CSE152 – Intro to Computer Vision – Assignment #2
Instructor: Prof. David Kriegman.
`http://cseweb.ucsd.edu/classes/sp12/cse152-a`
Due Date: Thu. May 11, 2012.

Instructions:

- Attempt all questions
- Submit your assignment electronically by email to sbranson@cs.ucsd.edu with the subject line *CSE152 Assignment 2*. The email should include 1) A 1) a written report that includes all necessary output images and written answers (e.g., as a Word file or PDF), and 2) all necessary Matlab code. Attach your code and report as a *zip* or *tar* file.

1 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 *coins.png* and turn in the output image in your report. [**9 points**]

Notes:

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

2 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 3×3 structuring element:

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

The 3×3 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, 4 and 10 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 7×7. 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 $N$ = 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 specied number of dilation passes. Display the dilated image after 1, 4 and 10 passes.[**6 points**]

3 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 *ucsd.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*? What does your connected component algorithm give as output for this image? Can you use one of the morphological operations in Question 1 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: 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: binarySmall = imresize(binary, 0.25, 'bilinear');

4 Edge Detection

In this question, you will experiment with the Canny edge detector described in class. You do not need to implement it and can use the built-in Matlab function:

edge(img,'canny',[τ_{low}, τ_{high}],σ)

Load in the image *ball.png* and experiment with different assignments to the parameter values τ_{low}, τ_{high}, σ. Find an assignment τ^1_{low}, τ^1_{high}, σ^1 where most of the rounded rectangles inside the ball appear as closed edges. Find a second assignment τ^2_{low}, τ^2_{high}, σ^2 where the rounded rectangles don't appear, but some of the vertical longitudal lines going up the ball appear as edges. Find a third assignment τ^3_{low}, τ^3_{high}, σ^3 where neither the rounded rectangles nor the longitudal lines appear, but the contour of the ball appears as a closed edge. Display your parameter values and the resulting output images of the Canny edge detector. Given your knowledge of the Canny edge detection algorithm, explain why these different parameter choices yield different types of detected edges. [**8 points**]