Answers to questions in
Lab 3: Image segmentation

Name: Panteleimon Myriokefalitakis 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?

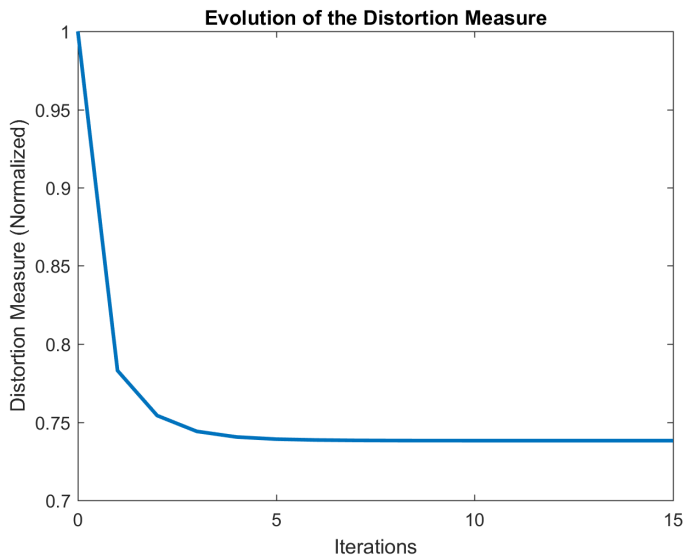
Answers:

K-means as being an EM procedure does not guarantee convergence to global optima. This makes the algorithm prone to the initialization of the cluster centers as it could converge to different local optima. Several methods have been proposed in the literature to overcome this issues. It seems that using heuristic/hierarchical algorithms results in clustering close to optimality. Nevertheless, these are usually computationally expensive or require careful implementation. So, we decided to overcome the initialization problem implicitly. We used the two more common and simple methods for cluster initialization. These are the Random Partitioning and the Forgy method. On the first one, we randomly divide the space in K clusters whereas on the second one we choose the initial clusters to be K randomly selected data-points. We found in most of the experiments that these two behave almost the same with the Forgy method to converge faster. Thus, we used this as initialization method and then, to gain optimality on the results we run the algorithm several times and we return the best results i.e. the run with the lower distortion.

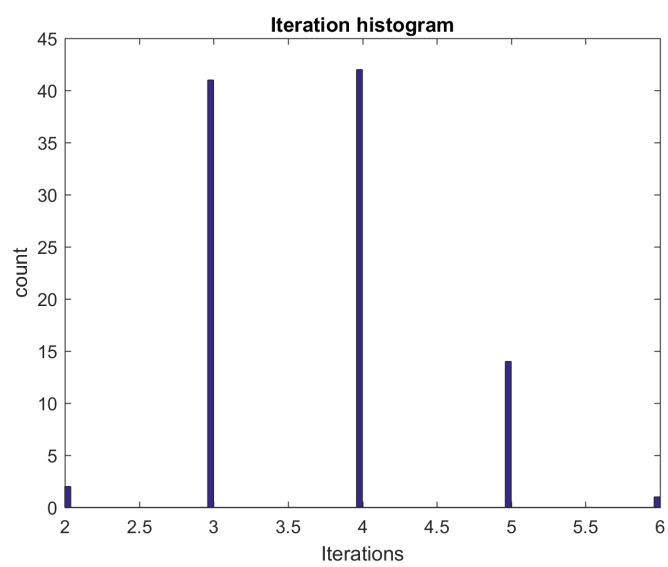
Question 2: How many iterations L do you typically need to reach convergence, that is the point where no additional iterations will affect the end results?

Answers:

In order to have a quantitatively measurement of convergence we computed the distortion measure. A random example for the image “tiger1” is shown in the figure below:



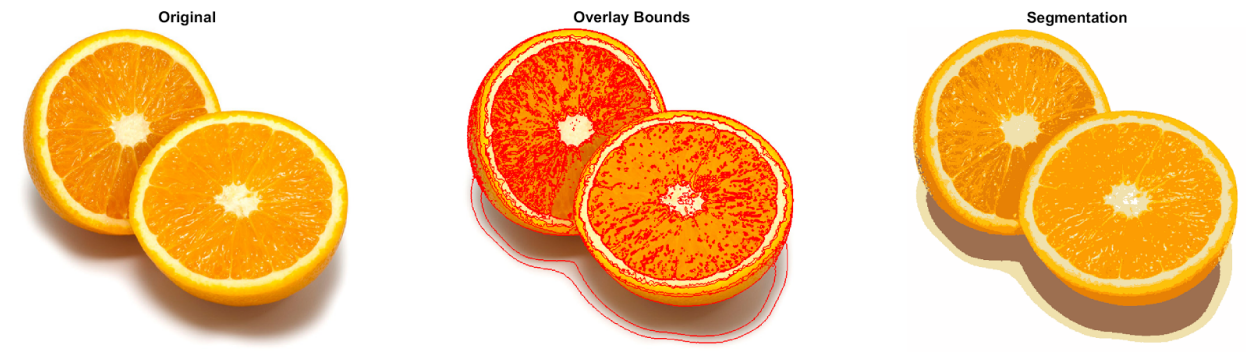
As we can see the algorithm converges fast enough i.e. in around 4 iterations. Nevertheless, in order to gain quantitative results, we set a threshold for convergence instead of setting a priori the number of iterations and we run 100 tests. The results are shown below and as we can see, our intuition about the number of iterations that are required for convergence was correct since we get an average of 4 iterations. Note that the k-means algorithm converges finitely to a local minima and this is one of its drawbacks.



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:

In order to get superpixels on the orange that does not cover both of the halfs we need to increase the number of K in values high enough that the extremely small difference between the color will result different superpixels and due to the spawn of very huge number of them we cannot infer if there is a true braking in the links or is due to line saturation. One way to overcome this is to let the parameter ‘image sigma’ to get values lower than 1. This way we increase the differences and we could get the superpixel distinction that we want. By doing that we used 7 classes to get appropriate results.



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

Answers:

For the orange we had an approximate uniform coloring object with its shadow in front of a white background, therefore we could segment the oranges when using 2 (including the shadows) or 3 classes. For the tigers this is not the case. The huge color variation on the object will cause the superpixels to brake. Also, k-means takes into account only the similarities between the colors of the pixel and thus it will cluster as the same group both parts of the object and the background. Furthermore, the fast change between colors in the object will result superpixels which cover small areas. A way to overcome this issue is to blur the image

beforehand. Nevertheless, we present here some results, that although they are cluster background and foreground in the same color clusters, we tried to create superpixels that include most of the object in them. Finally, we infer that when comes to multicoloring objects the k-means is not preferable to segment the object as it does not contain information on the pixel placement and thus will break the object to different segments based only on the coloring. Thus, k-means is not even close to optimal when comes to object segmentation. On the other hand is very useful as preprocessing tool to lower the colors of the image (vector quantization).

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.

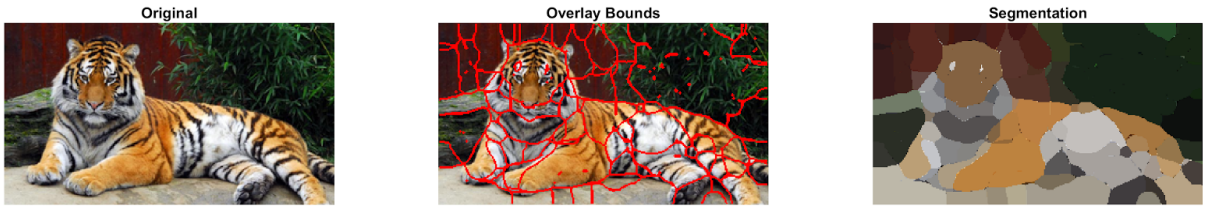
Answers:

The bigger the bandwidths, the bigger the gaussian kernel. Thus more data (information) is incorporated when the new mass center is about to be computed. Therefore, the bigger the bandwidths, the bigger the possibility of getting bigger area of the superpixels bue to the local minima escape. A more important aspect though is the ratio between the active radius of the spatial and color kernel. When the object is in high contrast with the background and the object has relatively small variation color-wise, a bigger window for the coloring is a sensible choice

and a low one for the spatial dimensions (like the orange image). When the contrast is low, the window should be kept small to be possible to group together neighbouring pixels. Finally, when we need to give more “attention” to the spatial closeness of the pixels the spatial bandwidth should be high.

For the tiger1 we have 2 types of backgrounds the top where more attention to the coloring is required and the bottom where the contrast is low and therefore less attention of the coloring is sensible choice (and vise versa for the spatial dimensions). On tiger 2 the upper and lower different backgrounds are inversed and now the contrast in the upper part is smaller. On tiger3 the contrast is high enough and with a combined information on the pixel position we can get really good results compared to the other tigers.

Bellow you can see a representative example for spatial and color radius of 6.5 and 30 respectively



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

Answers:

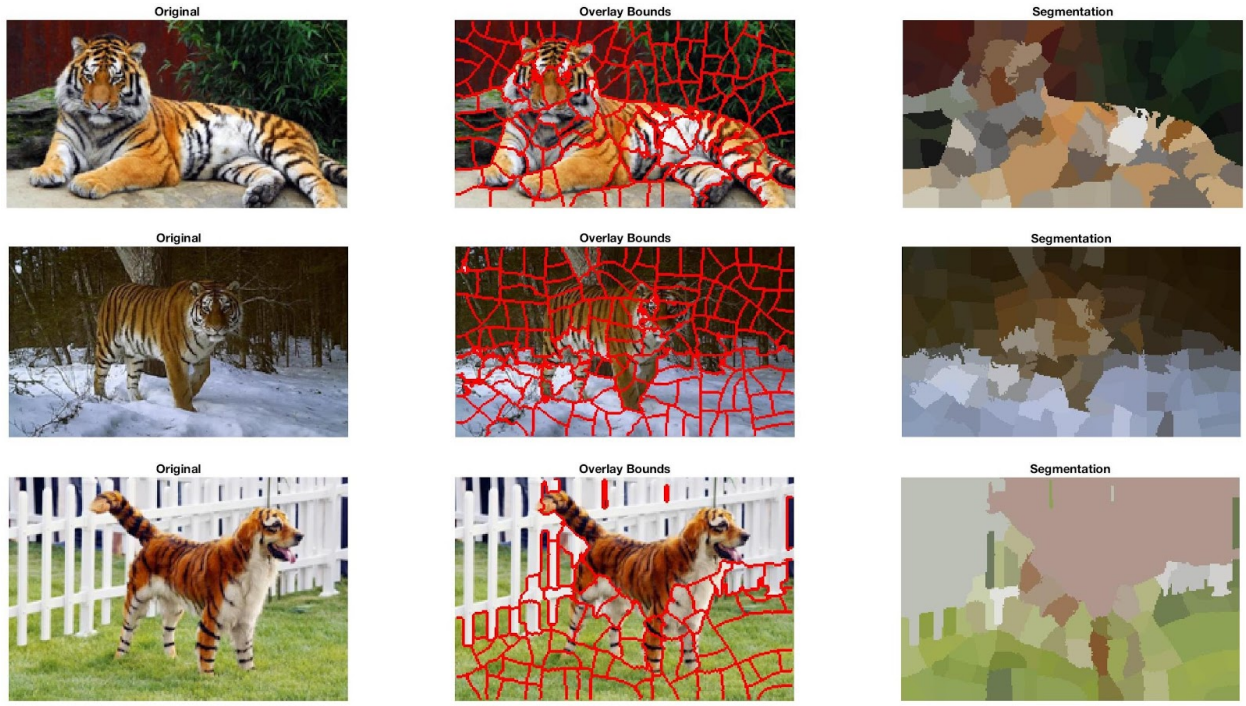
K-means and mean-shift are both unsupervised algorithms. They both depending on the inner density of the data-space and they guarantee to converge at a local optima. Especially in our case, the k-means does not incorporate spatial information whereas the mean shift does. K-means is a parametric method in the sense of needing definition of the number of clusters and this results completely different results for different K. In addition the sensitivity of the initialization is another drawback of k-means. Mean-Shift on the other hand is a non-parametric algorithm, so no prior information of the number of clusters required, but the number of segments i.e. the number of basins of attraction, depend on the window. The window dependency though, makes the algorithm sensitive to the predefined area of the window and in the case of 2 windows (color and spatial -like in our case-) lots of experiments required to optimize these parameters. Of course, this is one of the advantages of Mean-Shift compared to k-means as we can control the importance of the neighbouring while clustering but we cannot explicitly control the number of segments. Furthermore, the k-means is sensitive to outliers while the mean-shift is more robust. Also, k-means assumes equal covariance which yields spherical

clustering of the data-points. Mean shift does not have this problem but it does not scale well with the dimensionality of the feature space. Finally, while both algorithms are simple implementation-wise, Mean-Shift is more computationally expensive. To sum up, both algorithms are simple to implement and to conceptually understand and both are used in several applications, especially when vector quantisation (information compression) is in need. Those algorithms though are not optimal, in the sense of object segmentation. The reason for k-means has been introduced in the above questions (i.e. based only on coloring) and the Mean-Shift segments the object to several superpixels.

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:
It is not impossible to define parameters which will yield accepted results for different images, but this is not always the case and for sure this is not a rule. In general different images require different parameter setting and unfortunately in most cases these parameters will have to be defined through trial and error. There are many reasons for which each image needs different parameter settings. That is the color variation of the image and the relative contrast between different objects of it (which could be almost the same number in similar images), the state of the original image (i.e. noise, blur etc.), the size of the object of interest in the image as well as its inner contrast. Generally, the specification of the parameters depends the most on what information we want to acquire from the image. The figures below illustrate some of the results we got on the images in the course library using constant parameter settings.

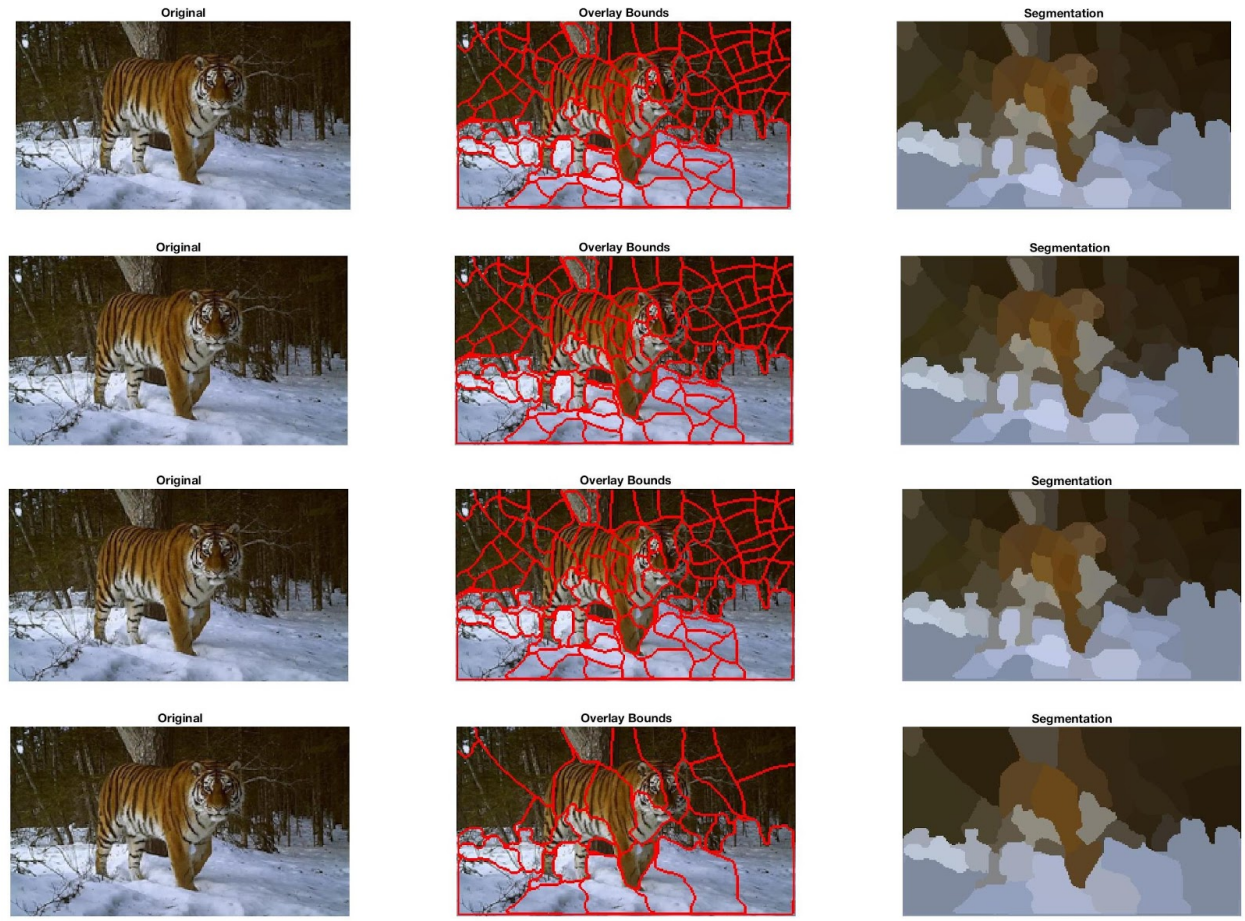
colour_bandwidth	**r**	**ncuts_thresh**	**min_area**	**max_depth**	**scale_fact**	**l_sigma**
60.0	4	20	200	10	0.4	0.00001



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

Answers:
The leading parameters responsible to control the recursive subdivision are the maximum splitting depth (max_depth), the cutting threshold (ncuts_thresh) and the minimum area of segment (min_area). The max depth specifies the depth of recursion and therefore the number of subdivision (if not another stopping criterion has been met). The cutting threshold is the upper limit (the largest Ncut(A,B) value) which will restrict the segment to further partitioning. This parameter ensures us that the process will not try to partition a segment that is highly likely to be an object in the image. Finally, the minimum area of segment determines the minimum number of points that a segment could have and thus we avoid to get too small partitions. Thus, although the max_depth explicitly controls the subdivision it's limitation to values <10 will result bad segmentation. The other two parameters control the subdivision as they pose restrictions on partitioning. Which of those is more important depends on the image content.

Generally, the bigger we set the cutting threshold and the larger we set the minimum area of segment, the bigger segments we will get. As expected, different images require different ncuts_thresh from the very first cut and unfortunately a proper setting which yields optimal results should be determined through trial and error. But, in any case we could say that if the segments are too small, we would like to increase ncuts_thresh or increase min_area.



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

Answers:

Minimizing normalized cut is proved to be equivalent to minimizing a Rayleigh quotient given by

$$mix_x Ncut(x) = min_y \frac{y^T(D-W)y}{y^T Dy}, \ subject\ to\ y(i) \in \{1, -b\}\ and\ y^T DI = 0$$

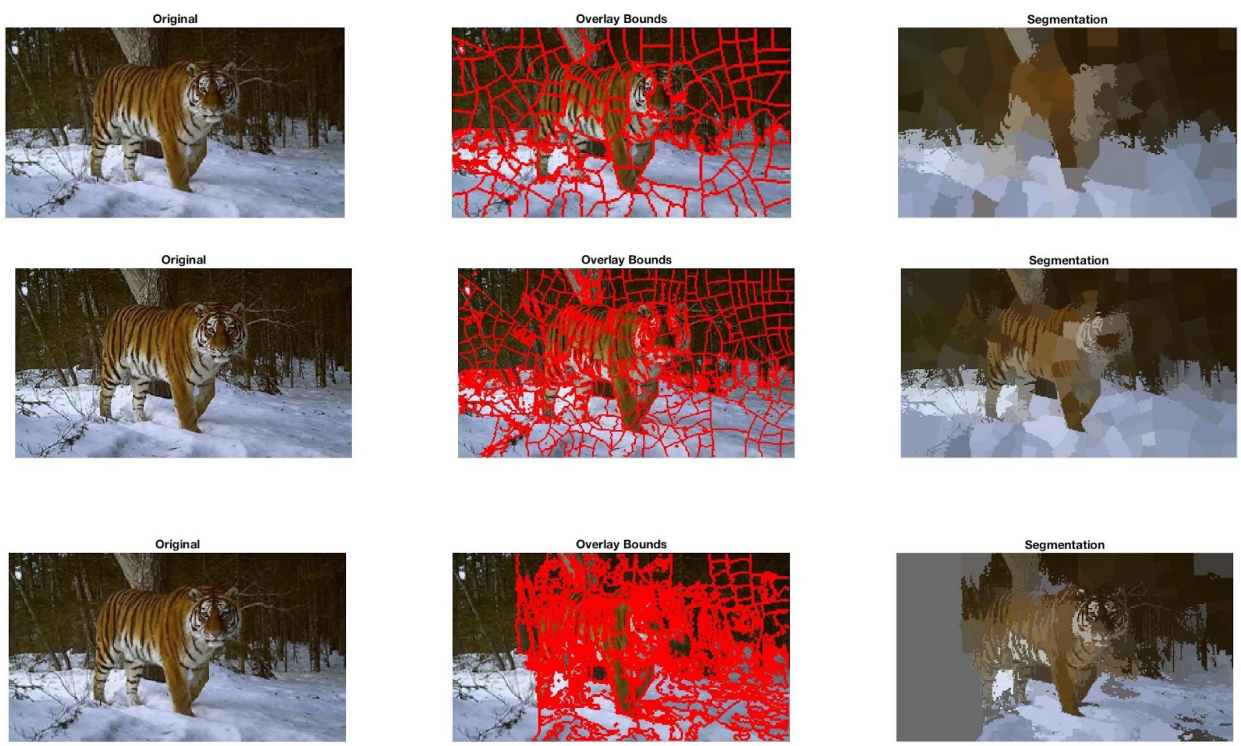
Therefore if *y* is relaxed to take on real values, we can minimize it by solving the generalized eigenvalue system

$$(D - W)y = \lambda Dy \qquad (1)$$

We know from Shi and Malik (2000) that the above quotient is minimized by the eigenvector y_i corresponding to the second minimum eigenvalue λ_i and this is the real valued solution to our normalized cut problem. Then, in our implementation, we convert the y_i from real to binary using as threshold the mean value. This yields, almost equal-size cuts -depending on the outliers and the symmetry of the distribution around the mean-. Note, that in the case of taking the median instead of the mean value, the splitting would be exactly in half but this results poor segmentation when comes to real images as groups of substantial differences do not impose equal areas.

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

Answers:
Indeed, the larger we set the radius the better segmentation we get as we include more information to the system. In general, setting *r* in smaller values the weight matrix becomes sparser. Thus it occupies smaller storage which implies shorter computational time. We tried an experiment (3rd figure) by setting r equal to 25 (which is high enough given our computers specification) and as expected the elapsed time was much longer (~563.5 sec) compared to r equals to 1 (~6.3 sec).
The radius in the figures below was set to 8, 14, 25 respectively all the other parameters remained constant.



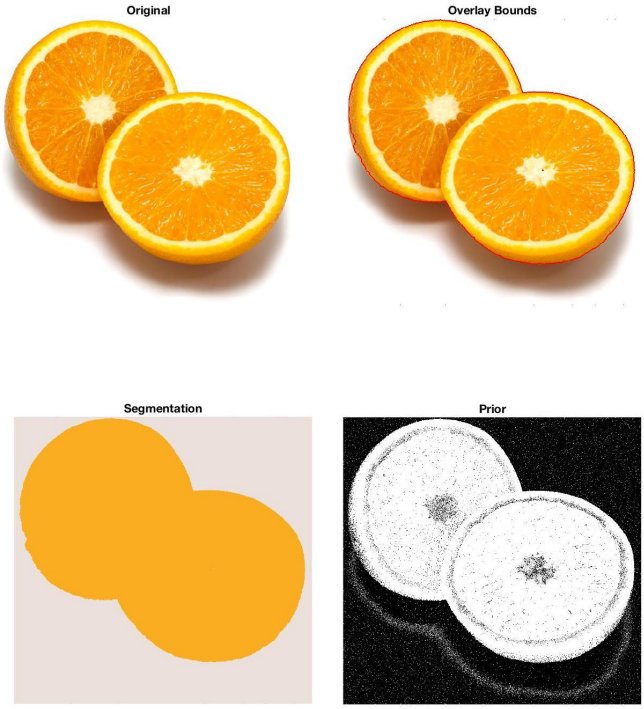
Question 11: Does the ideal choice of *alpha* and *sigma* vary a lot between different images? Illustrate with an example image with the parameters you prefer.

Answers:
In general, the proper choice of the parameters alpha (α) and sigma (σ) depends on the requested image and varies between different images. In order to get an idea about the role of these parameter let's interpret the equation which gives the pair-wise edge cost:

$$e_{i,j} = \frac{\alpha\,\sigma}{|c_i - c_j| + \sigma} \qquad (2)$$

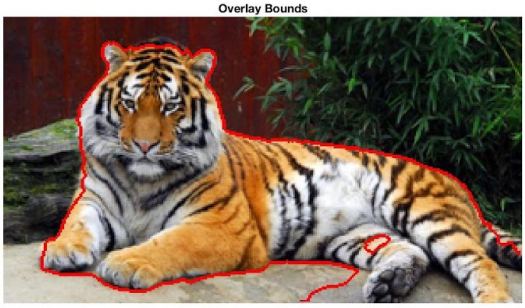
From (2) we can infer that σ control the strength of the similarity between the colors of two pixels which implies that the bigger the value of σ the stronger the under consideration similarity between them. As a result if we set it in high values it will result in more difficult segmentation between different (color-wise) adjacent pixels. Thus it is preferable to choose a value not too high (to allow further segmentation where needed) and at the same time not too small so as to not enhance the correlation between the colors of neighboring pixels at extrema. As far as the alpha parameter is concerned, it determines how strong the maximum connection between neighboring pixels should be (and thus the energy). Therefore, the higher the value of alpha the more difficult the connectivity will break. Keep in mind that we should not set it in too high values because it will not give to the fine details the proper attention.

In the experiments we did in the orange and tigers (tiger1,2,3) images we observed that indeed between the orange and tiger1 image we needed to set different values in the parameters to get satisfactory results (alpha=10 and sigma=4 for the orange and alpha = 40 and sigma = 5 for the tiger). However, concerning the tiger images we chose almost the same values (alpha = 40 , sigma = 5) and we still got good results. This is reasonable since the variety in colors in the object in these images is more or less similar.

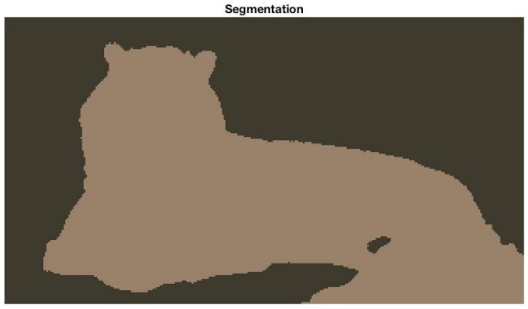

Original
Overlay Bounds
Segmentation
Prior

Original

Overlay Bounds



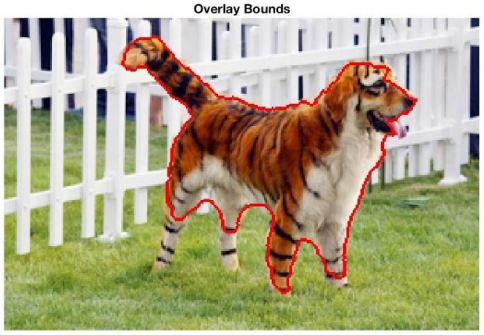
Segmentation



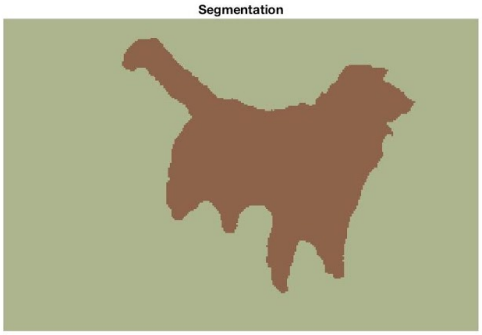
Prior

Original

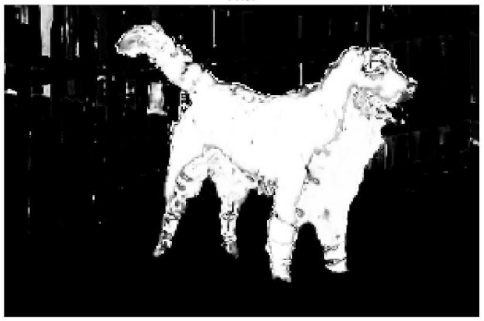
Overlay Bounds

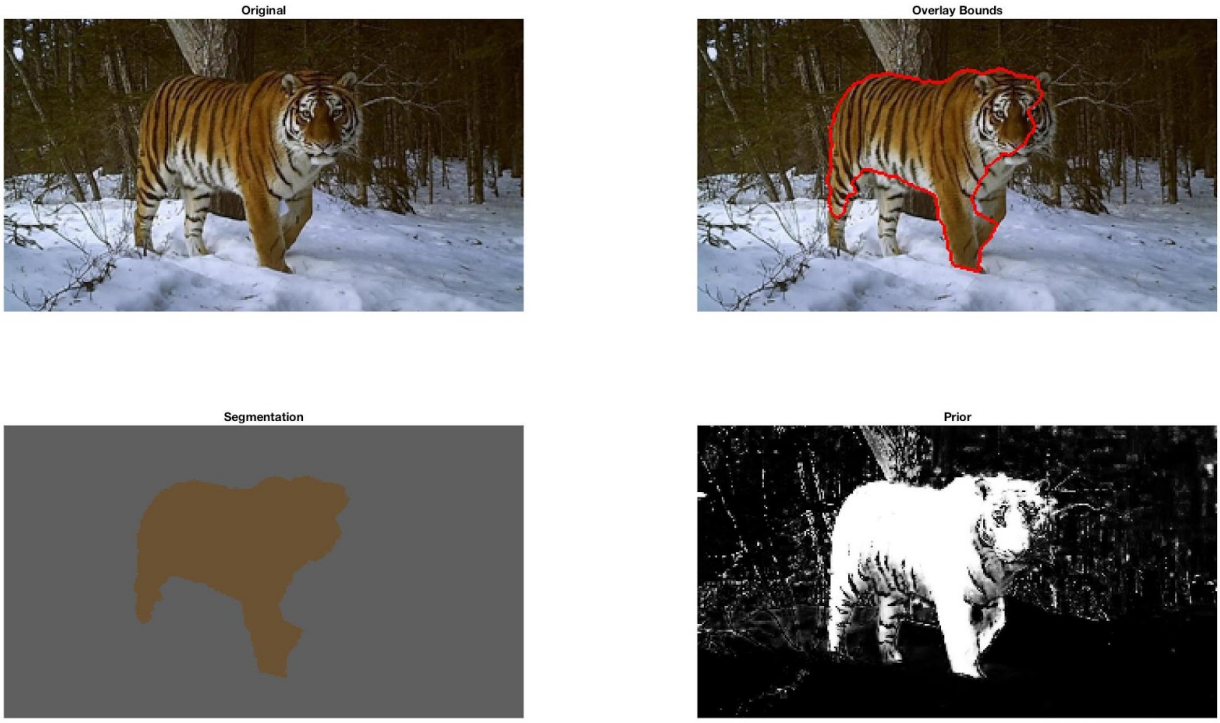


Segmentation



Prior





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

Answers:
To start with the orange, the lack of color multimodality results in incredibly good results even for K=2 as expected. On the other hand, the tiger's images include multimodal coloring in both the foreground and the background. Thus, the decrease of the GMM components will result in higher residual error from the true distribution when fitting. For the tiger1, the similarities between the bottom background area and the whitish colors of the tiger imply that more than 6 modes are required (as these are the basic colors of the image) and this is the hard limit that we found for this image experimentally. Following the same logic for the tiger2, we found a hard limit on 3 components, as below that limit, the algorithm fails to distinct the difference in the areas that resemble the background. Finally, for the tiger3, a hard lower threshold for the GMM components is 4, for the same reasons as above. Note that the experimental limits which mentioned above, were retrieved with optimal settings of the other parameters.

Question 13: Unlike the earlier method Graph Cut segmentation relies on some input from a user for defining a rectangle. Is the benefit you get of this worth the effort? Motivate!

Answers:

The results that we get with this method are without a doubt better than all the previous methods. Therefore it is definitely worth the effort! Nevertheless, we should note here that the clear difference in the optimality of the results originate in something deeper than the initialization of the parameters. When defining the rectangle, we do not just impose some constraints to the problem but we change the segmentation from unsupervised to supervised! The selected from the rectangle data-points are modeled (using GMMs) to fit the foreground distribution and the remaining to fit the background distribution. Then the weights from source to drain are proportional to the relative response of the nodes to the representative color distribution. Thus, we can no longer talk about unsupervised segmentation, and of course the results will be better as we incorporate a priori information to the system. Finally, note that the representative source and drain distributions could be trained on different images and this is what is done in practice.

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:

In this question the answer could be several pages, so we will just report here the key differences and only them. Note though that there are additional differences which differ conceptually, mathematically and experimentally.
To start from the last one i.e. energy based segmentation, it's main difference is the bayesian probabilistic approach which introduce a prior to the system and therefore makes it supervised, where the other ones are pure unsupervised methods. The K-means, is the only one that does not takes in count the spatial information of the image but it's bigger advantage is its conceptual and computational simplicity. Also, is the one that requires from the user to set the number of clusters (i.e. parametric approach) that will be created whereas in the other methods the number of segments is controlled implicitly. Mean-Shift resembles more the k-means method, and the differences and similarities between them are reported in gueston 6. The normalised cuts and the energy based normalized cuts, resemble each other in the formulation of the problem as they both use the linking between data-points as a measure of similarity/difference. All of the algorithms are trying to minimize some quantity. That is, K-means is trying to minimize the distances (or equivalent the RSD), normalized cuts the cuts (based on the affinity and color differences of the neighbourhood), energy-based the energy (represented by the similarities between the neighbourhoods and the priors) and Mean-Shift is following the gradient ascent which is equivalent to a minimization problem. All of them, do not quarante convergence to global optima and they are sensitive to hyperparameter initialization (in k-means the hyperparameters can be considered the initial cluster centers). In terms of performance, the energy based methods outperform the others when comes to semantic segmentation (as discussed above). Nevertheless, the other methods are better for vector quantization problems.

Thus, the optimality of each algorithm depends on the area of application. Finally, we should note that the exhaustive search for optimal parameters on the normalized-cuts and the mean-Shift make them cumbersome in use, as in some cases the hyperparameter optimization could be harder than the problem itself.