements, but it gives very complex boundaries, which might be hard to really use practically.

A close up of an animal

Description automatically generated

For example, for K = 9, the tiger is segmented quite well, but the boundaries are very convoluted.



For these kind of images with small detail, it might be a better idea to use more pre-blurring, as it at least gives clearer boundaries, but of course a lot of shape information is lost as it merges with the background.

**Tiger3** is again more contrasted, but there’s quite large color variety, so a larger K is advised. Also, the leg on the right side of the image tends to merge with the fence. If that’s to be avoided, seems at least K = 9 is needed.

A dog standing next to a fence

Description automatically generated

A close up of a dog

Description automatically generated

K-means seems good for objects that are well contrasted against the background. For most of these images a decently good segmentation can be achieved using a K value smaller than shown here, if there’s tolerance for losing some parts of the shape to the background, as most of them have some specific regions that don’t differ much from the nearby background. If precise shapes are priority, blurring should be avoided, as it makes these problematic regions even harder to segment. These larger K values also give complex boundaries which might be quite hard to work with. That can be mitigated to some extent with more blurring, in the expense of losing precise outer shape of the object.

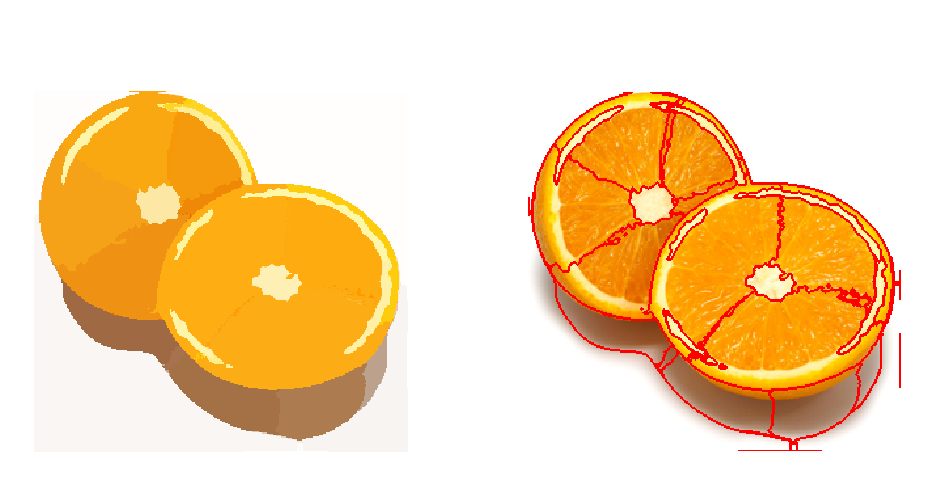
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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.

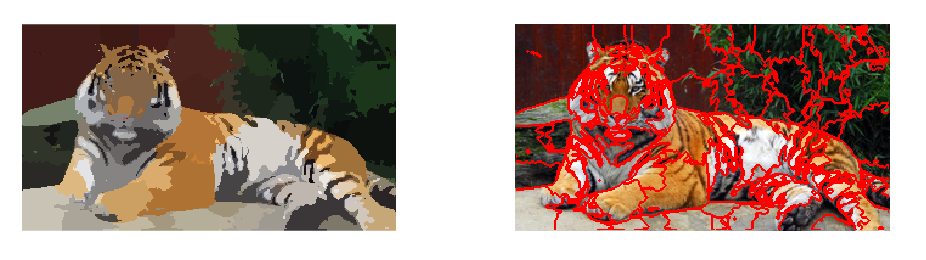
Answers:

A higher colour bandwidth widens the distribution of colours for the clusters, increasing tolerance for colour change. This tends to make the clusters less well defined shape-wise. There’s little growth spatially, but the shapes tend to round out, taking in some neighboring pixels.

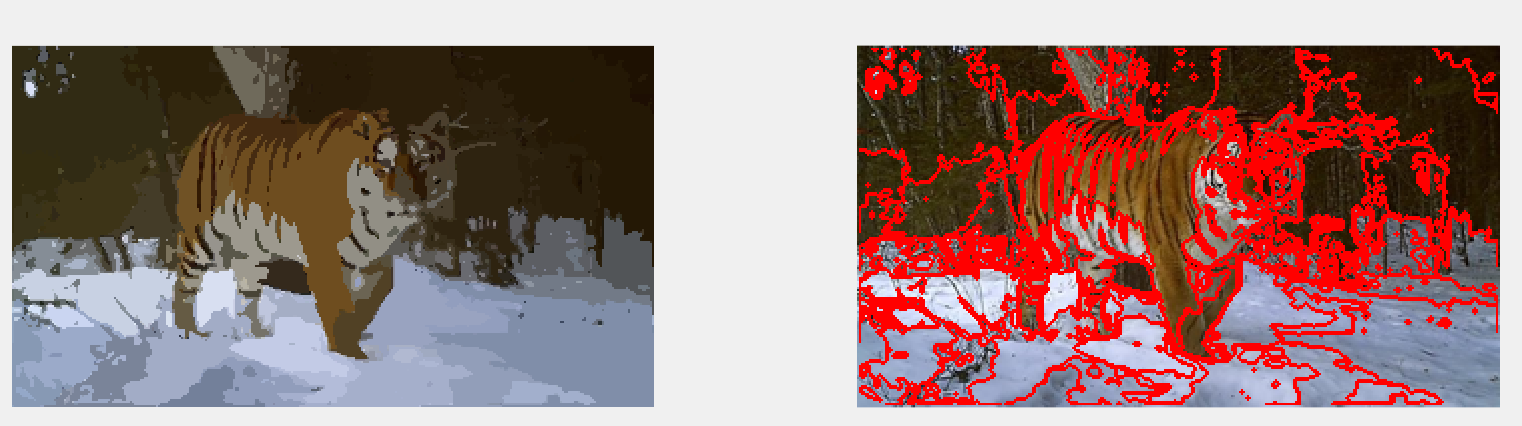
Increasing the spatial bandwidth, the clusters grow in size. Edges with sharp color change are preserved pretty well, with a low color bandwidth and high spatial bandwidth some clusters can actually be pretty small with complex boundaries, if there’s sharp colour change and they can’t grow out. But generally they tend to grow out, becoming more tolerant for spatial difference and taking in pixels with closer color values as the constraint on position becomes weaker and constraint and constraint on color remains the same.



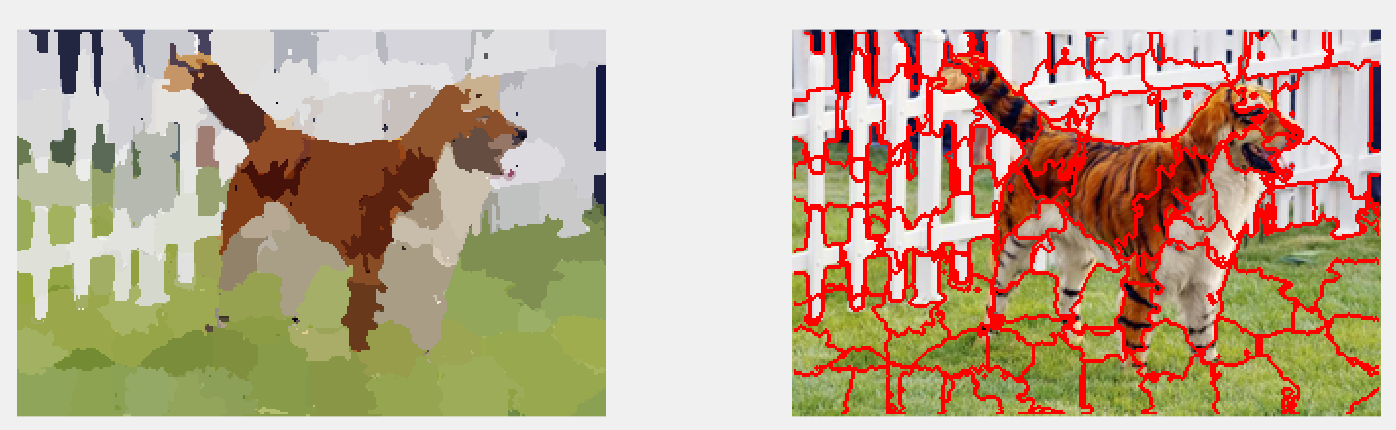
σs = 20.0, σc = 4.0



σs = 10.0, σc = 3.0



σs = 14.0, σc = 2.0



σs = 9.0, σc = 4.0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Answers:

K-means operates only with color information, ignoring spatiality, on a predefined number of clusters K. Clusters are initialized in the beginning, pixels are assigned to them solely based on the closest color match, cluster centers are updated and some pixels might switch clusters, but essentially it only refines the results. In the end all pixels in the image will be grouped based on color similarity, no matter where they are, so the resulting segments can be disconnected and all over the image.

Mean-shift takes into account spatial information. It doesn’t need a predetermined number of clusters, the algorithm ends up with some number of clusters based on the bandwidths used. Essentially, for each point the distance to other points is calculated and weighted based on a Gaussian kernel (for which the variances are based on the color and spatial bandwidths we’re giving the algorithm) centered on the point we’re currently looking at, therefore the larger the distance, the smaller the weight and the farther the other point is, the less it affects the shift of the point. The point is then shifted to the weighted average of other points. This is repeated for all points, and as the shift tends to shift locally closer points together due to the weighting, points flock to local peaks, which essentially give us the cluster. Therefore, the smaller bandwidths we use, the sharper Gaussians (with smaller variance) we get and the points are shifted only based on the other points that are really close, giving more total peaks and more small clusters. Spread out Gaussians on the other hand will result in larger clusters.

The other significant difference is that clusters in this algorithm will be connected, as the grouping of points in the high-dimensional space also contains the spatial information they have in 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:

Yes. Different images have different complexities and different points of interest. In some cases there might be one clearly defined foreground object that should be segmented from the background, in other cases there can be a complex scene with multiple objects of interest, color-wise poorly defined boundaries between them.

Therefore each type of image would need some finetuning of these parameters, as they directly influence the number (e.g. **max\_depth** to set how many times we cut) and size (e.g. **min\_area**, also **ncuts\_thresh** to decide how similar pixels can we group) of the segments and where the cuts are made.

A picture containing citrus

Description automatically generated

The following parameters were used for this segmentation:

**colour\_bandwidth = 22.0**

**radius = 8**

**ncuts\_thresh = 0.4**

**min\_area = 100**

**max\_depth = 4**

In this image we have two simple (yet overlapping) objects over a white and quite contrasting background, so less segments are desirable. For this, I chose to decrease the levels of recursion, as well as choose a somewhat smaller threshold to avoid too many cuts going over the middle of the orange slices.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Answers:

**Max\_depth** affects the subdivisions perhaps most directly. This was observed with the previous task. If the other parameters have been found to be suitable for the image, reducing **max\_depth** seems to give generally good results. It makes sense if we look at it the other way around, e.g. when the depth is increased, the already found segments are kept, only subdivided further, so if there are no bad segment edges at a larger depth, they won’t be there at a smaller depth as well.

**Max\_depth = 6**

A picture containing orange

Description automatically generated

**Max\_depth = 4**

A picture containing citrus

Description automatically generated

Choosing a smaller threshold **ncuts\_thresh** can also reduce subdivision, as it puts harder constraints on pixel similarity and cuts can’t be made as easily, but this can result in losing segmentation on some weaker edges.

**Min\_area** is also usable. If there are some small segments being found, increasing **min\_area** can avoid subdivision of those.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Answers:

As we have

where *assoc(A,V)* and *assoc(B,V)* are respectively total sum of connections from vertices in A to all other vertices in graph V, and total sum of connections from vertices in B to all other vertices in graph V, the algorithm can’t afford having the connection sum for one side to be particularly small, as that’d mean the sum of edges in the cut being proportionally large against it, resulting in a larger value from the fraction. This normalization is generally done to avoid cutting off single edge nodes. Naturally, for the sum of the fractions (which share the same numerator) to be as small as possible, the denominators should be as similar as possible. In this case it results in the cuts usually being somewhat similar in size. In practice it is not always the case, as there are other parameters in play and it also strongly depends on the image, but the general tendency is there that the images segmented with this method often have segments with similar areas.

For example:



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Answers:

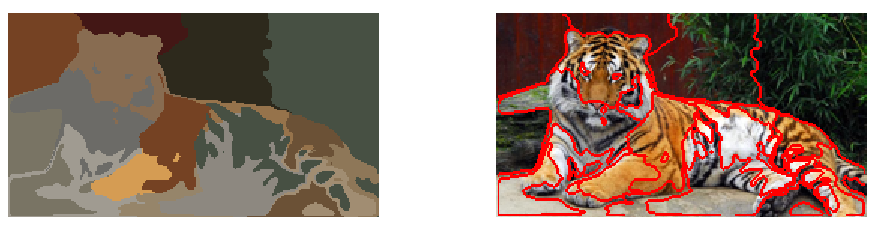
Radius = 5



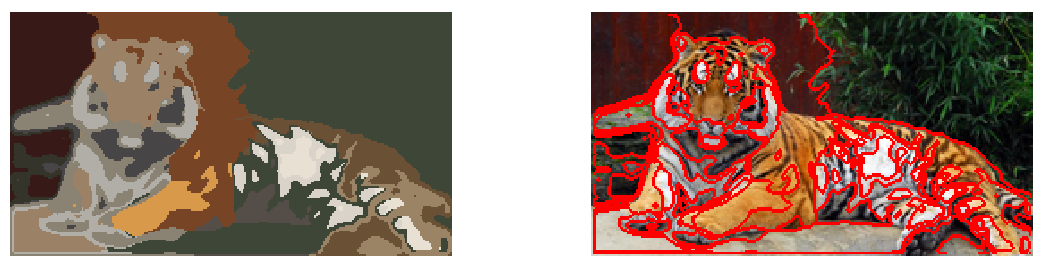
Radius = 10



Radius = 20



Radius = 30



Increasing the radius in general seems to give better results, as the cuts are in a sense more logical. Having a larger radius means having connections between pixels in a bigger region, reducing the effect of small local changes and taking more global features into account. A local divide which could previously result in a segment boundary might not do so with an increased radius, as the pixels on each side become “aware” of each other.

The main drawback is that an increased radius means an increased number of edges in the graph that have to be processed, this results in a huge performance hit.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Answers:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_