Aaron Mercier and Larry Gates

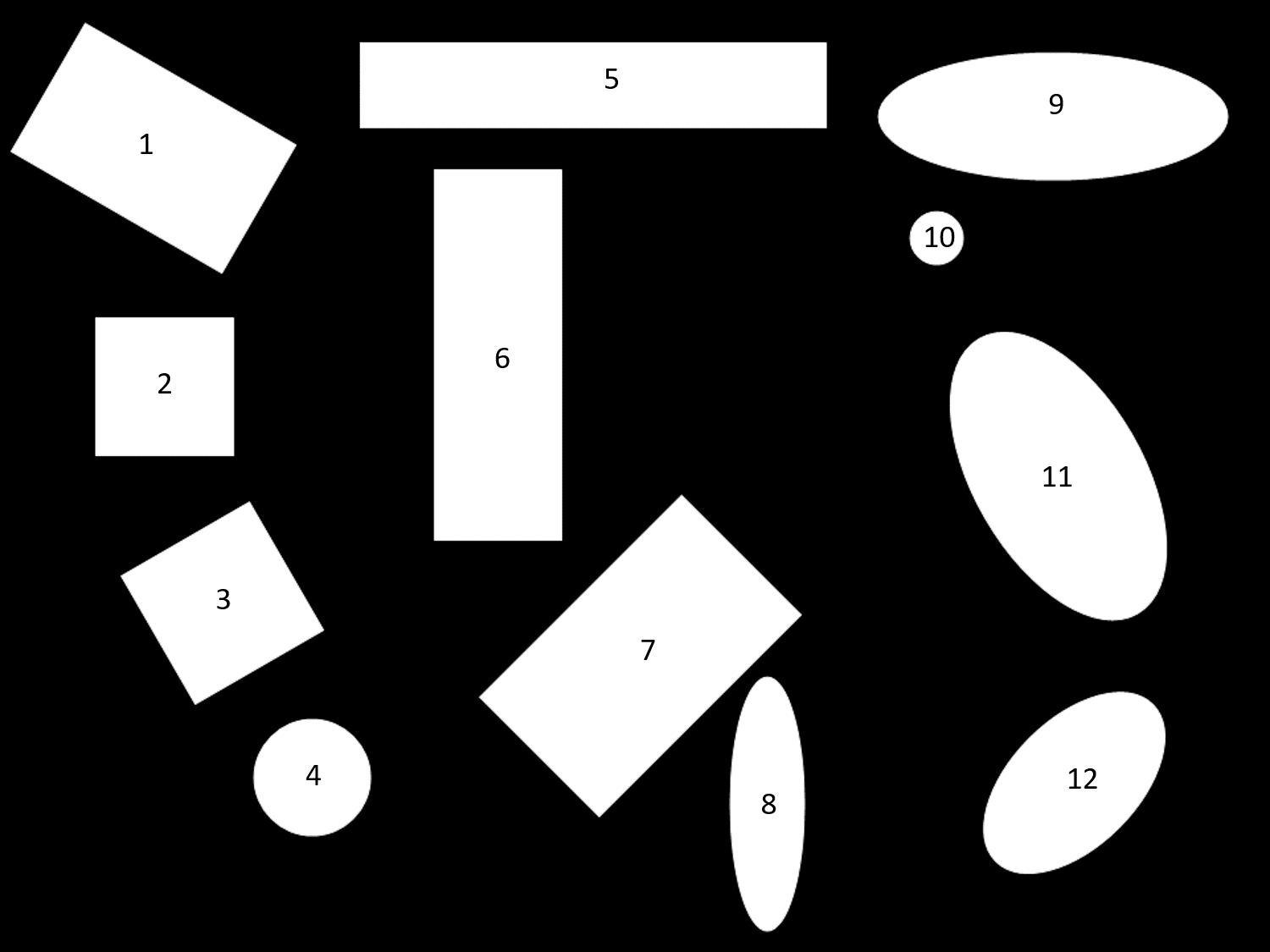
CM1193

CSSE463

**Lab 4**

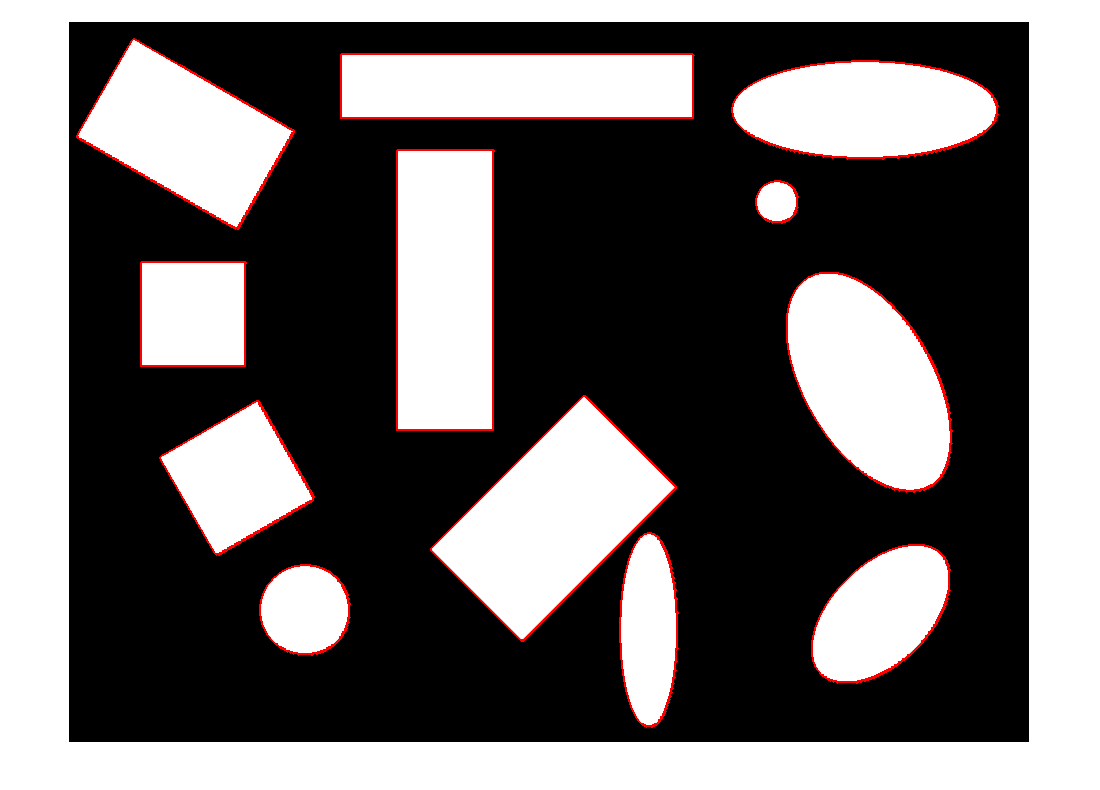
To increase the understanding of shape recognition, our task is to code the algorithm described in class and done by Matlab’s function, *regionprop*, and find the circularity of the identified objects in the image. We can then use the properties of these detected objects to classify them as a shape. The reason for students to tear apart a function that performs the functionality already is to engage the individual components that work behind the scenes. As students interested in image recognition, examining the details of a function allows us to increase our understanding of the tools and use the ideas in other areas.

The shapes are numbered in the order *bwlabel* declared the detected shapes. The numbering is explained in Figure 1. For each shape, we calculate the 4 entries into the covariance matrix, followed by determining the eigenvalues of the matrix. These eigenvalues are then used to determine the elongation of the shape.



**Figure 1:** The numbering order of the shapes in the table.

Afterwards, we find the perimeter pixels using *bwtraceboundary* as displayed in Figure 2. The starting point of a boundary was determined by finding the far left most point of a given region. Finding the perimeter length involved traversing the perimeter array and determining if the distance from the previous pixel was 1 or . The perimeter length and the area of the boundary was used to calculate the circularity () of the shape.



**Fig 2:** Shapes with the perimeter drawn by our algorithm.

In Table 1, the properties are displayed for each region. The process previously described determined the elongation and circularity. The centroid was determined by taking the mean of the columns and the mean of the rows for a region.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Shape** | **Elongation** | **Circularity (C1)** | **Centroid** | **True Class** | **Detected Class** |
| 1 | 1.6326 | 19.3134 | (116.5972, 112.6514) | Rectangle | Rectangle |
| 2 | 1 | 15.7722 | (125, 293) | Square | Square |
| 3 | 1.0006 | 18.0202 | (168.5915, 456.655) | Square | Square |
| 4 | 1.0005 | 13.8334 | (236.5604, 588.6314) | Circle | Circle |
| 5 | 5.4314 | 30.2414 | (449, 65) | Rectangle | Ellipses |
| 6 | 2.897 | 20.8023 | (377, 269) | Rectangle | Rectangle |
| 7 | 1.6739 | 16.9151 | (484.5043, 496.9989) | Rectangle | Rectangle |
| 8 | 3.3498 | 22.347 | (580.5692, 608.6421) | Ellipses | Ellipses |
| 9 | 2.717 | 19.4859 | (796.5547, 88.6193) | Ellipses | Ellipses |
| 10 | 1.0065 | 13.5577 | (708.5767, 180.6469) | Circle | Circle |
| 11 | 1.8601 | 15.9813 | (800.6075, 360.6639) | Ellipses | Ellipses |
| 12 | 1.7334 | 15.4711 | (812.5827, 592.6496) | Ellipses | Ellipses |

**Table 1:** Results of algorithm that detects the shape in form of table.

A scatter plot was used to help visualize and determine the thresholds needed to classify shapes (Graph 1). We determined that squares have a circularity greater than or equal to 15, and elongation between 1 and 1.5 inclusively. Circles have a circularity less than 15, and elongation between 1 and 1.5 inclusively. Using a trendline of the ellipses data points, we found the equation to be To offset the trendline, we changed the y-intercept to be 8.2 to shift upwards. Any points below the equation and had an elongation greater than 1.5 were classified as ellipses. Otherwise, unclassified regions were classified as rectangles.

The results classified all 12 shapes correctly besides a single rectangle. This rectangle had a larger than normal circularity and elongation, causing the data point to be below the equation threshold of an ellipse. Since this point appeared to be a base case or an outlier, the thresholds could not easily separate it.

**Graph 1:** Graph showing relationship between the elongation and circularity for the true class of each shape.