CS302 Data Structures

Spring 2012 – Dr. George Bebis

Programming Assignment 3

Due Date: 4/3/2012

This programming assignment will build on the results of the previous programming assignment; the goals are to represent the regions found by connected components in a *list* and compute a number of useful properties to characterize their position, orientation, shape, and intensity. You will apply this system to automatically count and characterize the galaxies in images taken by NASA's Hubble telescope (http://hubble.nasa.gov/). *Unsorted* and *sorted* lists will be the main data structures to be employed in this assignment. Specifically, the objectives of this assignment are the following:

- Improve your skills with manipulating lists
- Improve your skills with using templates
- Learn about object feature extraction and classification

Region properties: Typically, the next step after region extraction is region characterization. The description of a region may include many aspects; for example:

- Geometry
- Intensity
- Texture
- Relations with other regions

In this assignment, you will be computing geometric and intensity properties only.

Geometric properties: The geometry of a planar region concerns aspects like size, position, orientation, and shape. May of these aspects are covered by a family of parameters called *moments*. In probability theory, moments are used to characterize probability density functions; for example, mean (i.e., first order moment) variance and covariance (i.e., second order central moments). The moments of order (p,q) of a region R are:

$$M_{p,q} = \sum_{i,j \in R} i^p j^q$$

The area of a region is defined by the number of pixels in the region and can be computed using $M_{0,0}$. The first order moments $M_{1,0}$ and $M_{0,1}$ are related to the center of the region (i.e., centroid):

$$\bar{x} = \frac{M_{1,0}}{M_{0,0}} \text{ and } \bar{y} = \frac{M_{0,1}}{M_{0,0}}$$

In order to make the description independent of position, moments can be calculated with respect to the centroid. The results are the so-called central moments:

$$\mu_{p,q} = \sum_{i,j \in R} (i - \bar{x})^p (j - \bar{y})^q$$

The principal axes of a region can be computed using the *principal moments* which involve first and second order moments:

$$\lambda_{\max} = \frac{1}{2}(\mu_{2,0} + \mu_{0,2}) + \frac{1}{2}\sqrt{\mu_{2,0}^2 + \mu_{0,2}^2 - 2\mu_{0,2}\mu_{2,0} + 4\mu_{1,1}^2}$$

$$\lambda_{\min} = \frac{1}{2}(\mu_{2,0} + \mu_{0,2}) - \frac{1}{2}\sqrt{\mu_{2,0}^2 + \mu_{0,2}^2 - 2\mu_{0,2}\mu_{2,0} + 4\mu_{1,1}^2}$$

Using the principal moments, the principal axes can be computed as follows:

$$A_{\max} = \sqrt{\lambda_{\max}/M_{0,0}} \text{ and } A_{\min} = \sqrt{\lambda_{\min}/M_{0,0}}$$

The direction of the largest principal axis can be used to specify the orientation of a region:

$$\theta = \tan^{-1}(\frac{\lambda_{\max} - \mu_{2,0}}{\mu_{1,1}})$$

A useful measure of a region's circularity is its *eccentricity* which can be easily computed using the ratio between the principal axes:

$$eccentricity = \frac{A_{\max}}{A_{\min}} = \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}$$

Eccentricity could provide useful information about the region's roundness. Figure 1 shows several objects along with their principal axes and eccentricity values.
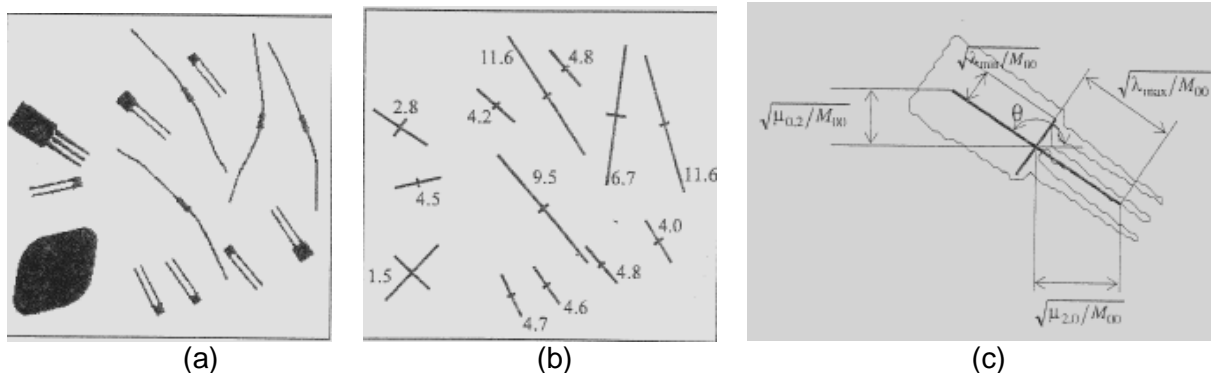

(a)   (b)   (c)

Figure 1. (a) thresholded original image, (b) principal axes and eccentricity, (c) specific example illustrating principal axes and angle θ

Intensity properties: Some simple measures of region intensity include the mean and median of the gray-levels, the minimum and maximum intensity values, and the number of pixels with values above and below the mean.

To compute the region properties, first you would need to store the regions in a list. This can be done by extending *computeComponents*.

int computeComponents(inputImage, labeledImage, listOfRegions): this function should "return" not only the labeled image and the number of components (i.e.,regions) but also a *sorted list* of regions (i.e., sorted with respect to region size, that is, the smallest region being first, the second smallest being second, etc.). For the purpose of this assignment, *computeComponents* can call either *findComponentDFS* or *findComponentBFS*. Additional information is provided in the PowerPoint slides accompanying the programming assignment.

Each node in the list of regions will store the following information:

- Geometric properties
- Intensity properties
- A sub-list containing the coordinates of the pixels in each region

The data structure to be implemented is shown in Figure 2. Each node of the list will be of type *RegionType* (i.e., you will need to define it) and should contain the following members: (1) geometric properties of the region (i.e., centroid, size, orientation, eccentricity), (2) intensity properties of the region (i.e., mean intensity value, min intensity value, and max intensity value), and (3) an unsorted list of type *PixelType* (i.e., you will need to define it) which should store the positions of the pixels contained in the region. You can implement the lists using either arrays or linked-lists. To compute the orientation of a region, use the **atan2()** function from the math library.
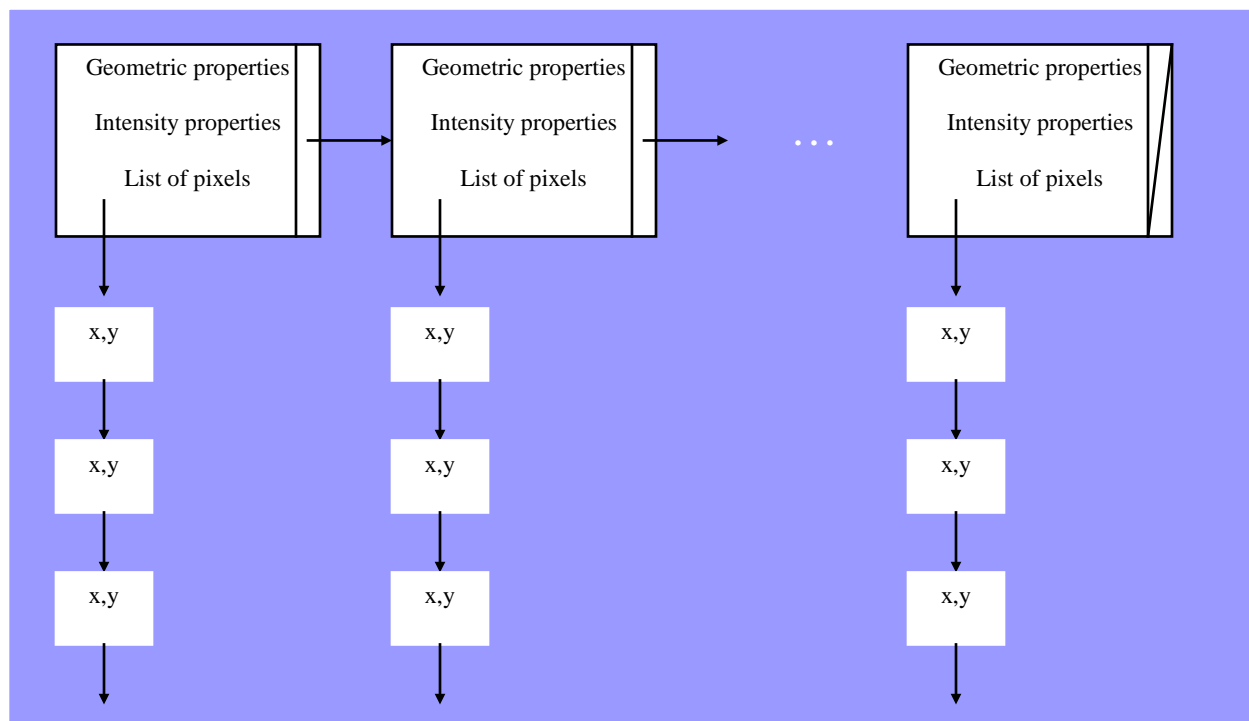

Figure 2. The list of regions.

void DeleteSmallComponents(listOfRegions, threshold): as you might have noticed from the previous assignment, the thresholded image might contain some small regions due to noise. Although erosion/dilation can remove most of them, there might still be some small regions left.

To get rid of them, simply step through the nodes of the list of regions and delete all the regions with size smaller than a specified threshold.

Classify regions: Once the regions have been processed and their properties have been computed, your program should print a summary of the regions found, their properties, and the min/max values of region size, eccentricity, and mean intensity found in the image. Also, it should print the following submenu on the screen:

(1) display regions with size between values A and B
(2) display regions with orientation between values A and B
(3) display regions with eccentricity between A and B
(4) display regions with mean intensity between A and B
(5) back to the main menu

Based on the option chosen by the user, your program should ask the user to provide two values (i.e., A and B); then, it will display an image containing only the galaxies from the category selected. You should use the original pixel values when displaying the galaxies in this step.

Instructions

Each function should be discussed in a separate section with the name of the section being the same as the name of the function. Functions that you have implemented in previous assignments do not need to be described again (only if you have made significant modifications). The sections should be clearly separated from each other.

Extra Credit

(10pts extra) Template the sorted and unsorted list types. The sorted type should be of type "RegionType" and the unsorted type should be of type "PixelType". You will need to define these types.