CSCI 381 (C++)

Project 2.2 Median Filter

Essam Yousry

Due Date of soft copy: 02/13/2018

Due Date of hard copy: 02/15/2018

Algorithm Steps for Median 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 🡨 sort neighborAry and put fifth elements in tempAry

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

#include <iostream>

#include <fstream>

using namespace std;

void selectionSort(int array[], int n){

    int min, temp;

    for (int i = 0; i < n - 1; i++){

        min = i;

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

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

        }

        if (min != i){

            temp = array[i];

            array[i] = array[min];

            array[min] = temp;

        }

    }

}

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

{

    int vars[4], row, col, min, max;

    int neighborAry[9];

    int sortedArray[9];

    static int newMin, newMax;

    ifstream myfile;

    myfile.open(argv[1]);

    for(int i = 0; i < 4; i++)

    myfile >> vars[i];

    cout << endl;

    row = vars[0];

    col = vars[1];

    min = vars[2];

    max = vars[3];

    cout << row << endl;

    cout << col << endl;

    cout << min << endl;

    cout << max << endl;

    ofstream myfile2;

    myfile2.open(argv[2]);

    int \*\*mirrorFramedAry = new int \*[row + 2];

    for (int i = 0; i < row + 2; i++)

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

    static int \*\*tempAry = new int \*[row + 2];

    for (int i = 0; i < row + 2; i++)

    tempAry[i] = new int [col + 2];

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

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

            myfile >> mirrorFramedAry[i][j];

        }

    }

    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 = 100;

    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];

            selectionSort(neighborAry, 9);

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

                sortedArray[i] = neighborAry[i];

            }

            tempAry[i][j] = sortedArray[4];

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

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

                    if (tempAry[m][t] < newMin) newMin = tempAry[i][j];

                    if (tempAry[m][t] < newMax) newMax = tempAry[i][j];

                }

            }

        }

    }

    myfile2 << row;

    myfile2 << ' ';

    myfile2 << col;

    myfile2 << ' ';

    myfile2 << newMin;

    myfile2 << ' ';

    myfile2 << newMax;

    myfile2 << endl;

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

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

            myfile2 << tempAry[i][j];

            myfile2 << ' ';

        }

        myfile2 << endl;

    }

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

        delete mirrorFramedAry[i];

    }

    delete[] mirrorFramedAry;

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

        delete tempAry[i];

    }

    delete[] tempAry;

    myfile.close();

    myfile2.close();

    return 0;

}

Output











