CS 308 Data Structures

Spring 2003 - Dr. George Bebis

Programming Assignment 3

Due date:04/15/03

In this assignment, you will use *linked-lists* to store appropriate information (i.e., centroid, size, pixels etc.) about each region found in an image by the connected components algorithms. The ultimate goal is to recognize the coins present in an image and report the total amount. Specifically, the objectives of this assignment are the following"

- Improve your skills with manipulating linked-lists.
- Improve your skills with using templates.
- Learn about object recognition.
- Learn to document and describe your programs.

How to choose a good threshold? Although it is a simple matter to convert an image to a binary image using thresholding, it is much harder to do it in such a way that important information in the image is preserved. A very high threshold will not preserve all the important information while a very low threshold will not separate the objects from the background satisfactorily. In certain cases, the histogram of the image can give us some useful hints regarding the selection of a good threshold. Let's consider, for example, the coins image whose histogram is shown below. Clearly, the histogram contains two peaks which illustrate that there are two distinct groups of pixels in the image: one corresponding to background pixels (left peak) and one corresponding to coin pixels (right peak). A good threshold in this case would be a value corresponding to the bottom of the valley of the histogram (around 160). In general, there should be a separate peak for each object "class" in an image, however, the peaks are not always clearly separated from each other due to pixel similarities and bad illumination conditions, making the choice of a good threshold very difficult. In this project, the user will choose the threshold by visually inspecting the histogram of the image to be thresholded. In other words, when the user chooses to threshold an image, your program should first display its histogram so the user can decide what threshold to use.

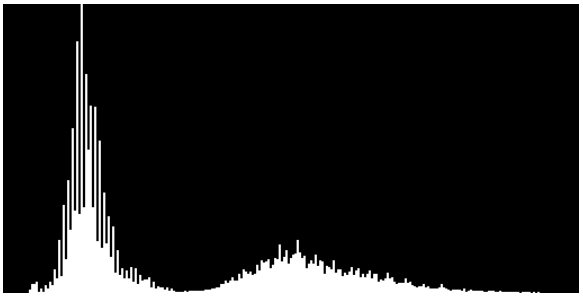

Figure 1. The histogram of an image with coins.

In the previous assignment, you implemented three connected component algorithms to count the

number of objects (i.e., coins) in an image. In this assignment, you will store appropriate information about each component in a linked-list. Then, you will pass this information to the recognition procedure (more details are given below). Your first task is to modify the *computeComponents* function.

int computeComponents(inputImage, labeledImage, listOfRegions): you need to modify this function (i.e., use either DFS or BFS) such that it returns not only the labeled image and the number of components (regions) but also a *sorted* list of regions (i.e., sorted in the size with the smallest region being the first, the second smallest second, etc.). You need to store the following information in each node:

size of the component (i.e., number of pixels)

centroid of the component ($x_c = \frac{\sum x_i}{N}$, $y_c = \frac{\sum y_i}{N}$)

the coordinates of the pixels in each component

The data structure you need to implement is shown in Figure 2. Each node of the list should be of type *RegionType* and should contain at least four data members: (1) the ID of the coin (to be determined later by the recognition procedure), (2) an "int" for the size, (3) two "floats" for the centroid, and (4) a pointer to linked list of type *PixelType* which should store the pixels contained in the region. Obviously, you would need to template the linked-list class.

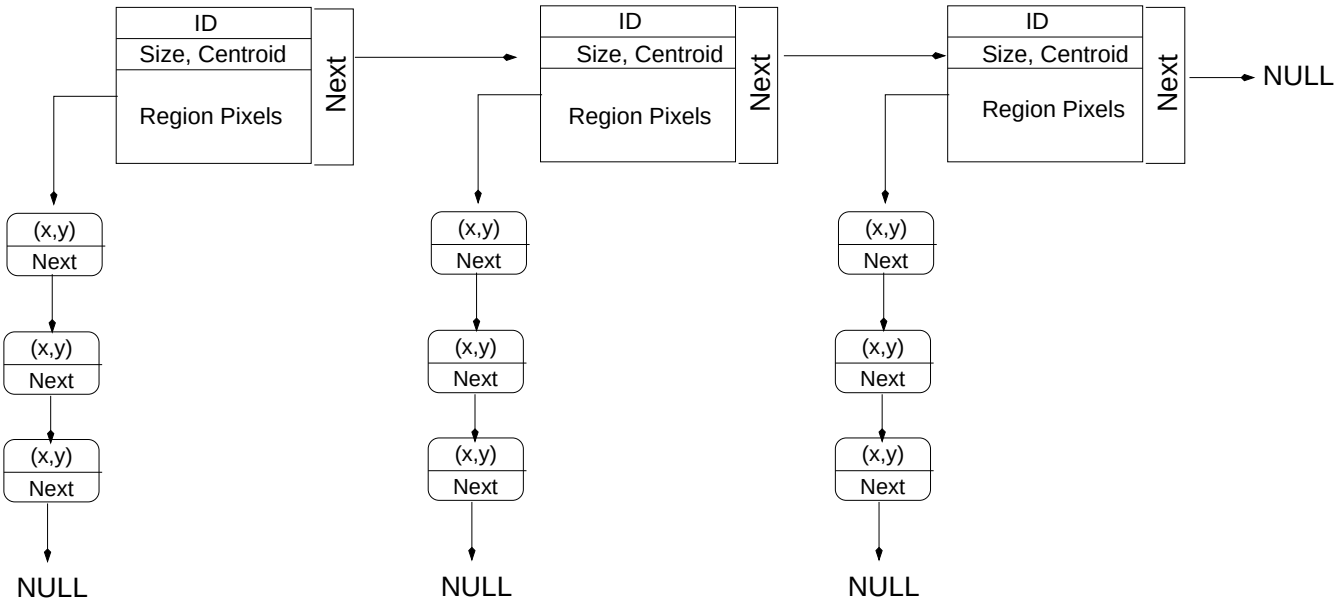

Figure 2. The linked-list of regions.

void DeleteSmallComponents(listOfRegions, threshold): as you might have noticed in the results you obtained from the previous assignment, there might be some very small regions present in the thresholded image due to noise. Although you can apply erosion/dilation to remove them before applying connected components, you can get rid of them much easier once you have

applied connected components. Simply, you need to step through the nodes of the list of regions and delete the nodes corresponding to very small regions (i.e., compared to a threshold).

Object Recognition

Your ultimate goal in this assignment is to recognize the coins in a given image and report the total amount. This is essentially a simple 2D object recognition problem. There are three main steps in every recognition system: **(1)** feature extraction/selection, **(2)** training, **(3)** recognition. In the first step, we need to decide what features to use for recognition and how to extract them. Examples of simple features include the *size* of the object, its *perimeter*, its *circularity*, etc.

During training, we process sample images containing instances of the objects to be recognized, extract the features of interest, and compute their values. The purpose of the training step is to collect enough evidence regarding the variation of the feature values for each object class. Consider, for example, the problem of recognizing the characters *a*, *b*, *c*, and *d* from their images. Let is suppose that we have selected three features: f_1, f_2, and f_3. The table below shows some sample values for each feature by processing two training images per character:

f1	f2	f3	Character
3	6	0	a
-5	9	1	c
4	5	1	d
7	-4	-10	b
1	10	0	a
2	6	1	d
2	2	1	c
-1	-3	-10	b

Once we have processed enough sample images per object, the next task is to derive some *rules* that would allow us to recognize the known characters in a new image correctly while rejecting unknown ones. One way to classify an unknown character is by finding the closest set of features in the training set. Alternatively, the decision can be based on single features or combinations of features. Looking at the above data, for example, we can observe that f_3=0 for the case of *a* or that f_1 is negative for the case of *c*. In general, selecting good features and deriving good rules for recognition are very challenging tasks. If you are interested in learning more about this subject, consider taking my *Pattern Recognition* course.

During recognition, a new image is presented to the system and its connected components are found. Then, the feature values for each component are computed and the objects (regions) are recognized by finding the most similar object(s) from the training set.

Coin Recognition

In this assignment, you will be using the size of the coins for recognition. This information should be sufficient in the context of our application since the orientation and position of the camera are fixed. Even though we have constrained our problem, there are a number of

parameters that can affect the size of a coin. For example, errors in the extraction of the coin regions due to noise will affect the size estimates. Also, placing a coin close to the periphery of an image will affect its size estimate due to distortions introduced by the imaging process. These errors in estimating the true size of a coin might have important implications when trying to separate coins having similar sizes, for example, dimes from pennies.

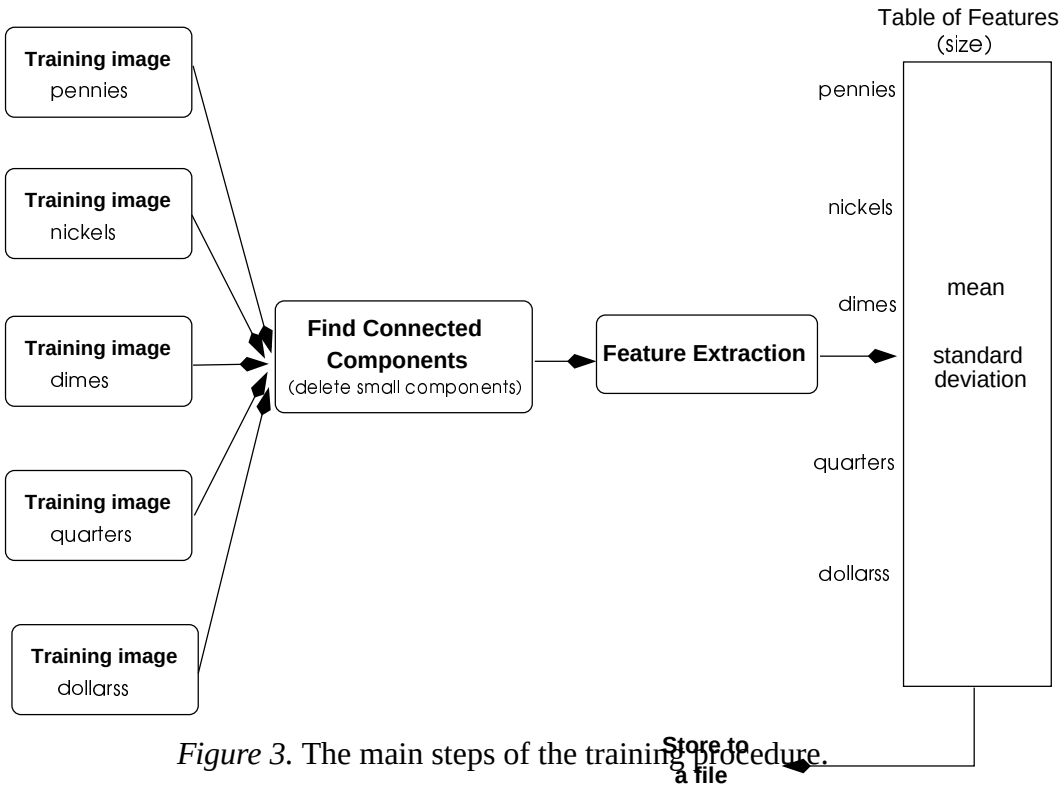

Figure 3. The main steps of the training procedure.

To deal with these issues more effectively, you will need to process several instances of coins from each class during training in order to compute the average size and standard deviation for each coin category. You will be using the images from *Image Gallery 3* in this step (*pennies.pgm, nickels.pgm, dimes.pgm, dollar.pgm*). The formulas for computing the mean and standard deviation are given below:

$$\bar{s} = \frac{1}{N}\sum_{i=1}^{N} s_i$$

$$\hat{s} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(s_i - \bar{s})^2}$$

This information per coin needs to be stored in a table, that is, the table needs to store the following information:

$\bar{s}_p$, $\hat{s}_p$
$\bar{s}_n$, $\hat{s}_n$
$\bar{s}_{di}$, $\hat{s}_{di}$
$\bar{s}_q$, $\hat{s}_q$
$\bar{s}_{do}$, $\hat{s}_{do}$

Store the contents of the table in a file so that you do not have to repeat the training procedure every time. Figure 3 illustrates the main steps of the training procedure.

During recognition, your program should first read in the table of features from the file. Then, given a new image of coins, it should recognize the coins and print the total amount on the screen. To recognize a given coin, you should first estimate its size *s*. Then, you should find the coin category whose size, computed during training, is closest to *s*. To accept match and avoid false positives, the error between *s* and the closest size should be within some tolerance which can be determined using the standard deviation. For example, suppose that the closest coin category is the *quarters* category. The following condition should hold true to accept the match as correct:

$$|s - \bar{s}_q| \leq const \ x \ \hat{s}_q$$

where *const* is a constant you have to determine to minimize miss classifications. If this condition is false, the coin should be classified as unknown. The recognition procedure is illustrated in Figure 4.

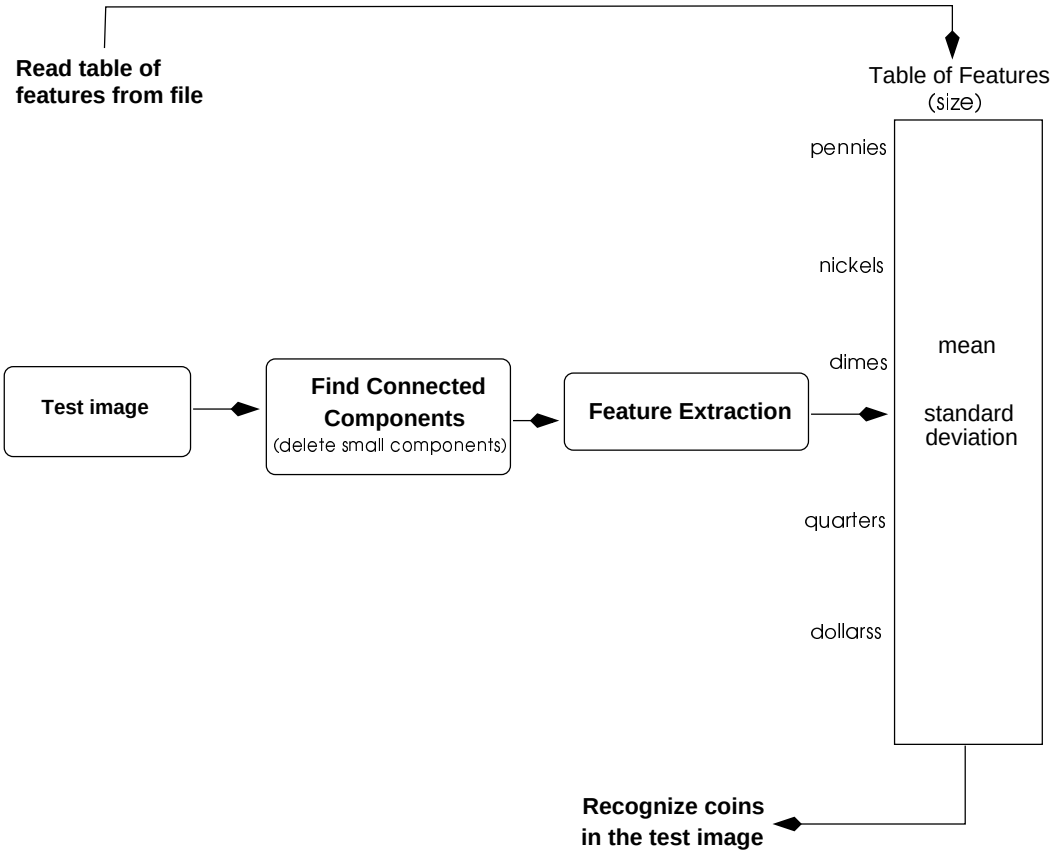

Figure 4. The recognition procedure.

Training and recognition should be implemented as two separate programs: *train.cpp* and *recognize.cpp* (i.e., client code).

train.cpp: this program should create the table of features using the coin images from *Image Gallery 3*. When running the program, the follow menu should appear on the screen:

(1) train pennies
(2) train nickels
(3) train dimes
(4) train quarters
(5) train dollars
(6) done with training

The user is supposed to select one of the above options and provide the filename of an image containing coins from the category selected for training. When option (6) is selected, the program should save the table of features in a file and exit.

recognize.cpp: this program should read the table of features from the file and ask the user for an input image. Then, it will try to recognize the coins in the input image and assign the appropriate value to the ID field in the corresponding node of the list of regions. Once all the coins have been recognized, the program should print the following menu on the screen:

(1) display the pennies only
(2) display the nickels only
(3) display the dimes only
(4) display the quarters only
(5) display the dollars only
(6) print total amount
(7) done with recognition

Based on the option chosen by the user, the program should display an image containing only the coins from the category selected by the user or the total amount (option 6). To display, let's say, the quarters only, you would need to traverse the list of regions, find the nodes corresponding to quarters, and draw the pixels of each quarter (i.e., their coordinates are stored in the list of pixels of that region) to a new image by copying their pixel values from the original image. You should use images from *Image Gallery 2* to test your program.

Instructions

Describe the implementation of *train.cpp* and *recognize.cpp* in sufficient detail. Discuss the data structures and functions used. New functions should be discussed in a separate section with the name of the section being the same as the name of the function. Functions you have implemented in the previous assignment do not need to be described again in this assignment. The sections should be clearly separated from each other.

Questions

Answering the following questions does not require that you have prior knowledge of image processing. Just spend some time thinking about them and give us your best possible answers along with some justification. No coding is required for this set of questions. Interesting answers will get **extra credit** !!

1. (2pts extra) Depending on the application, the problem of feature selection and extraction can be very challenging. What would be a good set of features according to your opinion? List a number of criteria if possible. Give some examples and justify your answer.

2. (2pts extra) In this assignment, you implemented a system to perform coin recognition using size information only. Will your algorithm work if we allow the camera to move closer or farther away (keeping the same orientation though). If not, can you suggest ways to improve your algorithm?

3. (2pts extra) Camera rotation and coin occlusion are two other very challenging problems. Will size information be enough in this case? Justify your answer. How would you deal with these problems?

4. (4pts extra) Given an unknown coin, you would need to find its size and compare it to the size range of each coin category to find the closest match. In general, the number of coin categories might be quite large (e.g., consider implementing a program to recognize coins from every country in the world!). In this case, you would need to compare the unknown coin to all known coins (e.g., maybe 1,000 different coin categories!). This will be very time consuming, can you suggest ways to deal with this problem more efficiently?