Jonathan Mathew

Algo steps:

Step 0: open inFile, maskFile via argv[]

open imgOutFile, AvgOutFile, MedianOutFile, GaussOutFile via argv[]

thrVal->get from argv[3] Step 1: numRows, numCols, minVal, maxVal->read from inFile

maskRows, maskCols, maskMin, maskMax->read from maskFile

Step 2: dynamically allocate all 1-D and 2-D arrays

Step 3: loadMaskAry (maskFile, maskAry) Step 4: loadImage (inFile, mirrorFramedAry) Step 5: mirrorFraming (mirrorFramedAry) Step 6: imgReformat (mirrorFramedAry, deBugFile)

Step 7: computeAvg (mirrorFramedAry, avgAry) imgReformat (avgAry, deBugFile) binaryThreshold (avgAry, thrAry) prettyPrint (thrAry, AvgOutFile)

Step 8: computeMedian (mirrorFramedAry, medianAry) imgReformat (medianAry, deBugFile) binaryThreshold (medianAry, thrAry) prettyPrint (thrAry, MedianOutFile)

Step 9: computeGauss (mirrorFramedAry, GaussAry) imgReformat (GaussAry, deBugFile) binaryThreshold (GaussAry, thrAry) prettyPrint (thrAry, GaussOutFile)

Step 10: close all files

Output1 mean

A picture containing text, device

Description automatically generated

Output1 median

Text

Description automatically generated

Output1 Gauss

A screenshot of a computer

Description automatically generated with low confidence

Debug.txt

numRows: 5 numCols: 5 minVal: 1 maxVal: 36

5 1 22 3 4

numRows: 5 numCols: 5 minVal: 1 maxVal: 36

8 10 10 10 9

numRows: 5 numCols: 5 minVal: 1 maxVal: 36

8 10 10 10 9

numRows: 5 numCols: 5 minVal: 1 maxVal: 36

8 10 10 10 9

Output2 mean

Chart

Description automatically generated

Output2 median

Chart

Description automatically generated

Output2 Gauss

Chart

Description automatically generated

Debug.txt

numRows: 46 numCols: 46 minVal: 1 maxVal: 63

1 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

numRows: 46 numCols: 46 minVal: 1 maxVal: 63

1 1 3 6 6 8 8 6 6 8 6 6 6 5 3 3 3 3 3 3 3 5 5 5 5 5 6 6 6 6 6 3 3 4 5 5 6 6 5 6 6 4 4 4 3 4

numRows: 46 numCols: 46 minVal: 1 maxVal: 63

1 1 3 6 6 8 8 6 6 8 6 6 6 5 3 3 3 3 3 3 3 5 5 5 5 5 6 6 6 6 6 3 3 4 5 5 6 6 5 6 6 4 4 4 3 4

numRows: 46 numCols: 46 minVal: 1 maxVal: 63

1 1 3 6 6 8 8 6 6 8 6 6 6 5 3 3 3 3 3 3 3 5 5 5 5 5 6 6 6 6 6 3 3 4 5 5 6 6 5 6 6 4 4 4 3 4

Main.cpp

#include <iostream>

#include <fstream>

#include <cmath>

#include <algorithm>

#include <sstream>

using *namespace* std;

*class* enhancement{

*public:*

*int* numRows, numCols, minVal, maxVal, maskRows, maskCols, maskMin, maskMax, thrVal,

\*\*mirrorFramedAry, \*\*avgAry, \*\*medianAry, \*\*gaussAry, \*\*thrAry, neighbor[25], mask[25], maskWeight;

enhancement(*int* *r*, *int* *c*, *int* *min*, *int* *max*, *int* *mr*, *int* *mc*, *int* *mmin*, *int* *mmax*, *int* *thr*){

numRows = *r*;

numCols = *c*;

minVal = *min*;

maxVal = *max*;

maskRows = *mr*;

maskCols = *mc*;

maskMin = *mmin*;

maskMax = *mmax*;

thrVal = *thr*;

mirrorFramedAry = new *int*\*[numRows+4];

avgAry = new *int*\*[numRows+4];

medianAry = new *int*\*[numRows+4];

gaussAry = new *int*\*[numRows+4];

thrAry = new *int*\*[numRows+4];

for (*int* i = 0; i < numRows+4; i++){

mirrorFramedAry[i] = new *int*[numCols];

avgAry[i] = new *int*[numCols];

medianAry[i] = new *int*[numCols];

gaussAry[i] = new *int*[numCols];

thrAry[i] = new *int*[numCols];

}

}

// void printArr(int \*\* arr, ofstream \*out){

// for (int i = 0; i < numRows +4; i++)

// {

// for (int j = 0; j < numCols+4; j++)

// {

// \*out << arr[i][j] << " ";

// }

// \*out << endl;

// }

// \*out << endl; \*out << endl;

// }

*void* binaryThreshold(*int* \*\**inAry*){

for (*int* i = 0; i < numRows+4; i++)

{

for (*int* j= 0; j < numCols+4; j++)

{

// cout<<inAry[i][j]<<" ";

if(*inAry*[i][j]>=thrVal){

thrAry[i][j]=1;

}else{

thrAry[i][j]=0;

}

}

// cout<<endl;

}

// printArr(thrAry);

}

*void* loadImage(ifstream \**infile*, ofstream \**outfile*){

string line;

*int* val;

*int* r=0;

while (getline(\**infile*, line)){

*int* c=1;

istringstream set(line);

while (set >> val ){

// cout << val << " ";

mirrorFramedAry[r+1][c+1] = val;

c++;

}

r++;

// cout <<endl;

}

}

*void* mirrorFraming(ofstream \**out*){

//TOP ROW

*int* r=2;

for (*int* c = 2; c < numCols+2; c++){

// cout << mirrorFramedAry[r][c] << " ";

mirrorFramedAry[r-1][c] = mirrorFramedAry[r][c];

mirrorFramedAry[r-2][c] = mirrorFramedAry[r][c];

}

//bottom row

r = numRows+1;

// cout <<endl << r <<endl;

for (*int* c = 2; c < numCols+2; c++){

// cout << mirrorFramedAry[r][c] << " ";

mirrorFramedAry[r+1][c] = mirrorFramedAry[r][c];

mirrorFramedAry[r+2][c] = mirrorFramedAry[r][c];

}

//leftmost

*int* c=2;

for (r = 0; r < numRows+4; r++){

// cout<<mirrorFramedAry[r][c] << endl;

mirrorFramedAry[r][c-1] = mirrorFramedAry[r][c];

mirrorFramedAry[r][c-2] = mirrorFramedAry[r][c];

}

//rightmost

c=numCols+1;

for (r = 0; r < numRows+4; r++){

mirrorFramedAry[r][c+1] = mirrorFramedAry[r][c];

mirrorFramedAry[r][c+2] = mirrorFramedAry[r][c];

}

// printArr(mirrorFramedAry,out);

}

*void* computeMedian(){

*int* i =2, j=2;

while (i < numRows+2){

while (j < numCols+2){

loadNeighbor(i, j);

sort(neighbor, neighbor+25);

medianAry[i][j] = neighbor[12];

j++;

}

i++;

}

}

*void* computeAvg(){

*int* i =2;

while (i < numRows+2){

*int* j=2;

while (j < numCols+2){

*int* total=0;

loadNeighbor(i,j);

for (*int* i = 0; i < 25; i++){

total += neighbor[i];

}

// cout<<"total is " << total<<endl <<"avg " << total/25 <<endl;

avgAry[i][j] = total/25;

j++;

}

i++;

}

// for (int i = 0; i < numRows+4; i++){

// for (int j = 0; j < numCols+4; j++)

// {

// cout << avgAry[i][j] << " ";

// }

// cout << endl;

// }

}

*void* computeGauss(){

*int* i =2;

while (i < numRows+2){

*int* j=2;

while (j < numCols+2){

loadNeighbor(i, j);

gaussAry[i][j] = convolution();

j++;

}

i++;

}

}

*int* convolution(){

*int* result =0,i=0;

while (i<25){

result +=neighbor[i] \* mask[i];

i++;

}

return result/maskWeight;

}

*void* imgReformat(*int* \*\**inAry*, ofstream \**output*){

\**output* << "numRows: " << numRows << " numCols: " << numCols << " minVal: " << minVal << " maxVal: " << maxVal<<endl;

string str = to\_string(maxVal);

*int* width = str.length();

*int* r =2;

//step 12

//step 4

*int* c =2;

while (r<numRows+2){

//step 10

while (c<numCols+2){

//step 5

\**output* << *inAry*[r][c];

//step 6

str = to\_string(*inAry*[r][c]);

*int* WW = str.length();

//step 8

while (WW<width+1){

//step 7

\**output* << " ";

WW++;

}

//step 9

c++;

}

\**output* << endl;

//step 11

r++;

}

}

*void* loadMaskAry(ifstream \**infile*){

*int* val;

*int* index =0;

maskWeight=0;

string line;

while (getline(\**infile*, line)){

istringstream set(line);

while (set >> val ){

// cout << val << " ";

mask[index++] = val;

maskWeight+=val;

}

// cout <<endl;

}

}

*void* loadNeighbor(*int* *i*, *int* *j*){

*int* count = 0;

// cout<<mirrorFramedAry[i][j]<<endl<<endl;

for (*int* r = *i*-2; r <*i*+3; r++){

for (*int* c = *j*-2; c <*j*+3; c++){

// cout<<mirrorFramedAry[r][c]<< " ";

neighbor[count++]= mirrorFramedAry[r][c];

}

// cout<<endl;

}

}

*void* prettyPrint(*int* \*\**ary*, ofstream \**out*){

for (*int* i = 0; i < numRows; i++)

{

for (*int* j = 0; j < numCols; j++)

{

if(*ary*[i][j]>0){

\**out*<<*ary*[i][j]<<" ";

}else{

\**out*<<" ";

}

}

\**out*<<endl;

}

}

};

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

ifstream input, mask;

input.open(*argv*[1]);

mask.open(*argv*[2]);

*int* thrval = stoi(*argv*[3]);

ofstream mean, median, gauss, debug;

mean.open(*argv*[4]);

median.open(*argv*[5]);

gauss.open(*argv*[6]);

debug.open(*argv*[7]);

*int* numRows, numCols, minVal, maxVal;

input >> numRows >> numCols>>minVal>>maxVal;

// cout << numRows<< " " << numCols<< " "<<minVal<< " "<<maxVal << endl;

*int* maskNumRows, maskNumCols, maskMinVal, maskMaxVal;

mask >> maskNumRows >> maskNumCols>>maskMinVal>>maskMaxVal;

// cout << maskNumRows<< " " << maskNumCols<< " "<<maskMinVal<< " "<<maskMaxVal << endl;

//step 2

enhancement \*proj2 = new enhancement(numRows, numCols, minVal, maxVal, maskNumRows, maskNumCols, maskMinVal, maskMaxVal, thrval);

// proj2->loadImage(&input, &debug);

// proj2->printArr(&debug);

// proj2->mirrorFraming(&debug);

// proj2->loadMaskAry(&mask);

// proj2->loadNeighbor(2,2);

// proj2->computeAvg();

// step 3

proj2->loadMaskAry(&mask);

//step4

proj2->loadImage(&input, &debug);

//step 5

proj2->mirrorFraming(&debug);

//step6

// cout<<"step 6";

proj2->imgReformat(proj2->mirrorFramedAry, &debug);

//step 7

// cout<<"step 7";

proj2->computeAvg();

proj2->imgReformat(proj2->avgAry, &debug);

proj2->binaryThreshold(proj2->avgAry);

proj2->prettyPrint(proj2->thrAry, &mean);

// //step 8

proj2->computeMedian();

proj2->imgReformat(proj2->avgAry, &debug);

proj2->binaryThreshold(proj2->avgAry);

proj2->prettyPrint(proj2->thrAry, &median);

//step 9

proj2->computeGauss();

proj2->imgReformat(proj2->avgAry, &debug);

proj2->binaryThreshold(proj2->gaussAry);

proj2->prettyPrint(proj2->thrAry, &gauss);

//step10

input.close();

mask.close();

mean.close();

median.close();

gauss.close();

debug.close();

return 0;

}