

Pattern Recognition Assignment

CS 480

Prof. Chang

Rasila Thapa

Project: Pattern Recognition - implementing your code without using an image processing library

- Gray-level image which was created by digitizing a picture which contains 2 objects. One is a rectangle, another is a circle.

```
1 3 5 7 9 3 4 4 5 6
1 20 25 24 3 5 6 4 2 4
1 22 35 24 3 5 6 4 5 7
1 20 28 34 2 5 6 4 8 9
1 3 5 7 9 3 4 4 5 6
1 3 5 7 9 3 6 7 4 5 6
1 3 5 7 9 7 8 5 4 9 4 5 6
1 3 5 7 9 9 9 9 8 5 4 5 6
1 3 5 7 9 3 6 4 4 5 6
1 3 5 7 9 3 4 4 5 6
```

- Binary image after applying [histogram analysis](#) and [thresholding](#)

```
0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 1 1 0 0
0 0 0 0 0 1 1 1 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0
```

- Before we can do pattern recognition we need to apply a [Connectivity Analysis](#) to identify the regions in the binary image, and assign a number to each region.

```
Code File Edit Selection View Go Run Terminal Window Help
ConnectivityAnalysis.java — Pattern_Recognition

OPEN EDITORS
Welcome
ConnectivityAnalysis.java
PATTERN_RECOGNITION
ConnectivityAnalysis.java
README.md

ConnectivityAnalysis.java
1 public class ConnectivityAnalysis {
2
3     public static void main(String[] args) {
4         // Example binary image represented as a 2D array
5         int[][] binaryImage = {
6             {0, 0, 0, 0, 0, 0, 0, 0, 0},
7             {0, 1, 1, 0, 0, 0, 0, 0, 0},
8             {0, 1, 1, 0, 0, 0, 0, 0, 0},
9             {0, 1, 1, 0, 0, 0, 0, 0, 0},
10            {0, 0, 0, 0, 0, 0, 0, 0, 0},
11            {0, 0, 0, 0, 1, 0, 0, 0, 0},
12            {0, 0, 0, 0, 1, 1, 0, 0, 0},
13            {0, 0, 0, 0, 1, 1, 0, 0, 0},
14            {0, 0, 0, 0, 1, 0, 0, 0, 0},
15            {0, 0, 0, 0, 0, 0, 0, 0, 0}
16        };
17
18        // Apply connectivity analysis
19        int[][] labeledImage = applyConnectivityAnalysis(binaryImage);
20
21        // Print the labeled image
22        printMatrix(labeledImage);
23    }
24
25    private static int[][] applyConnectivityAnalysis(int[][] binaryImage) {
26        int rows = binaryImage.length;
27        int cols = binaryImage[0].length;
28        int[][] labeledImage = new int[rows][cols];
29        int currentLabel = 0;
30
31        // First pass
32        for (int i = 0; i < rows; i++) {
33            for (int j = 0; j < cols; j++) {
34                if (binaryImage[i][j] == 1) {
35                    int[] neighbors = getNeighborsLabels(labeledImage, i, j);
36                    if (neighbors[0] == 0 && neighbors[1] == 0) {
37                        // Case: No neighbors have labels
38                        currentLabel++;
39                        labeledImage[i][j] = currentLabel;
40                    } else {
41                        // Case: At least one neighbor has a label
42                        int minNeighborLabel = findMinNeighborLabel(neighbors);
43                        labeledImage[i][j] = minNeighborLabel;
44                        updateEquivalenceTable(labeledImage, neighbors, minNeighborLabel);
45                    }
46                }
47            }
48        }
49    }
50}
```

```
Code File Edit Selection View Go Run Terminal Window Help
ConnectivityAnalysis.java — Pattern_Recognition

OPEN EDITORS
Welcome
ConnectivityAnalysis.java
PATTERN_RECOGNITION
ConnectivityAnalysis.java
README.md

ConnectivityAnalysis.java
51 // Second pass
52 for (int i = 0; i < rows; i++) {
53     for (int j = 0; j < cols; j++) {
54         if (labeledImage[i][j] != 0) {
55             labeledImage[i][j] = findRootLabel(labeledImage, labeledImage[i][j]);
56         }
57     }
58 }
59 return labeledImage;
60 }
61
62 private static int[] getNeighborsLabels(int[][] labeledImage, int row, int col) {
63     int[] neighbors = new int[2]; // West and North neighbors
64     if (col > 0) {
65         neighbors[0] = labeledImage[row][col - 1]; // West neighbor
66     }
67     if (row > 0) {
68         neighbors[1] = labeledImage[row - 1][col]; // North neighbor
69     }
70     return neighbors;
71 }
72
73 private static int findMinNeighborLabel(int[] neighbors) {
74     int minLabel = Integer.MAX_VALUE;
75     for (int neighbor : neighbors) {
76         if (neighbor > 0 && neighbor < minLabel) {
77             minLabel = neighbor;
78         }
79     }
80     return minLabel;
81 }
82
83 private static void updateEquivalenceTable(int[][] labeledImage, int[] neighbors, int minNeighborLabel) {
84     for (int neighbor : neighbors) {
85         if (neighbor > 0 && neighbor != minNeighborLabel) {
86             labeledImage[labeledImage.length - 1][neighbor - 1] = minNeighborLabel;
87         }
88     }
89 }
90
91 private static int findRootLabel(int[][] labeledImage, int label) {
92     while (labeledImage[labeledImage.length - 1][label - 1] != 0) {
93         label = labeledImage[labeledImage.length - 1][label - 1];
94     }
95     return label;
96 }
97
98 private static void printMatrix(int[][] matrix) {
99 }
```

```

ConnectivityAnalysis.java
83 private static void updateEquivalenceTable(int[][] labeledImage, int[] neighbors, int minNeighborLabel) {
84     for (int neighbor : neighbors) {
85         if (neighbor > 0 && neighbor != minNeighborLabel) {
86             labeledImage[labeledImage.length - 1][neighbor - 1] = minNeighborLabel;
87         }
88     }
89 }
90
91 private static int findRootLabel(int[][] labeledImage, int label) {
92     while (labeledImage[labeledImage.length - 1][label - 1] != 0) {
93         label = labeledImage[labeledImage.length - 1][label - 1];
94     }
95     return label;
96 }
97
98 private static void printMatrix(int[][] matrix) {
99     for (int[] row : matrix) {
100         for (int value : row) {
101             System.out.print(value + " ");
102         }
103         System.out.println();
104     }
105 }
106 }
107
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Run: ConnectivityAnalysis
/usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/rasilathapa/Desktop/Pattern_Recognition /bin ConnectivityAnalysis
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`
For more details, please visit https://support.apple.com/ab/af208959.
(base) Rasilas-MacBook-Pro:Pattern_Recognition rasilathapa$ /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/rasilathapa/Desktop/Pattern_Recognition /bin ConnectivityAnalysis
Error: Could not find or load main class ConnectivityAnalysis
Caused by: java.lang.ClassNotFoundException: ConnectivityAnalysis
one/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/rasilathapa/Desktop/Pattern_Recognition /bin ConnectivityAnalysis library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/H
0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 2 0 0 0
0 0 0 0 0 2 2 2 0 0
0 0 0 0 0 2 2 2 0 0
0 0 0 0 0 0 2 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
(base) Rasilas-MacBook-Pro:Pattern_Recognition rasilathapa$

```

Output:

```

0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 2 0 0 0
0 0 0 0 0 2 2 2 0 0
0 0 0 0 0 2 2 2 0 0
0 0 0 0 0 0 2 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

```

- Compute attributes and recognize objects. (Note: we assume that only two types of objects exist in the image, one is circular object, another is square object). For each isolated object compute

$$R = (4 * \pi * \text{AREA}) / (\text{PERIMETER} * \text{PERIMETER})$$

If R is equal to 1

then the object is circular

else if R is equal to  $\pi/4$

then the object is square

Note:

For square; Area =  $a^2$

Perimeter =  $4 * a$

$R = (4 * \pi * a^2) / (4 * a * 4 * a) = \pi/4$

For circle; Area =  $\pi * r^2$

Perimeter =  $2 * \pi * r$

$R = (4 * \pi * \pi * r^2) / (2 * \pi * r)^2 = 1$

To calculate the perimeter of an object, count the boundary points which are with up,

down, left, or right background pixels.

To calculate the area of an object, count all the pixels with the label of the object.

```
0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 2 0 0 0
0 0 0 0 0 2 2 2 0 0
0 0 0 0 0 2 2 2 0 0
0 0 0 0 0 0 2 0 0 0
0 0 0 0 0 0 0 0 0 0
```

For Image #a. (represented by 1)

Area = 9, perimeter= 8,  $r = \pi/4$  thus it is square

For image #b (represented by 2)

Area = 8, perimeter= 6,  $r = 2.29$  (if we carry out the math ) or close to 1, thus it is a circle.

Note:

In real situation, it is unlikely that the R value is very precise. Thus, the recognition criteria should be changed to:

If R is **closely** equal to 1

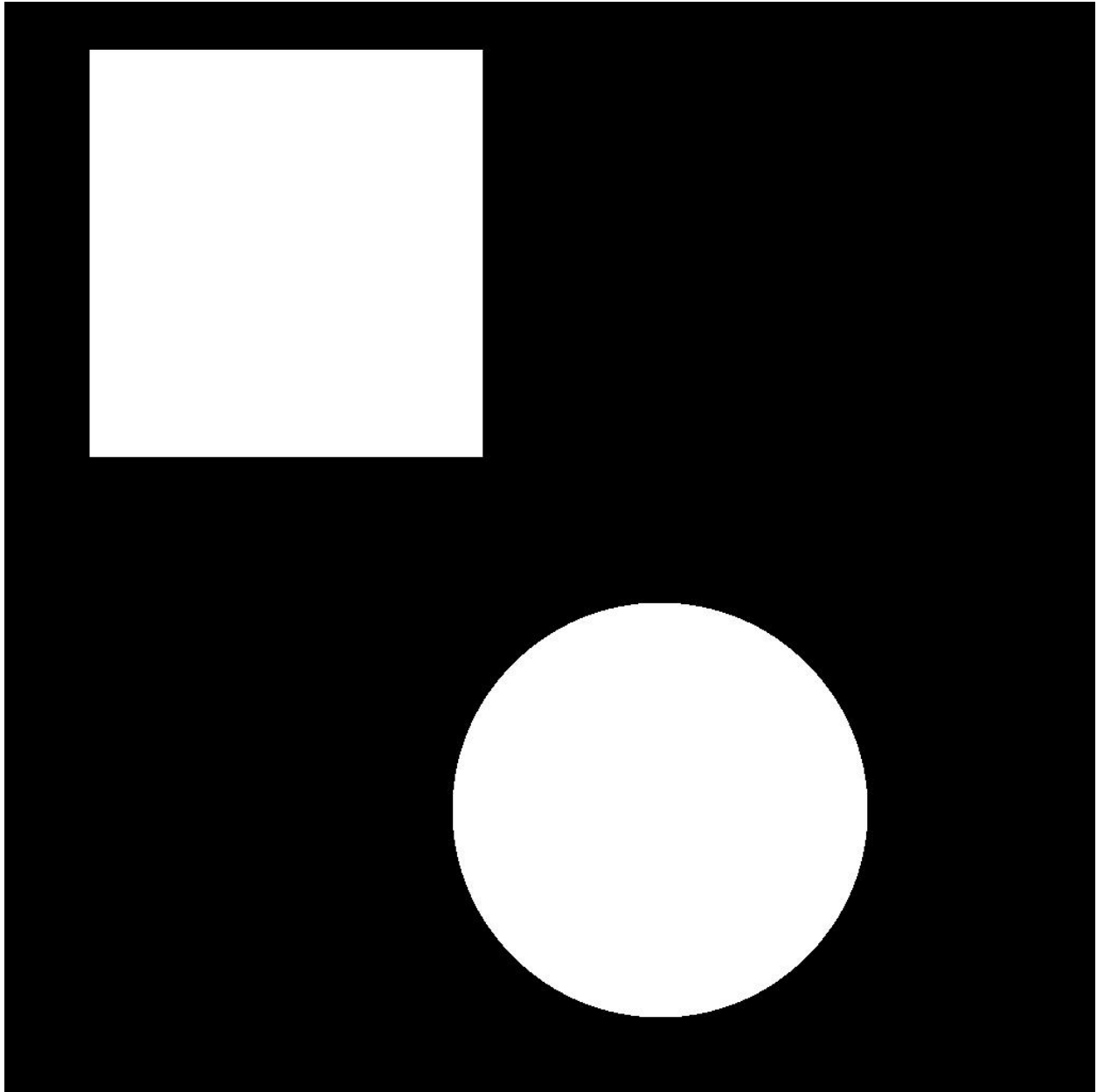
then the object is circular

else if R is **closely** equal to  $\pi/4$

then the object is square

- Apply pattern Recognition in following image

Path to digital image: `Users/rasilathapa/Desktop/Pattern_Recognition/SampleImage.jpeg`



```
1 import java.awt.image.BufferedImage;
2 import java.io.File;
3 import java.io.IOException;
4 import javax.imageio.ImageIO;
5
6 public class RecogImage {
7
8     public static void main(String[] args) {
9         try {
10             File imageFile = new File(System.getProperty("user.dir") + "/SampleImage.jpeg");
11             BufferedImage image = ImageIO.read(imageFile);
12
13             if (image == null) {
14                 System.out.println("Error: Unable to read the image.");
15                 return;
16             }
17
18             // Get image dimensions
19             int width = image.getWidth();
20             int height = image.getHeight();
21
22             // Create a 2D array to store pixel values
23             int[][] pixelArray = new int[width][height];
24
25             // Loop through each pixel and store its RGB value
26             for (int i = 0; i < width; i++) {
27                 for (int j = 0; j < height; j++) {
28                     // Get RGB value of the pixel
29                     int rgb = image.getRGB(i, j);
30
31                     // Store the RGB value in the array
32                     pixelArray[i][j] = rgb;
33                 }
34             }
35
36             // Now, pixelArray contains the RGB values of each pixel in the image
37             // You can further process this array as needed
38
39         } catch (IOException e) {
40             e.printStackTrace();
41         }
42     }
43 }
44 }
```

This will convert the digital image into its pixel array 2D representation, which prepares image data for pattern recognition. We can apply the steps given above in order to recognize the pattern in the digital image.