CS 485/685 – Computer Vision

Programming Assignment 2

(Due April 8)

1. (40 pts) Test and evaluate the Canny edge detector, by using the OpenCV function `cvCanny`. Images to be used in your experiments are available from the class webpage. Evaluate the sensitivity of the algorithm to the following parameters: low-threshold, high-threshold, and mask size. Show a representative set of results and discuss the performance of the Canny edge detector.

   Submit the following:
   - (hardcopy) a report showing and discussing your results
   - (email) one ZIP file containing:
     - the source code files
     - a README file with instructions on how to compile and run the program

2. (60 pts) Implement the Hough Transform for detecting circles. Images to be used in your experiments are available from the class webpage. For each detected circle, your program should print its center and radius, and its fitting error (see below). In addition, it must produce output images that show the detected circles superimposed on the original images (OpenCV provides the function `cvCircle` to draw a circle).
   You will probably need to discard some "spurious" circles – for example, very small circles, or circles that have little support, or circles that are concentric (i.e., have the same center within a threshold) which are likely to correspond to the same object.
   You may also want to start by using some synthetic images that contain perfect circles for debugging purposes.

   The steps for this problem are:
   a) Derive the algorithmic steps for circle detection using the Hough Transform.
   b) Apply edge detection on the input image. OpenCV provides a number of edge detectors, such as the Sobel edge detector (`cvSobel`).
   c) Implement the Hough Transform for circle detection. For each accumulator, use a linked-list to store the pixels voting for that accumulator.
   d) For each circle detected, use the corresponding linked-list to compute the fitting error (estimate how well the detected circle fits the data). The fitting error is determined by the distance of the points to the circle. The distance between a point $(x_i\ y_i)$ and a circle with center $(x_0\ y_0)$ and radius r is:

$$d_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} - r$$

The fitting error for a circle that has been supported by *n* points is:

$$E = \frac{1}{n}\sum_{i=1}^{n} d_i^2$$

Note: The trickiest part is figuring out how to partition the parameter space. If the bins are very small, none will ever accumulate enough votes due to noise. If they are too large, different circles may be confused.

Submit the following:

- (hardcopy) a report showing and discussing your results; you also need to describe your algorithm, the parameter space, the quantization, how "spurious" circles were discarded
- (email) one ZIP file containing:
    - the source code files
    - a README file with instructions on how to compile and run the program