CS 484, Fall 2017
Homework Assignment 3: Image Segmentation

Due: December 18, 2017

Image segmentation is a fundamental problem in computer vision. The goal of this assignment is to implement the mean shift procedure to segment images using color information. The mean shift algorithm performs segmentation by clustering the pixels in a selected feature space. A nice property of this algorithm is that it does not need the number of clusters as an input, unlike many other clustering based segmentation algorithms.

Part 1: Implementation of the mean shift procedure

The first step is to implement the mean shift procedure. First, read the following papers:

> D. Comaniciu, P. Meer, "Mean shift: a robust approach toward feature space analysis," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 24, num. 5, pp. 603–619, May 2002
>
> W. Tao, H. Jin, Y. Zhang, "Color Image Segmentation Based on Mean Shift and Normalized Cuts," IEEE Trans. on Systems, Man, And Cybernetics – Part B: Cybernetics, vol. 37, no. 5, pp. 1382–1389, October 2007

Then, write a function that takes an RGB image and a set of parameters as input, and returns the segmentation result in the form of a label matrix (where each pixel is assigned the integer id of the region that it belongs to) according to the guidelines below.

As discussed in the papers above, the mean shift algorithm is a generic procedure for clustering a d-dimensional data set by associating each point in the data set to a peak (mode) of the data set's probability density. For each point, mean shift computes its associated peak by first defining a window of a given size at the data point and computing the mean of the points that lie within the window. The algorithm then shifts the window to the mean and repeats until convergence, i.e., until the shift amount is less than a threshold. At each iteration the window will shift to a more densely populated portion of the data set until a peak is reached.

The procedure takes an $n \times d$ matrix as input where each row corresponds to one of the n data points and each data point is defined as a d-dimensional feature vector. The parameters of the procedure include r, the size of the search window, and t, the threshold for the stopping criterion. The entry point to your solution should be a function with the following prototype:

$$[\texttt{labels}, \texttt{peaks}] = \texttt{segmentImage}(\texttt{image}, \texttt{params})$$

where `image` is the input RGB image, `params` is a vector that includes the window size and the stopping threshold, `labels` is the label matrix that represents the segmentation result, and `peaks` is an $m \times d$ matrix where each row corresponds to one of the m resulting peaks and each peak is represented by its d dimensional feature vector. This function should call the actual mean shift procedure that is defined with the following prototype:

$$\texttt{peak} = \texttt{meanshift}(\texttt{data}, \texttt{i}, \texttt{params})$$

where `data` is the $n \times d$ data matrix described earlier, i is the index of the data point for which we wish to compute its associated density peak, `params` is the parameter vector, and `peak` is

the resulting d-dimensional peak vector associated with i. The `labels` matrix stores the id of the peak that the corresponding pixel's feature point converted to.

Note that, the peaks should be compared after each call to `meanshift` and similar peaks should be merged. In your implementation, you can consider two peaks to be similar if the distance between them is $\leq r/2$. Also, if the peak associated with a data point is found to already exist in the `peaks` matrix, then its computed peak should be discarded and it should be assigned the label of the already existing peak in `peaks`.

Another important note is that, you can run the mean shift procedure in a spectral-only feature space where each pixel is represented by its color values (e.g., as a vector of length $d = 3$), or you can use a spectral+spatial feature space where each pixel is represented by concatenating its color values with its location (row and column coordinates) in the image (e.g., as a vector of length $d = 3 + 2$). For the latter case, you have to define two separate windows, one for searching in the spectral (color) domain and one for searching in the spatial (image) domain. This corresponds to defining two separate kernels with different sizes as described in the references given above. In this homework assignment, you can use the uniform (box) kernel. When you have two kernels, the search window parameter r will consist of two separate size values, one for each kernel.

Also note that, the mean shift algorithm is often two slow to run when every single pixel is a data point. You can use the following speedup in your implementation. Upon finding a peak, each data point that is at a distance $\leq r$ from that peak is associated with the cluster defined by the peak. This speedup is known as the "basin of attraction" and is based on the intuition that points that are within one window size distance from the peak will, with high probability, converge to that peak. You may also want to reduce the image size if needed.

In this part, implement the mean shift procedure for image segmentation as outlined above.

Part 2: Parameter selection

The next step is to run the procedure on a subset of images with different parameter values and choose a final set of parameters that will be applied to all images given in the assignment. Note that, the mean shift procedure uses the Euclidean distance metric for finding the points in a search window. Unfortunately, Euclidean distance in the RGB color space does not correlate well to perceived differences in color by people. Thus, the LUV color space is often preferred as the color feature representation in the literature. You can run the mean shift segmentation by first converting the RGB image into the LUV color space. After trying the code on several images, fix the parameters and use the same settings for segmenting all images.

Part 3: Performance evaluation

The output of the segmentation procedure implemented in `segmentImage` is an integer label image where each pixel stores the id of the region it belongs to. You are given a set of 20 RGB images selected from the Berkeley segmentation data set (https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/) to evaluate your implementation.

Qualitative evaluation, that will be used to determine the common set of parameters that will be used for all images, can be done by visual inspection of the segmentation results on a subset of images.

Quantitative evaluation, that will be used to generate performance statistics, can be done by comparing the boundaries produced by the segmentation algorithm with the boundaries drawn by a human. The main ideas regarding the evaluation procedure are outlined in Section 3 of the following paper:

D. R. Martin, C. C. Fowlkes, J. Malik, "Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26

The procedure in that paper uses soft boundaries, but we will use binary boundary images in this assignment. The particular performance criteria that will be used include *precision*, that represents the probability that an algorithm-generated (detected) boundary pixel is a true boundary pixel as

$$\text{precision} = \frac{\text{number of detected boundary pixels that are true}}{\text{number of detected boundary pixels}},$$

and *recall*, that represents the probability that a true boundary pixel is detected by the algorithm as

$$\text{recall} = \frac{\text{number of detected boundary pixels that are true}}{\text{number of true boundary pixels in the ground truth}}.$$

For a selected set of parameters, you are asked to compute precision and recall for each image in the data set. Figure 1 shows an example result. True boundary pixels are the ones that belong to the boundaries in the given human segmentation (stored in the file named `imageid_seg.png` in the data set) (see Figure 1-c). Detected boundary pixels are the ones produced by the segmentation procedure (see Figure 1-b). In order to be tolerant to small perturbations in the locations of the detected boundaries, you can dilate the ground truth boundaries (see Figure 1-d) before computing the intersection with the detected boundaries. The intersection result is the set of detected pixels that are true boundary pixels.

Overall, as part of qualitative and quantitative evaluation, you are asked to provide:

(i) For at least 5 different images, the segmentation results for different parameter values (you can show the segmentation results by overlaying the segment boundaries on the images; make sure that the boundaries are visible on the image).

(ii) For all images, the segmentation results for the final common parameter values.

(iii) Performance statistics (precision and recall) for each image (as a table).

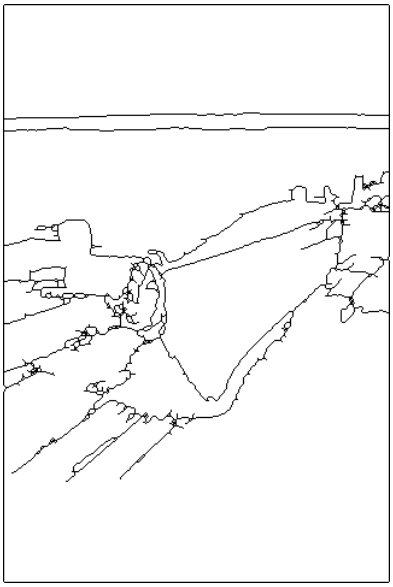
(iv) Illustration of how the performance statistics change for varying parameter values for the images used in (i) (as a plot of precision vs. parameter values and recall vs. parameter values) (note: you can fix the threshold for the stopping criterion but you should show different results by varying the search window size parameters).

(v) All evaluations in (i)-(iv) done separately when spectral features are used and when spectral+spatial features are used.

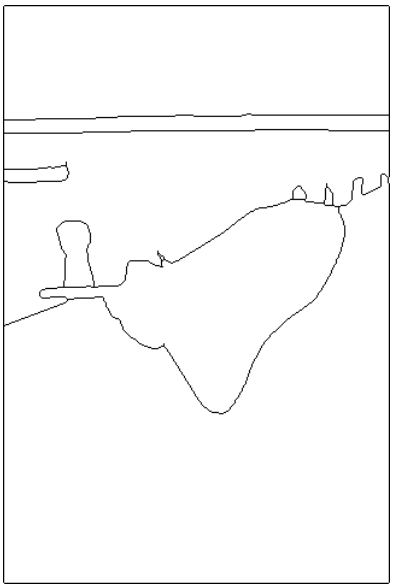
(vi) Discussion: based on your experiments and the resulting segmentations, answer the following questions:

- What effect does varying r seem to have on the resulting segmentations?
- What effect does adding position information as spatial features to color features have on the resulting segmentations?
- What are the advantages and disadvantages of using each type of feature vector?

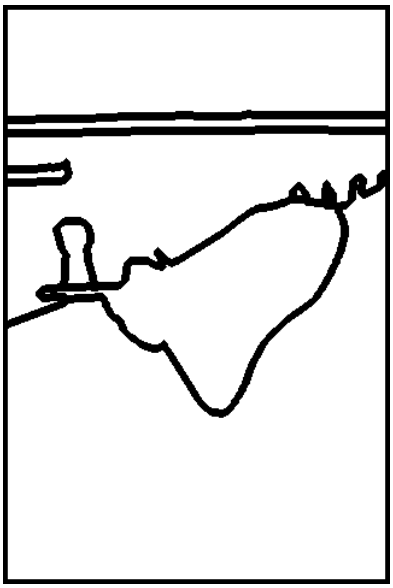
(a) Input image (`138078.jpg`)



(b) Output of mean shift segmentation



(c) Ground truth segmentation (boundaries from `138078_seg.png`)



(d) Ground truth segmentation, dilated

Figure 1: An example image, an example mean shift segmentation, and the ground truth segmentation to be used for performance evaluation. The segmentation boundaries are marked as black.

Submit:

1. Description of the implementation details.
2. Any source code that you wrote for the assignment (note: all code you use must be your own implementation, and you are not allowed to use code segments from other sources, including classmates).
3. All details asked in Part 3: Performance evaluation (items (i)-(vi)).

Notes:

This assignment is due by midnight on Monday, December 18, 2017. An important part of this assignment is the performance evaluation and the discussion of the results. You should upload your solutions as a **single archive file** that contains the **source code files** and a **single pdf file** that contains all the remaining details. Submit your solution using the online submission form on the course web page before the deadline. Please see the course syllabus for a discussion of the late homework policy as well as academic integrity. If you have any questions about what is allowed and what is not allowed in a solution, please check the course syllabus on the course web page.