# CS 4/791E Computer Vision

# Spring 2004 - Dr. George Bebis

# Programming Assignment 2

### Due date: 3/23/04

This assignment has two main parts. In the first part, you will use OpenCV to familiarize yourself with some of the algorithms we discussed in class. In the second part, you will implement the Hough Transform for circle detection.

**Part 1**

(1.1) *(25 pts)* In this part, you will test the *Canny* edge detector. The OpenCV function you would need to use is called *cvCanny* (p. 3-11 and 10-11). Evaluate the sensitivity of the algorithm to the following parameters: low-threshold, high-threshold, mask size. Show a representative set of results and provide some analysis and discussion (this part is the most important, don't expect lots of points if you do not do a good job here!). Put your results and discussion, and code (each file separately, do not tar or zip the files) on the web and email me the address.

(1.2) *(25 pts)* In this part, you will test the *deformable model* (snake) approach we discussed in class. The OpenCV function you would need to use is called *cvSnakeImage* (p. 2-15 and p. 9-11). Apply this algorithm on the test images provided on the web page of the course. Evaluate the sensitivity of the algorithm to the following parameters: initial contour points, number of contour points, $\alpha$, $\beta$, $\gamma$, and neighborhood size. Show a representative set of results and provide some analysis and discussion (this part is the most important, don't expect lots of points if you do not do a good job here!). Put your results and discussion, and code (each file separately, do not tar or zip the files) on the web and email me the address.

**Part 2**

(2) *(50 pts)* The objective of this part of the assignment is to implement the Hough Transform for detecting circular objects in an image (note that OpenCV provides an implementation of the Hough Transform for lines only -- see p. 3-14 and 10-17). Given a gray-scale image, your program should output the center and radius of all the circles found. In addition, it should output the fitting error in each case (see below). Circles that are concentric (i.e., have the same center within a threshold that you need to choose in a reasonable way) should be grouped together since it is likely that they arise from the same object. To show your results visually, overlay the circle(s) found by the Hough Transform on the input gray-scale image. OpenCV provides a number of

functions for drawing various primitives. You would need to use the function *cvCircle* (p. 14-69). I will put on the web several images for you to test your program. You may want, however, to make some preliminary test images that contain graphically generated perfect circles for debugging your program.

The steps for this part of the project are:

(1) Derive the algorithmic steps for circle detection using the Hough Transform. We have discussed the case of line detection in class. You would need follow similar ideas in order to derive the steps for circle detection.

(2) Apply edge detection on the input image. OpenCV provides a number of edge detectors such as the Sobel edge detector (*cvSobel*, p. 10-10).

(3) Implement the Hough Transform for circle detection. Use linked-lists to store the pixels voting for each accumulator.

(4) For each circle detected, use its corresponding linked-list (i.e., the linked list associated with the corresponding accumulator) to compute the fitting error (i.e., how well the circle parameters fit the data). To do this, you would need to compute the distance of the points from the circle. Let us suppose that the radius of the circle is $r$ and its center $(x_0, y_0)$. To compute the distance of a point $Q$ $(x_i, y_i)$ from the circle, first compute its distance $q$ from the center of the circle:

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

The distance $d$ from the circle is then given by:

$$d = q - r$$

The fitting error can then be computed as follows:

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

(4) Discussion and analysis.

- Use pseudocode to present the algorithmic steps of your algorithm.
- How do you quantize the accumulator array? Why?
- The effect of the quantization of parameter space in the accumulator array.
- Present a representative set of results.
- Analyze and discuss the results.

Put your report, results, and code (each file separately, do not tar or zip the files) on the web and email me the web-address (provide me with a README file too so I know how to compile and use your code too).

*Hint*

The trickiest part is figuring out how to partition the parameter space and detect the clusters. If your bins are very small, none will ever accumulate enough votes due to noise (and you might run out of memory). If they are too big, the different circles will be confused. We have addressed all of these issues in class.