CSCI 381 (Java)

Project 2.1 Average Filter

Essam Yousry

Due Date of soft copy: 02/11/2018

Due Date of hard copy: 02/13/2018

Algorithm Steps for Average Filter:

Step 0: open input and output files

Read image header (four numbers)

Dynamically allocate mirrorFramedAry

Dynamically allocate tempAry

Step 1: read input file and load into mirrorFramedAry begin at [1,1]

Step 2: mirrorFrame (mirrorFramedAry)

Step 3: process the mirrorFramedAry, from left to right and top to bottom using i, and j, begin at (1,1).

neighborAry 🡨Load mirrorFramedAry [I,j] with 3 by 3 neighbors

tempAry 🡨 compute the average of neighborAry

keep tracking the newMin and newMax of tempAry

Step 4: repeat step 3 until all pixels are processed

Step 5: output the image header to AVG3X3Out.txt

Step 6: output to AVG3X3Out.txt from tempAry, begin at [1,1], without the pixels on the boarder.

Step 7: close input and output file

Source Code

import java.util.Scanner;

import java.io.FileReader;

import java.io.IOException;

import java.io.FileWriter;

import java.io.File;

import java.io.FileOutputStream;

import java.io.PrintWriter;

public class AverageFilter{

  public static void main(String [] args){

    File inFile = new File (args[0]);

    File outFile = new File (args[1]);

      int row, col, min, max, sum, avg;

    int [] neighborAry = new int [9];

    int [] vars = new int [4];

    int [] sortedArray = new int [9];

    int newMin, newMax;

      try {

          Scanner sc = new Scanner(inFile);

          int c = 0;

            while (sc.hasNextInt() && c != 4){

                vars[c++] = sc.nextInt();

            }

            row = vars[0];

            col = vars[1];

            min = vars[2];

            max = vars[3];

            int [][] mirrorFramedAry = new int [row + 2][col + 2];

            int [][] tempAry = new int [row + 2][col + 2];

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

              for (int j = 1; j < col + 1; j++){

                mirrorFramedAry[i][j] = sc.nextInt();

              }

            }

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

              for (int j = 1; j < col + 1; j++){

                  mirrorFramedAry[0][j] = mirrorFramedAry[1][j];

                  mirrorFramedAry[row + 1][j] = mirrorFramedAry[row][j];

                  mirrorFramedAry[i][0] = mirrorFramedAry[i][1];

                  mirrorFramedAry[i][col + 1] = mirrorFramedAry[i][col];

              }

          }

          mirrorFramedAry[0][0] = mirrorFramedAry[0][1];

          mirrorFramedAry[0][col + 1] = mirrorFramedAry[0][col];

          mirrorFramedAry[row + 1][0] = mirrorFramedAry[row][0];

          mirrorFramedAry[row + 1][col + 1] = mirrorFramedAry[row][col];

          newMin = 1000;

          newMax = 0;

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

              for (int j = 1; j < col + 1; j++){

                  neighborAry[0] = mirrorFramedAry[i - 1][j - 1];

                  neighborAry[1] = mirrorFramedAry[i - 1][j];

                  neighborAry[2] = mirrorFramedAry[i - 1][j + 1];

                  neighborAry[3] = mirrorFramedAry[i][j + 1];

                  neighborAry[4] = mirrorFramedAry[i + 1][j + 1];

                  neighborAry[5] = mirrorFramedAry[i + 1][j];

                  neighborAry[6] = mirrorFramedAry[i + 1][j - 1];

                  neighborAry[7] = mirrorFramedAry[i - 1][j];

                  neighborAry[8] = mirrorFramedAry[i][j];

                  sum = 0;

                  avg = 0;

                  for (int s = 0; s < 9; s++){

                      sum += neighborAry[s];

                      avg = sum / 9;

                  }

                  tempAry[i][j] = (int) avg;

                  newMin = getMin(tempAry, i, j);

                  newMax = getMax(tempAry, i, j);

              }

          }

          PrintWriter pw = new PrintWriter(new FileWriter(outFile));

          pw.print(row);

          pw.print(' ');

          pw.print(col);

          pw.print(' ');

          pw.print(newMin);

          pw.print(' ');

          pw.print(newMax);

          pw.println();

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

              for (int j = 1; j < col + 1; j++){

                  pw.print(tempAry[i][j]);

                  pw.print(' ');

              }

              pw.println();

          }

          pw.close();

      }

      catch (Exception e) {

          System.out.println("Error " + e);

      }

  }

    public static int getMax (int[][] numbers, int row, int col){

        int max = numbers[1][1];

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

            for (int j = 1; j < col + 1; j++){

                if (numbers[i][j] > max) max = numbers[i][j];

            }

        }

        return max;

    }

    public static int getMin (int[][] numbers, int row, int col){

        int min = numbers[1][1];

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

            for (int j = 1; j < col + 1; j++){

                if (numbers[i][j] < min) min = numbers[i][j];

            }

        }

        return min;

    }

}

Output











