CS 378 / CS 395T Computer Vision
Fall 2007
Problem set 2
Due Thursday 10/11/07

Submit both as hardcopy and electronically through the turning program. If you have multiple files, tar them together, and use your first initial and last name to name the tar file. Then submit with:

turnin --submit svnaras pset2 <yourfilename>

1. Lining the football field (25 points)

Write a program that detects the significant lines in an image with the Hough transform, and apply it to the supplied image named 'field.jpg'. Your program should use Matlab's built-in implementation of the Hough transform to do this. There are three relevant functions: 'hough', 'houghpeaks', and 'houghlines'.

First read in the provided image, then convert it to grayscale. Convert the grayscale image into a binary one by detecting the edges (use the 'edge' function again—the 'canny' option is fine). Give these edges as input to the 'hough' function. Use the resulting accumulator array (H) to detect some number of peaks in the votes using the 'houghpeaks' function. The number of peaks is a parameter for the program (NUMPEAKS); experiment with a few values for this parameter based on the content of the image and choose something suitable. Finally, compute the edge segment locations based on these peaks: give the edge image and the THETA and RHO parameters returned by 'hough' to the function 'houghlines'. Display the strongest fitted lines on top of the original color image.

Start with the built in defaults for all parameters above. What could be improved here? Make some adjustments to the edge detection and/or accumulator array parameters to improve the results in some way. *Briefly explain what changes you made, what impact they have on the final result, and why.* Display the improved fitted lines on top of the original color image. Include both the original and improved figures with your submission.

2. Segmenting an image into regions (75 points)

For this problem, you will write a program to generate image regions by clustering pixel-wise color and position features.

Implement the k-means algorithm (refer to Forsyth and Ponce p. 316-317 for the algorithm outline). The clustering function should include a limit on the number of iterations that will be run, as well as a precision threshold specifying how close a cluster center and its updated position should be in order to consider that cluster unchanged from iteration *t* to iteration *t+1*. The method should stop iterating either once this convergence criterion is met, or when the maximum number of iterations is reached,

whatever happens first. Write this function so that it accepts the number of clusters (*k*) as an input along with the data to be clustered, and returns the cluster centers (means) and the cluster membership of each input data point.

Apply the clustering method to each of the provided four images ('flag.jpg', 'market.jpg', 'snake.jpg', and 'lizard.jpg') using the following features extracted at each pixel:

- RGB color values: each pixel has a vector [R G B]
- RGB color values + x and y position: each pixel has a vector [R G B X Y]
- Lab color values (see provided conversion code): each pixel has a vector [L a b]
- (optional) Texture representation from filter banks: each pixel has a *d*-dimensional vector listing the local response at that position to *d* filters.

The optional texture features are for extra credit. If you choose to try them, it's fine to use freely available code for the extraction. (For example, Matlab code for steerable filters is available here: http://www.cns.nyu.edu/~eero/steerpyr/.)

Matlab tip: if the variable 'im' is a 3d matrix containing a color image with 'numpixels' pixels, 'X = reshape(im, numpixels, 3);' will give a matrix X with the RGB features as its rows.

Based on the image content, choose a value *k* that seems reasonable for each image; this can differ from image to image. Experiment with a few values if necessary.

After clustering the pixel features for a single image, group together the pixels belonging to the same cluster. To display a clustering result for any of the color-based features, construct an image where every pixel is colored with the *mean color* for its assigned cluster. (To display a clustering result for the texture features, label the pixels according to their cluster assignments with different graylevels of your choosing.) Include the figures displaying the clusters found with your submission.

Comment on the following:

- How do the clusters formed with the different features look? Explain why they differ the way they do.
- How sensitive is the clustering to the random initialization of the cluster centers? Run k-means a few times on each image and check.