DIP Homework Assignment #4

姓名：鍾毓安

學號：B01902040

系級：資訊四

Email: [iamyuanchung@gmail.com](mailto:iamyuanchung@gmail.com)

繳交日期：2016.5.31

**Execution**

To reproduce the result, simply execute README.m under Matlab environment.

**Problem 1: Shape Analysis**

Sample1.raw is a gray-level image that contains some characters. Please design an algorithm to recognize different characters and count the number of occurrence of each character using TrainingSet.raw as the training set. Please provide the flow chart and details of your algorithm, and discuss the result in the report.

For problem 1, the following two functions were implemented:

1. shapeAnalysis.m: The function takes Sample1.raw and TrainingSet.raw, denoted as S1 and TS, respectively, as inputs and performs a series of steps, which will be described in detailed later in Figure 1-3, to deal with the required task. Finally, for each instance that appears in S1, the function outputs the class of shape it belongs to.
2. signSegment.m: The function takes S1 as input and segments S1 into small pieces recursively such that each piece is an independent instance to be classified. The function returns a cell array Ins, where the i-th instance (an image matrix) in S1 can be accessed by Ins{i}. Figure 1-1 depicts the concept of the recursive segmentation performed by signSegment.m.

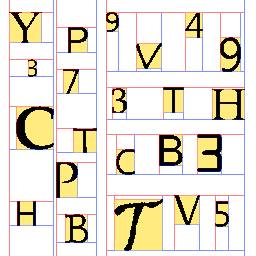


Figure 1-1: Starting from the leftmost column of S1, the function finds the first column that contains at least one black pixel, as suggested by the red vertical line. Then, from that red vertical line, the function searches for the first column that contains all white pixels, as suggested by the blue vertical line. The segment between the red and the blue vertical lines is then taken as an independent input image by another signSegment.m and the same task described previously is performed again, except that for now the function starts from the top row and looks for the first row that contains at least one black pixel and so on.

Figure 1-2 displays some instances segmented by signSegment.m.

rslt_images/Q1_instances/instance1.png rslt_images/Q1_instances/instance2.png rslt_images/Q1_instances/instance11.png rslt_images/Q1_instances/instance19.png rslt_images/Q1_instances/instance20.png

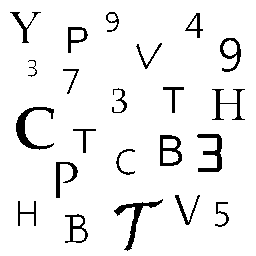
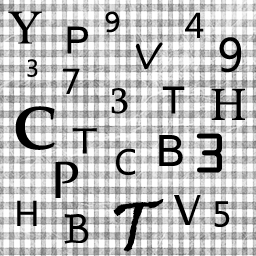
Figure 1-2: From left to right are Ins{1}, Ins{2}, Ins{11}, Ins{19}, and Ins{20}.

We can see that they can have very different sizes even they belong to the same category.

Undoubtedly, this will be one of the difficulties we will encounter during classification.

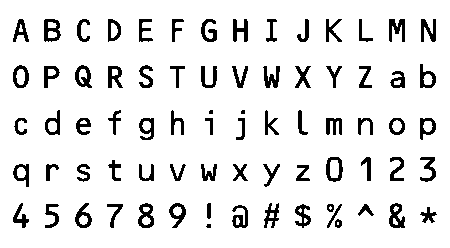
Figure 1-3 is the work flow of shapeAnalysis.m. The following is the description of each step:

1. Sample1.raw, denoted as S1 is taken as input. However, the background of S1 contains noise. We first remove it and denote the clean version as S1\_clean.



S1 S1\_clean

1. S1\_clean is then fed in the function signSegment.m, which segments S1\_clean into separate instances to be classified. signSegment.m returns a cell array Ins, where each entry Ins{i} is the i-th instance (an image matrix). Some examples of the segmented instances are shown in Figure 1-2.
2. The TrainingSet.raw, denoted as TS, contains some pixels whose values are not 0 or 255. To make the later matching more convenient, TS is binarized such that the pixel values in the resultant image, denoted as TS\_bin, are either 0 or 255.



TS TS\_bin

1. The classes of shapes provided in TS\_bin are so well-organized that we can separate them heuristically. By setting a window size and slicing it through TS\_bin, we obtain another cell array Cls, where each entry Cls{j} is the image matrix of the j-th class (counting from left to right, top to bottom). To make the later matching more convenient, each class Cls{j} is further trimmed such that the surrounding spaces are removed. Figure 1-4 displays some examples of classes before and after trimming.



Figure 1-4: Three pairs of before-&-after examples are shown.

The outer spaces are removed after Cls{j} is trimmed.

1. We now have two cell arrays: Ins and Cls. For each instance Ins{i}, we want to classify it into one of the seventy classes provided in TS. This is done by computing the similarity between Ins{i} with each Cls{j}: Ins{i} is first resized to the same size as Cls{j}. Then, the similarity between Ins{i} with current Cls{j} is measured by the overlapping percentage, that is, the number of pixels that have the same values divided by the area of Cls{j}. The larger the overlapping percentage is, the more similar they are. Ins{i} will then be classified as class . The reason why I use the overlapping percentage instead of the counts of pixels with the same values is that Cls{j} with larger size always has a better chance to win over another whose size is smaller, this cause the classification to be inaccurate.
2. By taking these steps, 16 out of 22 instances are classified correctly, that is, accuracy =. Those six misclassified instances are somewhat originally hard to be classified and the misclassifications are understandable, for example:

rslt_images/Q1_instances/instance9.png is classified as 8, and rslt_images/Q1_instances/instance20.pngis classified as 7.

**Problem 2: Morphological Processing**

Given a binary image Sample2.raw, please try to produce the same images as illustrated in Fig. 4 by adopting appropriate morphological processing. Please describe the designed algorithm in detail for each case.

https://www.csie.ntu.edu.tw/~b01902040/doc/DIP\_hw4\_Q2\_Skeletonize.gif

**Problem 3: Texture Analysis**

Let’s denote Sample.raw as I. Please generate several images by the instructions below.

1. Transfer I to frequency domain by DFT (Discrete Fourier Transform) with centering and output the result as D.
2. Apply an ideal low-pass filter to D with and . Output results as and , respectively.
3. Apply a Gaussian low-pass filter to D with and . Output results as and , respectively.
4. Transfer , , , and back to spatial domain by Inverse DFT. Please compare the results and provide some discussions in the report.