Computer Vision Programming Language: Java

Project #2 Bi-Gaussian Auto Threshold

Andres Quintero

Due Date:

Soft copy: 02/13/2020

Hard copy: 02/13/2020

Submission Date:

Soft copy: 02/13/2020

Hard copy: 02/13/2020

Algorithm steps:

Main():

Step 0: Get input and outputfile names from command line arguments

Step 1: Read header values from inFile and assign to respective variables

Step 2: Set offset to (maxVal – minVal) / 10 and set threshVal to offset

Step 3: Dynamically allocate histAry, GaussAry, histGraph, GaussGraph, gapGraph and initialize all values to zero

Step 4: Call loadHist() method

Step 5: Call plotHistGraph() and prettyPrint(histGraph);

Step 6: Call biMeanGauss() method and assign return value to threVal

Step 7: Call bestThreshPlot() method and prettyPrint(GaussGraph)

Step 8: Call prettyPrint(gapGraph)

Step 9: Close all files

**Source code: autoThreshold.java**

import java.util.\*;

import java.io.\*;

import java.lang.Math;

public class autoThreshold{

public static void main(String[] args) {

// Declaring varibles and opening all files

int numRows, numCols, minVal, maxVal;

int offset, thresVal, maxHeight;

Scanner inFile = null;

PrintWriter outFile1 = null;

PrintWriter outFile2 = null;

PrintWriter outFile3 = null;

// Error handling

try {

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

} catch (FileNotFoundException err) {

System.out.println("Error in opening inputFile: " + err);

}

try {

outFile1 = new PrintWriter(args[1]);

} catch (FileNotFoundException err) {

System.out.println("Error in opening outFile1: " + err);

}

try {

outFile2 = new PrintWriter(args[2]);

} catch (FileNotFoundException err) {

System.out.println("Error in opening outFile2: " + err);

}

try {

outFile3 = new PrintWriter(args[3]);

} catch (FileNotFoundException err) {

System.out.println("Error in opening outFile2: " + err);

}

// Reading header and setting values

numRows = inFile.nextInt();

numCols = inFile.nextInt();

minVal = inFile.nextInt();

maxVal = inFile.nextInt();

outFile1.println("Header: " + numRows + " "+ numCols+ " " + minVal+ " " + maxVal);

// Setting offSet and thresVal

offset = (int) (maxVal-minVal) / 10;

thresVal = offset;

// Creating histAry and initializing to zeros

int[] histAry = new int[maxVal+1];

set1DZero(histAry);

// Setting maxHeight from histAry

// Reading from inFile into histAry to get maxHeight

loadHist(histAry, inFile);

maxHeight = 0;

for(int i=0; i<histAry.length; i++){ if(histAry[i] > maxHeight) {maxHeight = histAry[i];}}

// Creating GaussAry and initializing to zeros

int[] GaussAry = new int [maxVal+1];

set1DZero(GaussAry);

// Creating histGraph and initializing to zeros

int[][] histGraph = new int[maxVal+1][maxHeight+1];

set2DZero(histGraph);

// Creating GaussGraph and initializing to zeros

int[][] GaussGraph = new int[maxVal+1][maxHeight+1];

set2DZero(GaussGraph);

// Creating gapGraph and initializing to zeros

int[][] gapGraph = new int[maxVal+1][maxHeight+1];

set2DZero(gapGraph);

// Writing histGraph with prettyPrint and caption

plotHistGraph(histGraph, histAry);

outFile1.println("histGraph: ");

prettyPrint(histGraph, outFile1);

// Computing the best threshold value

thresVal = biMeanGauss(thresVal, histAry, GaussAry, GaussGraph, gapGraph, minVal, maxVal, maxHeight, offset, outFile2, outFile3);

outFile1.println("The best threshold value is: " + thresVal);

bestThresPlot(thresVal, minVal, maxVal, maxHeight, histAry, GaussAry, GaussGraph, gapGraph);

outFile1.println("The Gauss Graph: ");

prettyPrint(GaussGraph, outFile1);

outFile1.println("The Gap Graph: ");

prettyPrint(GaussGraph, outFile1);

// Closing Files

inFile.close();

outFile1.close();

outFile2.close();

outFile3.close();

}

// Functions

static void plotGaps(int[] histAry, int[] GaussAry, int[][] gapGraph, int minVal, int maxVal){

int index, first, last;

index = minVal;

first = Math.min(histAry[index], GaussAry[index]);

last = Math.max(histAry[index], GaussAry[index]);

while(index < maxVal){

while(first < last){

gapGraph[index][first] = 1;

first++;

}

index++;

}

}

static void bestThresPlot(int thresVal, int minVal, int maxVal, int maxHeight, int[] histAry, int[] GaussAry, int[][] GaussGraph, int[][] gapGraph){

double sum1, sum2;

set1DZero(GaussAry);

set2DZero(GaussGraph);

set2DZero(gapGraph);

sum1 = fitGauss(0, thresVal, GaussAry, GaussGraph, histAry, maxHeight);

sum2 = fitGauss(thresVal, maxVal, GaussAry, GaussGraph, histAry, maxHeight);

plotGaps(histAry, GaussAry, gapGraph, minVal, maxVal);

}

static double modifiedGauss(int x, double mean, double var, int maxHeight){

return (double) (maxHeight \* Math.exp(- (((x-mean)\*(x-mean))/ (2\*var))) );

}

static double computeVar(int leftIndex, int rightIndex, double mean, int[] histAry) {

double sum = 0.0;

int numPixels = 0;

int index = leftIndex;

while(index < rightIndex){

sum += (double) histAry[index] \* Math.pow(((double) index - mean), 2);

numPixels += histAry[index];

index++;

}

return (double) sum / (double) numPixels;

}

static double computeMean(int leftIndex, int rightIndex, int maxHeight, int[] histAry) {

int sum, numPixels, index;

maxHeight = 0;

sum = 0;

numPixels = 0;

index = leftIndex;

while(index < rightIndex){

sum += (histAry[index] \* index);

numPixels += histAry[index];

if(histAry[index] > maxHeight) {

maxHeight = histAry[index];

}

index++;

}

return (double) sum / (double) numPixels;

}

static double fitGauss(int leftIndex, int rightIndex, int[] GaussAry, int[][] GaussGraph, int[] histAry, int maxHeight){

double mean, var, sum, Gval, maxGval;

int x, index;

sum = 0.0;

mean = computeMean(leftIndex, rightIndex, maxHeight, histAry);

var = computeVar(leftIndex, rightIndex, mean, histAry);

index = leftIndex;

while(index <= rightIndex){

Gval = modifiedGauss(index, mean, var, maxHeight);

sum += Math.abs(Gval + (double) histAry[index]);

GaussAry[index] = (int) Gval;

GaussGraph[index][(int) Gval] = 1;

index++;

}

return sum;

}

static int biMeanGauss(int thresVal, int[] histAry, int[] GaussAry, int[][] GaussGraph, int[][]gapGraph, int minVal, int maxVal, int maxHeight, int offset, PrintWriter outFile2, PrintWriter outFile3){

double sum1, sum2, total, minSumDiff;

int bestVal = thresVal;

minSumDiff = 999999.0;

while(thresVal < (maxVal - offset)){

set1DZero(GaussAry);

set2DZero(GaussGraph);

set2DZero(gapGraph);

sum1 = fitGauss(0, thresVal, GaussAry, GaussGraph, histAry, maxHeight);

sum2 = fitGauss(thresVal, maxVal, GaussAry, GaussGraph, histAry, maxHeight);

total = sum1 + sum2;

if(total < minSumDiff){

minSumDiff = total;

bestVal = thresVal;

}

thresVal++;

prettyPrint(GaussGraph, outFile2);

plotGaps(histAry, GaussAry, gapGraph, minVal, maxVal);

prettyPrint(gapGraph, outFile3);

}

return bestVal;

}

static void prettyPrint(int[][] graph, PrintWriter outFile){

for (int i=0; i<graph.length; i++) {

for(int j=0; j<graph[0].length;j++){

if(graph[i][j] <= 0){

outFile.print(" ");

} else {

outFile.print("\*");

}

}

outFile.println();

}

}

static void plotHistGraph(int[][] histGraph, int[] histAry){

for (int i=0; i<histGraph.length; i++) {

for(int j=0; j<histGraph[0].length;j++){

histGraph[i][histAry[i]] = 1;

}

}

}

static void loadHist(int[] histAry, Scanner inFile){

while(inFile.hasNext()){

int index = inFile.nextInt();

int value = inFile.nextInt();

histAry[index] = value;

}

}

static void set2DZero(int[][] Ary2D){

for (int i=0; i<Ary2D.length; i++) {

for(int j=0; j<Ary2D[0].length;j++){

Ary2D[i][j] = 0;

}

}

}

static void set1DZero(int[] Ary1D){

for(int i=0; i < Ary1D.length; i++) {Ary1D[i] = 0;}

}

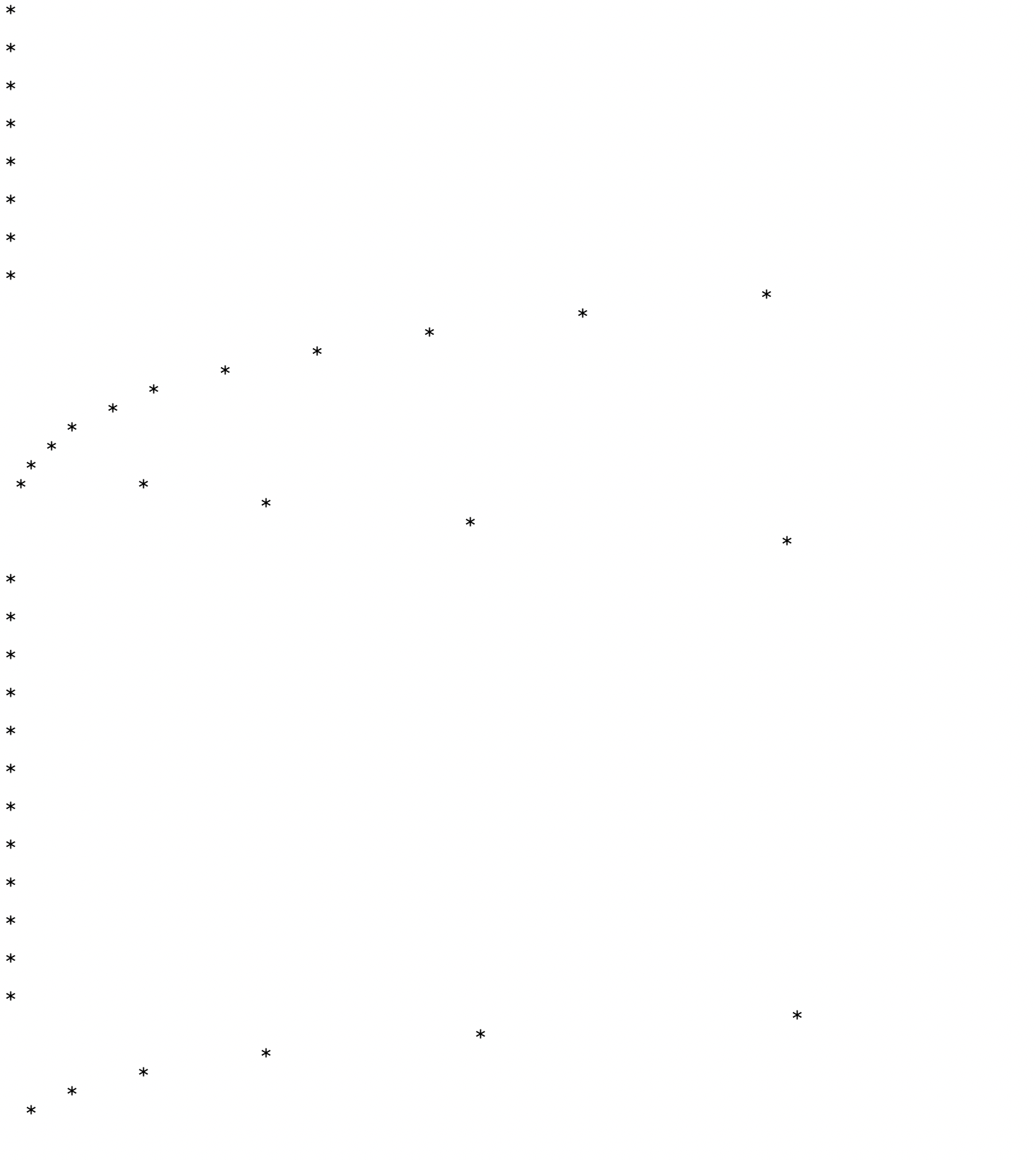
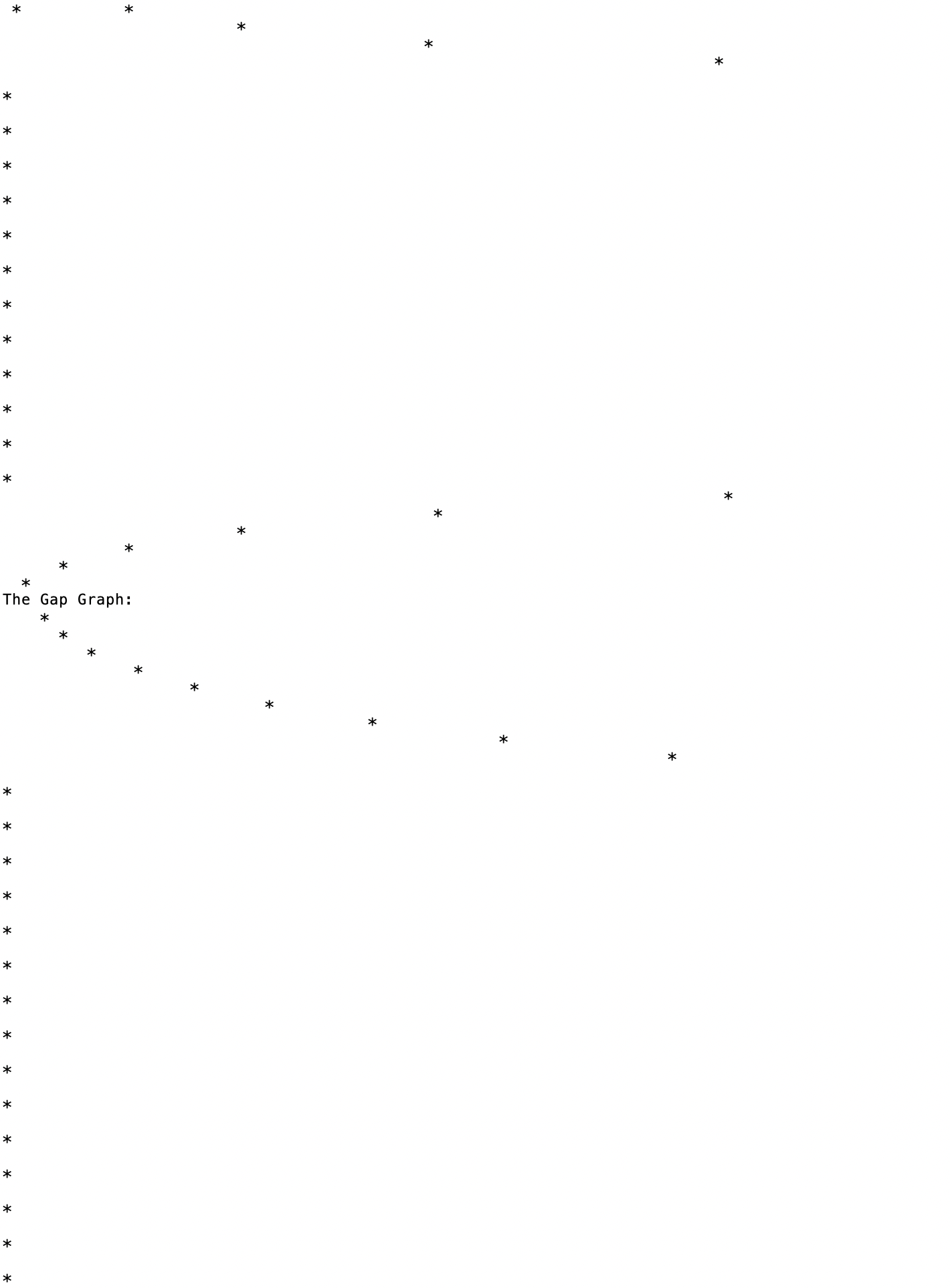
}

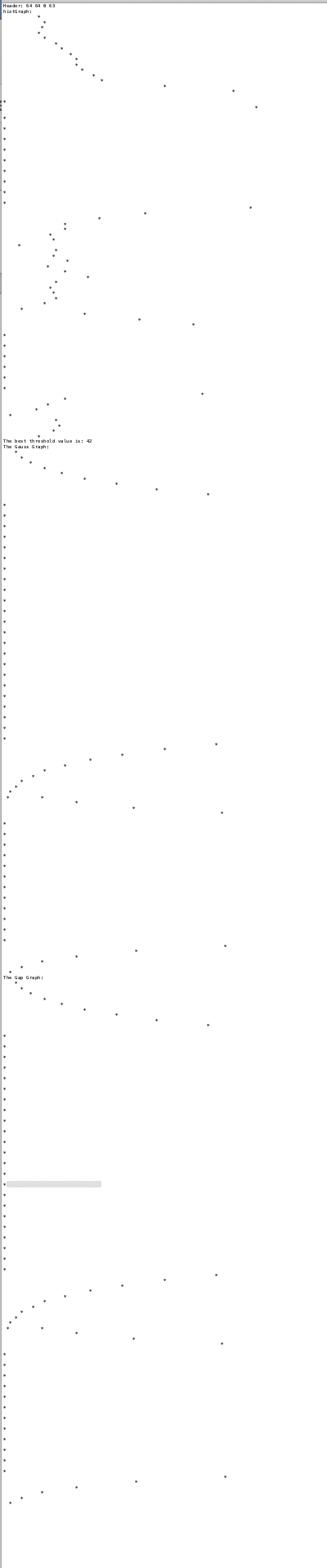
**Output File 1: output1.txt**

A picture containing text, map, sky, large

Description automatically generated

A close up of a logo

Description automatically generated

**Zoom out**