CSC 381 Computer Vision ( C++ )

Project 5: 8-connected component & bounding boxes.

Han Wen Loh

Soft copy: 03/27/2019

Hard copy: 04/2/2019

Algorithm Steps:

step 0: inFile 🡨 open the input file

numRows, numCols, minVal, maxVal 🡨 read the image header

dynamically allocate zeroFramedAry and initialize to zero.

outFile1, outFile1, outFile3, outFile4 🡨 open

step 1: loadImage(inFile, zeroFramedAry)

// read from input file and write to zeroFramedAry begin at(1,1)

step 2: 8ConnectCCPass1 (...) // as taught in class

prettyPrint (outFile1) // the result of pass1

printEQAry (outFile1)// with index up to newLable with proper caption

step 3: 8ConnectCCPass2 (...)// as taught in class

prettyPrint (outFile1) // the result of pass2

printEQAry (outFile1) // with index up to newLable with caption

step 4: manageEQAry (...)// algorithm was given in class.

printEQAry (outFile1)// with index up to newLable with caption

step 5: 8ConnectCCPass3 (...) // as taught in class

prettyPrint (outFile1) // the result of pass3

printEQAry (outFile1) // with index up to newLable with caption

step 6: output numRows, numCols, newMin, newMax to outFile2

step 7: Output the result of pass3 from zeroFramedAry to outFile2,

begins at (1, 1) and ending at ??

step 8: printCCproperty (...)to outFile 3

step 9: drawBoxes(zeroFramedAry, CCproperty)

step 10: output zeroFrameAry to outFile4

step 11: close all files

#include <iostream>

#include <fstream>

using namespace std;

class ConnectedComp {

private:

int numRows, numCols, minVal, maxVal, newMin, newMax, smallestLabel, largestLabel, newLabel, numNb;

int\*\* zeroFramedAry;

int\* neighborAry;

int\* EQAry;

int\*\* CCproperty; //[label][numpixels, minRow, minCol, maxRow, maxCol]

int checkFrontNeighbor(int i, int j) {

smallestLabel = largestLabel + 1;

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

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

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

neighborAry[3] = zeroFramedAry[i][j - 1];

int sameLabel = 1, diffLabel = 0;

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

smallestLabel = (neighborAry[i] > 0 && neighborAry[i] < smallestLabel) ? neighborAry[i] : smallestLabel;

if (neighborAry[i] > 0 && sameLabel == 1) {

if (diffLabel > 0)

sameLabel = diffLabel != neighborAry[i] ? 0 : 1;

else

diffLabel = neighborAry[i];

}

if (sameLabel <= 0) //if found any different label in neighbor return case #2

return 2;

}

if (diffLabel <= 0) //if all label are 0 return case #0

return 0;

return 1; //if all label are same return case #1

}

int checkBackNeighbor(int i, int j) {

smallestLabel = largestLabel + 1;

neighborAry[0] = zeroFramedAry[i][j + 1];

neighborAry[1] = zeroFramedAry[i + 1][j - 1];

neighborAry[2] = zeroFramedAry[i + 1][j];

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

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

int sameLabel = 1, diffLabel = 0;

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

smallestLabel = (neighborAry[i] > 0 && neighborAry[i] < smallestLabel) ? neighborAry[i] : smallestLabel;

if (neighborAry[i] > 0 && sameLabel == 1) {

if (diffLabel > 0)

sameLabel = diffLabel != neighborAry[i] ? 0 : 1;

else

diffLabel = neighborAry[i];

}

if (sameLabel <= 0)

return 2;

}

if (diffLabel == 0)

return 0;

return 1;

}

void updateEQAry(int p, int label) {

EQAry[p] = label;

}

void prettyPrint(ofstream& outFile) {

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

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

if (zeroFramedAry[i][j] >= 10)

outFile << zeroFramedAry[i][j];

else if (zeroFramedAry[i][j] > 0)

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

else

outFile << " ";

}

outFile << endl;

}

}

void printEQAry(ofstream& outFile) {

outFile << "\n |";

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

if (i < 10)

outFile << i << " |";

else

outFile << i << "|";

outFile << "\n |";

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

outFile << "--+";

outFile << "\nEQAry|";

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

if (EQAry[i] < 10)

outFile << EQAry[i] << " |";

else

outFile << EQAry[i] << "|";

}

outFile << "\n\n";

}

~ConnectedComp() {

delete[] neighborAry;

delete[] EQAry;

delete[] CCproperty;

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

delete[] zeroFramedAry[i];

}

public:

ConnectedComp(ifstream & inFile) {

inFile >> numRows >> numCols >> minVal >> maxVal;

newMin = newMax = newLabel = 0;

numNb = 4;

neighborAry = new int[numNb+1]();

largestLabel = numRows \* numCols / 2;

EQAry = new int[largestLabel];

smallestLabel = largestLabel;

for (int i = 0; i < largestLabel; ++i)

EQAry[i] = i;

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

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

zeroFramedAry[i] = new int[numCols]();

CCproperty = NULL;

}

void loadImg(ifstream& inFile) {

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

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

inFile >> zeroFramedAry[i][j];

}

void eightCCCpass1(ofstream& outFile) {

int caseNum; //0 = all neighbor are 0, 1 = neighbor has same label, 2 = neighbor has different

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

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

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

caseNum = checkFrontNeighbor(i, j);

if (caseNum == 0) {

++newLabel;

zeroFramedAry[i][j] = EQAry[newLabel];

}

else if (caseNum == 1){

zeroFramedAry[i][j] = EQAry[smallestLabel];

}

else {

zeroFramedAry[i][j] = EQAry[smallestLabel];

}

}

}

}

outFile << "Pass #1:\n";

prettyPrint(outFile);

printEQAry(outFile);

}

void eightCCCpass2(ofstream& outFile) {

int caseNum; //0 = all neighbor are 0, 1 = neighbor has same label, 2 = neighbor has different

for (int i = numRows; i > 0; --i) {

for (int j = numCols; j > 0; --j) {

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

caseNum = checkBackNeighbor(i, j);

if(caseNum >= 2) {

updateEQAry(zeroFramedAry[i][j], smallestLabel);

zeroFramedAry[i][j] = EQAry[smallestLabel];

}

}

}

}

outFile << "Pass #2:\n";

prettyPrint(outFile);

printEQAry(outFile);

}

void eightCCCpass3(ofstream& outFile) {

int currentLabel = 0;

CCproperty = new int\* [newMax];

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

CCproperty[i] = new int[5]();

CCproperty[i][1] = numRows-1;

CCproperty[i][2] = numCols-1;

}

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

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

zeroFramedAry[i][j] = EQAry[zeroFramedAry[i][j]];

currentLabel = zeroFramedAry[i][j]-1;

if (currentLabel >= 0) {

++CCproperty[currentLabel][0];

if (i - 1 < CCproperty[currentLabel][1]) { CCproperty[currentLabel][1] = i - 1; }

if (j - 1 < CCproperty[currentLabel][2]) { CCproperty[currentLabel][2] = j - 1; }

if (i - 1 > CCproperty[currentLabel][3]) { CCproperty[currentLabel][3] = i - 1; }

if (j - 1 > CCproperty[currentLabel][4]) { CCproperty[currentLabel][4] = j - 1; }

}

}

}

outFile << "Pass #3:\n";

prettyPrint(outFile);

printEQAry(outFile);

}

void manageEQAry(ofstream& outFile) {

int currentMax = 0;

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

if (EQAry[i] > currentMax) {

currentMax = EQAry[i];

EQAry[i] = ++newMax;

}

else {

EQAry[i] = EQAry[EQAry[i]];

}

}

outFile << "Managed EQAry:\n";

printEQAry(outFile);

}

void printLabeledCC(ofstream& outFile) {

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

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

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

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

}

outFile << endl;

}

}

void printCCproperty(ofstream& outFile) {

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

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

outFile << i + 1 << endl << CCproperty[i][0] << endl << CCproperty[i][1] << " " << CCproperty[i][2] << endl << CCproperty[i][3] << " " << CCproperty[i][4] << endl;

}

}

void drawBoxes(ofstream& outFile) {

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

for (int c = CCproperty[i][2] + 1; c <= CCproperty[i][4] + 1; ++c) {

if(zeroFramedAry[CCproperty[i][1] + 1][c] <= i)

zeroFramedAry[CCproperty[i][1]+1][c] = i+1;

if(zeroFramedAry[CCproperty[i][3] + 1][c] <= i)

zeroFramedAry[CCproperty[i][3]+1][c] = i+1;

}

for (int r = CCproperty[i][1] + 1; r <= CCproperty[i][3] + 1; ++r) {

if(zeroFramedAry[r][CCproperty[i][2] + 1] <= i)

zeroFramedAry[r][CCproperty[i][2]+1] = i+1;

if(zeroFramedAry[r][CCproperty[i][4] + 1] <= i)

zeroFramedAry[r][CCproperty[i][4]+1] = i+1;

}

}

prettyPrint(outFile);

}

};

int main(int args, char\*\* argv) {

ifstream inFile;

ofstream outFile1, outFile2, outFile3, outFile4;

inFile.open(argv[1]);

if (inFile.fail()) {

cout << "ERROR: cannot find \"" << argv[1] << "\"\n";

exit(1);

}

outFile1.open(argv[2]);

outFile2.open(argv[3]);

outFile3.open(argv[4]);

outFile4.open(argv[5]);

ConnectedComp\* connectedComp = new ConnectedComp(inFile);

connectedComp->loadImg(inFile);

connectedComp->eightCCCpass1(outFile1);

connectedComp->eightCCCpass2(outFile1);

connectedComp->manageEQAry(outFile1);

connectedComp->eightCCCpass3(outFile1);

connectedComp->printLabeledCC(outFile2);

connectedComp->printCCproperty(outFile3);

connectedComp->drawBoxes(outFile4);

inFile.close();

outFile1.close();

outFile2.close();

outFile3.close();

outFile4.close();

return 0;

}

3passes.txt

Pass #1:

1

2 2 2 1 1 1 3 3 3

2 2 2 1 1 1 1 1 3 3 3

2 2 2 1 1 1 1 1 3 3 3

2 2 2 1 1 1 1 1 1 1 3 3 3

4 2 2 2 2 2 1 1 1 1 1 1 1 1 1 3 3 3

4 2 2 2 1 1 1 1 1 1 1 1 1 1 1 3 3 3

4 4 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 3

2 1 1 1 1 1 1 1 1 1 1 1 1 1 3

5 5 5 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 3

5 5 5 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1

5 5 5 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1

5 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1

5 1 1 1 1 1 1 1 1 1 1 1 1 6

1 1 1 1 1 1 1 1 6 6

7 1 1 1 1 1 6 6

8 7 9 1 1 1 10 116 6

8 7 7 7 7 1 1 1 10 6 6 6 6

8 7 7 7 7 7 6 6 6 6

7 7 7 7 12 6 6 6 6

12 6 6 6

121212 6 6 6

121212 6

|0 |1 |2 |3 |4 |5 |6 |7 |8 |9 |10|11|12|

|--+--+--+--+--+--+--+--+--+--+--+--+--+

EQAry|0 |1 |2 |3 |4 |5 |6 |7 |8 |9 |10|11|12|

Pass #2:

1

1 1 1 1 1 1 3 3 3

1 1 1 1 1 1 1 1 3 3 3

1 1 1 1 1 1 1 1 3 3 3

1 1 1 1 1 1 1 1 1 1 3 3 3

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3

2 1 1 1 1 1 1 1 1 1 1 1 1 1 3

5 5 5 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 3

5 5 5 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1

5 5 5 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1

5 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1

5 1 1 1 1 1 1 1 1 1 1 1 1 6

1 1 1 1 1 1 1 1 6 6

7 1 1 1 1 1 6 6

7 7 7 1 1 1 1 6 6 6

7 7 7 7 7 1 1 1 10 6 6 6 6

7 7 7 7 7 7 6 6 6 6

7 7 7 7 12 6 6 6 6

12 6 6 6

121212 6 6 6

121212 6

|0 |1 |2 |3 |4 |5 |6 |7 |8 |9 |10|11|12|

|--+--+--+--+--+--+--+--+--+--+--+--+--+

EQAry|0 |1 |1 |3 |1 |5 |6 |7 |7 |7 |1 |6 |12|

Managed EQAry:

|0 |1 |2 |3 |4 |5 |6 |7 |8 |9 |10|11|12|

|--+--+--+--+--+--+--+--+--+--+--+--+--+

EQAry|0 |1 |1 |2 |1 |3 |4 |5 |5 |5 |1 |4 |6 |

Pass #3:

1

1 1 1 1 1 1 2 2 2

1 1 1 1 1 1 1 1 2 2 2

1 1 1 1 1 1 1 1 2 2 2

1 1 1 1 1 1 1 1 1 1 2 2 2

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2

1 1 1 1 1 1 1 1 1 1 1 1 1 1 2

3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2

3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1

3 1 1 1 1 1 1 1 1 1 1 1 1 4

1 1 1 1 1 1 1 1 4 4

5 1 1 1 1 1 4 4

5 5 5 1 1 1 1 4 4 4

5 5 5 5 5 1 1 1 1 4 4 4 4

5 5 5 5 5 5 4 4 4 4

5 5 5 5 6 4 4 4 4

6 4 4 4

6 6 6 4 4 4

6 6 6 4

|0 |1 |2 |3 |4 |5 |6 |7 |8 |9 |10|11|12|

|--+--+--+--+--+--+--+--+--+--+--+--+--+

EQAry|0 |1 |1 |2 |1 |3 |4 |5 |5 |5 |1 |4 |6 |

Connected\_components.txt

24 31 0 6

6

1

186

1 4

18 22

2

21

2 23

10 28

3

13

10 2

14 4

4

27

14 23

23 29

5

19

16 2

20 7

6

8

20 14

23 16

Bounding\_boxes.txt

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 2 2 2 2 2 2

1 1 1 1 1 1 1 1 1 1 2 2 2 2 2

1 1 1 1 1 1 1 1 1 1 2 2 2 2 2

1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2

3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2

3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1

3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4

1 1 1 1 1 1 1 1 1 1 4 4 4

5 5 5 5 5 5 1 1 1 1 1 1 4 4 4

5 5 5 1 1 1 1 1 4 4 4 4

5 5 5 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 4 4

5 5 5 5 5 5 4 4 4 4 4 4

5 5 5 5 5 5 6 6 6 4 4 4 4 4 4

6 6 6 4 4 4 4

6 6 6 4 4 4 4 4

6 6 6 4 4 4 4 4 4 4