Student: Simon Kong

Project Due date: 11/16/2021

Algorithm Steps

```
*******************************
IV. main (…)
*******************************
Step 0:   inFile ← open input file from args
          outFile1, outFile2 ← open from args
          numRows, numCols, minVal, maxVal ←- read from inFile
          HoughAngle ← 180
          HoughDist ←2 * (the diagonal of the input image)
          imgAry ← dynamically allocate
          HoughAry ← dynamically allocate HoughAry, size of
                     HoughDist by HoughAngle and initialize to zero

Step 1: loadImage (inFile)
Step 2: buildHoughSpace (...) // See algorithm below.
Step 3: prettyPrint (HoughAry, outFile1)
Step 4: determineMinMax (HoughAry)
Step 5: outFile2 ← HoughDist, HoughAngle, HoughMinVal, HoughMaxVal to outFile2
                   // as the header of Hough image
step 6: ary2File (HoughAry, outFile2) // output HoughAry to outFile2

Step 7: close all files
```

Source Code

Main Class

```
package Project_7;
import java.io.BufferedReader;

public class Main {
    public static void main(String[] args) throws IOException {
            if(args.length != 3) {
                System.out.println("Invalid number of arguments.");
                System.exit(0);
            }
    //  Initialize variables
            String inputFile = args[ 0 ];
            String prettyPrintFile = args[ 1 ];
            String houghFile = args[ 2 ];

    //  Open input
            FileReader inputReader = new FileReader( inputFile ) ;
            BufferedReader buffInReader = new BufferedReader( inputReader) ;
            Scanner input = new Scanner( buffInReader ) ;

            FileWriter prettyPrintWriter = new FileWriter(prettyPrintFile);
            BufferedWriter prettyPrintOutput = new BufferedWriter(prettyPrintWriter);

            FileWriter houghWriter = new FileWriter(houghFile);
            BufferedWriter houghOutput = new BufferedWriter(houghWriter);


        try{
    //  initialize variables
            int numRows = 0 ;
            int numCols = 0 ;
            int minVal = 0 ;
            int maxVal = 0 ;

            if( input.hasNextInt() ) numRows = input.nextInt() ;
            if( input.hasNextInt() ) numCols = input.nextInt() ;
            if( input.hasNextInt() ) minVal = input.nextInt() ;
            if( input.hasNextInt() ) maxVal = input.nextInt() ;

            HoughTransform houghObj = new HoughTransform( numRows, numCols, minVal, maxVal ) ;
            houghObj.loadImage(input);
            houghObj.buildHoughSpace();
            houghObj.prettyPrint(houghObj.houghDist, houghObj.houghAngle, houghObj.houghAry, prettyPrintOutput);
            houghObj.determineMinMax(houghObj.houghAry);
            houghObj.printHeader(houghObj.houghDist, houghObj.houghAngle, houghObj.houghMinVal, houghObj.houghMaxVal, houghOutput);
            houghObj.aryToFile(houghObj.houghAry, houghOutput);

        }finally {
            if( input != null ) input.close() ;
            if( prettyPrintOutput != null ) prettyPrintOutput.close();
            if( houghOutput != null ) houghOutput.close();

        }
    }
}
```

HoughTransform

```
package Project_7;

import java.io.BufferedWriter;
import java.io.IOException;
import java.util.Scanner;

public class HoughTransform {
    public int numRows, numCols, minVal, maxVal, houghMinVal, houghMaxVal, houghDist, houghAngle, angleInDegree, offset;
    public int[][] imgAry, houghAry;
    public double angleInRadians;

    public HoughTransform(int rows, int cols, int min, int max) {
        this.numRows = rows;
        this.numCols = cols;
        this.minVal = min;
        this.maxVal = max;
        this.houghMinVal = 999;
        this.houghMaxVal = 0;
        this.houghAngle = 180;
        this.offset = (int) Math.ceil( Math.sqrt( Math.pow(this.numRows, 2) + Math.pow(this.numCols, 2) ));
        this.houghDist = 2 * this.offset;
        this.imgAry = new int [this.numRows][this.numCols];
        this.houghAry = new int [this.houghDist][this.houghAngle];
    }

    public void loadImage(Scanner input) {
        for(int i = 0; i < this.numRows; i++) {
            for(int j = 0; j < this.numCols; j++) {
                if( input.hasNextInt() ) this.imgAry[i][j] = input.nextInt();
            }
        }
    }

    public void buildHoughSpace(){
        for(int x = 0; x < this.numRows; x++) {
            for(int y = 0; y < this.numCols; y++) {
                int p = this.imgAry[x][y];
                if(p > 0) {
                    computeSinusoid(x, y);
                }
            }
        }
    }
```

```java
public double polarDistance(int x, int y, double radians){
    return ( x * Math.cos(radians) ) + ( y * Math.sin(radians) ) + this.offset;
}

public void prettyPrint(int rows, int cols, int[][] ary, BufferedWriter output) throws IOException {
    for(int i = 0; i < rows; i++) {
        for( int j = 0; j < cols; j++) {
            if(ary[i][j] > 0) output.write(Integer.toString(ary[i][j]) + " ");
            else output.write(". ");
        }
        output.write("\n");
    }
}

public void determineMinMax(int[][] arr) {
    for(int i = 0; i < this.houghDist; i++) {
        for(int j = 0; j < this.houghAngle; j++) {
            if(arr[i][j] > this.houghMaxVal) this.houghMaxVal = arr[i][j];
            else if(arr[i][j] < this.houghMinVal) this.houghMinVal = arr[i][j];
        }
    }
}

public void printHeader(int rows, int cols, int min, int max, BufferedWriter output) throws IOException {
    String r = Integer.toString(rows);
    String c = Integer.toString(cols);
    String l = Integer.toString(min);
    String h = Integer.toString(max);
    output.write(r + " " + c + " " + l + " " + h + "\n");
}

public void aryToFile(int[][] arr, BufferedWriter output) throws IOException {
    for(int i = 0; i < this.houghDist; i++) {
        for(int j = 0; j < this.houghAngle; j++) {
            output.write( Integer.toString(arr[i][j]) + " ");
        }
        output.write("\n");
    }
}
```

```java
        public void computeSinusoid(int x, int y) {
            this.angleInDegree = 0;

            while( this.angleInDegree <= 179) {
                this.angleInRadians = (this.angleInDegree / 180.00) * Math.PI;
                double dist = polarDistance(x, y, this.angleInRadians);
                int distInt = (int) dist;
                this.houghAry[distInt][angleInDegree]++;
                this.angleInDegree++;
            }
        }
    }
```

Output Files

Output images are too large to be properly analyzed on Word. I have uploaded them to Google Drive for a better view.

Note: The image is not aligned properly if the values are greater than 9.

Note 2: Links have been tested using incognito mode.

Image 1 Pretty Print File

Image 1 Hough File

Image 2 Pretty Print File

Image 2 Hough File

Image 3 Pretty Print File

Image 3 Hough File

Image 4 Pretty Print File

Image 4 Hough File

Image 5 Pretty Print File

Image 5 Hough File