**CSCI 381 – Computer Vision (C++)**

**Program: Project 2.1: GaussianHistogramCurve**

**Name: Isaac Gordon**

**Due Date:**

**Soft copy: 2/12/2019 Tuesday before midnight**

**Hard copy: 2/14/2019 Thursday in class**

step -1: make sure all inputted arguments are valid

step 0: inFile <-- argv[1]

outFile1 <-- argv[2]

outFile2 <-- argv[3]

outFile3 <-- argv[4]

step 1: numRows, numCols, minVal, maxVal <-- read from inFile

step 2: dynamically allocate 1D hist array of sizev maxVal+1

step 3: maxCount <-- loadHist ( )

step 4: dynamically allocate histImg of size (maxVal+1) by (maxCount+1)

dynamically allocate GaussImg of size (maxVal+1) by (maxCount+1)

dynamically allocate GapImg of size (maxVal+1) by (maxCount+1)

// Do Not forget to initialize all three arrays to zero!!!!

step 5: plotHist ( )

prettyPrint (histImg, outFile1) // pretty print histImg to outFile1

step 6: plotGauss( ) // see algorithm below

prettyPrint (GaussImg, outFile2)// pretty print GaussImg to outFile2

step 7: plotGaps( ) // see algorithm below

prettyPrint (GapImg, outFile3) // pretty print GapImg to outFile3

step 8: Close all files

**Code:**

#include<fstream>

#include<iostream>

#include<cmath>

#include <string>

using namespace std;

//data structs

ifstream inFile1;

ofstream outFile1, outFile2, outFile3;

int numRows, numCols, minVal, maxVal, grayScale, maxCount, grayCount;

int\* histAry;

int\* gaussAry;

int\*\* histImg;

int\*\* gaussImg;

int\*\* gapImg;

//function headers

bool endsWith(string str, string ex);

int loadHist();

double computeMean();

double computeVar();

void plotHist();

void plotGauss();

int gaussianFunc(int index, double mu, double sigma);

void plotGaps();

void prettyPrint(int\*\* imgArr, ofstream &outputStream);

int main(int argc, char \*argv[]){

//set arg error message

string BAD\_ARGS = "Correct arguement format is \"<inputFile> <outputFileHist> <outputFileGauss> <outputFileGaps>\".\nBoth should end in \'.txt\'.";

//check for correct number of args

if(argc != 5){

cout << "Wrong number of arguements.\n" << BAD\_ARGS << endl;

exit(1);

}//if

//make sure they are all text files

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

if(!endsWith(argv[i], ".txt")){

cout << argv[i] << " is not a .txt file. Try again." << endl;

exit(1);

}//if

}//for

//STEP 0

inFile1.open(argv[1]);

outFile1.open(argv[2]);

outFile2.open(argv[3]);

outFile3.open(argv[4]);

//STEP 1

inFile1 >> numRows;

inFile1 >> numCols;

inFile1 >> minVal;

inFile1 >> maxVal;

int header[] = {numRows, numCols, minVal, maxVal};

//STEP 2

histAry = new int[maxVal + 1]; //a 1D integer array, size of maxVal + 1

// need to be dynamically allocated at run time.

gaussAry = new int[maxVal + 1]; //a 1D integer array, size of maxVal + 1

// need to be dynamically allocated at run time.

//STEP 3

maxCount = loadHist();

//STEP 4

histImg = new int\*[maxVal + 1]; // a 2-D integer array, initialize to 0

gaussImg = new int\*[maxVal + 1];; // a 2-D integer array, initialize to 0

gapImg = new int\*[maxVal + 1];; // a 2-D integer array, initialize to 0

for(int i = 0; i < maxVal + 1; i++){

histImg[i] = new int[maxCount +1];

gaussImg[i] = new int[maxCount +1];

gapImg[i] = new int[maxCount +1];

//initliza every element on this new array row to 0

for(int j = 0; j < maxCount+1; j++){

histImg[i][j] = 0;

gaussImg[i][j] = 0;

gapImg[i][j] = 0;

}//for

}//for

//STEP 5

plotHist();

prettyPrint(histImg, outFile1);

//STEP 6

plotGauss();

prettyPrint(gaussImg, outFile2);

//STEP 7

plotGaps();

prettyPrint(gapImg, outFile3);

//STEP 8

inFile1.close();

outFile1.close();

outFile2.close();

outFile3.close();

return 0;

}//main

bool endsWith(string str, string ex){

int pos = str.find(ex);

if(pos != str.size() - 4) return false;

return true;

}//endsWith

int loadHist(){

int maxGrayCount = 0;

int sum = 0;

while(!inFile1.eof()){

int index = 0;

int val = 0;

inFile1 >> index;

inFile1 >> val;

sum += val;

histAry[index] = val;

if(val > maxGrayCount) maxGrayCount = val;

}//while

grayCount = sum;

return maxGrayCount;

}//loadHist

double computeMean(){

double m = 0;

for(int i = 0; i < maxVal+1; i++)

m += i \* histAry[i];

return (double) (m/grayCount);

}//computeMean

double computeVar(double mu){

int sum = 0;

for(int i = 0; i < maxVal + 1; i++){

sum += (i - mu)\*(i - mu)\*histAry[i];

}//for

return (double) sum/grayCount;

}//computeVar

void plotHist(){

for(int i = 0; i < maxVal + 1; i++){

histImg[i][histAry[i]] = 1;

}//for

}//plotHist

void plotGauss(){

double mean, var;

mean = computeMean();

var = computeVar(mean);

//compute each gaussian

for(int index = 0; index < maxVal + 1; index++){

int gval = gaussianFunc(index, mean, var);

gaussAry[index] = gval;

gaussImg[index][gval] = 1;

}//for

}//plotGauss

int gaussianFunc(int index, double mu, double sigma2){

int x = index;

double a = 1 / (sqrt(2 \* M\_PI \* sigma2));

double p = -0.5 \* (pow(x - mu,2)/(sigma2));

double g = a \* exp(p);

g \*= grayCount;

return (int) g;

}//gaussianFunc

void plotGaps(){

for(int i = 0; i < maxVal + 1; i++){

int first = min(histAry[i], gaussAry[i]);

int last = max(histAry[i], gaussAry[i]);

while(first < last){

gapImg[i][first] = 1;

first++;

}//while

}//for

}//plotGaps

void prettyPrint(int\*\* imgArr, ofstream &outputStream){

for(int i = 0; i < maxVal + 1; i++){

for(int j = 0; j < maxCount + 1; j++){

int v = imgArr[i][j];

switch (v){

case 1:

outputStream << ".";

break;

default:

outputStream << " ";

break;

}//switch

}//for

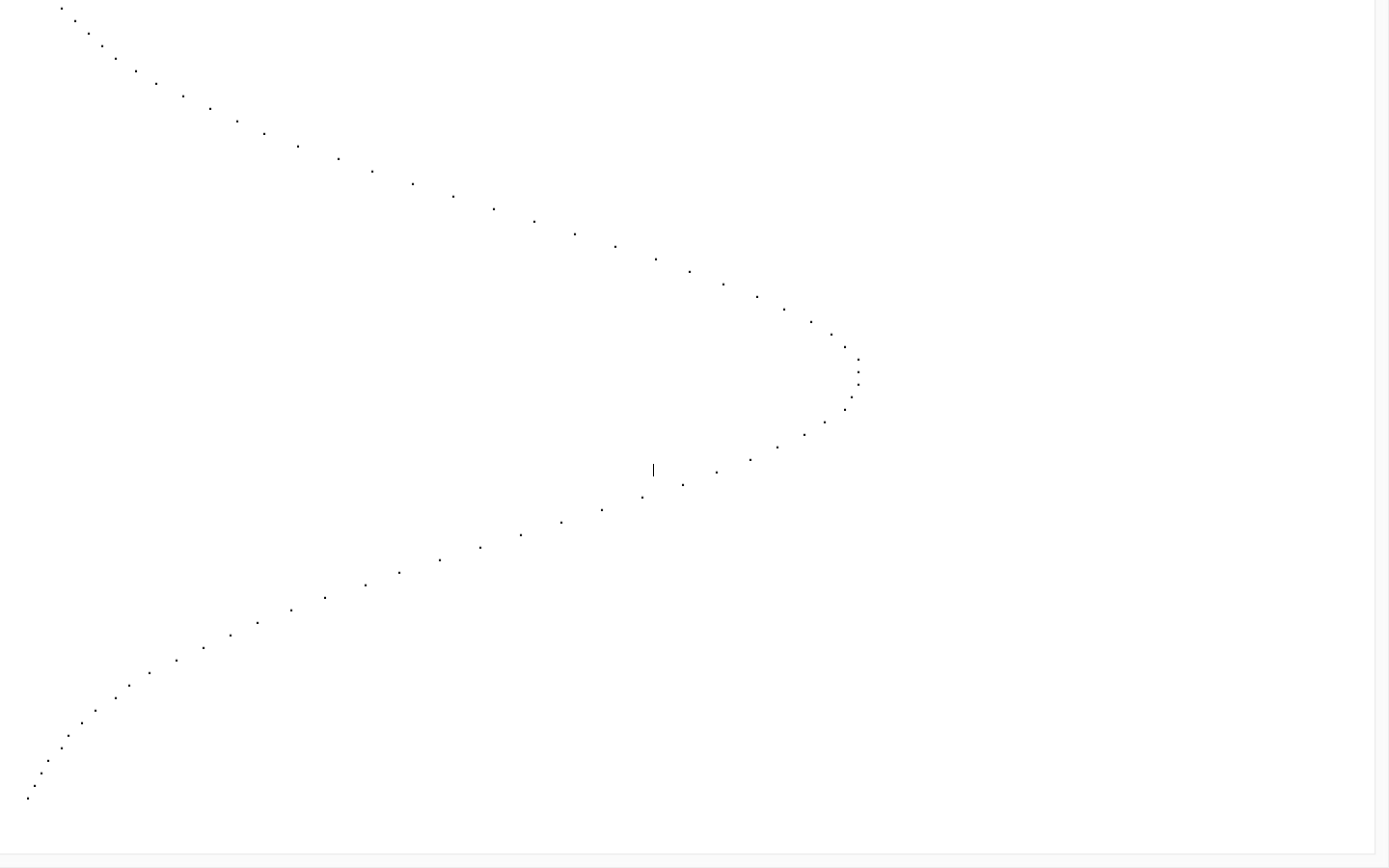
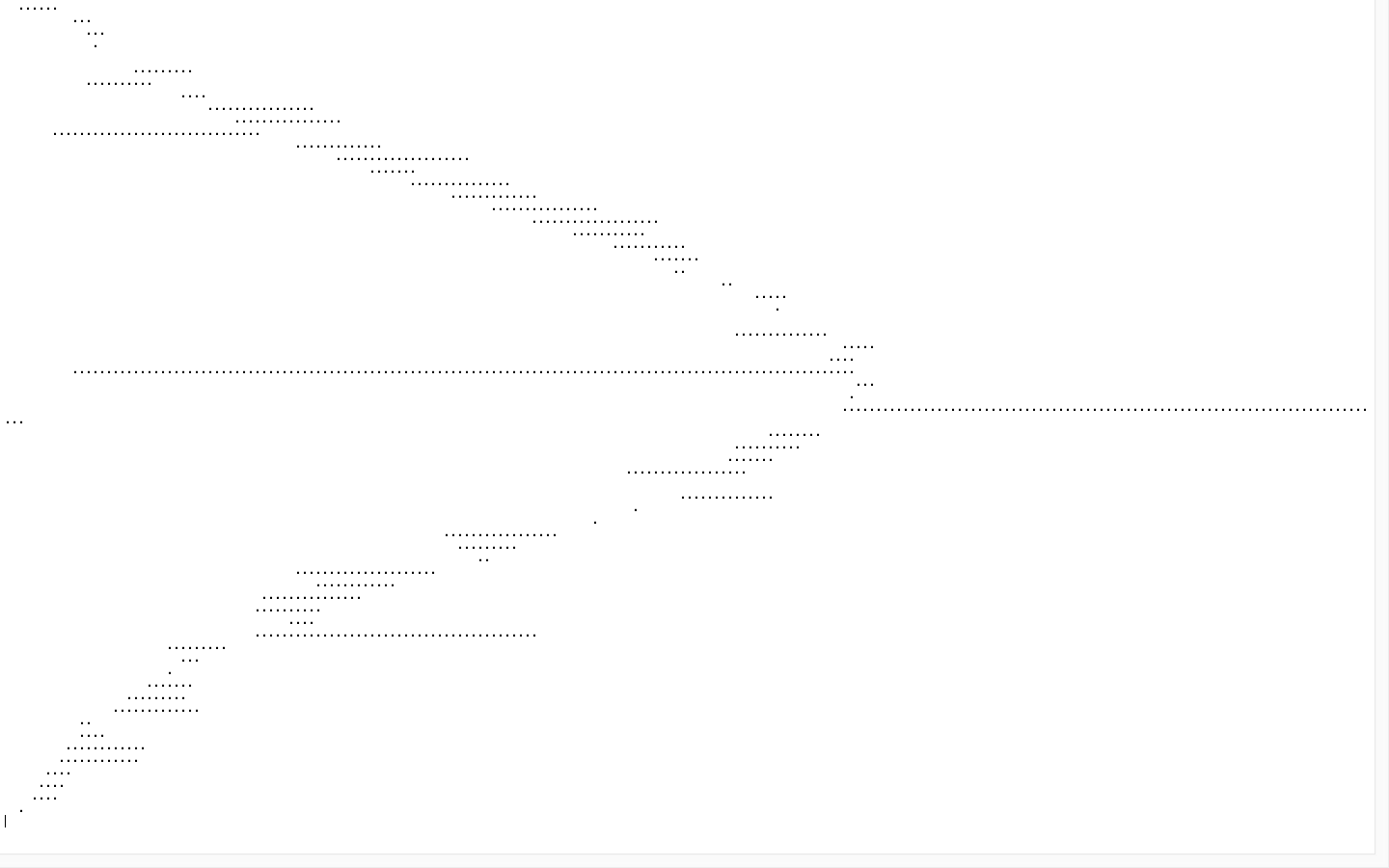
outputStream << endl;

}//for

}//prettyPrint

**Output: Histogram (rotated screenshot)**

**Output: Gaussian (rotated screenshot)**

** Outline: Gaps (rotated screenshot)**