Homework 2
Computational Photography, Fall 2018
Due Date: October 8, 2018
Total Points: 20

This homework contains one written problem, one programming walkthrough and one programming challenge. All submissions are due at the beginning of class on **October 8, 2018**. The written assignment should be turned in as a hard copy at the beginning of class, and the walkthrough and challenge should be submitted according to the instructions in the document titled "**Guidelines for Programming Assignments**" before the beginning of class.

Written Assignments

Problem 1: Show that the extreme (minimum and maximum) values of moment of inertia (E) for a 2D binary object are given by:

$$E_{max} = \frac{1}{2}\left[a + c + \sqrt{b^2 + (a-c)^2}\right],$$

$$E_{min} = \frac{1}{2}\left[a + c - \sqrt{b^2 + (a-c)^2}\right].$$

where a, b, c, and E are as defined in the lecture notes. **(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 Assignments

This programming assignment has one walkthrough and one challenge (each with its own subset of unit tests). Instructions and summary corresponding to these are given below. **runHw2.m** will be your main interface for running your code. Parameters for the different programs or unit tests can also be set in that file. Before submission, make sure you can run all your programs with the command `runHw2("all")` with no errors.

Walkthrough 1: This walkthrough demonstrates how to convert a gray-level image to a binary image and how to use morphological operations (erosion, dilation) to "clean" a binary image. Complete **hw2_walkthrough1.m** and include both the completed script and the generated outputs in your submission. **(2 points)**

Challenge 1: Your task is to develop a vision system that recognizes two-dimensional objects in images. For example, we might be interested in detecting the two objects shown in **two_objects.png** in another image such as **many_objects_1.png**. The vision system should not only determine whether the objects are present in the image, but also compute their positions and orientations.

You are allowed to use any functions demonstrated in the walkthrough or `demoMATLABTricksFun.m` and any other functions explicitly permitted. You may not use any other functions from the Image Processing or Computer Vision toolboxes, nor any functions explicitly prohibited. Refer to **Guidelines for Programming Assignments** for details.

The recognition pipeline is divided into four sub-parts, each corresponding to a program you need to write.

a. Write a program named `generateLabeledImage` that converts a gray-level image to a binary image using a threshold value and segments the binary image into several connected regions:
```
labeled_img = generateLabeledImage(gray_img, threshold)
```
Select any threshold that results in "clean" binary images for the gray-level ones given to you. You should be able to use the same threshold value for all the images. You are allowed to use `im2bw` to threshold the image and `bwlabel` to generate the labeled image. In the labeled image, the background should be labeled as 0, and the maximum value of a label should be equal to the total number of objects. **(2 points)**
Functions not allowed: `im*()`, `bwconncomp()`

b. Write a program named `compute2DProperties` that takes a labeled image from the previous step and computes properties for each labeled object in the image. Store these properties in an objects database.
```
[obj_db, out_img] = compute2DProperties(gray_img,
labeled_img)
```
The generated object database `obj_db` should be a 2D matrix, with each column corresponding to an object and each row corresponding to a property. The first six rows should correspond to the following properties:
1. Object label,
2. Row position of the center,

3. Column position of the center,
4. The minimum moment of inertia,
5. The orientation (angle in degrees between the axis of minimum inertia and the horizontal axis, positive = clockwise from the horizontal axis),
6. The roundness.

You can compute any additional properties if you want. However, these should appear after the six properties mentioned above. Describe the additional properties in your **README** file. The computed properties for all the training objects will serve as your object model database. The output image `out_img` should display the positions and orientations of objects on the original image `gray_img`. Use a dot or star to annotate the position and a short line segment originating from the dot for orientation (refer to `demoMATLABTricksFun` for examples to draw dots, lines, and to save an annotated image). Apply `compute2DProperties` to the labeled image of **two_objects.png**. **(5 points)**
Functions not allowed: `regionprop()`

c. Now you have all the tools needed to develop the object recognition system. Write a program named `recognizeObjects` that recognizes objects from the database:
```
output_img = recognizeObjects(gray_img, labeled_img, database)
```
Your program should compare (using your own comparison criteria) the properties 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 of only the recognized objects on the original image (using dots and line segments, as before). Using the object database generated from **two_objects.png**, test your program on the images **many_objects_1.png** and **many_objects_2.png**. In addition, use **many_objects_1.png** as your object database and find the corresponding objects in the other images. In your **README** file, state the combination criteria and thresholds that you used. **(5 points)**

References

[1] E. W. Weisstein, "Eigenvalue," [Online]. Available: http://mathworld.wolfram.com/Eigenvalue.html.