CSC 381 Computer Vision ( C++ )

Project 8: Document text-line detection

Han Wen Loh

Soft copy: 05/04/2019

Hard copy: 05/09/2019

Algorithm Steps:

step 0: inFile 🡨 open input file

outFile 🡨 given

thrVal 🡨 given

numRows, numCols, minVal, maxVal 🡨 read from inFile

dynamically allocate imgAry size of numRows by numCols

dynamically allocate HPP, VPP, HPPBin, VPPBin, HPPMorph, and VPPMorph

size of numRows+2 by numCol+2

Step 1: computeHPP(HPP)

computeVPP(VPP)

printPP(HPP, outFile)

printPP(VPP, outFile)

Step 2: threshold(HPP, HPPBin)

Threshold(VPP, VPPBin)

printPP(HPPBin, outFile)

printPP(VPPBin, outFile)

Step 3: morphClosing(HPPBin, HPPMorph)

morphClosing(VPPBin, VPPMorph)

printPP(HPPMorph, outFile)

printPP(VPPMorph, outFile)

Step 4: readingDir 🡨 determineReadingDir(HPPMorph, VPPMorph)

printReadingDir(readingDir, outFile)

Step 5: boxQ 🡨 findLineBoxes()

Step 6: printBoxesQ(boxQ, outFile)

Step 7: close all files

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class ImagePP {

public:

class Box {

public:

int minR, minC, maxR, maxC;

Box() {

minR = minC = maxR = maxC = 0;

}

Box(int miR, int miC) {

minR = miR;

minC = miC;

maxR = maxC = 0;

}

};

class BoxNode {

public:

int boxType; //1 - doc box, 2 - paragraph, 3 - textLine

Box\* bBox;

BoxNode\* next;

BoxNode() {

boxType = -1;

bBox = NULL;

next = NULL;

}

BoxNode(int boxT, Box\* bB) {

boxType = boxT;

bBox = bB;

next = NULL;

}

};

class BoxQ {

public:

BoxNode\* qFront;

BoxNode\* qBack;

BoxQ(BoxNode\* front) {

qFront = front;

qBack = qFront;

}

void insert(BoxNode\* newBoxNode) {

qBack->next = newBoxNode;

qBack = newBoxNode;

}

};

int\*\* imgAry;

Box\* imgBox;

int numRows, numCols, minVal, maxVal;

int thrVal, hppRuns, vppRuns;

int\* hpp;

int\* vpp;

int\* hppBin;

int\* vppBin;

int\* hppMorph;

int\* vppMorph;

string readingDir;

BoxQ\* boxQ;

ImagePP(ifstream& inFile, int t) {

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

imgBox = new Box();

imgBox->minR = numRows;

imgBox->minC = numCols;

imgAry = new int\* [numRows];

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

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

thrVal = t;

hppRuns = vppRuns = 0;

hpp = new int[numRows + 2]();

vpp = new int[numCols + 2]();

hppBin = new int[numRows + 2]();

vppBin = new int[numCols + 2]();

hppMorph = new int[numRows + 2]();

vppMorph = new int[numCols + 2]();

readingDir = "";

boxQ = new BoxQ(new BoxNode());

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

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

inFile >> imgAry[i][j];

}

}

void computePP(ofstream &outFile) {

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

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

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

++hpp[i + 1];

++vpp[j + 1];

}

}

outFile << "HPP\n";

printPP(hpp, numRows, outFile);

outFile << "VPP\n";

printPP(vpp, numCols, outFile);

}

void printPP(int\* pp, int size, ofstream& outFile) {

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

outFile << pp[i] << " ";

}

outFile << endl << endl;

}

void threshold(ofstream& outFile) {

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

if (hpp[i] > thrVal) {

hppBin[i] = 1;

//compute image horizontal boundary box

imgBox->minR = imgBox->minR < i ? imgBox->minR : i;

imgBox->maxR = imgBox->maxR > i ? imgBox->maxR : i;

}

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

if (vpp[j] > thrVal) {

vppBin[j] = 1;

//compute image vertical boundary box

imgBox->minC = imgBox->minC < j ? imgBox->minC : j;

imgBox->maxC = imgBox->maxC > j ? imgBox->maxC : j;

}

--imgBox->minR;

--imgBox->minC;

--imgBox->maxR;

--imgBox->maxC;

outFile << "HPPBinary\n";

printPP(hppBin, numRows, outFile);

outFile << "VPPBinary\n";

printPP(vppBin, numCols, outFile);

}

void morphClosing(ofstream& outFile) {

//strcElep[] = { 1, 1, 1 };

int\* hppTempAry = new int[numRows + 2]();

int\* vppTempAry = new int[numCols + 2]();

dilation(hppBin, hppTempAry, numRows);

dilation(vppBin, vppTempAry, numCols);

erosion(hppTempAry, hppMorph, numRows);

erosion(vppTempAry, vppMorph, numCols);

outFile << "HPPMorph\n";

printPP(hppMorph, numRows, outFile);

outFile << "VPPMorph\n";

printPP(vppMorph, numCols, outFile);

}

void dilation(int\* pp, int\* processedPP, int size) {

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

if (pp[i] > 0) {

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

processedPP[j] = 1;

}

}

}

}

void erosion(int\* pp, int\* processedPP, int size) {

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

if (pp[i] > 0 && pp[i - 1] > 0 && pp[i + 1] > 0)

processedPP[i] = 1;

}

void determineReadingDir(ofstream& outFile) {

int factor = 3;

hppRuns = computeRuns(hppMorph, numRows);

vppRuns = computeRuns(vppMorph, numCols);

if (hppRuns >= factor \* vppRuns) {

readingDir = "horizontal";

outFile << "Reading Direction : " << readingDir << endl << endl;

}else if (vppRuns >= factor \* hppRuns) {

readingDir = "vertical";

outFile << "Reading Direction: " << readingDir << endl << endl;

}else {

cout << "Cannot determine the reading direction.\n";

exit(1);

}

}

int computeRuns(int\* pp, int size) {

int numRuns = 0;

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

if (pp[i] > 0 && pp[i + 1] <= 0)

++numRuns;

return numRuns;

}

void findLineBoxes(ofstream& outFile) {

if (readingDir.compare("horizontal") == 0) {

findLineBoxes(hppMorph,imgBox->minR, imgBox->minC, imgBox->maxR, imgBox->maxC);

}

else {

findLineBoxes(vppMorph, imgBox->minC, imgBox->minR, imgBox->maxC, imgBox->maxR);

}

outFile << "text-line bounding box\n";

printBoxQueue(outFile);

}

void findLineBoxes(int\* pp, int minSize, int otherMinSize, int maxSize, int otherMaxSize) {

BoxNode\* newBoxNode;

Box\* box;

for (int i = minSize; i <= maxSize; ++i) {

if (pp[i + 1] > 0) {

box = new Box(numRows, numCols);

newBoxNode = new BoxNode(3, box);

while (pp[i + 1] > 0) {

for (int j = otherMinSize; j <= otherMaxSize; ++j) {

if (readingDir.compare("horizontal") == 0) {

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

box->minR = box->minR < i ? box->minR : i;

box->minC = box->minC < j ? box->minC : j;

box->maxR = box->maxR > i ? box->maxR : i;

box->maxC = box->maxC > j ? box->maxC : j;

}

}

else {

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

box->minR = box->minR < j ? box->minR : j;

box->minC = box->minC < i ? box->minC : i;

box->maxR = box->maxR > j ? box->maxR : j;

box->maxC = box->maxC > i ? box->maxC : i;

}

}

}

++i;

}

boxQ->insert(newBoxNode);

}

}

}

void printBoxQueue(ofstream& outFile) {

BoxNode\* current = boxQ->qFront->next;

while (current) {

outFile << current->boxType << endl;

outFile << current->bBox->minR << " " << current->bBox->minC << " ";

outFile << current->bBox->maxR << " " << current->bBox->maxC << endl << endl;

current = current->next;

}

}

};

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

ifstream inFile;

ofstream outFile;

int thrVal;

if (argc < 3) {

cout << "ERROR: missing argument(s).\n<input file> <threshold value> <output file>";

exit(1);

}

inFile.open(argv[1]);

if (inFile.fail()) {

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

exit(1);

}

thrVal = atoi(argv[2]);

outFile.open(argv[3]);

ImagePP imagePP(inFile, thrVal);

inFile.close();

imagePP.computePP(outFile);

imagePP.threshold(outFile);

imagePP.morphClosing(outFile);

imagePP.determineReadingDir(outFile);

imagePP.findLineBoxes(outFile);

outFile.close();

}

textLineDetection\_data1.txt

45 50 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0

0 0 1 0 1 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0

0 0 0 1 1 1 0 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 0 0 1 1 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 1 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 0

0 0 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0

0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0

0 0 1 0 1 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0

0 0 0 1 1 1 0 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 0 0 1 1 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0

0 1 1 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 0

0 0 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0

0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0

0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 0 0 0 0 1 0 1 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0

0 0 1 0 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0

0 0 0 1 1 1 0 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 0 1 1 1 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 1 1 1 0 0 0 1 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 0

0 0 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0

0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0

0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

textLineDetection\_data1\_output.txt

HPP

0 1 1 19 24 25 1 1 1 24 27 26 4 2 2 1 1 8 23 24 27 1 1 2 24 27 26 8 2 1 2 1 10 22 26 26 0 1 0 25 27 26 18 1 0

VPP

0 7 14 11 25 10 7 3 9 10 10 13 14 8 10 11 6 9 15 16 14 9 6 5 6 9 11 16 18 20 14 8 13 8 12 10 13 16 16 10 10 4 4 8 18 18 12 3 0 0

HPPBinary

0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0

VPPBinary

0 1 1 1 1 1 1 0 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 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0

HPPMorph

0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0

VPPMorph

0 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0

Reading Direction : horizontal

text-line bounding box

3

3 2 5 46

3

9 1 11 46

3

17 2 20 46

3

24 1 27 46

3

32 2 35 46

3

39 1 42 46

textLineDetection\_data2.txt

45 50 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 0 0 1 1 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0

0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0

0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 0 1 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 0 0 1 1 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0

0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0

0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0

0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 0 0 1 1 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0

0 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0

0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0

0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

textLineDetection\_data2\_output.txt

HPP

0 7 18 21 20 21 16 3 8 10 13 18 15 9 2 0 7 17 21 20 21 16 3 8 10 13 17 15 8 2 0 7 17 21 20 21 16 3 9 11 14 17 16 11 2

VPP

0 0 0 18 27 24 12 2 1 0 16 19 22 15 15 9 0 1 3 24 30 27 12 6 1 1 0 9 18 24 21 9 15 0 0 0 15 21 24 30 18 3 0 1 9 15 15 12 0 0

HPPBinary

0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0

VPPBinary

0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 0

HPPMorph

0 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0

VPPMorph

0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 0

Reading Direction: vertical

text-line bounding box

3

1 3 42 6

3

1 10 43 15

3

1 19 43 23

3

2 27 43 32

3

2 36 43 40

3

5 44 39 47