

EE435: Biometric Signal Processing
Project 5: Feature Extraction & Segmentation

Assigned: Tues 2/22/11

Due: Tues 3/1/11

I. Extracting Features from a Synthetic Binary Image

1. Download the “proj05_1.bmp” binary image. This is a binary image I created for you to learn a way to extract features from a binary image. Imagine that this image was created through some thresholding and other preprocessing of a grayscale iris image. If you view it, it contains a number of white objects (blobs), which range from very small to much larger. For this exercise, you will be doing the programming required to automatically segment the object that best represents the pupil.

What information about pupils can help you look for this pupil?

A pupil is very round, almost circular...maybe not perfectly round, but close to it.

For this image resolution and the distance from the camera to the eye, the radius of the pupil should always fall somewhere between 20 and 30 pixels.

2. First, use morphology to clean up the binary image. Use `imread` to input the image into MATLAB and then follow these steps:
 - a. Fill in gaps in the objects in the image. Use `bwfill` with the “holes” option.
 - b. Get rid of noise (objects much smaller than what you’re looking for). Use `imerode` on the result from the previous step with a suitable structuring element. You can use `strel` to create a structuring element if you wish; or make up a suitable array of 1s and 0s on your own.
 - c. Undo the erosion of the objects that are not noise using `imdilate` with the same structuring element you used in the previous step on the binary image you created in the previous step.
 - d. With the remaining objects, create a label image as follows: take the binary result from the previous step and apply `bwlabel`. This will take a binary image and return a grayscale image, where each object is a different grayscale value: the first object will have all of its pixels being grayscale value 1, the second object will have all values being grayscale value 2, etc. Use `imshow` with the `[]` option to view this label image.
3. After using `bwlabel`, now you can extract geometric features from the label image using `regionprops` (short for region properties). The function `regionprops` will take a label image and can compute a number of useful features about each object (or region—these are the region properties). Type `>> help regionprops` to see what properties it can compute. Some of the region properties that may be useful here are:

-Centroid (the row, column of the center of mass of the object—used to determine its center location)

-MajorAxisLength (the length of the major axis)

-MinorAxisLength (the length of the minor axis, which is perpendicular to the major axis)

-Area (the area of the object...meaning the number of pixels on the object)

-Eccentricity (the roundness of the object, with a value 0.0 up to 1.0...a circle is perfectly round and has eccentricity 0, while a line segment has eccentricity 1).

-EquivDiameter (the equivalent diameter, based on the area computed and $area = \pi r^2$)

To specify which region properties you want MATLAB to compute, use `regionprops` as follows. Suppose variable `bw` is the label image. Then to compute the Centroid and Area of each object in `bw`, use:

```
>> rp=regionprops(bw, 'Centroid', 'Area');
```

This command will create an array of structures called `rp`. The array will have as many structures as there are objects in the label image. To access any of these individual object properties, you must specify which object (i.e., which structure), and what property (or properties). For example, to see all of the computed properties of the 3rd object, you would use:

```
>> rp(3)
```

In an `if` statement, to see if the area of the 5th object is > 50 , you can use a command like:

```
>> if rp(5).Area > 50
```

In an `if` statement, to see if the major axis length of the k^{th} object is > 15 , you can use a command like:

```
>> if rp(k).MajorAxisLength > 15
```

Use `regionprops` to extract some useful features that will help you decide which one of the objects is actually the pupil. Area and Eccentricity can help determine this. You should loop through each object to check its properties, until you find one that is most pupil-like. In the loop, when you decide you've found the pupil, record the index of the pupil object...this index value is the same as its shade of gray in the label image. Once the pupil is determined, we need to know its center row and column, and the radius.

Record your center row/column and radius for the pupil. Convert all of these values to integers. Note that the region property "Centroid" is given as [column # row #] vice [row # column #], and these may not be integers, so you should force them to be integers (e.g., using the `round` function).

Center row = _____ Center column = _____ Radius = _____

4. Now redo the original binary image so that the pupil stands out. One way to do this is to draw a circle in the original image, but with a circle that stands out in color, and has the center location and radius you found. Perform the following steps:

- a. Download the `create_shifted_circle.c` and `create_shifted_circle.m` files from the course website. These are functions I created that will draw a binary circle (white) in a image of all zeros. This is your introduction to a MATLAB "mex" function. A "mex" function is a c-code function that is modified so you can use it as if it were a MATLAB function. These are used to speed up processing. We will address these types of functions more in a later project.

Open the `create_shifted_circle.c` program in a text editor so that you can see how this C program links to MATLAB.

In order to use this mex function, you must compile the c-program in MATLAB. At the MATLAB command line, type:

```
>> mex create_shifted_circle.c
```

It will begin to compile, and you may have to choose which compiler to use. If the Visual Studio compiler appears, choose that one. Once compiled, there will be a file that is called `create_shifted_circle_c.mexw32` in your directory—this is the compiled function. With each mex file, I also write an associated m-file, so that the MATLAB help can be used. Type:

```
>> help create_shifted_circle
```

This shows the help comments that are in the m-file. Open the m-file, and you will see that it calls the mex function to run. Now you have a function that will create a circle anywhere you want, even if it would fall off the edge of your image.

Create a shifted circle image based on where you think the pupil is, and its radius. The dimensions of this image should match the dimensions of the original image you started with. Convert this shifted circle image to logical.

- b. Now create a color image that shows a red circle where you found the pupil to be. Convert the original binary image (“proj05_1.bmp”) to a uint8 image with values 0 or 255 (convert it to double, multiply by 255, then convert back to uint8). It should still appear black and white when you view it using `imshow`. Call this grayscale uint8 image “A”.
 - create a red, green and blue plane of data that looks like A: `R=A, G=A, B=A`;
 - use the logical shifted circle image you created earlier to set the appropriate red values in “R” to 255, and the same G (green) and B (blue) values to 0.
 - concatenate R, G, and B to make a new color image using the `cat` function.
 - `imshow` this color image...the pupil boundary should be red, while all the other objects and the rest of the pupil are white, and the background is black. You may have to zoom in to see the red circle around the pupil, because it is small and the circle is only 1 pixel wide.
- c. Viewing this color image, did you segment the pupil correctly? If the red does not perfectly cover the pupil (I don’t think it will), why not?

II. Segmenting the Pupil from an Iris Image

The goal of this section is to write a complete segmentation program that will work on a number of different iris images. Once your code is written for the 1st image, for all other images after that, you should ONLY change the iris image filename.

1. Download the “0019_L_0003_vga.bmp” binary image. This is an actual iris image. For this image resolution and the distance from the camera to the eye, the radius of the pupil should always fall somewhere between 30 and 90 pixels. These are real eye images, so even though the eye is looking almost directly into the camera, the pupil will not be truly round...be flexible on the value of eccentricity in determining the pupil.
2. Create a binary image that shows the segmentation of the pupil as follows:
 - a. To make the pupil stand out more, invert the iris image so that the pupil is now light instead of dark (e.g. compute the photo-negative). Convert this to a double array. Now, to make the pupil stand out even more, square this inverted image pixel-by-pixel. Use `imshow` with the `[]` option to see what this looks like.
 - b. Create a threshold which is equal to (the mean of the overall inverted and squared image) + two times (the overall standard deviation) ($\mu+2\sigma$...use `mean2` and `std2` to find these). DO NOT HARD-CODE A THRESHOLD VALUE INTO YOUR MATLAB PROGRAM—your program should compute and use this value automatically, and you needn’t even know what the value is! This is important, as you will use your code on other images when this one is done. Threshold the image you created in the previous step with this threshold value.
 - c. Go through the same morphological steps and feature extraction steps you did in part I, steps 2-3.
 - d. Using your pupil center and radius, create a shifted circle logical image and then create a color image using the grayscale iris image (0019_L_0003_vga.bmp). That is, work through step I.4.b again with this grayscale iris image being variable A. Use the logical circle to create a red circle in the color image you just created to display how well you segmented the pupil. Write out this color image as a .bmp file using `imwrite`. Be sure that you can view the image OUTSIDE of MATLAB (e.g., in Windows).
 - e. Record your center location and radius of the pupil found:

Center row = _____ Center column = _____ Radius = _____
 - f. What was the largest (area) object in the image after thresholding—that is, what did it correspond to in the original image?

- g. Now run your program on the other 6 iris images on the course website. Your program should work on all of these images. If your program fails to segment properly, what is the problem? Record the pupil center and radius of each in the table that follows.

Iris Image	Pupil Row Center	Pupil Column Center	Pupil Radius
0001_R_0001_vga			
0002_L_0003_vga			
0019_L_0003_dark_vga			
0025_R_0015_vga			
0026_L_0004_vga			
0030_L_0007_vga			

For this project, fill in the blanks and answer questions in parts I and II, print out all your code you used to create the color image in the last step of part II, and email me your color image of the iris with a red circle around the pupil you found. If you cannot correctly segment the pupil in any of the iris images, let me know so we can fix it.