**CSCI 381 – Computer Vision (C++)**

**Program: Project 2.2: Bi-Means Gaussian Histogram**

**Name: Isaac Gordon**

**Due Date:**

**Soft copy: 2/20/2019 Wednesday before midnight**

**Hard copy: 2/21/2019 Thursday in class**

step -1: check for valid args

step 0: inFile <-- argv[1]

outFile1 <-- argv[2]

outFile2 <-- argv[3]

outFile3 <-- argv[4]

outFile4 <-- argv[5]

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

step 2: dynamically allocate 1D histAry of size maxVal+1

set1DZero(histAry)

dynamically allocate 1D GaussAry of size maxVal+1

set1DZero(GaussAry)

offSet <-- (int) (maxVal / 10)

thrVal <-- offSet

step 3: maxCount <-- loadHist ( ) // maxCount is the largest hist[i]

step 4: histImg <-- dynamically allocate 2D histImg array of size

// (maxVal+1) by (maxCount+1)

set2DZero(histImg)

GaussImg <-- dynamically allocate 2D GaussImg array of size

// (maxVal+1) by (maxCount+1)

set2DZero(GaussImg)

GapImg <-- dynamically allocate GapImg array of size

// (maxVal+1) by (maxCount+1)

set2DZero(GapImg)

step 5: plotHist ( )

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

// write caption to indicate it is the original histogram

step 6: median1D ( ) // to smooth the original histogram

step 7: copyArys ( ) // copy the smoothedHistAry to histAry

step 8: set2DZero(histImg) // reset histImg to zero

step 9: plotHist (histAry)

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

// write caption to indicate it is after median filtered

step 10: bestThrVal <-- biMeanGauss (thrVal)

// bestThrVal is the principle method that

// determines the best threshold selection

step 11: output bestThrVal to outFile4 // include caption

step 12: bestThrPlot (bestThrVal) // plotting the result

Step 13: prettyPrint(GaussImg, outFile4)

// output to outFile4 to see the best fitting 2 Gaussian curves

step 14: prettyPrint(GapImg, outFile4)

// output to outFile4 to see the gaps between the 3 best fitting //Gaussian curves and the histogram

step 15: close all files

step 16: delete dynamically allocated resources

**CODE:**

#include<iostream>

#include<fstream>

#include<string>

#include<cmath>

using namespace std;

//data structs

ifstream inFile1;

ofstream outFile1, outFile2, outFile3, outFile4;

int numRows, numCols, minVal, maxVal;

int maxCount; // The largest histAry[i] in the input histogram

int grayCount;

double minDiff; // the minimum sum of absolute “distances” between

// the bi-Gaussians curves and the histogram curve.

int offSet; // offSet is set to one-tenth of the maxVal.

// the assumption: in a bimodal histogram, the first

// modal occupies at least one-tenth from the beginning of the histogram

int thrVal; // Initially, ThrValue is set to offSet,

// the final value of thrVal is the selected threshold value.

int\* histAry; //a 1D integer array (histogram), size of maxVal + 1

// need to be dynamically allocated at run time.

int\* smoothedHistAry; //a 1D integer array (smoothed histogram),

//size of maxVal + 1 need to be dynamically allocated at run time.

int\* GaussAry; //a 1D integer array size of maxVal + 1

// for displaying digital Gaussian curve, need to be dynamically allocated at run time.

int\*\* histImg; // a 2-D integer array of size

//(maxVal+1) by (maxCount+1), initialize to 0, a 2D plot of histogram, for visualization only.

int\*\* GaussImg; // a 2-D integer array of size

// (maxVal+1) by (maxCount+1), initialize to 0

// a 2D plot of Gaussian curve, for visualization only.

int\*\* GapImg; // a 2-D integer array of size

// (maxVal+1) by (maxCount+1), initialize to 0 a 2D plot shows the gaps between Gaussian

//curves and histogram, for visualization only.

//function headers

bool endsWith(string str, string ex);

int loadHist();

void median1D();

void copyArys();

double computeMean(int leftIndex, int rightIndex, int maxCount);

double computeVar(int leftIndex, int rightIndex, double mean);

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

void set1DZero(int \*&ary);

void set2DZero(int \*\*&imgAry);

int biMeanGauss (int thr);

void bestThrPlot(int bestThrVal);

void plotHist();

double fitGauss(int leftIndex, int rightIndex);

void plotGaps(int leftIndex, int rightIndex);

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

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

int c = 0;

//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 != 6){

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

return 1;// 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;

return 1;// exit(1);

}//if

}//for

//STEP 0

inFile1.open(argv[1]);

outFile1.open(argv[2]);

outFile2.open(argv[3]);

outFile3.open(argv[4]);

outFile4.open(argv[5]);

//STEP 1

inFile1 >> numRows;

inFile1 >> numCols;

inFile1 >> minVal;

inFile1 >> maxVal;

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

//STEP 2

set1DZero(histAry);

set1DZero(GaussAry);

offSet = (int) maxVal/10;

thrVal = offSet;

//STEP 3

maxCount = loadHist();

//STEP 4

set2DZero(histImg);

set2DZero(GaussImg);

set2DZero(GapImg);

//STEP 5

plotHist();

outFile1 << "ORIGINAL HISTOGRAM:" << endl << endl;

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

//STEP 6

median1D(); //to smooth the original histogram

//STEP 7

copyArys(); //copy the smoothedHistAry to histAry

//STEP 8

set2DZero(histImg); //reset histImg to zero

//STEP 9

plotHist();

outFile1 << endl << "POST-MEDIAN FILTER HISTOGRAM:" << endl <<endl;

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

//STEP 10

int bestThrVal = biMeanGauss(thrVal);

//STEP 11

outFile4 << "BEST THRESHOLD VALUE: " << bestThrVal << endl << endl;

//STEP 12

bestThrPlot(bestThrVal);

//STEP 13

prettyPrint(GaussImg, outFile4); //output to outFile4 to see the best fitting 2 Gaussian curves

outFile4 << endl << endl;

outFile4 << "============================================================================================================================================================"<<endl;

outFile4 << "x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+x+"<<endl;

outFile4 << "============================================================================================================================================================"<<endl;

//STEP 14

prettyPrint(GapImg, outFile4); //output to outFile4 to see the gaps between the 3 best fitting

//Gaussian curves and the histogram

//STEP 15

inFile1.close();

outFile1.close();

outFile2.close();

outFile3.close();

outFile4.close();

//STEP 16

delete[] GaussAry;

delete[] histAry;

delete[] smoothedHistAry;

delete[] GapImg;

delete[] GaussImg;

delete[] histImg;

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

void median1D(){

smoothedHistAry = new int[maxVal + 1];

for(int i = 2; i <= maxVal - 2; i++){

int x[5];

int index = 0;

for(int j = i - 2; j <= i + 2; j++){

x[index++] = histAry[j];

}//for

//bubble sort array x

bool swapped;

for(int k = 0; k < 4; k++){

for(int r = 0; r < 4 - k - 1; r++){

if(x[r] > x[r+1]){

int temp = x[r];

x[r] = x[r+1];

x[r+1] = temp;

swapped = true;

}//if-then-swap

}//inner-for

if(!swapped) break;

}//outer-for

//set median value into smoothed histogram

smoothedHistAry[i] = x[2];

}//for

}//median1D

void copyArys(){

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

histAry[i] = smoothedHistAry[i];

}//for

} //copyArys

double computeMean(int leftIndex, int rightIndex, int maxCount){

double m = 0;

int count = 0;

for(int i = leftIndex; i < rightIndex + 1; i++){

m += i \* histAry[i];

count += histAry[i];

}//for

grayCount = count;

return (double) (m/count);

}//computeMean

double computeVar(int leftIndex, int rightIndex, double mu){

double sum = 0;

int count = 0;

for(int i = leftIndex; i < rightIndex + 1; i++){

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

count += histAry[i];

}//for

return (double) sum/count;

}//computeVar

void set1DZero(int \*&ary){

delete[] ary;

ary = new int[maxVal + 1];

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

ary[i] = 0;

}//set1DZero

void set2DZero(int \*\*&imgAry){

delete[] imgAry;

imgAry = new int\*[maxVal +1];

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

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

for(int j = 0; j < maxCount + 1; j++) imgAry[i][j] = 0;

}//for

}//set2DZero

int biMeanGauss(int thr){

double sum1, sum2, total;

int bestThr = thr;

minDiff = 999999.0;

//find threshold val that minimizes disparity between origional ans gauss curves

while(thr < (maxVal - offSet)){

set1DZero(GaussAry);

sum1 = fitGauss(0, thr);

sum2 = fitGauss(thr, maxVal);

total = sum1 + sum2;

if(total < minDiff){

minDiff = total;

bestThr = thr;

} //if

thr++;

}//while

return bestThr;

}//biMeansGauss

void bestThrPlot(int bestThrVal){

double sum1, sum2;

set1DZero(GaussAry);

set2DZero(GaussImg);

set2DZero(GapImg);

sum1 = fitGauss(0, bestThrVal);

plotGaps(0, thrVal);

sum2 = fitGauss(bestThrVal, maxVal);

plotGaps(thrVal, maxVal);

}//bestThrPlot

void plotHist(){

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

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

}//for

}//plotHist

double fitGauss(int leftIndex, int rightIndex){

double mean, var, sum;

mean = computeMean(leftIndex, rightIndex, maxCount);

var = computeVar(leftIndex, rightIndex, mean);

sum = 0.0;

for(int i = leftIndex; i <= rightIndex; i++){

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

sum += abs(gval - histAry[i]);

GaussAry[i] = gval;

GaussImg[i][gval] = 1;

}//for

return sum;

}//fitGauss

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(int leftIndex, int rightIndex){

int index = leftIndex;

while(index <= rightIndex){

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

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

while(first < last){

GapImg[index][first] = 1;

first++;

}//while

index++;

}//while

}//plotGaps

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

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

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

int v = plot[i][j];

switch (v){

case 1:

outputStream << "o";

break;

default:

outputStream << " ";

break;

}//switch

}//for

outputStream << endl;

}//for

}//prettyPrint

**Output: Histogram**

Original Smoothed

**Output: Bi-Mean Gaussian**

Best Threshold Value: 33



**Output: Gaps Filled Btwn Histogram and Bi-Mean Gaussian**

