Pattern Recognition Assignment
CS 480
Prof. Chang

Rasila Thapa

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

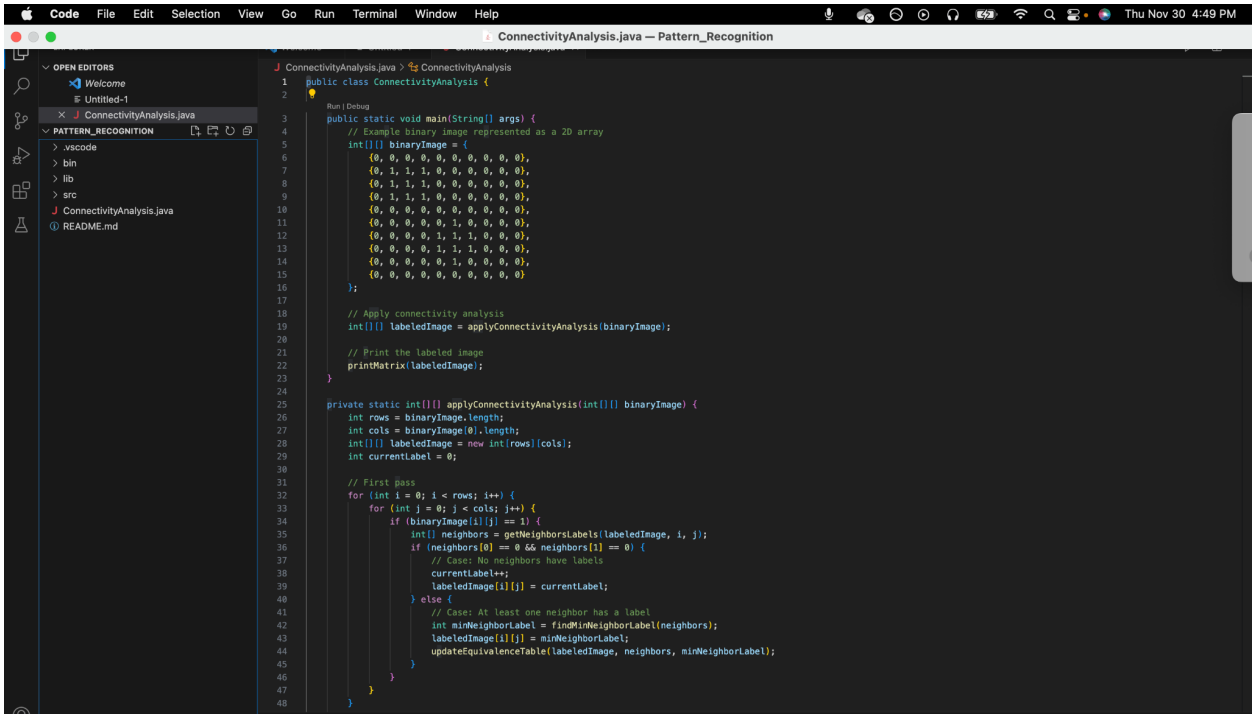
- Gray-level image which was created by digitizing a picture which contains 2 objects. One is a rectangle, another is a circle.

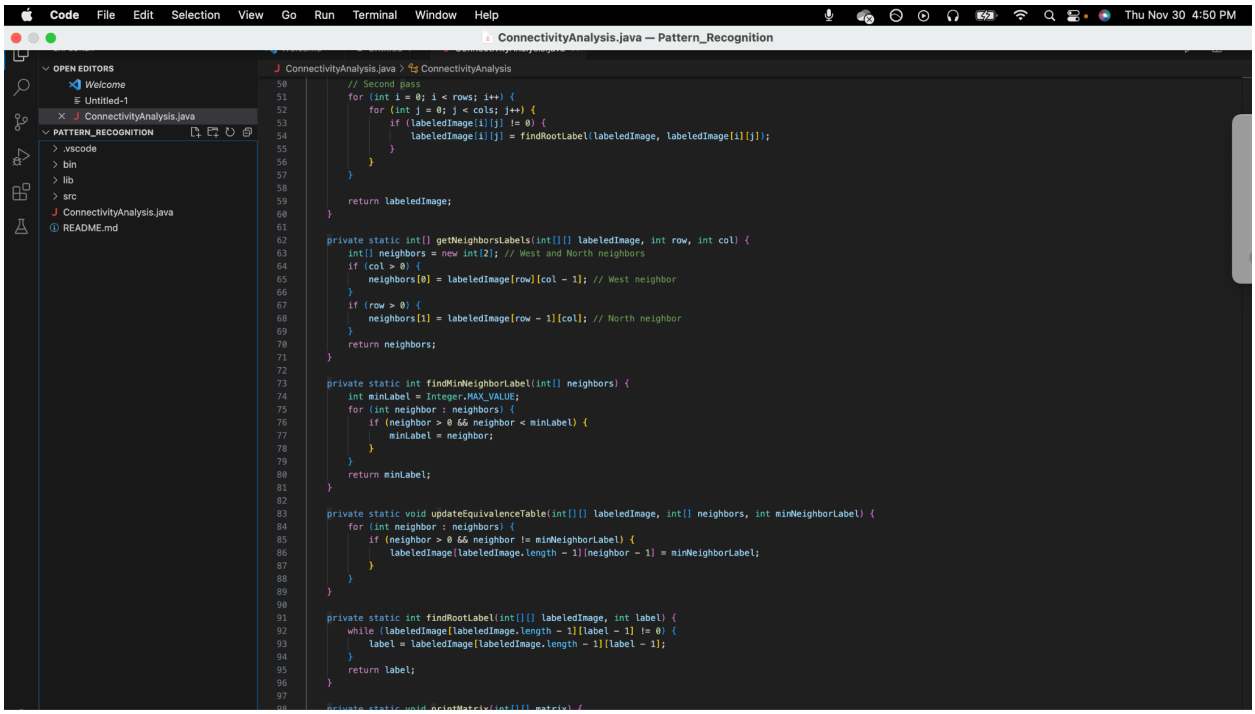
```
      1   3  5  7  9  3  4   4  5  6
1  20 25 24  3  5  6   4  2  4
1  22 35 24  3  5  6   4  5  7
1  20 28 34  2  5  6   4  8  9
1   3  5  7  9  3  4   4  5  6
1   3  5  7  9  3 67   4  5  6
1   3  5  7  9 78 54  94  5  6
1   3  5  7  9 99 98  54  5  6
1   3  5  7  9  3 64   4  5  6
1   3  5  7  9  3  4   4  5  6
```

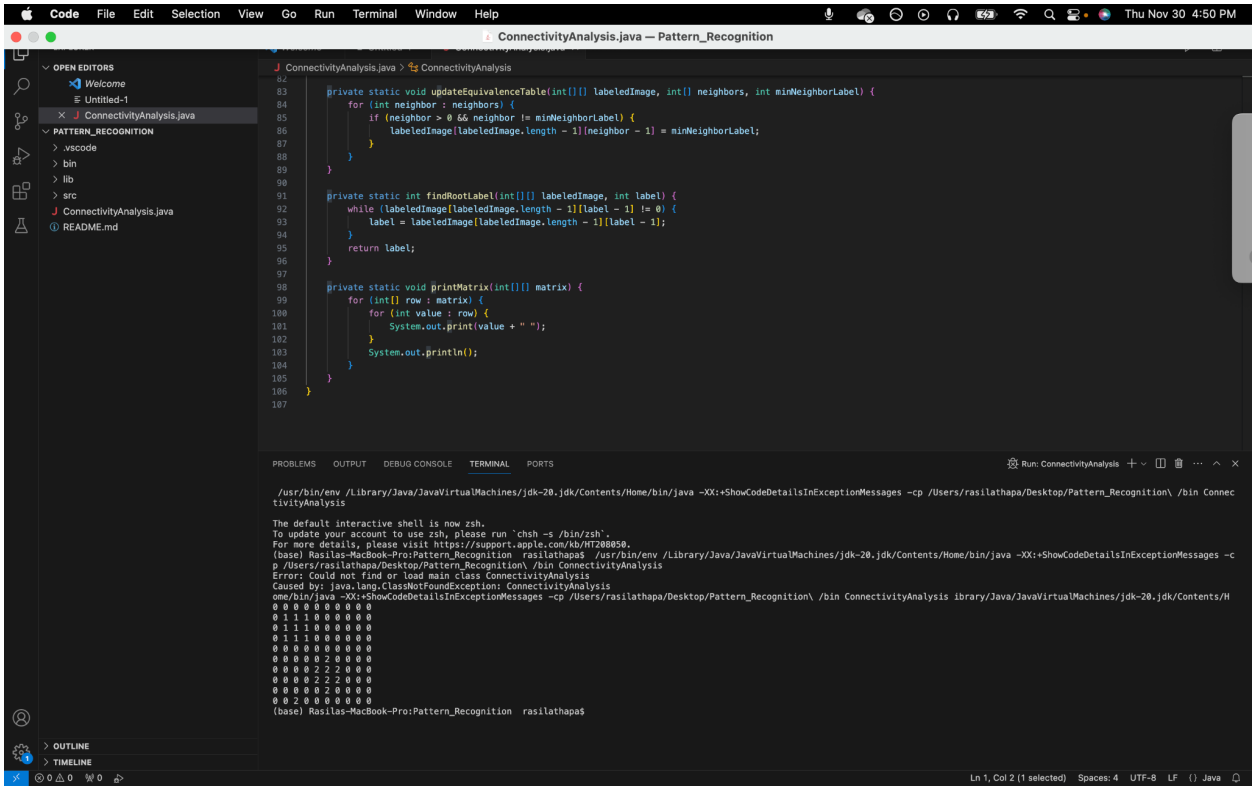
- Binary image after applying histogram analysis and thresholding

```
      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
Thu Nov 30 4:49 PM
ConnectivityAnalysis.java — Pattern_Recognition
OPEN EDITORS
Welcome
Untitled-1
ConnectivityAnalysis.java
PATTERN_RECOGNITION
.vscode
bin
lib
src
ConnectivityAnalysis.java
README.md
ConnectivityAnalysis.java > ConnectivityAnalysis
public class ConnectivityAnalysis {
Run | Debug
public static void main(String[] args) {
// Example binary image represented as a 2D array
int[][] binaryImage = {
{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, 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, 0}
};
// Apply connectivity analysis
int[][] labeledImage = applyConnectivityAnalysis(binaryImage);
// Print the labeled image
printMatrix(labeledImage);
}
private static int[][] applyConnectivityAnalysis(int[][] binaryImage) {
int rows = binaryImage.length;
int cols = binaryImage[0].length;
int[][] labeledImage = new int[rows][cols];
int currentLabel = 0;
// First pass
for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
if (binaryImage[i][j] == 1) {
int[] neighbors = getNeighborsLabels(labeledImage, i, j);
if (neighbors[0] == 0 && neighbors[1] == 0) {
// Case: No neighbors have labels
currentLabel++;
labeledImage[i][j] = currentLabel;
} else {
// Case: At least one neighbor has a label
int minNeighborLabel = findMinNeighborLabel(neighbors);
labeledImage[i][j] = minNeighborLabel;
updateEquivalenceTable(labeledImage, neighbors, minNeighborLabel);
}
}
}
}


Code File Edit Selection View Go Run Terminal Window Help
Thu Nov 30 4:50 PM
ConnectivityAnalysis.java — Pattern_Recognition
OPEN EDITORS
Welcome
Untitled-1
ConnectivityAnalysis.java
PATTERN_RECOGNITION
.vscode
bin
lib
src
ConnectivityAnalysis.java
README.md
ConnectivityAnalysis.java > ConnectivityAnalysis
// Second pass
for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
if (labeledImage[i][j] != 0) {
labeledImage[i][j] = findRootLabel(labeledImage, labeledImage[i][j]);
}
}
}
return labeledImage;
}
private static int[] getNeighborsLabels(int[][] labeledImage, int row, int col) {
int[] neighbors = new int[2]; // West and North neighbors
if (col > 0) {
neighbors[0] = labeledImage[row][col - 1]; // West neighbor
}
if (row > 0) {
neighbors[1] = labeledImage[row - 1][col]; // North neighbor
}
return neighbors;
}
private static int findMinNeighborLabel(int[] neighbors) {
int minLabel = Integer.MAX_VALUE;
for (int neighbor : neighbors) {
if (neighbor > 0 && neighbor < minLabel) {
minLabel = neighbor;
}
}
return minLabel;
}
private static void updateEquivalenceTable(int[][] labeledImage, int[] neighbors, int minNeighborLabel) {
for (int neighbor : neighbors) {
if (neighbor > 0 && neighbor != minNeighborLabel) {
labeledImage[labeledImage.length - 1][neighbor - 1] = minNeighborLabel;
}
}
}
private static int findRootLabel(int[][] labeledImage, int label) {
while (labeledImage[labeledImage.length - 1][label - 1] != 0) {
label = labeledImage[labeledImage.length - 1][label - 1];
}
return label;
}



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 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

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

R = (4 * PI * AREA) / (PERIMETER * 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
```

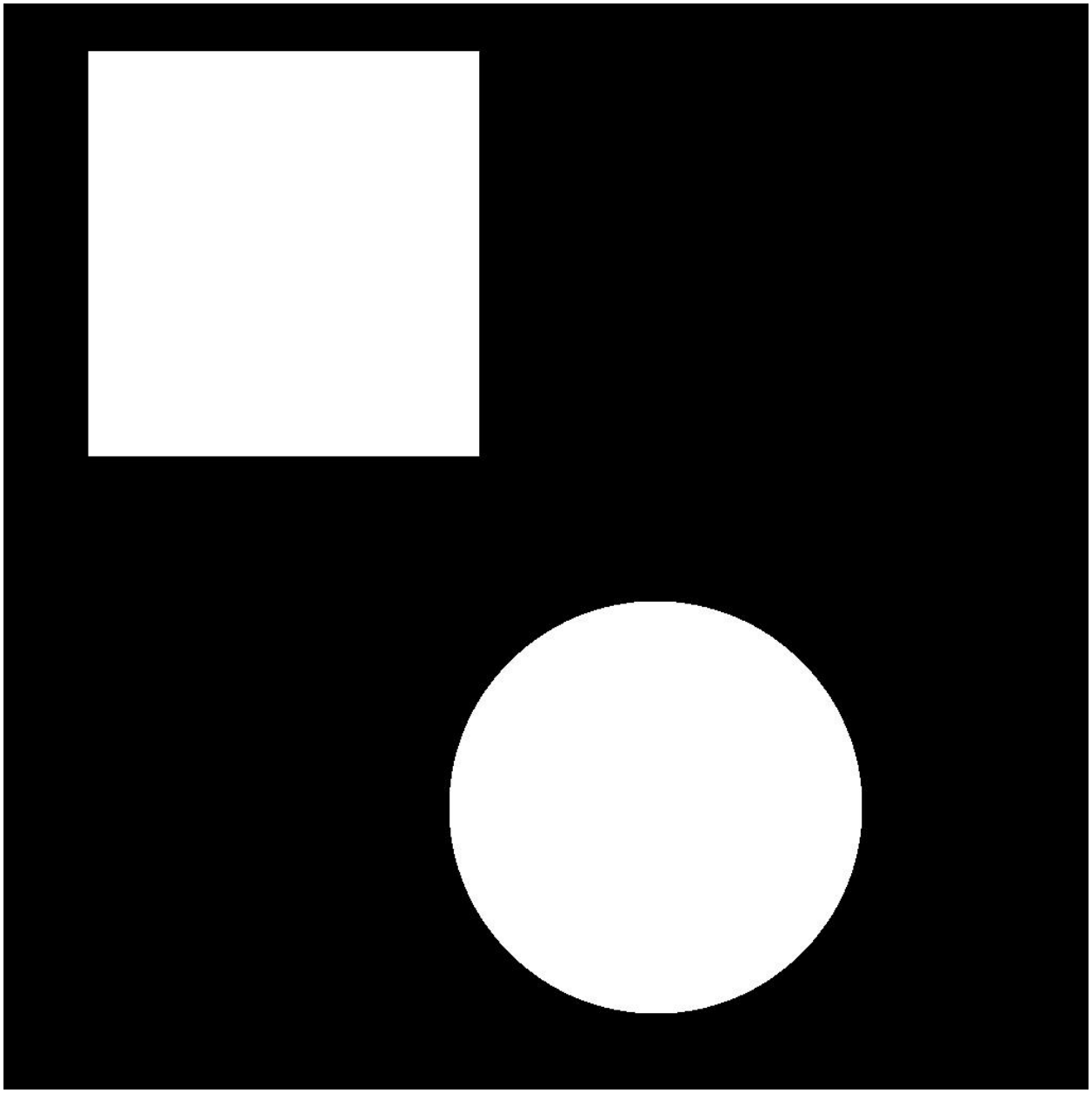
For Image #a. (represented by 1)
Area = 9, perimeter= 8, r= pie/4 thus it is square

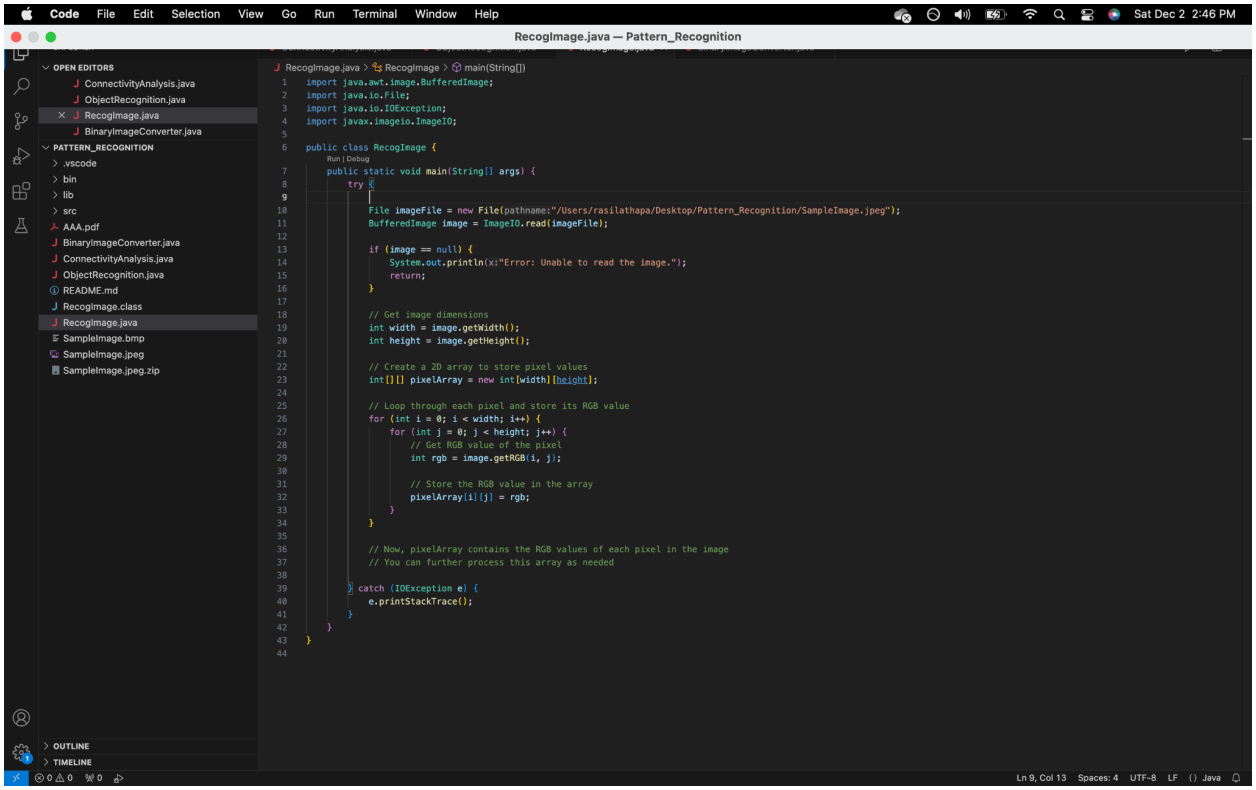
For image #b (represented by 2)
Area = 8, perimeter= 6, r= 2.29 (if we carry out the math) or close to 1, thus it is a circle.

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`





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.