Instructions:

- Attempt all questions
- Submit your assignment electronically by email to obeijbom@es.ucsd.edu with the subject line CSE152 Assignment 2. The email should include 1) A a written report that includes all necessary output images and written answers (in PDF format) 2) all necessary Matlab code. Attach your code and report as a zip or tar file.
- No physical hand-in for this assignment.

1 Introduction

In this homework we will build a coins - cap object classification system. Given an image of a number of coins (quarter) and caps (permanent marker caps) you program should correctly classify each item as either a coin or a cap. We will use three images for this: coin.jpg, cap.jpg, and mix.jpg. The high-level idea is that coin.jpg and cap.jpg will be used as training data and then the objects in mix.jpg will be classified by your system. To achieve this we need several simple programs. We will start by looking at each separately and then put everything together in the end. We will further use two files: hole.png and coins.png to practice the different components of the pipeline. Good luck!

2 Binarization:

Write a Matlab function to implement the 'peakiness' detection algorithm described in class. This algorithm should automatically determine an intensity level to threshold an image to segment out the foreground from the background. The output of your function should be a binary image that is 0 for all background pixels and 1 for all foreground pixels. Apply this function to the image mix.jpg and turn in the output image in your report. [9 points]

Notes:

- Load in an image as a grayscale, double-precision image, e.g.
  \[
  \text{img} = \text{im2double(rgb2gray(imread('coins.png')))};
  \]
- You can use the Matlab function \text{hist(img(:,),numBins)} to create a histogram of pixel intensities as a first step to the peakiness detection algorithm. Using \text{numBins=10} and a minimum distance of 3 bins between peaks should work.

3 Morphological Image Processing

Morphology refers to application of set operations such as union and intersection on an image. Morphological algorithms take as input an image and a structuring element, which encodes the shape characteristics based on which the input image must be processed. Usually, the structuring element is a \((2k + 1) \times (2k + 1)\) array, where \(k\) is a small number (say, between 1 and 5).

In general, a morphological operation is performed by making a copy of the input image. The structuring element, \(S\), is compared to a \((2k + 1) \times (2k + 1)\) window neighborhood, \(W\), of each pixel of the input image. If \(W\) exactly matches \(S\), then the corresponding pixel of the copy image is set to a particular value (0 or 1, depending on particular operation), else that pixel in the copy is left unchanged.
(a) **Erosion**: The desired effect in erosion is that any foreground pixel which has a background pixel as a neighbor is set to background. Consider the $3 \times 3$ structuring element:

$$S_1 = \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{bmatrix}$$

The $3 \times 3$ neighborhood of each pixel in the input image is compared to this structuring element $S_1$ and the corresponding pixel in the copy image is set to background if they do not match, else it is left unchanged. Thus, it causes the boundary of the foreground to shrink. Write Matlab code to perform erosion using the element $S_1$ on the image `hole.png`. Remember to convert the input to a binary image first. Display your output after the image has been eroded by 1, 3 and 8 passes of the erosion algorithm. [8 points]

(b) Now repeat question (a) with a structuring element $S'_1$ of the same form as $S_1$, but of size $7 \times 7$. Also, give a qualitative description of the effect of increasing the size of the structuring element. [4 points]

(c) **Dilation**: The dual problem to erosion is called dilation. Here the aim is to use a morphological operation to expand the boundary of the foreground region of an image. One way to do this is the following:

for each background pixel $p$
   Let $N = \text{neighborhood of } p$
   if $N$ contains only background pixels
      leave $p$ unchanged
   else
      set $p$ to foreground
   end
end

Describe the process and structuring element, $S_2$, you would use to perform dilation on an input image. Write Matlab code that takes as input the image `hole.png`, the structuring element you have devised and desired number of passes. The output should be the dilated image after the specified number of dilation passes. Display the dilated image after 1, 3 and 8 passes. [6 points]

4 **Connected Components**

(a) Write Matlab code to implement the connected component labeling algorithm discussed in class, assuming 8-connectedness. Your function should take as input a binary image (computed using your algorithm from question 1) and output a 2D matrix of the same size where each connected region is marked with a distinct positive number (e.g. 1, 2, 3). On the image `mix.jpg`, display an image of detected connected components where connected region is mapped to a distinct color (using the function `imagesc` in Matlab). [9 points]

(b) How many components do you think are in the image `coins.png` (note, this is not the mix.jpg image, but another one containing a bunch of coins)? What does your connected component algorithm give as output for this image? Can you use one of the morphological operations to help separate out the individual coins in this image? Write Matlab code combining your morphological operator and connected components algorithm, and use it to count the number of coins. Include any relevant figures for this separation in your report. [6 points]

Notes:
To avoid Matlab recursion limits, you may find it necessary to increase the recursion limit:
\[
\text{set(0,'RecursionLimit',1000)};
\]
For the coin image, it may help to reduce the size of the image before running the connected components algorithm but after converting the image to a binary image:
\[
\text{binarySmall = imresize(binary, 0.25, 'bilinear')};
\]

5 Image moments

We now have code to (1) binarize the image (2) clean up the binary image by morphological operations (3) use connected components to enumerate the objects. The final building block is to create object descriptors using image moments. For this problem we will use the \( I_1 \) and \( I_2 \) moments described in section 'Rotation invariant moments' on this wikipedia page: [http://en.wikipedia.org/wiki/Image_moment](http://en.wikipedia.org/wiki/Image_moment). The derivation of these are slightly different from the class notes, but don’t worry about that.

Write code that takes a binary image and calculates the \( I_1 \) and \( I_2 \) moments. Use this code to calculate these moments for each object in the mix.jpg image. Plot the results with blue markers for the caps and red for the coins. Your plot should have the \( I_1 \) on the x-axis and \( I_2 \) on the y-axis and contain one marker for each object (6 total). What can you tell from this plot? What operation did you use from the ones we implemented in the previous problems? Discuss. [12 points]

6 Object classification

We are now ready to put everything together and create an object classification system. We will use cap.jpg and coin.jpg as training data and mix.jpg at testdata. We will thus pretend that we do not know what types of objects are in mix.jpg, and that we want to write a program that figures this out for us. This might seem silly (because it’s very easy to just open the image and look), but imagine we had 1000 such images, then it’d be quite nice with an automated system. Your program structure should be something like this:

1. Load cap.jpg, we call this class 1.
2. Extract the \( I_1 \) and \( I_2 \) moments, store in a vector \( x_1 \). This is commonly called a feature vector.
3. Load coin.jpg, we call this class 2.
4. Extract the \( I_1 \) and \( I_2 \) moments, store in a vector \( x_2 \).
5. You now have two ‘training’ points
6. Load mix.jpg
7. For each object in this image:
   (a) extract the \( I_1 \) and \( I_2 \) moments, store in a vector \( z \).
   (b) classify the object as: \( \arg\min_i \| z - x_i \|^2 \). This means: assign the test object to the category whose train data feature vector is closest to the test object’s feature vector using the euclidean distance.

Describe what steps you took to go from the image file to the feature vector, show plots of your traindata and testdata points, and how the objects in mix.jpg were classified. Did it work? [12 points]