CSCI 381 (CPP)

Section 31

Project 8.1 kCurvature

Essam Yousry

Due Date of soft copy: 04/28/2018

Due Date of hard copy: 05/1/2018

Algorithm Steps in main

step 0: - inFile 🡨 open input files

(numRows, numCols, minVal, maxVal, label) 🡨 get from inFile

- numPts 🡨 countPts (inFile)

- close inFile

- inFile 🡨 open the input file the second time.

- K 🡨 get from the user from console

- index 🡨 0

step 1: (x, y) 🡨 read from inFile

storePt (x, y, index)// store x, y to PtAry[index]

printPtAry()

step 2: index ++;

step 3: repeat step 1 and step 2 until the end of inFile

step 4: Q 🡨 0

P 🡨K

R 🡨 2\* K

step 5: index <-- P

curvature <-- computeCurvature (Q, P, R)

store curvature to PtAry[index]

print Q, P, R, index, x, y, curvature of PtAry[index]to argv[4]

step 6: Increment Q, P, R by 1

step 7: repeat step 5 to step 6 until P == K-1

step 8: print the info (x, y, curvature) of the entire PtAry to argv[4]

step 9: computeLocalMaxima (PtAry) for all point in PtAry[index], index from 0 to numPts-1

step 10: setCorner (PtAry) do for all point in boundPtAry[index], index from 0 to numPts-1

step 11: output only (x, y, corner) of the entire PtAry to argv[2]

step 12: Img 🡨 create an image of size numRows by numCols

step 13: plotPt2Img()

step 14: prettyPrint (img) to argv[3]

step 15: close all files

Source Code

#include <iostream>

#include <fstream>

#include <string>

#include <algorithm>

#include <cmath>

using namespace std;

class point{

private:

    int row, col, localMax, corner;

    double curvature;

public:

    point(int x = 0, int y = 0){

        row = x;

        col = y;

    }

    void setCurvature(double cur){

        curvature = cur;

    }

    void setLocalMax(int lmax){

        localMax = lmax;

    }

    void setCorner(int corn){

        corner = corn;

    }

    int getCorner(){ return corner; }

    int getLocalMax(){ return localMax; }

    double getCurvature(){ return curvature; }

    int getRow(){ return row; }

    int getCol(){ return col; }

};

void storePt(int x, int y, int index, point \*ptAry){

    point p = point(x, y);

    ptAry[index] = p;

}

void plotPt2Img(int \*\*imgAry, point \*ptAry, int numPts){

    int index = 0;

    while (index < numPts){

        imgAry[ptAry[index].getRow()][ptAry[index].getCol()] = ptAry[index].getCorner();

        index++;

    }

}

double computeCurvature(int Q, int P, int R, point \*ptAry){

    double curvature;

    int r1 = ptAry[Q].getRow();

    int c1 = ptAry[Q].getCol();

    int r2 = ptAry[P].getRow();

    int c2 = ptAry[P].getCol();

    int r3 = ptAry[R].getRow();

    int c3 = ptAry[R].getCol();

    if (((r1 - r2) == 0) || ((r2 - r3) == 0)){

        curvature = abs(((double)(c1 - c2) / ((double)(r1 - r2) + 0.01))) - abs(((double)(c2 - c3) / ((double)(r2 - r3) + 0.01)));

    }

    else {

        double a = ((double) (c1 - c2))/(double) (r1 - r2);

        double b = double(c2 - c3)/(double)(r2 - r3);

        curvature = a - b;

    }

    return abs(curvature);

}

bool computeLocalMaxima(point \*ptAry, int i, int numPts){

    bool check = true;

    double curvature = ptAry[i].getCurvature();

    for (int a = i - 2; a <= i + 2; a++){

        if (curvature < ptAry[a % numPts].getCurvature()){

            check = false;

            break;

        }

    }

    if (check == true){

        ptAry[i].setLocalMax(1);

        return true;

    }

    else{

        ptAry[i].setLocalMax(0);

        return false;

    }

}

int SetCorner(point \*ptAry, int i){

    bool check1 = false;

    if (ptAry[i].getLocalMax() == 1){

        check1 = true;

    }

    bool check2 = false;

    if ((ptAry[i + 2].getLocalMax() == 0) && (ptAry[i - 2].getLocalMax() == 0)){

        check2 = true;

    }

    if ((check1 == true) && (check2 == true)){

        ptAry[i].setCorner(8);

        return 8;

    }

    else {

        ptAry[i].setCorner(1);

        return 1;

    }

}

void prettyPrint(int \*\*imgAry, int row, int col, ofstream &myfile){

    for(int x = 0; x < row; x++){

        for (int y = 0; y < col; y++){

            if (imgAry[x][y] > 0){

                myfile << (imgAry[x][y]);

                myfile << ' ';

            }

            else {

                myfile << ' ';

                myfile << ' ';

            }

        }

        myfile << endl;

    }

}

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

{

    int vars[5], row, col, min, max, label;

    int x, Q, P, R;

    int numPts = 0;

    ifstream myfile;

    myfile.open(argv[1]);

    for(int i = 0; i < 5; i++)

        myfile >> vars[i];

    cout << endl;

    row = vars[0];

    col = vars[1];

    min = vars[2];

    max = vars[3];

    label = vars[4];

    int count = 0;

    while (myfile >> x){

        ++count;

    }

    numPts = count / 2;

    myfile.close();

    myfile.open(argv[1]);

    string fileName = argv[1];

    string fileNameWithoutExtension = fileName.substr(0, fileName.rfind("."));

    ofstream myfile2;

    myfile2.open(argv[2]);

    ofstream myfile3;

    myfile3.open(argv[3]);

    ofstream myfile4;

    myfile4.open(argv[4]);

    int i = 0;

    while(i < 5){

        myfile >> x;

        i++;

    }

    point \*ptAry = new point [numPts];

    int index = 0;

    int r;

    int c;

    while (index < numPts){

        myfile >> r;

        myfile >> c;

        storePt(r, c, index, ptAry);

        index++;

    }

    int K;

    cout << ("Please Enter a Value For K: ") << endl;

    cin >> K;

    Q = 0;

    P = K;

    R = 2 \* K;

    int ind;

    double curvature;

    do{

        ind = P;

        curvature = computeCurvature(Q, P, R, ptAry);

        ptAry[ind].setCurvature(curvature);

        myfile4 << Q;

        myfile4 << ' ';

        myfile4 << P;

        myfile4 << ' ';

        myfile4 << R;

        myfile4 << ' ';

        myfile4 << ind;

        myfile4 << ' ';

        myfile4 << ptAry[ind].getRow();

        myfile4 << ' ';

        myfile4 << ptAry[ind].getCol();

        myfile4 << ' ';

        myfile4 << (double) curvature;

        myfile4 << endl;

        Q++;

        P = (P + 1) % numPts;

        R = (R + 1) % numPts;

    } while (P != K);

    myfile4 << "------------------------------------";

    myfile4 << endl;

    for (int t = 0; t < numPts; t++){

        myfile4 << ptAry[t].getRow();

        myfile4 << ' ';

        myfile4 << ptAry[t].getCol();

        myfile4 << ' ';

        myfile4 << ptAry[t].getCurvature();

        myfile4 << endl;

    }

    for (int i = 0; i < numPts; i++){

        computeLocalMaxima(ptAry, i, numPts);

    }

    for (int j = 0; j < numPts; j++){

        SetCorner(ptAry, j);

    }

    myfile2 << row;

    myfile2 << ' ';

    myfile2 << col;

    myfile2 << ' ';

    myfile2 << min;

    myfile2 << ' ';

    myfile2 << max;

    myfile2 << endl;

    myfile2 << label;

    myfile2 << endl;

    myfile2 << numPts;

    myfile2 << endl;

    for (int k = 0; k < numPts; k++){

        myfile2 << ptAry[k].getRow();

        myfile2 << ' ';

        myfile2 << ptAry[k].getCol();

        myfile2 << ' ';

        myfile2 << ptAry[k].getCorner();

        myfile2 << endl;

    }

    int \*\*imgAry = new int \*[row];

    for (int i = 0; i < row; i++)

        imgAry[i] = new int [col];

    plotPt2Img(imgAry, ptAry, numPts);

    prettyPrint(imgAry, row, col, myfile3);

    for (int i = 0; i < row; i++){

        delete imgAry[i];

    }

    delete[] imgAry;

    delete[] ptAry;

    myfile.close();

    myfile2.close();

    myfile3.close();

    myfile4.close();

    return 0;

}

Output







