Computer Vision Programming Language: C++

Project #6 Component Connectness

Andres Quintero

Due Date:

Soft copy: 3/22/2020

Hard copy: 3/22/2020

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 0: inFile 🡨 open the input file

prettyPrintFile, labelFile, propertyFile 🡨 open from argc[]

numRows, numCols, minVal, maxVal 🡨 read from inFile

dynamically allocate zeroFramedAry.

step 1: zero2D (zeroFramedAry)

step 2: loadImage(inFile, zeroFramedAry)

step 3: Connectness 🡨 from argv[2]

step 4: newLabel 🡨 connectPass1 (Connectness, zeroFramedAry, NonZeroNeighborAry)

step 5: prettyPrint (prettyPrintFile)

printEQAry (newLable, prettyPrintFile)

step 6: connectPass2 (Connectness, zeroFramedAry, NonZeroNeighborAry)

step 7: prettyPrint (prettyPrintFile

printEQAry (newLable, prettyPrintFile)

step 8: manageEQAry (EQAry, newLabel)

printEQAry (numCCLable, prettyPrintFile)

step 9: connectPass3 (...) // See algorithm below

prettyPrint (prettyPrintFile

printEQAry (numCCLable, prettyPrintFile)

step 10: output numRows, numCols, newMin, newMax to labelFile

step 11: printImg (labelFile)

step 12: printCCproperty (propertyFile

step 13: drawBoxes(zeroFramedAry, CCproperty)

step 14: prettyPrint (prettyPrintFile)

step 15: close all files

**Source Code:**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

//DEBUGSTUFF

void printArrayDebug(int\*\* array, int rows, int cols){

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

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

cout << array[i][j] << " ";

}

cout << endl;

}

}

struct Property{

int label, numpixels;

int upperLftR = INT\_MAX;

int upperLftC= INT\_MAX;

int lowerRgtR, lowerRgtC;

};

// Prototypes

void zero2DAry(int\*\* array, int rows, int cols);

void loadImage(fstream& imgFile, int\*\* zeroFrameAry, int row, int col);

int connectPass1(int Connectness, int\*\* Ary, int\* NonZeroNeighborAry, int rows, int cols, int\* EQArray);

int loadNonZeroPass1(int\*\* Ary, int Connectness, int i, int j, int\* NonZeroNeighborAry, bool &diffFlag, int &nonZeroCount);

void minus1D(int\* Ary1D);

void updateEQ(int\* EQArray, int\* NonZeroNeighborAry, int minLabel, int nonZeroCount);

void prettyPrint(int\*\* array, fstream& outFile, int rows, int cols);

void printEQAry(int newLabel, int\* EQArray, fstream& outFile);

void connectPass2(int Connectness, int\*\* Ary, int\* NonZeroNeighborAry, int rows, int cols, int\* EQArray);

int loadNonZeroPass2(int\*\* Ary, int Connectness, int i, int j, int\* NonZeroNeighborAry, bool &diffFlag, int &nonZeroCount);

void manageEQAry(int\* EQArray, int newLabel);

void connectPass3(int\* EQArray, int\*\* Ary, Property\* CCproperty, int rows, int cols);

void PrintImg(int\*\* Ary, fstream& outFile, int rows, int cols);

void printCCProperty(Property\* CCproperty, fstream& outFile, int numOfCC);

void drawBoxes(int\*\* zeroFrameAry, Property\* CCproperty, int numOfCC);

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

//CLI inputs

fstream inFile(argv[1]);

// connectness integer is from argv[2]

fstream prettyPrintFile(argv[3], fstream::out);

fstream labelFile(argv[4], fstream::out);

fstream propertyFile(argv[5], fstream::out);

// Variables

int numRows, numCols, minVal, maxVal, Connectness, newLabel, trueNumCC, numNb;

inFile >> numRows;

inFile >> numCols;

inFile >> minVal;

inFile >> maxVal;

newLabel = 0;

numNb = 5;

int \*NonZeroNeighborAry = new int[numNb];

int EQSize = (numRows\*numCols)/2;

int \*EQArray = new int[EQSize];

//Setting all EQArray[i] = i

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

EQArray[i] = i;

}

//Dynamically allocating zeroFrameAry

int \*\*zeroFrameAry = new int\*[numRows+2];

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

zeroFrameAry[i] = new int[numCols+2];

}

//Step 1

zero2DAry(zeroFrameAry, numRows+2, numCols+2);

//Step 2

loadImage(inFile, zeroFrameAry, numRows, numCols);

// printArrayDebug(zeroFrameAry, numRows+2, numCols+2); //DEBUGSTUFF

//Step 3

Connectness = stoi(argv[2]);

//Step 4

newLabel = connectPass1(Connectness, zeroFrameAry, NonZeroNeighborAry, numRows, numCols, EQArray);

//Step 5

prettyPrintFile << "zeroFrameAry after Pass-1: " << endl;

prettyPrint(zeroFrameAry, prettyPrintFile, numRows+2, numCols+2); // zeroFrameAry to prettyPrintFile

prettyPrintFile << "EQArray after Pass-1: " << endl;

printEQAry(newLabel, EQArray, prettyPrintFile); // print EQArray up to newLabel to pretty

//Step 6

connectPass2(Connectness, zeroFrameAry, NonZeroNeighborAry, numRows, numCols, EQArray);

//Step 7

prettyPrintFile << "zeroFrameAry after Pass-2: " << endl;

prettyPrint(zeroFrameAry, prettyPrintFile, numRows+2, numCols+2);

prettyPrintFile << "EQArray after Pass-2: " << endl;

printEQAry(newLabel, EQArray, prettyPrintFile); //

//Step 8

manageEQAry(EQArray, newLabel);

prettyPrintFile << "EQArray after manageEQAry: " << endl;

printEQAry(newLabel, EQArray, prettyPrintFile);

//Step 8.5 Creating CCproperty

int numOfCC = -1;

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

if(EQArray[i] > numOfCC){ numOfCC = EQArray[i];}

}

Property\* CCproperty = new Property[numOfCC+1];

//Step 9

connectPass3(EQArray, zeroFrameAry, CCproperty, numRows, numCols);

prettyPrintFile << "zeroFrameAry after Pass-3: " << endl;

prettyPrint(zeroFrameAry, prettyPrintFile, numRows+2, numCols+2);

prettyPrintFile << "EQArray after Pass-3 (Up to numOfCC): " << endl;

printEQAry(numOfCC, EQArray, prettyPrintFile);

//Step 9.5 Finding newMin and newMax

int newMin = INT\_MAX, newMax;

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

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

if(zeroFrameAry[i][j] > newMax) {newMax = zeroFrameAry[i][j];}

if(zeroFrameAry[i][j] < newMin) {newMin = zeroFrameAry[i][j];}

}

}

//Step 10

labelFile << numRows << " " << numCols << " " << newMin << " " << newMax << endl;

//Step 11

PrintImg(zeroFrameAry, labelFile, numRows, numCols);

//Step 12

propertyFile << numRows << " " << numCols << " " << newMin << " " << newMax << endl;

propertyFile << numOfCC << endl;

printCCProperty(CCproperty, propertyFile, numOfCC);

//Step 13

drawBoxes(zeroFrameAry, CCproperty, numOfCC);

//Step 14

prettyPrintFile << "zeroFrameAry after drawBoxes(): " << endl;

prettyPrint(zeroFrameAry, prettyPrintFile, numRows+2, numCols+2);

//Step 15

inFile.close();

prettyPrintFile.close();

labelFile.close();

propertyFile.close();

} // end of main

// Functions

void drawBoxes(int\*\* zeroFrameAry, Property\* CCproperty, int numOfCC){

int index = 1;

while(index < numOfCC+1){

int minRow = CCproperty[index].upperLftR + 1;

int minCol = CCproperty[index].upperLftC + 1;

int maxRow = CCproperty[index].lowerRgtR + 1;

int maxCol = CCproperty[index].lowerRgtC + 1;

int label = CCproperty[index].label;

for(int i = minCol; i < maxCol; i++){ zeroFrameAry[minRow][i] = label;}

for(int i = minCol; i < maxCol; i++){ zeroFrameAry[maxRow][i] = label;}

for(int i = minRow; i < maxRow; i++){ zeroFrameAry[i][minCol] = label;}

for(int i = minRow; i < maxRow; i++){ zeroFrameAry[i][maxCol] = label;}

index++;

}

}

void printCCProperty(Property\* CCproperty, fstream& outFile, int numOfCC){

int sizeCCP = numOfCC+1;

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

outFile << CCproperty[i].label << endl;

outFile << CCproperty[i].numpixels << endl;

outFile << CCproperty[i].upperLftR << " " << CCproperty[i].upperLftC << endl;

outFile << CCproperty[i].lowerRgtR << " " << CCproperty[i].lowerRgtC << endl;

}

}

void PrintImg(int\*\* Ary, fstream& outFile, int rows, int cols){

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

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

outFile << Ary[i][j] << " ";

}

outFile << endl;

}

}

void connectPass3(int\* EQArray, int\*\* Ary, Property\* CCproperty, int rows, int cols){

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

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

if(Ary[i][j] > 0){

Ary[i][j] = EQArray[ Ary[i][j] ];

CCproperty[ Ary[i][j] ].label = Ary[i][j];

CCproperty[ Ary[i][j] ].numpixels++;

if(i < CCproperty[ Ary[i][j] ].upperLftR) {CCproperty[Ary[i][j]].upperLftR = i;} // lowest i

if(j < CCproperty[ Ary[i][j] ].upperLftC) {CCproperty[Ary[i][j]].upperLftC = j;} // lowest j

if(i > CCproperty[ Ary[i][j] ].lowerRgtR) {CCproperty[Ary[i][j]].lowerRgtR = i;} // highest i

if(j > CCproperty[ Ary[i][j] ].lowerRgtC) {CCproperty[Ary[i][j]].lowerRgtC = j;} // highest j

}

}

}

}

void manageEQAry(int\* EQArray, int newLabel){

//0

int realLabel = 0;

//1

int index = 1;

//2-3

while(index <= newLabel){

//2

if(index != EQArray[index]){

EQArray[index] = EQArray[EQArray[index]];

} else {

realLabel++;

EQArray[index] = realLabel;

}

//3

index++;

}// 4(loop)

}

int loadNonZeroPass2(int\*\* Ary, int Connectness, int i, int j, int\* NonZeroNeighborAry, bool &diffFlag, int &nonZeroCount){

minus1D(NonZeroNeighborAry);

nonZeroCount = 0;

//1

NonZeroNeighborAry[nonZeroCount] = Ary[i][j];

nonZeroCount++;

//2

if(Ary[i+1][j] > 0){

NonZeroNeighborAry[nonZeroCount] = Ary[i+1][j];

nonZeroCount++;

}

//3

if(Ary[i][j+1] > 0){

NonZeroNeighborAry[nonZeroCount] = Ary[i][j+1];

nonZeroCount++;

}

//4

if(Connectness == 8){

if(Ary[i+1][j-1] > 0){

NonZeroNeighborAry[nonZeroCount] = Ary[i+1][j-1];

nonZeroCount++;

}

if(Ary[i+1][j+1] > 0){

NonZeroNeighborAry[nonZeroCount] = Ary[i+1][j+1];

nonZeroCount++;

}

}

//5

int minLabel = NonZeroNeighborAry[0];

diffFlag = false;

//6

int index = 1;

// 7-8

while(index < nonZeroCount){

//7 (new specs)

if(minLabel != NonZeroNeighborAry[index]){

diffFlag = true;

}

if(minLabel > NonZeroNeighborAry[index]){

minLabel = NonZeroNeighborAry[index];

}

//8

index++;

}// 9(loop)

//10

return minLabel;

}

void connectPass2(int Connectness, int\*\* Ary, int\* NonZeroNeighborAry, int rows, int cols, int\* EQArray){

int minLabel;

int nonZeroCount;

bool diffFlag;

for(int I = rows; I > 0; i--){ //R-L

for(int j = cols; j > 0; j--){//B-T

if(Ary[i][j] > 0){

minLabel = loadNonZeroPass2(Ary, Connectness, I, j, NonZeroNeighborAry, diffFlag, nonZeroCount);

if(minLabel != Ary[i][j]){

Ary[i][j] = minLabel;

}

if(diffFlag == true){

updateEQ(EQArray, NonZeroNeighborAry, minLabel, nonZeroCount);

}

}

}

}

}

void printEQAry(int newLabel, int\* EQArray, fstream& outFile){

for(int I = 0; I < newLabel; i++){

outFile << EQArray[i] << “ “;

}

outFile << endl;

}

void prettyPrint(int\*\* array, fstream& outFile, int rows, int cols){

for(int I = 0; I < rows; i++){

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

if(array[i][j] == 0){

outFile << “. “;

} else {

outFile << array[i][j] << “ “;

}

}

outFile << endl;

}

outFile << endl;

}

void updateEQ(int\* EQArray, int\* NonZeroNeighborAry, int minLabel, int nonZeroCount){

int index = 0;

while(index < nonZeroCount && NonZeroNeighborAry[index] != -1){

EQArray[NonZeroNeighborAry[index]] = minLabel;

index++;

}

}

void minus1D(int\* Ary1D){

for(int I = 0; I < 5; i++){ //Size has been hardcoded

Ary1D[i] = -1;

}

}

int loadNonZeroPass1(int\*\* Ary, int Connectness, int I, int j, int\* NonZeroNeighborAry, bool &diffFlag, int &nonZeroCount){

minus1D(NonZeroNeighborAry);

nonZeroCount = 0;

//1

if(Ary[i-1][j] > 0){

NonZeroNeighborAry[nonZeroCount] = Ary[i-1][j];

nonZeroCount++;

}

//2

if(Ary[i][j-1] > 0){

NonZeroNeighborAry[nonZeroCount] = Ary[i][j-1];

nonZeroCount++;

}

//3

if(Connectness == 8){

if(Ary[i-1][j-1] > 0){

NonZeroNeighborAry[nonZeroCount] = Ary[i-1][j-1];

nonZeroCount++;

}

if(Ary[i-1][j+1] > 0){

NonZeroNeighborAry[nonZeroCount] = Ary[i-1][j+1];

nonZeroCount++;

}

}

//4

if(nonZeroCount <= 0){return 0;}

//5

int minLabel = NonZeroNeighborAry[0];

diffFlag = false;

//6

int index = 1;

//7-8

while(index < nonZeroCount){

//7 (new specs)

if(minLabel != NonZeroNeighborAry[index]){

diffFlag = true;

}

if(minLabel > NonZeroNeighborAry[index]){

minLabel = NonZeroNeighborAry[index];

}

//8

index++;

}// 9(loop)

//10

return minLabel;

}

int connectPass1(int Connectness, int\*\* Ary, int\* NonZeroNeighborAry, int rows, int cols, int\* EQArray){

int newLabel = 0;

int minLabel;

int nonZeroCount;

bool diffFlag;

for(int I = 1; I < rows+1; i++){

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

if (Ary[i][j] > 0){

minLabel = loadNonZeroPass1(Ary, Connectness, I, j, NonZeroNeighborAry, diffFlag, nonZeroCount);

if(minLabel == 0){

newLabel++;

Ary[i][j] = newLabel;

} else if (minLabel > 0){

Ary[i][j] = minLabel;

if (diffFlag == true){

updateEQ(EQArray, NonZeroNeighborAry, minLabel, nonZeroCount);

}

}

}

}

} // end of loop

return newLabel;

}

void loadImage(fstream& inFile, int\*\* zeroFrameAry, int rows, int cols){

int value;

for(int I = 1; I < rows+1; i++){

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

inFile >> value;

zeroFrameAry[i][j] = value;

}

}

}

void zero2Dary(int\*\* array, int rows, int cols){

for(int I = 0; I < rows; i++){

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

array[i][j] = 0;

}

}

}

**END OF SOURCE CODE**

**Output for 4-Connectness:**

**prettyPrintFile**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a social media post

Description automatically generated**

**END OF PRETTY\_4**

**labelFile**

**A picture containing microwave, computer, clock

Description automatically generated**

**END OF LABEL\_4**

**A close up of a logo

Description automatically generatedpropertyFile**

**A screenshot of a cell phone

Description automatically generated**

**END OF PROPERTY\_4**

**END OF CONNECTNESS 4 OUTPUT**

**Output for 4-Connectness:**

**prettyPrintFile**

**A close up of text on a white background

Description automatically generated**

**A screenshot of a social media post

Description automatically generated**

**END OF PRETTY\_8**

**labelFile**

**A picture containing white, microwave

Description automatically generated**

**END OF LABEL\_8**

**propertyFile**

**A screenshot of a cell phone

Description automatically generated**

**END OF PROPERTY\_8**

**END OF CONNECTNESS 8 OUTPUT**