CS 8001 Biomedical Image Processing **Spring 2017**
Out: Mon Feb 13
Due: Wed Mar 1

Assignment #3 [20tpts]

Segmentation of individual cells or nuclei is an important and challenging job for biomedical image analysis. Analysis of cell morphology (shape, structure, color, texture), cell distribution, cell motility and behavior, or cell-to-cell interactions, *ALL* heavily rely on identification of individual cells. For this assignment we will explore different steps involved in segmentation of individual cells, nuclei, or particles.

The goals of this assignment are to understand:

(i) A basic cell/nuclei/particle segmentation pipeline;
(ii) Clustering-based segmentation (K-means method);
(iii) Clustering-based segmentation (Meanshift method);
LATER: Segmentation of overlapping/touching cells.

Assignment: Implement the following tasks and report the results for each of the given test images.

Cell/nuclei segmentation using Meanshift clustering (20pts):

Segment the nuclei from the background using Meanshift clustering method.

See the following source for further details on mean shift algorithm:

- Slides for Lecture 8,
- D. Comaniciu, P. Meer: *Mean Shift: A Robust Approach toward Feature Space Analysis*, IEEE Trans. Pattern Analysis Machine Intell., Vol. 24, No. 5, 603-619, 2002.

The Mean Shift algorithm clusters an n-dimensional data set (i.e., each data point is described by a feature vector of n values) by associating each point with a peak of the data set's probability density. For each point, Mean Shift computes its associated peak by first defining a spherical window at the data point of radius r 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 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.

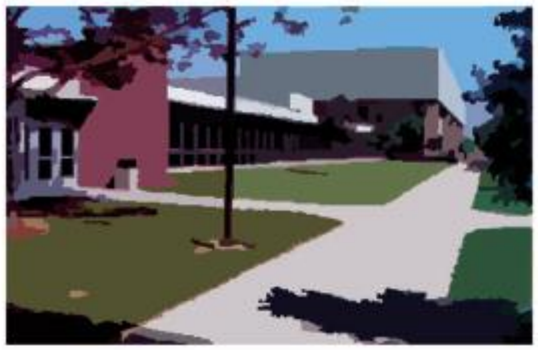
Part A) Implement mean shift function which calls the FindMode function described below for each point and the assigns a label to each point according to its mode/peak. **Follow the program structure guidelines given below.**

Report should include

- source code and comments,

- results for the given test images for a few different R values,
- interpretation of the results. How do the results change with R?
- following figures:
  a) Plot points in the data set (for feature space higher than 2D, show only two selected dimensions).
  b) Plot color-coded clustered results.
  c) For a few data points, plot the trajectories from initial estimate to final mode on the original data plot. Hint: Use ylist returned from FindMode function.
  d) Show original color image, and resulting segmented image as below.

Note: in your implementation, avoid using loops as much as possible, but some of the loops will be unavoidable and this first naïve implementation will be very slow, so first test the program with small images. For Part B, I will describe ways to make this code more efficient.



Steps of the process:

1. Convert your image to L*a*b* space, or keep the original RGB colors.
2. For every pixel (L; a; b / R; G; B), compute the weighted mean of its neighbors using either a unit ball (Epanechnikov kernel) or finite-radius Gaussian, or some other kernel of your choosing.
3. Replace the current value with this weighted mean and iterate until either the motion is below a threshold or a finite number of steps has been taken.

Program Structure:

Your program needs to include the following functions:

1. **function [Nlist]=FindNeighbors(data, y, R)**
   data: (npoints X vector_size) data matrix
   y: (1 X vector_size) one point in feature space
   R: radius of the kernel
   Nlist: (npoints X 1) vector

   It finds the data points that are within distance R from the point y in feature space. Nlist is a binary 0/1 list. Nlist(i)=1 indicates that data(i,:) is within distance R from the point y. Using this function :

2. **function [y_next]=Neighbors2Mean(data, Nlist, y)**
   data: (npoints X vector_size) data matrix
   Nlist: (npoints X 1) binary vector indicating whether a data point is within distance R from y or not.
   y: (1 X vector_size) one point in feature space (current estimate of mode)
   y_next: (1 X vector_size) one point in feature space (updated estimate of mode)

$$y_{k+1} = y_k + m(y_k) = \frac{\sum_i x_i G(y_k - x_i)}{\sum_i G(y_k - x_i)}$$

3. **function [ylist]=FindMode(data,ind,R,max_iter)**
   data: (npoints X vector_size) data matrix
   ind: a scalar index of the starting point in data.
   R: radius of the kernel.
   max_iter: maximum number of iterations
   ylist: (k X vector_size) mode estimates.
   ylist(1,:) = initial estimate data(ind,:)
   ylist(end,:) = final estimate for mode

   Using functions FindNeighbors and Neighbors2Mean, it iteratively estimates mode/peak that the data point ind converges to. Iteration is stopped when mean shift is small or when number of iterations reaches max_iter.