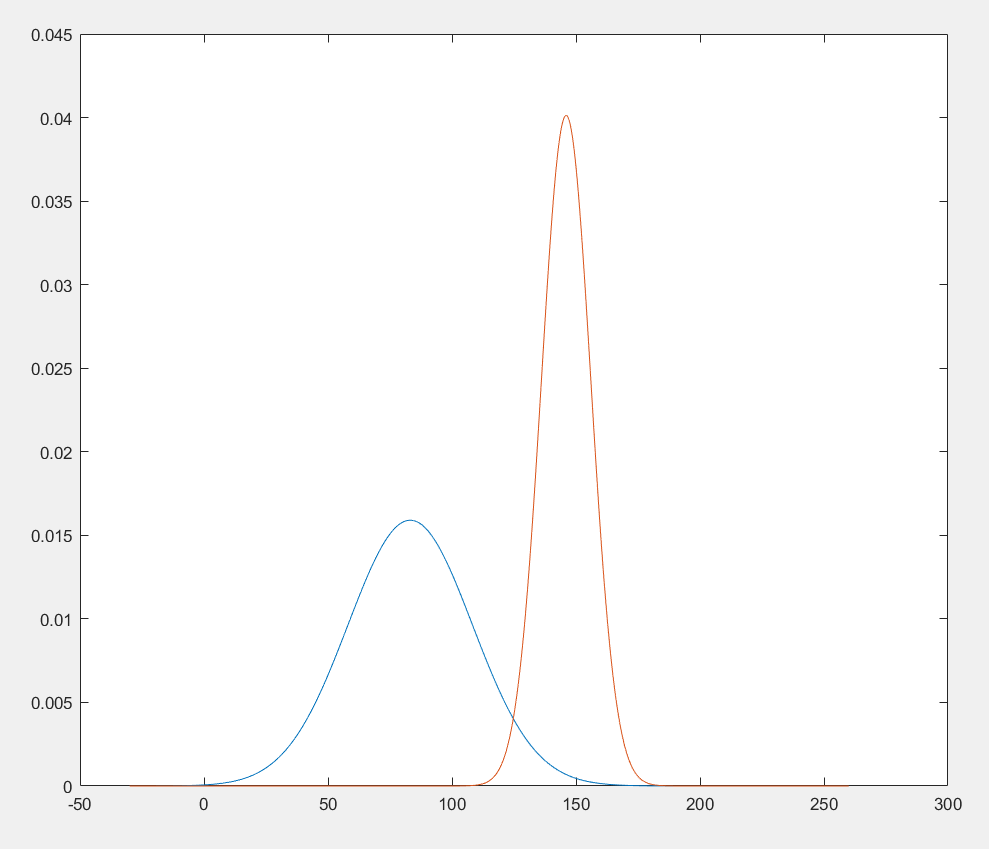
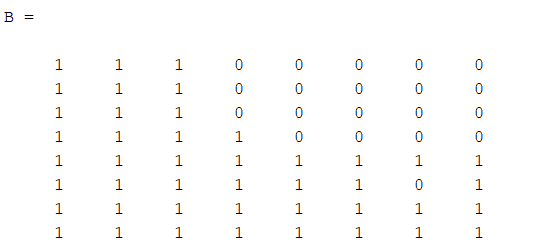
1. Chapter 1.1:
2. For the first image, 12 2x2 neighbors were checked and found that there were 8 external corners, 0 internal corners, and 4 neither. Using the formula number of holes = (E-I)/4 , we get that number of holes = (8-4)/4 = 2. This matches the image, which only has 2 holes.
3. For the second image, 9 2x2 neighbors were checked and found that there were 6 external corners, 0 internal corners, and 3 neither. Using the formula number of holes = (E-I)/4 , we get that number of holes = (6-0)/4 = 1.5. This does not match the image, which has 2 holes.
4. For the third image, 12 2x2 neighbors were checked and found that there were 6 external corners, 2 internal corners, 4 neither. Using the formula number of holes = (E-I)/4 , we get that number of holes = (6-2)/4 = 1. This matches the image, which only has 1 hole.
5. In conclusion, the holecounting formula **works for the first and third images**, but **fails for the second image.**
6. Chapter 1.3, I used 5 squares per inch graph paper for this question after talking with Dr. Liu since I had 8 squares per inch graph paper unavailable to me
   1. For area, using method a) I decided whether a pixel is part of the quarter or not by looking to see if any part of the pixel was inside the quarter. If it was, I counted it. All work is shown on the graph paper, attached in .zip. Using this method, I got the following areas for Quarter 1 (Q1) and Quarter(Q2):
      1. Area of Q1 = 25 pixels units, Area of Q2 = 30 pixel units
      2. Mean for method a) is 27.5 pixel units. Standard deviation is 3.5355
   2. For area using method b), I estimated every boundary pixel to 1/10th and added up all the pixels.
      1. Area of Q1 = 17.9 pixel units, Area of Q2 = 19.2 pixel units
      2. Mean for method b) is 18.55 pixel units. Standard deviation is 0.9192
   3. For circumference using method a) by counting each horizontal and vertical unit of the boundary pixels:
      1. Circumference of Q1 = 34 pixel units, Circumference of Q2 = 38 pixel units
      2. Mean for method a) is 36 pixel units. Standard deviation is 2.8284
   4. For circumference using method b) by counting ever 90 degree angle on the boundary as 1.4 instead of 2:
      1. Circumference of Q1 = 25 pixel units, Circumference of Q2 = 26.6 units
      2. Mean for method a) 25.8 pixel units. Standard deviation is 1.1314
7. Chapter 2.3:
   1. Since the eyeball is a sphere, the formula for its surface area A = 4πr2 can be used. Diameter is 1 inch, so radius is 0.5 inches. So A = 4π(0.5)2 = π. Since the rods and cones only populate 1/ π portion of the inner surface area, the area covered by the rods and cones is π\*(1/ π) = 1 inch. Since, there are **108 rods and cones**, the average size of area covered by a single receptor is 1/108 = **10-8 inches**.We have to keep in mind that foveal receptors are more densely packed so their average area of coverage will be even smaller, while the peripheral receptors are more sparse so their average area of coverage will be larger.
   2. Let us now compare the above to the density of 800x800 square cells in a 1cm x 1cm chip. The density or total number of square cells are (800)2 = **640000/cm2**. The total area of the chip is 1cm2. So, the average size of area covered by a single square cell is 1/640000 = 1.5625 x 10-6 cm. We can convert this to inches for a slightly better comparison by diving the value by 2.54, giving us the value 6.15 x 10-**7** inches**.**
   3. As seen above, the density of sensing elements in the chip are far sparser by multiple orders of magnitudes than the eye.
8. Chapter 2.4:
   1. The area of the rectangle in pixels depends upon what criteria is used to determine whether or not a pixel is 0 or 1 for the binary image output. If we say that a pixel has to fully covered by black for it to count, then the rectangle measure 5.9 x 8.1 pixels would only be read as 5 x 8 pixels, giving us the **smallest area in pixels of 40**. If we say that any portion of a pixel covered by a black counts, then the same rectangle would be read as 6 x 9, giving us the **largest area in pixels of 54**.
9. MATLAB file “visionhw\_q5.m” contains work and graphs for this problem. I have included pictures here as well for reference.
   1. Using MATLAB, the mean of all intensities less than 128 is equal to **83.15**. The standard deviation of all intensities less than 128 is equal to **25.087**.
   2. Using MATLAB, the mean of all intensities ≥ 128 is equal to **146.0227**. The standard deviation of all the intensities ≥ 128 is equal to  **9.9428**.
   3. The gaussian distributions are automatically plotted once the file “visionhw\_q5.m” is run on MATLAB. But, shown below is a picture of the two curves. Blue curve is for all intensities < 128 and orange curve is for all intensities ≥ 128:

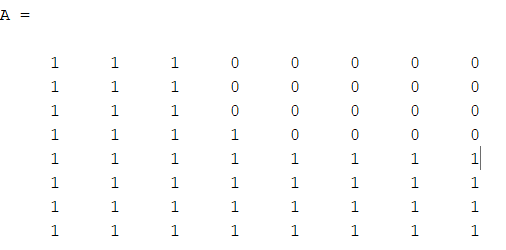


The overlapping of the two curves represents values 0-255 that can be fit into either 0 or 1 after binary image processing based on the threshold value chosen. These are the values that can easily fall from 0 to 1 or 1 to 0 as they lie at the edge of both gaussian distributions.

* 1. The table below, taken from MATLAB, shows an binary array of the original sample with a threshold of 128:



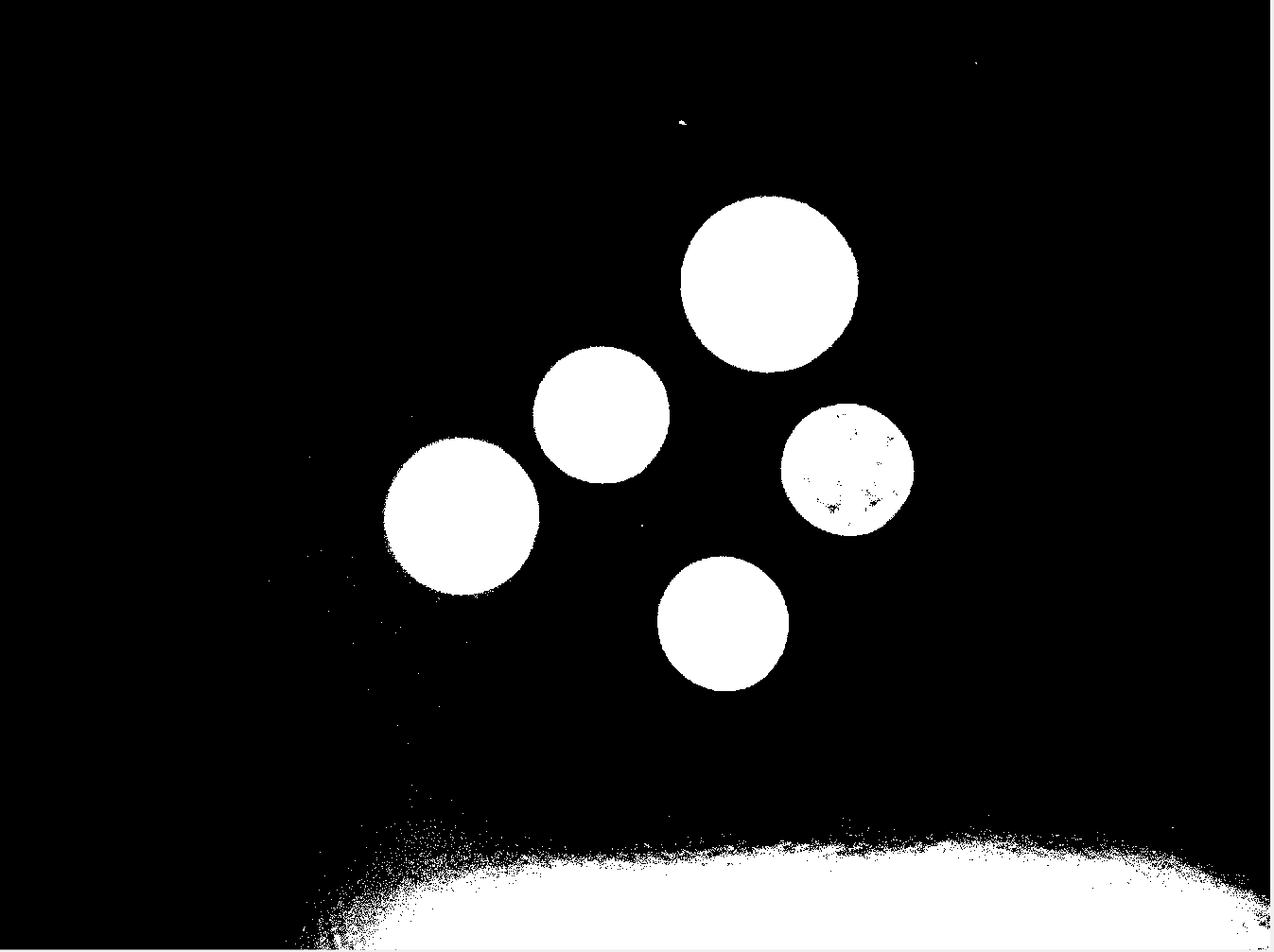
The problematic pixel is at row 5 and column 6 – here there appears to be a 0 surrounded by 1’s. **Thresholding at 128 does not create a “clean” segmentation as this spot has a hole**. So there must be a better threshold such that this inconsistency is taken care of. Looking at the original array in the book at row 5 column 6, we see that this value is 125. So if we put the **threshold at < 125**, we can get rid of this 0 and get a better separation of the white and dark regions of the eye, as shown below (taken from MATLAB):



1. MATLAB file “visionhw\_q6.m” contains the work for this problem.
   1. My original digital image of the coins was .jpg. I used MATLAB to convert to .tif and an online converter to get .gif. The images are inside the .zip file attached with this homework. The files are different sizes. My original .jpg is 3.55 MB in size. The .tif files is 33.5 MB in size. The .gif file is 6.86 MB in size. As for the appearance, they all look the same side-by-side on my desktop screen.

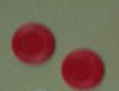


* 1. I chose a **threshold of 60**, which separated the coins from the background. Image is automatically shown when the program “visionhw\_q6.m” is run, but it is shown below:

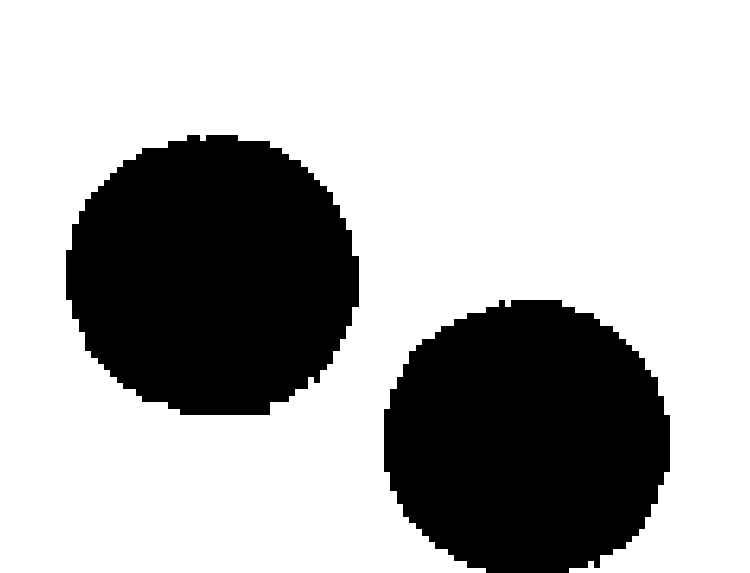


In the image above, we see the 5 coins separated from the background at threshold 50. All of the coins do not look the same however, although they look similar. Most of the coins all look white since they contrast well against the dark background originally. The foreground objects easily surpass the threshold and therefore appear white and the dark background fails to meet the threshold and therefore appears black. The middle right is the only one that looks somewhat different as it has dark patches inside due to the lighting in the original picture.

* 1. I used MATLAB to convert the .gif provided by the professor to a .tif and .jpg. They are also in the .zip file. My work is shown in visionhw\_q6\_part2.m file. The original .gif file is 6.9 KB in size. The converted .tif file is 32.2 KB in size, and the converted .jpg file is 1.5KB in size. Again, they all look the same on the desktop. I combined all color planes into a single intensity image by converting to grayscale first and then applying the thresholds. Below is the original image:



At **threshold < 0.4**, we can isolate the two foreground objects, as shown below. MATLAB file visionhw\_q6\_part2.m automatically shows this picture once run:



Both objects look similar after thresholding possibly because the objects are close in color originally.

* 1. My original picture of coins has **6 “regions**”. 5 of the regions are the coins themselves, but there is also a 6th region at the bottom of the image that was caused by excess lighting when the picture was taken. The brightness of this region at the bottom of the picture bypassed the threshold of 60 and showed up as an extra region. For the instructor provided image, there are only 2 regions, the above seen foreground objects.