CSCI 381 – Computer Vision (C++)
Program: Project 3: 2d Median & Gaussian Filters
Name: Isaac Gordon
Due Date:
Soft copy: 2/27/2019 Wednesday before midnight
Hard copy: 2/18/2019 Thursday in class

---

step -1: make sure all inputted arguments are valid

step 0: 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

step 1: create Image object
- Within constructor
  - Step 1.1: Open files
  - Step 1.2: loadImage(…)
    - Step 1.2.1: dynamically allocate 2D arrays
    - Step 1.2.2: load data into imgAry from inFile1
    - Step 1.2.3: close inFile1
  - Step 1.3 loadTemplate(…)
    - Step 1.3.1: dynamically allocate 2D gaussTemplate
    - Step 1.3.2: load template data into gaussTemplate from inFile2
    - Step 1.3.3: close inFile2

Step 2: medinFilter()

Step 3: output median filter data
- Within function
  - Step 3.1: print header to outFile1
  - Step 3.2: print medianAry to outFile1
  - Step 3.3: close outFile1

Step 4: output gauss filter data
- Within function
  - Step 4.1: print header to outFile2
  - Step 4.2: print gaussAry to outFile2
  - Step 4.3: close outFile2

Code: MedianGaussFilter.java

```
import java.io.FileNotFoundException;

public class MedianGaussFilter {

	public static void main(String[] args) {

		final String ARG_ERROR_STRING = "Improper arguements. Correct
syntax is: \n>> ... <input1.txt> <input2.txt> <output1.txt> <output2.txt>"
				+ " \n\twhere: \n\tinput1.txt is a grey-scale image
with header \n\tinput2.txt is a Gaussian Template \n\toutput1.txt is a file
to "
				+ " print the result of the Median filter to
\n\toutput2.txt is a file to print the result of the Gaussian Filter.";

		//STEP -1: Check for bad arguements
		if(args.length != 4) {
			System.out.println(ARG_ERROR_STRING);
			System.exit(0);
		}//if

		for(String s: args) {
			if(!s.endsWith("txt")) {
				System.out.println(ARG_ERROR_STRING);
				System.exit(0);
			}//if
		}//for

		String inFile1 = args[0];
		String inFile2 = args[1];
		String outFile1 = args[2];
		String outFile2 = args[3];

		Image image;
		try {
			image = new Image(inFile1, inFile2);

			//output new header details to outFile1, output medianAry
to outFile1
			image.medianFilter();
			image.printMedianOutput(outFile1);

			//run gaussFilter and print new header and the new image to
outFile2
			image.gaussianFilter();
			image.printGaussOutput(outFile2);
		} catch (FileNotFoundException e) {
			System.out.println("1 or more of your arguements could not
be found. Check its path.");
			e.printStackTrace();
		} catch (IllegalArgumentException e) {
			System.out.println("1 or more of your arguements were not a
text file.");
			e.printStackTrace();  }
	}//main
}//class
```

Code: Image.java

```
import java.io.*;
import java.util.*;

public class Image {

	int numRows;
	int numCols;
	int minVal;
	int maxVal;
	int newMin;
	int newMax;

	int templateRows;
	int templateCols;
	int totalWeight;

	int[][] imgAry;
	int[][] gaussTemplate;
	int[][] medianAry;
	int[][] gaussAry;
	int[] neighborAry = new int[9];

	public Image(String imgFilePath, String templateFilePath) throws
FileNotFoundException, IllegalArgumentException {
		//check for wrong file types
		if(!imgFilePath.endsWith("txt")) throw new
IllegalArgumentException("Not a text file...");
		if(!templateFilePath.endsWith("txt")) throw new
IllegalArgumentException("Not a text file...");

		//get files and load data
		File img = new File(imgFilePath);
		File template = new File(templateFilePath);
		loadImage(img);
		loadTemplate(template);
	}//constructor

	private void loadImage(File imgFile) throws FileNotFoundException {
		Scanner inputReader = new Scanner(new FileReader(imgFile));

		//get header values
		numRows = inputReader.nextInt();
		numCols = inputReader.nextInt();
		minVal = inputReader.nextInt();
		maxVal = inputReader.nextInt();

		//allocate all 2D img arrays
		imgAry = new int[numRows][numCols];
		medianAry = new int[numRows][numCols];
		gaussAry = new int[numRows][numCols];

		//load img data into imgArray, L->R, T->B
		int i = 0;
		int j = 0;
		while(inputReader.hasNextInt()) {
```

```
                  if(i == numRows) {
                        i = 0;
                        j++;
                  }//if
                  if(j == numCols) {
                        System.out.println("Shit might be messed up... Check
loadImage().");
                  }//if

                  imgAry[i][j] = inputReader.nextInt();
                  //TODO: might load all this info into medianAry and
gaussAry
                  i++;
            }//while
      inputReader.close();
      }//loadImage

      private void loadTemplate(File templateFile) throws
FileNotFoundException {
            Scanner inputReader = new Scanner(new FileReader(templateFile));

            //get header values
            templateRows = inputReader.nextInt();
            templateCols = inputReader.nextInt();

            //allocate 2D template array
            gaussTemplate = new int[templateRows][templateCols];

            //load template data into gaussTemplate, L->R, T->B
            int i = 0;
            int j = 0;
            while(inputReader.hasNextInt()) {
                  if(i == templateRows) {
                        i = 0;
                        j++;
                  }//if
                  if(j == templateCols) {
                        System.out.println("Shit might be messed up... Check
loadTemplate().");
                  }//if

                  gaussTemplate[i][j] = inputReader.nextInt();
                  i++;
            }//while
            inputReader.close();
      }//loadTemplate

      public void medianFilter() {
            newMin = maxVal;
            newMax = minVal;

            //process imgAry using a median filter
            for(int i = 1; i <= numRows -2 ; i++) {
                  for(int j = 1; j <= numCols - 2; j++) {
                        loadNeighbors(i,j);
                        selectionSort5x(neighborAry);
                        medianAry[i][j] = neighborAry[4];
```

```
                        //get newMin and newMax
                        if(neighborAry[4] < newMin) newMin = neighborAry[4];
                        if(neighborAry[4] > newMax) newMax = neighborAry[4];
                  }//for
            }//for
      }//medianFilter

      private void selectionSort5x(int[] arr) {
            int currMin = arr[0];
            //run selection sort only 5x because we only need the 5th
smallest number
            for(int p = 0; p < 5; p++){
                  for(int i = p + 1; i < arr.length; i++) {
                        if(arr[i] < currMin) {
                              currMin = arr[i];
                              int temp = arr[p];
                              arr[p] = arr[i];
                              arr[i] = temp;
                        }
                        p++;
                  }//for
            }//for
      }//selectionSort

      private void loadNeighbors(int i, int j) {
            neighborAry[0] = imgAry[i-1][j-1];
            neighborAry[1] = imgAry[i][j-1];
            neighborAry[2] = imgAry[i+1][j-1];
            neighborAry[3] = imgAry[i-1][j];
            neighborAry[4] = imgAry[i][j];
            neighborAry[5] = imgAry[i+1][j];
            neighborAry[6] = imgAry[i-1][j+1];
            neighborAry[7] = imgAry[i][j+1];
            neighborAry[8] = imgAry[i+1][j+1];
      }//loadNeighbors

      public void gaussianFilter() {
            newMin = maxVal;
            newMax = minVal;
            totalWeight = computeTemplateWeight();


            //process imgAry with a gaussian filter from L->R, T->B
            for(int i = 2; i <= numRows - 3; i++) {
                  for(int j = 3; j <= numCols - 3; j++) {
                        int c = convolution(i, j);

                        gaussAry[i][j] = (int)(c / totalWeight);
                        if(gaussAry[i][j] < newMin) newMin = gaussAry[i][j];
                        if(gaussAry[i][j] > newMax) newMax = gaussAry[i][j];
                  }//for
            }//for
      }//gausssianFilter

      private int convolution(int i, int j) {
            int result = 0;
```

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

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


            for(int m = 0; m < templateRows; m++) {
                  for(int n = 0; n < templateCols; n++) {
                        result += imgAry[iOffset + m][jOffset + n] *
gaussTemplate[m][n];
                  }//for
            }//for

            return result;
      }//convolution

      private int computeTemplateWeight() {
            int sum = 0;
            for(int i = 0; i < templateRows; i++) {
                  for( int j = 0; j < templateCols; j++) {
                         sum += gaussTemplate[i][j];
                  }//for
            }//for
            return sum;
      }//computeTemplateWeight

      public void printMedianOutput(String outFile) throws
FileNotFoundException {
            File file = new File(outFile);
            PrintWriter outputStream = new PrintWriter(file);
            outputStream.println(numRows + " " + numCols  + " " + newMin  + "
" + newMax);
            for(int i = 0; i < numRows; i++) {
                  for(int j = 0; j < numCols; j++) {
                        outputStream.print(medianAry[i][j]);
                        if(j!= numCols - 1)outputStream.print(" ");
                  }//for
                  outputStream.println();
            }//for
            outputStream.close();
      }//printMedianOutput

      public void printGaussOutput(String outFile) throws
FileNotFoundException {
            File file = new File(outFile);
            PrintWriter outputStream = new PrintWriter(file);
            outputStream.println(numRows + " " + numCols  + " " + newMin  + "
" + newMax);
            for(int i = 0; i < numRows; i++) {
                  for(int j = 0; j < numCols; j++) {
                        outputStream.print(gaussAry[i][j]);
                        if(j!= numCols - 1)outputStream.print(" ");
                  }//for
                  outputStream.println();
            }//for
            outputStream.close();
      }//printOutput
}//class
```

Output: Histograms

noFilter

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

MedianHist

46 46 1 63
0 180
1 251
2 252
3 244
4 280
5 200
6 2
7 2
8 31
9 2
10 1
11 7
12 8
13 6
14 9
15 3
16 4
17 0
18 13
19 1
20 3
21 4
22 7
23 3
24 7
25 4
26 0
27 3
28 18
29 6
30 7
31 9
32 7
33 2
34 7
35 9
36 0
37 0
38 17
39 0
40 4
41 11
42 8
43 12
44 13
45 7
46 3
47 2
48 374
49 0
50 1
51 6
52 4
53 1
54 16
55 12
56 0
57 0
58 12
59 0
60 8
61 1
62 2
63 10

GaussHist

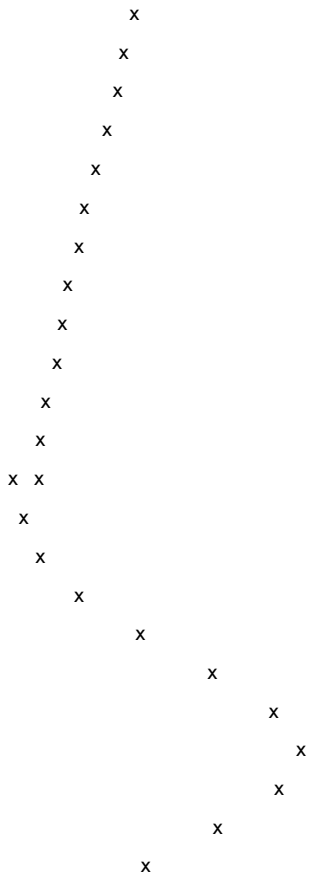
46 46 2 52
0 394
1 0
2 63
3 233
4 145
5 91
6 102
7 89
8 67
9 54
10 61
11 53
12 38

13 26
14 32
15 13
16 11
17 6
18 16
19 13
20 24
21 7
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 46
44 46
45 50
46 27
47 21
48 7
49 3
50 4
51 3
52 1

```
Output: noFilterThr
```

BEST THRESHOLD VALUE: 53

```
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
```



Output: MedianThr

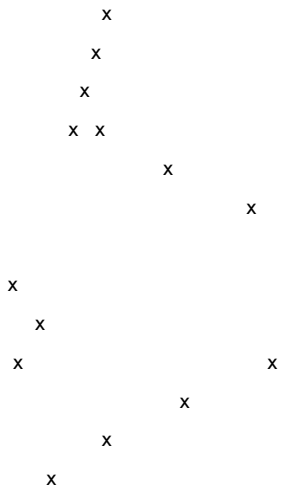
BEST THRESHOLD VALUE: 14

```
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
```

Output: GaussThr

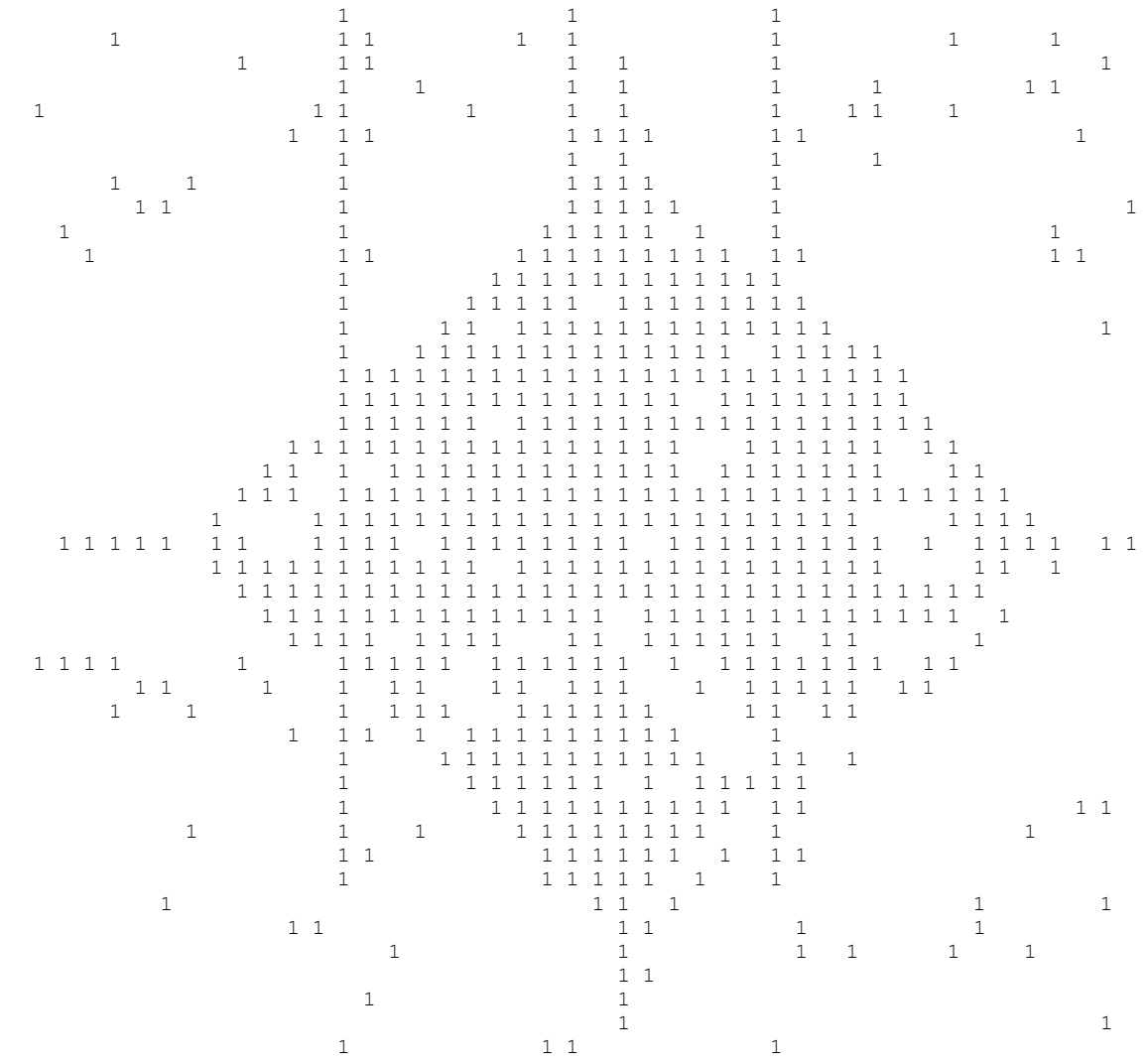
BEST THRESHOLD VALUE: 43

```
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
```



```
Output: noFilterBinary

    1





       1                          1




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

                                      1


                1      1 1

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


           1  1    1 1 1


                                      1
    1           1 1  1   1  1  1 1
                  1


                                    1 1

                                 1     1
         1

                                1
```

Output: MedianBinary



Output: GaussBinary