CS 216: Image Understanding

Homework 3, Spring 2014

Due: a PDF file containing your writeup and result images and a zip file containing your code should be uploaded to the EEE dropbox before 11:59pm on Friday 5/9/2014.

Reading: your class notes, papers on the website, Forsyth and Ponce Chapter 8,9 and Szeleski Chapter 5

1. Color quantization: Although k-means clustering on color alone doesn't provide very satisfying segmentations, it can be used to reduce the color palette of an image. Write a script which takes a color image and a value for k and returns a new version of the image which uses only k distinct colors. Your code should cluster the pixel values using k-means and then produce a new image where each pixel is replaced with the color of the closest cluster. Demonstrate your code on a (colorful) image for different values of $k = 2, 5, 10$. Describe what would happen to the clustering if you scale one of the feature coordinates, say the R (red) value by a factor of 100? You can test this by first multiplying the R value by 100 and performing the clustering and then dividing the R value in the result image by 100 prior to displaying it.

2. Filterbank: In order to analyze texture, we would like to describe the image in terms of distributions of filter outputs. Implement a filter bank function which takes an image as input and filters the image with Gaussian derivatives in the horizontal and vertical directions at 3 different scales $\sigma = 1, 2, 4$. Also create an additional center surround filter by taking the different of two isotropic Gaussian functions at two different scales, e.g. $G_2(x, y) - G_1(x, y)$ and $G_4(x, y) - G_2(x, y)$.

   Your filterbank function should take as input one grayscale image and return 8 filter response images. Feel free to use the tricks we've discussed for making this process fast such as separability. Submit your code and include an image in your writeup which shows your 8 filter kernels.

3. Filter Distributions: Now convince yourself that the distribution of filter outputs really captures something about the texture. Use the

image:

`http://www.ics.uci.edu/~fowlkes/class/cs216/hwk3/zebra_small.jpg`

Select a image patch (say 40x40 pixels) over the zebra's neck and compute the mean absolute response of each of the 8 filter responses in that region [1]. Compute the mean absolute responses of the filterbank channels for similar regions in the tree leaves above the zebra's back and also on the grass in front of the zebra. Print out or visualize (e.g., using the `bar` function) the three mean response vectors for these regions and explain the differences you see in terms of your filters. Important Note: you may need to experiment with resizing image to find the best scale at which to capture the texture. In your writeup, explain how you chose the scale.

4. Textons: A nice way to represent the joint distribution of filter responses is to vector quantize them. Use your same k-means procedure from the first problem to quantize your filter bank output (now your feature vectors will be of dimension 8 instead of 3) using $k = 20$. Rather than output the cluster centers, create an image where each pixel is assigned its cluster label $1 \ldots 20$. Submit the resulting texton image (use `imagesc` and `colormap jet` to visualize the labels similar to figures we showed in class).

5. MRF Segmentation: 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/hwk3/mincut.zip`

   There are precompiled mex files for different platforms but you may find it necessary to recompile using the `GCMex_compile.m` script.

   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 `hw3_demo.m` contains the basic skeleton code you can start from in order to complete the problem.

[1] If you don't take an absolute value then the positive and negative components will tend to cancel out and give you mean 0

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 the paper by Boykov & Jolly linked on the course website) which controls the relative importance of the st 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 or color feature code developed in the first two questions in order to build foreground and background models based on histograms. You can either build a histogram of quantized colors (use $k = 20$) or a histogram of quantized filter responses (textons) or both (by just concatenating the two histograms together). For each pixel, you should compute a histogram of the quantized colors/textons that occur in a square window centered at that pixel. You can then define the s/t weights by comparing the histogram for that pixel with the foreground and background models. Keep the neighbor edge weights based on color differences as before.

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` included in the zip file. Also include at least one other image where you get a nice looking result. 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.

6. Project Proposal:

Put together a short description of a proposed project for the class and include it in your writeup. I will post some links to ideas and papers on the course website but feel free to suggest something else which interests you. You will need to be careful to choose something which is interesting enough to impress and intrigue your classmates (not too trivial), but also that you can successfully implement in a few weeks time (not too big).