\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CSCI 381-26 Project 5: Robert, Sobel and Gradient Edge Detectors Language: JAVA

Name: Akshar Patel

Due date: Soft copy: 2/10/2020

Hard copy: 2/12/2020

Submission date: Soft copy: 2/10/2020

Hard copy: 2/12/2020

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Part 1: Algorithm**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IV. main (...)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 0: open the image and read the image header

dynamically allocate mirrorFramedAry and all the edge arrays

step 1: loadImage (mirrorFramedAry)

// load input file to mirrorFramedAry begin at (1,1)

step 2: mirrowFramed (mirrorFramedAry)

step 3: process the mirrorFramedAry, from left to right and top to bottom

begin at (1, 1) // process all pixels!!!

RobertRightDiag(i,j)🡨 abs(convoluteRobert (i,j, maskRobertRightDiag))

RobertLeftDiag(i,j) 🡨 abs (convoluteRobert (i,j, maskRobertLeftDiag))

SobelRightDiag(i,j) 🡨 abs(convoluteSobel (i,j, maskSobelRightDiag))

SobelLeftDiag(i,j)🡨 abs (convoluteSobel (i,j, maskSobelLeftDiag))

GradiantEdge(i,j) 🡨 computeGradient(i,j)

step 4: repeat step 3 until all pixels inside of the frame are processed.

step 5: addTwoArys (RobertRightDiag, RobertLeftDiag, edgeSum)

output RobertRightDiag to deBugOut file // with caption

output RobertLeftDiag to deBugOut file // with caption

output input image header to RobertEdgeOut file

output edgeSum to RobertEdgeOut file // begin at edgeSum[1][1]

step 6: addTwoArys (SobelRightDiag, SobelLeftDiag, edgeSum)

output SobelRightDiag to deBugOut file // with caption

output SobelLeftDiag to deBugOut file // with caption

output input image header to SobelEdgeOut file

output edgeSum to SobelEdgeOut file // begin at edgeSum[1][1]

Step 7: output input image header to GradiantEdgeOut file

output GradiantEdge to GradiantEdgeOut file //begin at GradiantEdge[1][1]

step 8: close all files

**Part 2: Source code**

import java.io.\*;

import java.util.\*;

class Main {

public static int numRows, numCols, minVal, maxVal;

public static int max = 0;

public static int min = 999999;

public static int[][] mirrorFramedAry;

public static int[][] maskRobertRightDiag = {{0,1},{-1,0}};

public static int[][] maskRobertLeftDiag = {{1,0},{0,-1}};

public static int[][] maskSobelRightDiag = {{2,1,0},{1,0,-1},{0,-1,-2}};

public static int[][] maskSobelLeftDiag = {{0,1,2},{-1,0,1},{-2,-1,0}};

public static int[][] RobertRightDiag, RobertLeftDiag, SobelRightDiag, SobelLeftDiag, GradiantEdge,

edgeSum;

static void set2DZero(int[][] Ary){

for(int i=0; i<numRows+2 ; i++){

for(int j=0; j<numCols+2 ; j++){

Ary[i][j]=0;

}

}

}

static void loadImage (int[][] Ary, Scanner file){

for(int i=1; i<numRows+1 ; i++){

for(int j=1 ; j<numCols+1 ; j++){

Ary[i][j] = file.nextInt();

}

}

}

static void mirrowFramed(int[][] Ary){

for(int i=0; i<numRows+2; i++){

Ary[i][0]=Ary[i][1];

Ary[i][numCols+1]=Ary[i][numCols];

}

for(int j=0; j<numCols+2; j++){

Ary[0][j]=Ary[1][j];

Ary[numRows+1][j]=Ary[numRows][j];

}

}

static int convoluteRobert(int i, int j, int[][] Ary){

int sum = 0, temp = j;

for(int r = 0 ; r < 2 ; r++){

for(int c = 0 ; c < 2 ; c++){

sum += (Ary[r][c] \* mirrorFramedAry[i][j]);

j++;

}

i++;

j = temp;

}

return sum;

}

static int convoluteSobel(int i, int j, int[][] Ary){

int sum=0,temp = j;

for(int r = 0 ; r < 3 ; r++){

for(int c = 0 ; c < 3 ; c++){

sum += (Ary[r][c] \* mirrorFramedAry[i-1][j-1]);

j++;

}

j = temp;

i++;

}

return sum;

}

static int computeGradient(int i, int j){

int sum=0;

int x = mirrorFramedAry[i][j], r = mirrorFramedAry[i+1][j], c = mirrorFramedAry[i][j+1];

sum = (int) Math.sqrt(((x-r)\*(x-r))+((x-c)\*(x-c)));

return sum;

}

static void addTwoArys(int[][] Ary1,int[][] Ary2,int[][] Ary3){

for(int i=1; i<numRows+1; i++){

for(int j=1; j<numCols+1; j++){

Ary3[i][j] = Ary2[i][j] + Ary1[i][j];

if(Ary3[i][j] > max){

max = Ary3[i][j];

}

if (Ary3[i][j] < min) {

min = Ary3[i][j];

}

}

}

}

static void imgOut(int[][] Ary, PrintWriter file){

int newmin = min;

int newmax = max;

file.println(numRows+" "+numCols+" "+newmin+" "+newmax);

for(int i=1; i<numRows+1; i++){

for(int j=1 ; j<numCols+1 ; j++){

file.print(Ary[i][j]+" ");

}

file.println();

}

}

static void prettyprint(int[][] Ary, PrintWriter file){

for(int i=1; i<numRows+1; i++){

for(int j=1 ; j<numCols+1 ; j++){

file.print(Ary[i][j]+" ");

}

file.println();

}

}

public static void main(String[] args) throws IOException {

Scanner inFile = new Scanner(new FileInputStream(args[0]));

PrintWriter outFile1 = new PrintWriter(new FileOutputStream(args[1]));

PrintWriter outFile2 = new PrintWriter(new FileOutputStream(args[2]));

PrintWriter outFile3 = new PrintWriter(new FileOutputStream(args[3]));

PrintWriter outFile4 = new PrintWriter(new FileOutputStream(args[4]));

numRows = inFile.nextInt();

numCols = inFile.nextInt();

minVal = inFile.nextInt();

maxVal = inFile.nextInt();

mirrorFramedAry = new int[numRows + 2][numCols + 2];

RobertRightDiag = new int[numRows + 2][numCols + 2];

RobertLeftDiag = new int[numRows + 2][numCols + 2];

SobelRightDiag = new int[numRows + 2][numCols + 2];

SobelLeftDiag = new int[numRows + 2][numCols + 2];

GradiantEdge = new int[numRows + 2][numCols + 2];

edgeSum = new int[numRows + 2][numCols + 2];

loadImage(mirrorFramedAry, inFile);

mirrowFramed(mirrorFramedAry);

for(int i=1; i<numRows+1; i++){

for(int j=1 ; j<numCols+1 ; j++){

RobertRightDiag[i][j] = Math.abs(convoluteRobert(i, j, maskRobertRightDiag));

RobertLeftDiag[i][j] = Math.abs(convoluteRobert(i, j, maskRobertLeftDiag));

SobelRightDiag[i][j] = Math.abs(convoluteSobel(i, j, maskSobelRightDiag));

SobelLeftDiag[i][j] = Math.abs(convoluteSobel(i, j, maskSobelLeftDiag));

GradiantEdge[i][j] = computeGradient(i,j);

if(GradiantEdge[i][j] > max){

max = GradiantEdge[i][j];

}

if (GradiantEdge[i][j] < min) {

min = GradiantEdge[i][j];

}

}

}

imgOut(GradiantEdge, outFile3);

addTwoArys(RobertRightDiag, RobertLeftDiag, edgeSum);

outFile4.println("--------RobertLeftDiag to pretty print file ------"+"\n");

prettyprint(RobertLeftDiag, outFile4);

outFile4.println("\n"+"-------- RobertRightDiag to pretty print file -------"+"\n");

prettyprint(RobertRightDiag, outFile4);

imgOut(edgeSum, outFile1);

set2DZero(edgeSum);

min = 9999999;

max = 0;

addTwoArys(SobelRightDiag, SobelLeftDiag, edgeSum);

outFile4.println("\n"+"-------- SobelLeftDiag to pretty print file ------"+"\n");

prettyprint(SobelLeftDiag, outFile4);

outFile4.println("\n"+"-------- SobelRightDiag to pretty print file -------"+"\n");

prettyprint(SobelRightDiag, outFile4);

imgOut(edgeSum, outFile2);

outFile4.println("\n"+"--------- GradiantEdge to pretty print file -------"+"\n");

prettyprint(GradiantEdge, outFile4);

inFile.close();

outFile1.close();

outFile2.close();

outFile3.close();

outFile4.close();

}

}

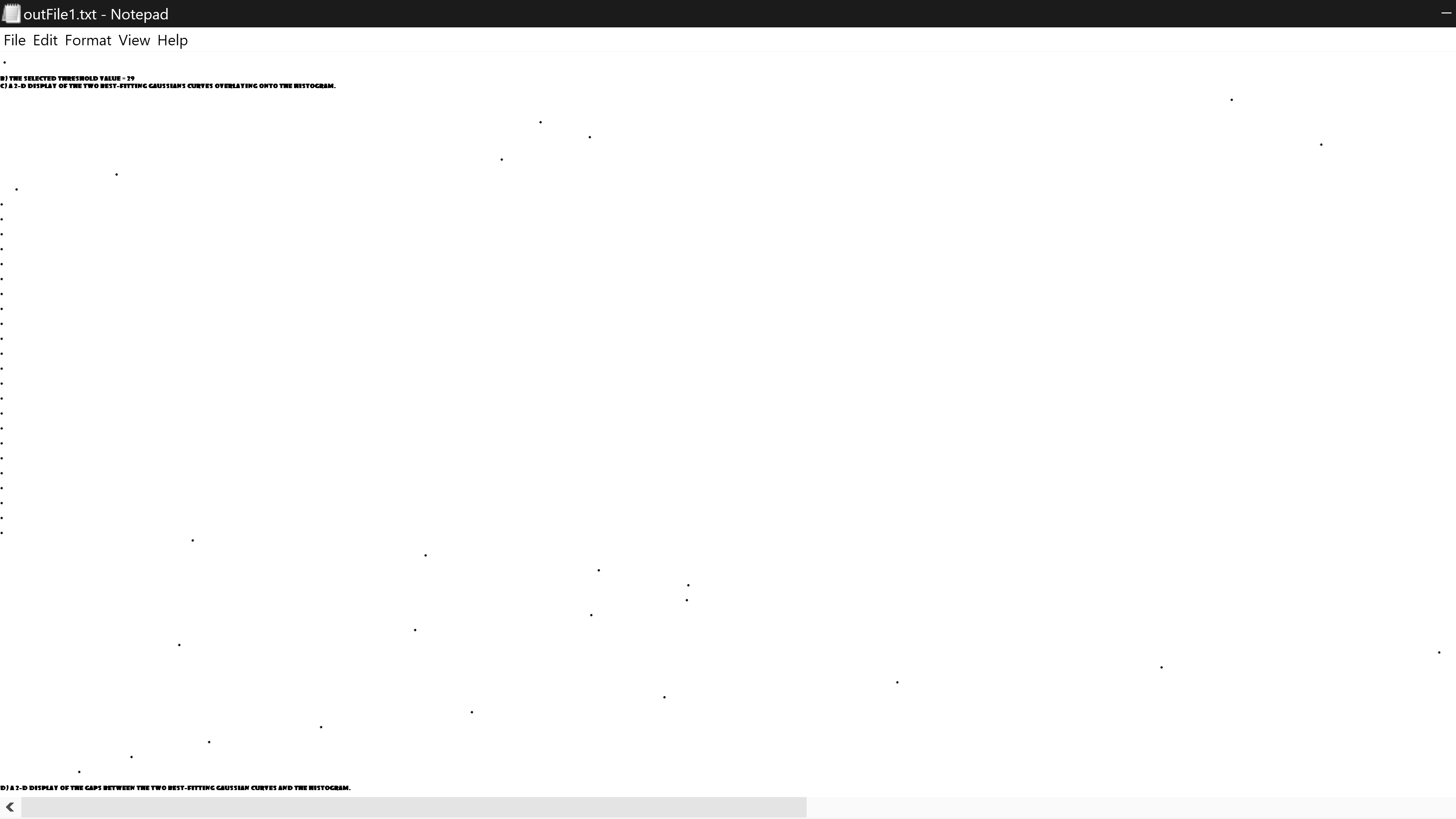
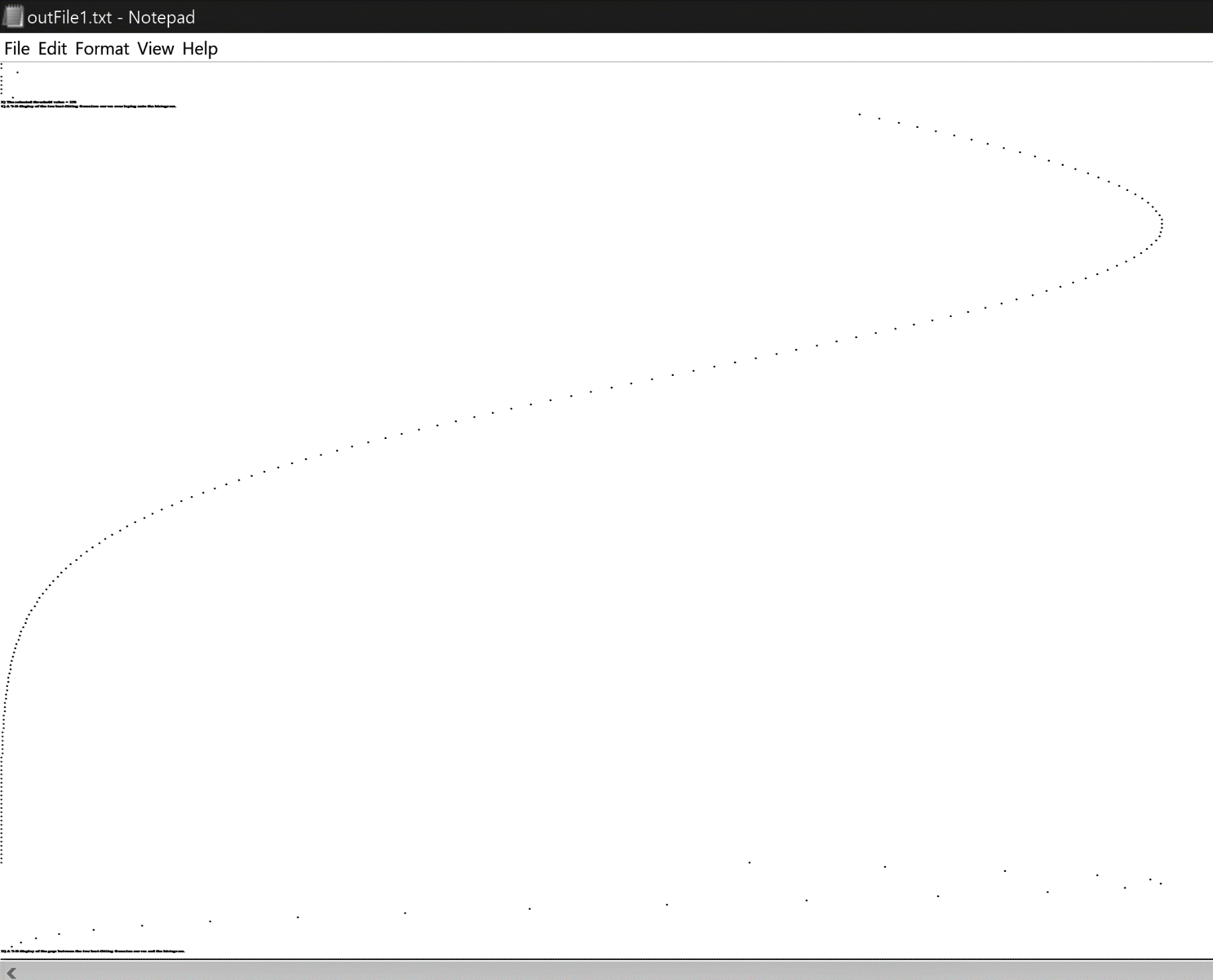
**Part 3: Output**

|  |  |  |
| --- | --- | --- |
| **histogram of RobertEdgeOut** | **histogram of SobelEdgeOut** | **histogram of GradientEdgeOut** |
| 45 45 0 60  0 1897  1 0  2 1  3 0  4 4  5 0  6 0  7 0  8 0  9 0  10 0  11 0  12 0  13 0  14 0  15 0  16 0  17 0  18 0  19 0  20 0  21 0  22 0  23 0  24 0  25 0  26 0  27 0  28 0  29 0  30 0  31 0  32 0  33 0  34 0  35 0  36 0  37 0  38 0  39 0  40 0  41 0  42 0  43 0  44 0  45 0  46 0  47 0  48 0  49 0  50 0  51 0  52 0  53 0  54 0  55 0  56 0  57 0  58 3  59 0  60 121 | 45 45 0 198  0 0  1 0  2 0  3 0  4 0  5 0  6 81  7 0  8 1  9 0  10 0  11 0  12 1007  13 0  14 0  15 0  16 0  17 0  18 593  19 0  :: :  57 0  58 1  59 0  60 56  61 0  62 1  63 0  64 0  65 0  66 2  67 0  68 0  69 0  70 0  71 0  72 35  73 0  74 0  75 0  76 0  77 0  78 21  79 0  80 0  81 0  82 0  83 0  84 0  85 0  86 0  87 0  88 0  89 0  90 0  91 0  92 0  93 0  94 0  95 0  96 0  97 0  98 0  99 0  100 0  101 0  102 2  103 0  104 0  105 0  106 0  107 0  108 3  109 0  110 0  111 0  112 0  113 0  114 1  115 0  116 0  117 0  118 1  119 0  120 62  121 0  122 1  123 0  124 2  125 0  126 1  127 0  128 0  129 0  130 0  131 0  132 38  133 0  134 0  135 0  136 0  137 0  138 21  139 0  ::: :  161 0  162 10  163 0  164 0  165 0  166 0  167 0  168 14  169 0  170 0  171 0  172 0  173 0  174 0  175 0  176 0  177 0  178 0  179 0  180 48  181 0  182 0  183 0  184 0  185 0  186 0  187 0  188 0  189 0  190 0  191 0  192 14  193 0  194 0  195 0  196 0  197 0  198 10 | 45 45 0 45  0 44  1 1493  2 3  3 1  4 340  5 0  6 0  7 0  8 0  9 0  10 0  11 0  12 0  13 0  14 0  15 0  16 0  17 0  18 0  19 0  20 0  21 0  22 0  23 0  24 0  25 0  26 2  27 0  28 0  29 27  30 57  31 26  32 0  33 0  34 2  35 0  36 0  37 0  38 0  39 2  40 0  41 13  42 1  43 13  44 0  45 2 |

Note: In sobel’s histogram, pixel value from 19 to 57 pixel count is 0 and also from 139 to 161 pixel count is 0.

* **overlay bi-Gaussian curves on histogram of**

|  |  |  |
| --- | --- | --- |
| **RobertEdgeOut** | **SobelEdgeOut** | **GradientEdgeOut** |

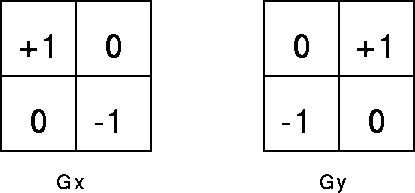


* **pretty print the best threshold result**

|  |  |
| --- | --- |
| **RobertEdgeOut** | **SobelEdgeOut** |
| **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1**  **1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1** | **1 1 1**  **1 1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1**  **1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1**  **1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1**  **1 1 1 1 1**  **1 1 1** |

|  |
| --- |
| **GradientEdgeOut** |
| **1 1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1 1 1 1 1 1 1 1 1 1 1 1 1 1**  **1 1**  **1 1**  **1 1**  **1 1**  **1 1**  **1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1**  **1 1**  **1 1**  **1 1**  **1 1**  **1 1**  **1 1 1 1 1 1 1 1 1 1 1 1 1 1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1 1**  **1 1**  **1 1 1**  **1 1** |

* **with new Robert mask**



|  |  |
| --- | --- |
| 45 45 0 62  0 44  1 0  2 1467  3 1  4 2  5 0  6 0  7 0  8 336  9 0  10 0  11 0  12 0  13 0  14 0  15 0  16 0  17 0  18 0  19 0  20 0  21 0  22 0  23 0  24 0  25 0  26 0  27 0  28 0  29 1  30 60  31 1  32 53  33 1  34 0  35 0  36 0  37 0  38 8  39 0  40 0  41 0  42 0  43 0  44 0  45 0  46 0  47 0  48 0  49 0  50 0  51 0  52 0  53 0  54 0  55 0  56 0  57 0  58 13  59 0  60 26  61 0  62 13 | 1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  1 1  1 1  1 1  1 1  1 1  1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1  1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1  1 1  1 1  1 1  1 1  1 1  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 1 1  1 1 |