CSC 381 Computer Vision ( Java )

Project 3: Median Gaussian Filter

Han Wen Loh

Soft copy: 02/24/2019

Hard copy: 02/28/2019

Algorithm Steps:

step 0: inFile1 <-- open the input image file

inFile2 <-- open the template file

outFile1, outFile2 <-- open output files

numRows, numCols, minVal, maxVal <-- read from inFile1

TemplateRows, TemplateCols <-- read from inFile2

dynamically allocate all arrays, initialize all arrays to zero

step 1: loadImage (...)

step 2: loadTemplate (...)

step 3: medianFilter(...) // see algorithm below

step 4: output (numRows, numCols, newMin, newMax) to outFile1

step 5: output medianAry to outFile1

step 6: GaussianFilter(...) // see algorithm below

step 7: output (numRows, numCols, newMin, newMax) to outFile2

step 8: output GaussAry to outFile2

step 9: close all files

Filter.java

import java.io.PrintWriter;

import java.util.Scanner;

public class Filter {

private int numRows, numCols, minVal, maxVal, newMin, newMax, templateRows, templateCols, totalWeight;

private int[][] imgArg;

private int[][] gaussTemplate;

private int[][] medianAry;

private int[][] gaussAry;

private int[] neighborAry;

public Filter(Scanner inFile1, Scanner inFile2) {

numRows = numCols = minVal = maxVal = newMin = newMax = templateRows = templateCols = totalWeight = 0;

imgArg = gaussTemplate = medianAry = gaussAry = null;

numRows = inFile1.nextInt();

numCols = inFile1.nextInt();

minVal = inFile1.nextInt();

maxVal = inFile1.nextInt();

templateRows = inFile2.nextInt();

templateCols = inFile2.nextInt();

neighborAry = new int[9];

imgArg = new int[numRows][numCols];

medianAry = new int[numRows][numCols];

gaussAry = new int[numRows][numCols];

gaussTemplate = new int[templateRows][templateCols];

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

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

imgArg[i][j] = 0;

medianAry[i][j] = 0;

gaussAry[i][j] = 0;

}

}

for (int i = 0; i < 9; ++i)

neighborAry[i] = 0;

for (int i = 0; i < templateRows; ++i)

for (int j = 0; j < templateCols; ++j)

gaussTemplate[i][j] = 0;

}

public void loadImage(Scanner inFile1) {

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

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

imgArg[r][c] = inFile1.nextInt();

medianAry[r][c] = imgArg[r][c];

gaussAry[r][c] = imgArg[r][c];

}

}

}

public void loadTemplate(Scanner inFile2) {

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

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

gaussTemplate[r][c] = inFile2.nextInt();

}

}

}

public void medianFilter() {

newMin = minVal;

newMax = maxVal;

for (int r = 1; r < numRows - 1; ++r) {

for (int c = 1; c < numCols - 1; ++c) {

loadNeighbors(r, c);

sort(neighborAry);

medianAry[r][c] = neighborAry[4];

newMin = newMin > neighborAry[4] ? neighborAry[4] : newMin;

newMax = newMax < neighborAry[4] ? neighborAry[4] : newMax;

}

}

}

private void loadNeighbors(int i, int j) {

int x = 0;

for (int r = i - 1; r <= i + 1; ++r) {

for (int c = j - 1; c <= j + 1; ++c) {

neighborAry[x++] = imgArg[r][c];

}

}

}

private void sort(int[] array) {

int min, t;

for (int i = 0; i < 5; ++i) {

min = i;

for (int j = i + 1; j < 9; ++j) {

if (array[j] < array[min])

min = j;

}

t = array[i];

array[i] = array[min];

array[min] = t;

}

}

public void printMedianAry(PrintWriter outFile1) {

outFile1.format("%d %d %d %d\n", numRows, numCols, newMin, newMax);

for (int[] r : medianAry) {

for (int c : r)

outFile1.format("%d ", c);

outFile1.println();

}

}

public void gaussianFilter() {

newMin = minVal;

newMax = maxVal;

totalWeight = computeWeight();

for (int r = 2; r < numRows - 2; ++r) {

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

gaussAry[r][c] = convolution(r, c) / totalWeight;

newMin = newMin > gaussAry[r][c] ? gaussAry[r][c] : newMin;

newMax = newMax < gaussAry[r][c] ? gaussAry[r][c] : newMax;

}

}

}

private int computeWeight() {

int total = 0;

for (int[] i : gaussTemplate) {

for (int x : i) {

total += x;

}

}

return total;

}

private int convolution(int i, int j) {

int result = 0;

int iOffset = i - (templateRows / 2); // 5/2 = 2

int jOffset = j - (templateCols / 2); // 5/2 = 2

for (int m = 0; m < templateRows; ++m) {

for (int n = 0; n < templateCols; ++n) {

result += imgArg[iOffset + m][jOffset + n] \* gaussTemplate[m][n];

}

}

return result;

}

public void printGaussianAry(PrintWriter outFile2) {

outFile2.format("%d %d %d %d\n", numRows, numCols, newMin, newMax);

for (int[] r : gaussAry) {

for (int c : r)

outFile2.format("%d ", c);

outFile2.println();

}

}

}

MedianGaussFilter.java (Main)

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Scanner;

public class MedianGaussFilter {

public static void main(String[] args) {

try {

Scanner inFile1 = new Scanner(new File(args[0]));

Scanner inFile2 = new Scanner(new File(args[1]));

PrintWriter outFile1 = new PrintWriter(new BufferedWriter(new FileWriter(new File(args[2]))));

PrintWriter outFile2 = new PrintWriter(new BufferedWriter(new FileWriter(new File(args[3]))));

Filter medianGaussFilter = new Filter(inFile1, inFile2);

medianGaussFilter.loadImage(inFile1);

medianGaussFilter.loadTemplate(inFile2);

medianGaussFilter.medianFilter();

medianGaussFilter.printMedianAry(outFile1);

medianGaussFilter.gaussianFilter();

medianGaussFilter.printGaussianAry(outFile2);

inFile1.close();

inFile2.close();

outFile1.close();

outFile2.close();

} catch (IOException e) {

System.out.printf("ERROR: %s", e);

}

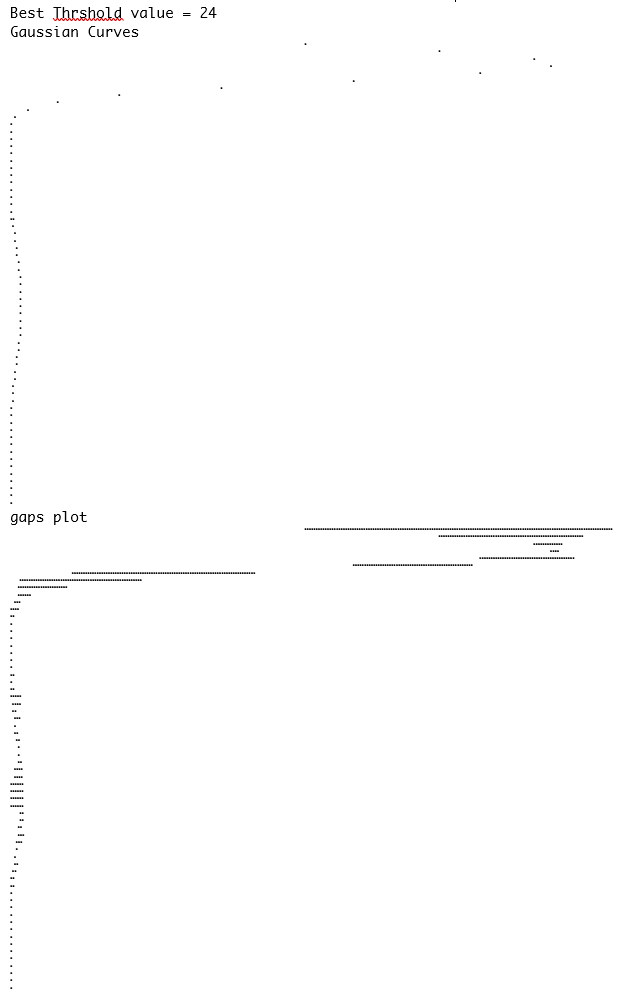
}

}

|  |  |
| --- | --- |
| 46 46 1 63  0 0  1 23  2 481  3 293  4 477  5 236  6 5  7 4  8 33  9 4  10 4  11 0  12 1  13 3  14 0  15 0  16 0  17 0  18 2  19 0  20 1  21 0  22 4  23 0  24 8  25 4  26 0  27 2  28 17  29 1  30 4  31 4  32 5  33 2  34 2 | 35 0  36 0  37 0  38 8  39 0  40 6  41 8  42 5  43 2  44 6  45 3  46 0  47 2  48 450  49 0  50 0  51 0  52 0  53 0  54 2  55 0  56 0  57 0  58 4  59 0  60 0  61 0  62 0  63 0 |

MedianHist.txt

MedianThr.txt



prettyMedianBinary.txt

1 1

1 1 1 1

1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1

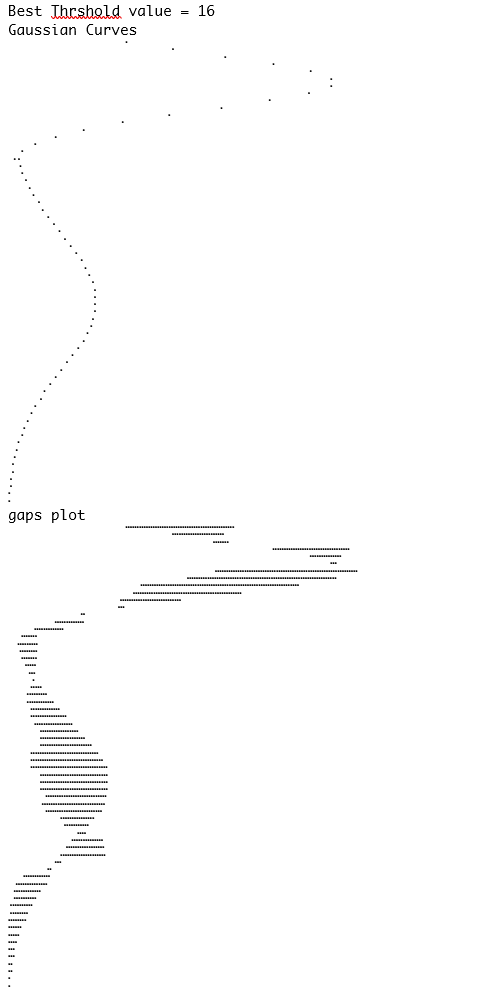
1 1 1 1 1

1 1 1

GaussHist.txt

|  |  |
| --- | --- |
| 46 46 1 63  0 0  1 77  2 110  3 286  4 227  5 175  6 111  7 96  8 71  9 60  10 67  11 59  12 39  13 26  14 37  15 13  16 11  17 6  18 16  19 13  20 24  21 10  22 6  23 17  24 12  25 10  26 5  27 14  28 17  29 22  30 27  31 12  32 10  33 12  34 17 | 35 20  36 17  37 16  38 28  39 20  40 18  41 30  42 40  43 47  44 47  45 50  46 27  47 21  48 8  49 3  50 4  51 3  52 1  53 0  54 0  55 1  56 0  57 0  58 0  59 0  60 0  61 0  62 0  63 0 |

GaussThr.txt



prettyGaussBinary.txt

1 1

1

1

1 1

1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1

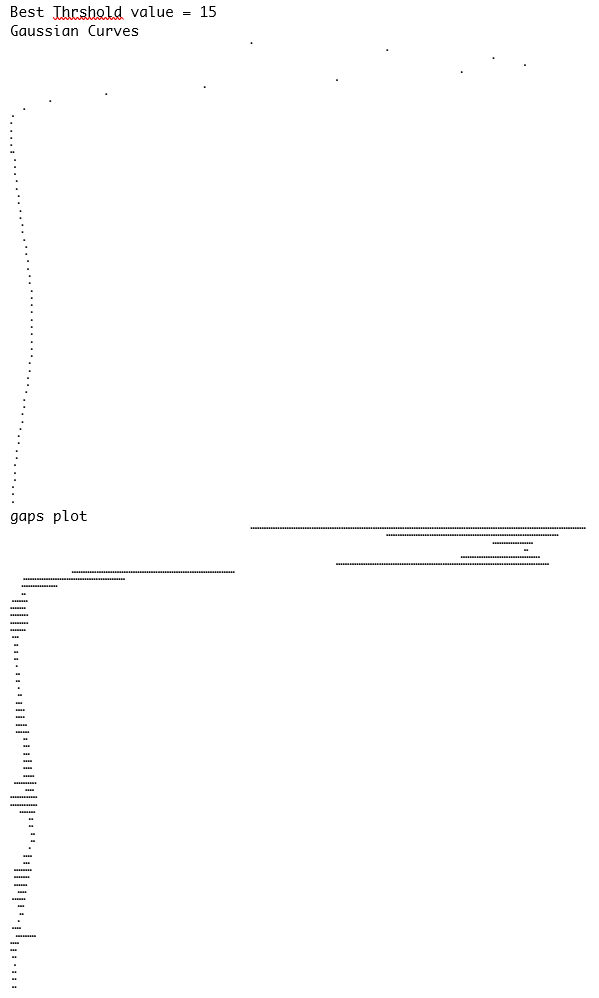
1 1 1

1 1

nonFilterHist.txt

|  |  |
| --- | --- |
| 46 46 1 63  0 0  1 277  2 276  3 268  4 306  5 277  6 7  7 6  8 33  9 6  10 5  11 7  12 8  13 6  14 9  15 3  16 3  17 0  18 12  19 1  20 3  21 4  22 7  23 3  24 7  25 3  26 0  27 3  28 15  29 3  30 7  31 7  32 7  33 2  34 10 | 35 8  36 0  37 0  38 16  39 0  40 5  41 12  42 10  43 16  44 14  45 7  46 2  47 2  48 363  49 0  50 1  51 6  52 4  53 1  54 14  55 11  56 0  57 0  58 14  59 0  60 8  61 1  62 2  63 8 |

nonFilterThr.txt



prettyNonFilterBinary.txt

1 1

1 1 1

1 1 1

1 1 1 1 1

1 1 1

1 1 1 1

1 1 1

1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1

1 1

1 1 1 1 1

1 1