

CS 302 Data Structures
Spring 2008 - Dr. George Bebis
Programming Assignment 4

Due date: 4/22/2008

In this assignment, you will use *lists* to store appropriate information about each coin found in an image. The ultimate goal would be to recognize the coins present in an image and output the total amount. Specifically, the objectives of this assignment are the following"

- Improve your skills with manipulating lists.
- Improve your skills with using templates.
- Improve your skills file input/output
- Learn about object recognition.
- Learn to document and describe your programs.

In the previous assignment, you used connected components to count the number of coins in an image. In this assignment, you will store specific information about each coin in a list. Then, you will provide this information to the recognition algorithm - more details are given below. Your first task would be to modify the *computeComponents* function.

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 coin size, that is, the smallest coin being first, the second smallest being second, etc.). For the purpose of this assignment, *computeComponents* can call either *findComponentDFS* or *findComponentBFS*. You would need to store the following information in each node of the list of regions:

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

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

a sub-list containing the coordinates of the pixels in each regions

The data structure you would need to implement is shown in Figure 2. Each node should be of type *RegionType* (i.e., you would need to define it) 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* (i.e., you would need to define it) which should store the positions of the pixel contained in the region.

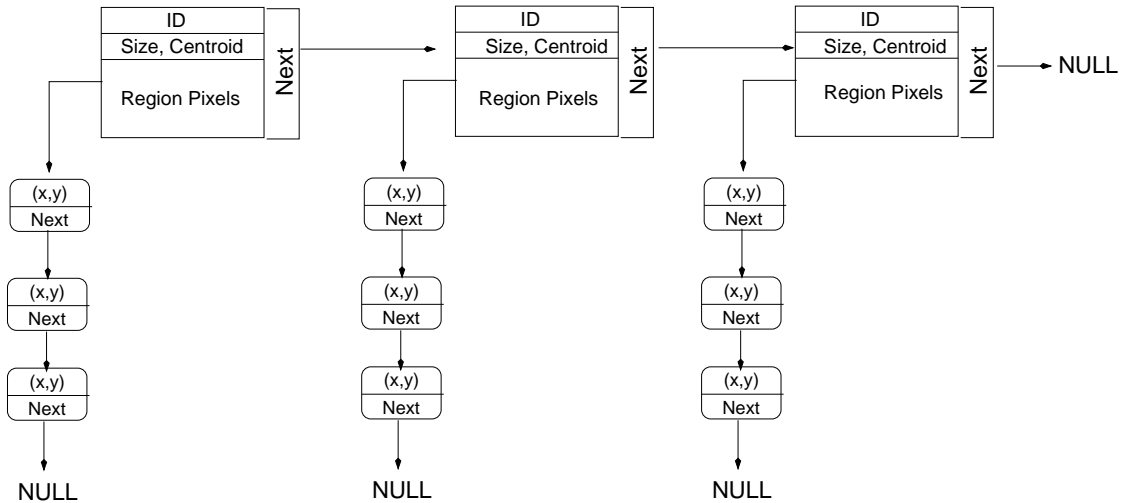


Figure 2. The linked-list of regions.

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 regions whose size is smaller than a given threshold.

Object Recognition

Your ultimate goal in this assignment is to recognize the coins in a given image and output the total amount. This is essentially a simple 2D object recognition problem. In general, recognition systems include three main steps: (1) feature extraction/selection, (2) training, (3) recognition. The first step decide what features to use for recognition and how to extract them. Some simple features include the *size* of an 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. Once we have processed enough sample images per object, the next task is to derive some *rules* that would allow us to recognize known objects in a new image correctly while rejecting unknown ones. 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 (CS479/679).

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, coins will be recognized using their "size". Coin size provides sufficient

information for recognition purposes in the context of our application since the orientation and position of the camera are fixed. It should be mentioned that even though we have constrained our problem, there are still a number of issues that could affect recognition performance. For example, errors in the extraction of the coin regions due to noise could affect the size estimate of a coin. Also, placing a coin close to the periphery of an image could 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 physical size (e.g., dimes vs pennies).

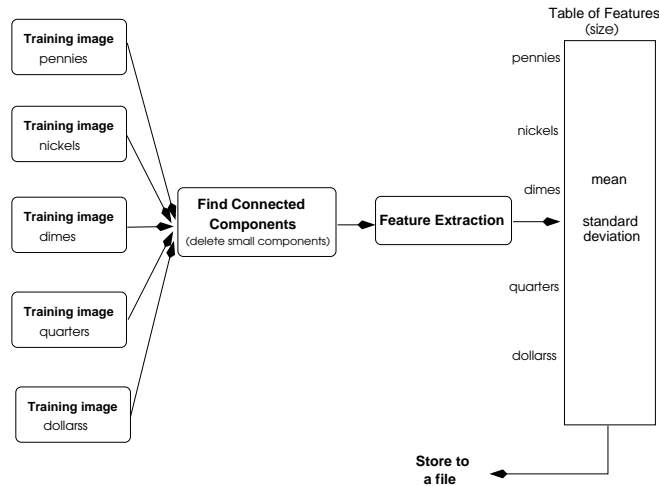


Figure 3. The main steps of the training procedure.

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 4* in this step (*pennies.pgm*, *nickels.pgm*, *dimes.pgm*, *dollar.pgm*). The equations to compute the mean size and standard deviation are shown 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:

$$\begin{aligned} &\bar{s}_p, \hat{s}_p \\ &\bar{s}_n, \hat{s}_n \\ &\bar{s}_{di}, \hat{s}_{di} \\ &\bar{s}_q, \hat{s}_q \\ &\bar{s}_{do}, \hat{s}_{do} \end{aligned}$$

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 a 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:

$$|s - \bar{s}_q| \leq c \cdot \hat{s}_q$$

where c is a constant that you have to determine in order to minimize misclassifications. If this condition is false, the coin should be classified as unknown. The recognition procedure is illustrated in Figure 4.

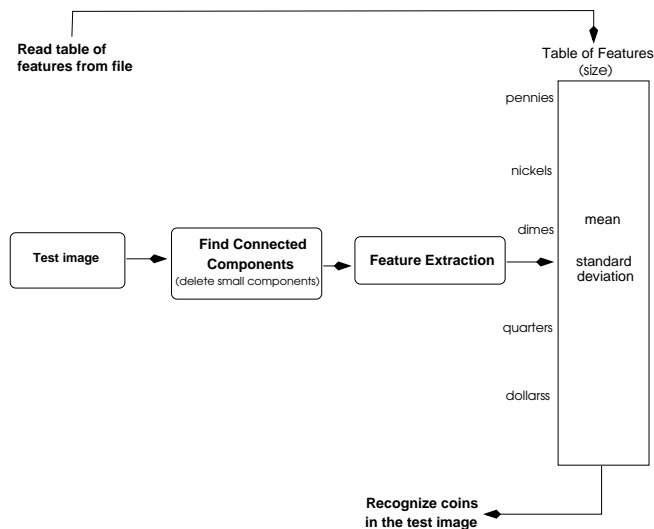


Figure 4. The recognition procedure.

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 4*. 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 3* 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

1. (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 would need to define these types.