Homework 2
Computer Vision CS 4731, Fall 2011
Due Date: Sept. 29, 2011
Total Points: 26

Note 1: Both the analytical problems and the programming assignments are due at the beginning of class on Sept. 29, 2011. The analytical problems should be turned in as a hard copy at the beginning of class, and the programming assignments should be posted to the "Homework 2 Dropbox" folder on CourseWorks by the beginning of class. Please start working on your assignment early and note that there is no credit for late submissions.
Note 2: Read the guidelines for the programming assignments carefully. They are available on CourseWorks at Class Files/Shared Files/Programming Guidelines.

Problem 1: Show that the extreme values of the moment of inertia (E) are given by the eigen values of the 2×2 matrix :

$$\begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix}$$

where a, b, c, and E are as defined in the lecture notes (or as in Horn). (3 points)
Argue that E is real and non-negative, and hence prove that $4ac \geq b^2$. (2 points)
When is E equal to zero? (1 point)

Programming Assigment

Our goal is to develop a vision system that recognizes two-dimensional objects in images. The two objects we are interested in are shown in the image **two_objects.pgm**. (All the images given to you are gray-level PGM images and can be downloaded from CourseWorks) Given an image such as **many_objects_1.pgm**, we would like our vision system to determine if the two objects are in the image and if so compute their positions and orientations. This information is valuable as it can be used by a robot to sort and assemble manufactured parts.

The task is divided into four parts, each corresponding to a program you need to write and submit. Each program must accept arguments in the format specified as

program_name {1st argument} {2nd argument} ... {N*th* argument}

- **Program 1:** Write a program named **p1** that converts a gray-level image to a binary one using a threshold value:

  p1 {input gray-level image} {input gray-level threshold} {output binary image}

  Select any threshold that results in "clean" binary images for the gray-level ones given to you. (A binary image can be saved as a PGM file with 1 as the number of colors in the header.) You should be able to use the same threshold value for all the images. (When submitting your assignment, please indicate the value you used in a separate README file.) Apply program **p1** to the image **two_objects.pgm**. (2 points)

- **Program 2:** Implement the sequential labeling algorithm (and name the program **p2**) that segments a binary image into several connected regions:

  p2 {input binary image} {output labeled image}

  Note that you may have to make two passes of the image to resolve possible equivalences in the labels. In the "labeled" output image each object region should be painted with a different gray-level: the gray-level assigned to an object is its label. The labeled images can be displayed to check the results produced by your program. (To make gray-levels look significantly different, you may want to use consecutive natural numbers as labels in your output file, and set the number of colors in the PGM header to the total number of objects.) Note that your program must be able to produce correct results given *any* binary image. You can test it on the images given to you. Apply program **p2** to the binary version of the image **two_objects.pgm**. (6 points)

- **Program 3:** Write a program named **p3** that takes a labeled image and computes object attributes, and generates the objects database:

  p3 {input labeled image} {output database} {output image}

The generated object database should include a line for each of the objects with the following values, separated by blanks:

a. object label,
b. row position of the center, column position of the center,
c. the minimum moment of inertia,
d. the orientation (angle in degrees between the axis of minimum inertia and the vertical axis),
e. the roundness.

You can compute any additional properties if you want. However these should be appear after (to the right) the properties mentioned above. Mention the additional properties in your README file. These attributes will serve as your object model database. The output image should display positions and orientations of objects in the input image using a dot for the position and a short line segment originating from the dot for orientation. (To draw a line, you can use the code provided to you in vision_utilities.c). Apply program **p3** to the labeled image of **two_objects.pgm**. (6 points)

- **Program 4:** Now you have all the tools needed to develop the object recognition system. Write a program named **p4** that recognizes objects from the database:

  p4 {input labeled image} {input database} {output image}

  Your program should compare (using your own comparison criteria) the attributes of each object in a labeled image file with those from the object model database. It should produce an output image which would display the positions and orientations (using dots and line segments, as before) of only those objects that have been recognized. Using the object database generated from **two_objects.pgm**, test your program on the images **many_objects_1.pgm** and **many_objects_2.pgm** In your README file, state the comparison criteria and thresholds that you used. (6 points)

Notes:

- The data for the programming assignment can be downloaded from CourseWorks as hw2data.zip. This zip file contains the test images and a Makefile which contains targets for building the programs (you will need to fill some values, such as the names of your files, thresholds, etc.) and a target for testing it out ('make test').

- You should also download the vision utilities, available from CourseWorks. This contains useful functions for reading and writing images, accessing image data, and drawing lines. Be sure to copy the files **vision_utilities.c** and **vision_utilities.h** to your working directory so that you can compile and use them in your program.