

CS 216/295: Image Understanding
Fall 2008
Homework 3 Due at beginning of class, 11/18

Review: your class notes, papers on the website, and Forsyth and Ponce Chapter 14

Suggested reading: Forsyth and Ponce Chapter 16 (EM algorithm)

Submit your writeups and results for the following problems electronically via the EEE website "hwk3" dropbox. If you are not registered for the class, you can send me the assignment via email.

1. MRF: In this problem you will build an MRF model for segmenting an image into foreground/background and use an st-mincut solver to find the MAP assignment of pixels to foreground and background. You can download the code for the mincut solver from:

<http://www.ics.uci.edu/~fowlkes/class/cs216/hwk/mincut.zip>.

I have written a MEX wrapper which will let you invoke it from within MATLAB. I have compiled a version on my linux machine but you may need to recompile the code by executing the command

```
mex st_mincut.cc maxflow.cc graph.cc
```

in your own MATLAB environment.

You will need write code that loads in an image, computes the appropriate weights between neighboring pixels as well as the edge connections to the s and t nodes. The file `skeleton.m` contains the basic skeleton code you can start from in order to complete the problem.

Your code should display the image and prompt the user to click on two seed points (use `ginput` to get the mouse clicks). You should use these seed points to do two things. First, use the color of the pixels in the vicinity of the two points to initialize the foreground and background color models. Second, set the st connections so that these seed points are constrained to be in the foreground and background of the final solution (i.e. give them large weights)

- (a) Build a model in which the foreground and background are assumed to be constant color. Your st edges should have weights

which are proportional to the distance in RGB between the pixel and your foreground or background color. Connections between neighbors should be proportional to the difference in color between neighboring pixels. There is a free parameter (λ in eqn 1 of Boykov & Jolly) which controls the relative importance of the self edges and the neighbor edges in the graph. You will want to experiment to find a setting of this parameter that gives you a good segmentation result for the test image.

- (b) Modify your model to use your texture/color feature code developed in the previous assignment in order to build foreground and background models based on texture histograms. For each pixel, you can define the self weights by comparing the histogram in a window around the pixel to your foreground and background models. Keep the neighbor edge weights based on color differences as before.
- (c) Bonus: Once you have found the MAP estimate of the pixel labels, you can go back and reestimate your foreground and background models based on these assignments and then reiterate. Implement this iterative approach and see if it improves your segmentation results. Note that this is similar to Lloyd's algorithm for k-means but now with a spatial prior built in to the cluster assignment step.

You should submit your code for both parts along with example results of the segmentations produced from the test images `segtest1.jpg` and `segtest2.jpg` in the website hwk directory as well as any others of interest. Make sure and use the `plot` command to plot 'x's on the original image indicating where your seed points were for each example. Comment on the successes and failures of this approach.

2. Normalized Cut: Implement the spectral relaxation to normalized cut we discussed in class. As you'll recall, given the weight matrix W , you'll first compute the normalized affinity $N = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ where D is a diagonal matrix whose entries are the row sums of W . You will then need to find the k leading eigenvectors of the matrix N (use the matlab function `eigs`). Take the resulting set of eigenvectors and cluster them using k-means where the feature vector is given by the entries of the eigenvector (i.e. the i th component of the l th eigenvector gives the l th coordinate of the i th pixel).

- (a) Use the code on the website

`http://www.ics.uci.edu/~fowlkes/class/cs216/hwk/ncuttest.m`

to generate a set of data points in R^2 . Use the squared exponential kernel $W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}$ to compute the entries of the similarity matrix W which define the edge weights between each pair of points. The points are listed in order of the 3 clusters so you should be able to see the nice block structure in W . Compute the leading 3 eigenvectors and plot them as functions of the data point index. Are they piecewise constant? How does varying the parameter σ affect the eigenvectors? Run kmeans and indicate the result by plotting the points in the three recovered clusters using different symbols. Are you able to recover the original clusters? Compare the results of the spectral clustering (k-means on the eigenvectors) to running k-means on the raw data points x themselves. How do the results of the two clustering algorithms differ? How do you account for the (hopefully superior) behavior of the spectral clustering algorithm?

- (b) Build a W_{ij} matrix for an image where the entries are given by the squared exponential kernel applied to pixel color. You should only compute W_{ij} for pixels which are neighbors in the image (assume all other edges in the graph have 0 weight). Since there are a large number of pixels, you will either have to limit yourself to very small images or else use MATLAB's `sparse` function to build a sparse matrix. Compute the leading k eigenvectors of the normalized affinity matrix. Reshape each vector into an image and visualize it. Run k-means on the set of eigenvectors in order to produce segmentations of the two test images.

You should submit your code for both parts, the plots of the eigenvectors for different values of sigma (as 1d plots for the point set problem and shaped as grayscale images for the image problem) and the clustering / segmentation results.