CV Programming Language: CPP

Project #10 Chain Code

Andres Quintero

Due Date:

Soft copy: 4/23/2020

Hard copy: 4/23/2020

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

Step 1: labelFile <-- open label file from argv[1]

propFile <-- open property file from argv[2]

output image header to ChainCodeFile

output image header to deBugFile // per text line

imageAry <-- dynamically allocated

loadImage (imageAry )

CCAry <-- dynamically allocated

Step 2: CC <-- get the next connected component from the property file

Step 3: CClabel <-- get the label of CC

Step 4: clearCC () // zero out the old CClabel for next cc

Step 5: loadCC (CClabel, CCAry)

Step 6: getChainCode (CC, CCAry) // see algorithm below

Step 7: repeat step 2 to step 5 until all connected components are processed.

Step 8: close all files

**Hand tracing of Img1**

A close up of text on a white background

Description automatically generated

**Source code:**

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

class Image{

public:

int numRows;

int numCols;

int minVal;

int maxVal;

int\*\* imgAry;

int\*\* CCAry;

Image(ifstream& labelFile){

labelFile >> numRows;

labelFile >> numCols;

labelFile >> minVal;

labelFile >> maxVal;

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

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

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

}

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

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

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

}

zeroFramed();

}

void zeroFramed(){

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

for(int j = 0; j < numCols+2; j++){

imgAry[i][j] = 0;

CCAry[i][j] = 0;

}

}

}

void loadImage(ifstream& inFile){

int value;

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

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

inFile >> value;

imgAry[i][j] = value;

}

}

}

};

class connectCC{

public:

int label;

int numPixels;

int minRow;

int minCol;

int maxRow;

int maxCol;

connectCC(int labelNum, int pixelNum, int rowMin, int colMin, int rowMax, int colMax){

label = labelNum;

numPixels = pixelNum;

minRow = rowMin;

minCol = colMin;

maxRow = rowMax;

maxCol = colMax;

}

void clearCC(int\*\* CCAry, int numRows, int numCols){

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

for(int j = 0; j < numCols+2; j++){

CCAry[i][j] = 0;

}

}

}

void loadCC(int\*\* imgAry, int\*\* CCAry){

for(int i = minRow; i < maxRow+2; i++){

for(int j= minCol; j < maxCol+2; j++){

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

CCAry[i][j] = label;

}

}

}

}

};

class ChainCode{

public:

class Point{

public:

int row;

int col;

Point(int x, int y){

row = x;

col = y;

}

bool isEqual(Point& second){

bool rowEqual, colEqual;

rowEqual = this->row == second.row;

colEqual = this->col == second.col;

return rowEqual && colEqual;

}

};

Point neighborCoord[8] = {Point(-1,-1), Point(-1,-1), Point(-1,-1), Point(-1,-1), Point(-1,-1), Point(-1,-1), Point(-1,-1), Point(-1,-1) };

Point startP = Point(-1,-1);

Point currentP = Point(-1,-1);

Point nextP = Point(-1,-1);

int lastQ;

int zeroTable[8] = {6,0,0,2,2,4,4,6};

int nextDir;

int pChainDir;

ChainCode(){

}

void getChainCode(connectCC CC, int\*\* CCAry, int\*\* imgAry, ofstream& ChainCodeFile, ofstream& debugFile){

int label = CC.label;

bool found = false;

for(int iRow = CC.minRow+1; iRow < CC.maxRow+2; iRow++){

for(int jCol = CC.minCol+1; jCol < CC.maxCol+2; jCol++){

if(CCAry[iRow][jCol] == label && !found){

ChainCodeFile << label << " " << iRow << " " << jCol << " " ;

debugFile << label << " " << iRow << " " << jCol << endl ;

startP = Point(iRow, jCol);

currentP = Point(iRow, jCol);

lastQ = 4;

found = true;

}

}

}

int debugCount = 0;

int doOnce = 1; // at the begining the currentP and startP are the same

int nextQ;

while(doOnce > 0 || !(currentP.isEqual(startP)) ){

doOnce = 0;

pChainDir = findNextP(currentP, lastQ, imgAry);

currentP.row = -currentP.row;

currentP.col = -currentP.col;

ChainCodeFile << pChainDir;

debugFile << pChainDir << " ";

if(++debugCount == 20){

debugFile << endl;

debugCount = 0;

}

lastQ = zeroTable[(pChainDir+7) % 8];

currentP.row = nextP.row;

currentP.col = nextP.col;

}

ChainCodeFile << endl;

debugFile << endl;

}

int findNextP(Point currentP, int lastQ, int\*\* imgAry){

loadNeighborCoord(currentP);

int chainDir = ++lastQ;

int loop = 0;

int i = currentP.row, j = currentP.col;

while(loop < 8){

switch(chainDir){

case 0:

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

chainDir = 0;

nextP = neighborCoord[chainDir];

return chainDir;

}

break;

case 1:

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

chainDir = 1;

nextP = neighborCoord[chainDir];

return chainDir;

}

break;

case 2:

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

chainDir = 2;

nextP = neighborCoord[chainDir];

return chainDir;

}

break;

case 3:

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

chainDir = 3;

nextP = neighborCoord[chainDir];

return chainDir;

}

break;

case 4:

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

chainDir = 4;

nextP = neighborCoord[chainDir];

return chainDir;

}

break;

case 5:

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

chainDir = 5;

nextP = neighborCoord[chainDir];

return chainDir;

}

break;

case 6:

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

chainDir = 6;

nextP = neighborCoord[chainDir];

return chainDir;

}

break;

case 7:

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

chainDir = 7;

nextP = neighborCoord[chainDir];

return chainDir;

}

break;

}

chainDir = (chainDir+1) % 8;

loop++;

}

return -1;

}

void loadNeighborCoord(Point p){

int i = p.row;

int j = p.col;

neighborCoord[0] = Point(i,j+1);

neighborCoord[1] = Point(i-1,j+1);

neighborCoord[2] = Point(i-1,j);

neighborCoord[3] = Point(i-1,j-1);

neighborCoord[4] = Point(i,j-1);

neighborCoord[5] = Point(i+1,j-1);

neighborCoord[6] = Point(i+1,j);

neighborCoord[7] = Point(i+1,j+1);

}

void prettyPrint(ofstream& outFile){

} // Not called?

};

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

ifstream labelFile(argv[1]);

ifstream propFile(argv[2]);

ofstream ChainCodeFile(argv[3]);

ofstream debugFile(argv[4]);

Image image(labelFile);

ChainCodeFile << image.numRows << " " << image.numCols << " " << image.minVal << " " << image.maxVal << endl;

debugFile << image.numRows << " " << image.numCols << " " << image.minVal << " " << image.maxVal << endl;

image.loadImage(labelFile);

int dummyRead; // to get the correct spot

propFile >> dummyRead; propFile >> dummyRead; propFile >> dummyRead; propFile >> dummyRead; // imageHeader

int totalCC;

int proccessedCC = 0;

propFile >> totalCC;

int label, numPixels, minRow, minCol, maxRow, maxCol;

// StartOfLoop

while(proccessedCC < totalCC){

propFile >> label;

propFile >> numPixels;

propFile >> minRow;

propFile >> minCol;

propFile >> maxRow;

propFile >> maxCol;

connectCC CC(label, numPixels, minRow, minCol, maxRow, maxCol);

CC.clearCC(image.CCAry, image.numRows, image.numCols);

CC.loadCC(image.imgAry, image.CCAry);

ChainCode chainCode;

chainCode.getChainCode(CC, image.CCAry, image.imgAry, ChainCodeFile, debugFile);

proccessedCC++;

}

labelFile.close();

propFile.close();

ChainCodeFile.close();

debugFile.close();

}

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

IMG1 outputs

A screenshot of a social media post

Description automatically generated

IMG2 outputs

A screenshot of a cell phone

Description automatically generated