Project 4 (C++) : Implementation of the four basic Morphology Operations

You will be given two images:

- Morphology\_Img1.txt,

- Morphology\_Img2.txt

and two structuring elements:

- Morphology\_StrucElem1.txt

- Morphology\_StrucElem2.txt

\*\*\* Run your program four times using the following four tests:

test-1: Morphology\_Img1.txt and Morphology\_StrucElem1.txt

test-2: Morphology\_Img1.txt and Morphology\_StrucElem2.txt

test-3: Morphology\_Img2.txt and Morphology\_StrucElem1.txt

test-4: Morphology\_Img2.txt and Morphology\_StrucElem2.txt

Your hard copies include:

- cover sheet

- program source code

- print all output files for test-1

- print all output files for test-2

- print all output files for test-3

- print all output files for test-4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Language: C++

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Project points: 10 pts

Due Date: Soft copy: 2/25/2020 Tuesday before midnight

1 day late: -1 pt 2/26/2019 Wednesday before midnight

1 and 1/2 days late: -2 pts 2/27/2020 Thursday before NOON

-10 pts: after 2/27/2020

Due Date: All Hard copy: 2/27/2020 Thursday in class

-1 pt for late hard copy submission.

\*\*\*\* All projects without hard copy will receive 0 pts even you have submitted soft copy on time.

Hard copy includes:

* Cover sheet
* Source code

- All output files of data set 1

- All output files of data set 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

I. Inputs: There are two input files.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

a) imgFile (argv[1]): a txt file representing a binary image with header information.

b). structFile (argv[2]): a txt file representing a binary image of a structuring element with header and the origin of the structuring element. The format of the structuring element is as follows: 1st text line are the four integer of the header;

2nd text line is the position (w.r.t. index) of the origin of the structuring element then follows by the rows and column of the structuring element.

For example (2-D structuring element):

5 5 0 1 // 5 rows, 5 columns, min is 0, max is 1:

2 2 // origin is at row index 2 and column index 2.

0 0 1 0 0

0 0 1 0 0

1 1 1 1 1

0 0 1 0 0

0 0 1 0 0

\*\* Note: when a structure element contains zeros,

only those 1’s to be used in dilation and erosion!

Another example (2-D structuring element):

3 3 1 1 // 3 rows, 3 columns, min is 1, max is 1:

1 1 // origin is at row index 1 and column index 1.

1 1 1

1 1 1

1 1 1

Another example (1-D structuring element):

1 5 1 1 // 1 row, 5 columns, min is 1, max is 1:

0 2 // origin is at row index 0 and column index 2.

1 1 1 1 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

II. Outputs: (All of the following output files need to be included in your hard copies!)

- dilateOutFile (argv[3]): the result of dilation image with header, the same dimension as imgFile

- erodeOutFile (args[4]): the result of erosion image with header, the same dimension as imgFile

- closingOutFile (args[5]): the result of closing image with header, the same dimension as imgFile

- openingOutFile (args[6]): the result of opening image with header, the same dimension as imgFile

- prettyPrintFile (args[7]): pretty print which are stated in the algorithm steps

\*\*\* Note: When you run your program, please name your output files as given in the above.

\*\*\* NO HARD coded file names in the program, you will receive the score of 0 , if you hard code file name in this project!!!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

III. Data structure:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- numImgRows (int)

- numImgCols (int)

- imgMin (int)

- imgMax (int)

- numStructRows (int)

- numStructCols (int)

- structMin (int)

- structMax (int)

- rowOrigin (int)

- colOrigin (int)

- rowFrameSize (int) // set to numStructRows / 2

- colFrameSize (int) // set to numStructCols / 2

- extraRows (int) // set to rowFrameSize \* 2

- extraCols (int) // set to colFrameSize \* 2

- zeroFramedAry (int \*\*) // a 2D array of size (numImgRows + extraRows) by (numImgCols + extraCols)

// for the input image, needs to dynamically allocate at run time

- morphAry (int \*\*) // Same size as zeroFramedAry

- structAry (int \*\*) //a 2D array of size numStructRows by numStructCols,

// needs to dynamically allocate at run time

Methods:

- loadImg (imgFile, zeroFramedAry ) // On your own!

// load imgFile to zeroFramedAry, begins at (rowFrameSize, colFrameSize) and ends at ??

- loadstruct (structFile, structAry) // On your own!

// load structFile to structAry

- zero2DAry(…) // set a given 2D array to zero

- ComputeDilation (...) // see algorithm below

// process the entire ary

- ComputeErostion (...)// see algorithm below

// process the entire ary

- ComputeOpening (...) // on your own

// process the entire ary

- ComputeClosing (...) // on your own

// process the entire ary

- dilation (i,j) // if zeroFramedAry (i,j) > 0, apply dilation on zeroFramedAry (i,j) with structAry.

// on your own

- erosion (i,j) // if zeroFramedAry (i,j) > 0, apply erosion on zeroFramedAry (i,j) with structAry

// on your own

- closing (i,j) // calling computeDilation then follows by calling ComputeErosion

// on your own

- opening (i,j) // callint computeErosion then follows by calling ComputeDilation

// on your own

- AryToFile (Ary, outFile) // output the img header then the rows and cols of Ary to outFile

- prettyPrint (Ary, outFile) // Remark: use “Courier new” font and small font size to fit in the page.

// if Ary[i, j] == 0 output “ .” // a period

// else output Ary[i, j]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IV. Main(...)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 0: imgFile, structFile, dilateOutFile, erodeOutFile, openingOutFile, closingOutFile, prettyPrintFile 🡨 open

step 1: numImgRows, numImgCols, imgMin, imgMax 🡨 read from imgFile

numStructRows, numStructCols, structMin, structMax 🡨 read from structFile

rowOrigin, colOrigin 🡨 read from strucFile

step 2: zeroFramedAry, structAry, morphAry 🡨 dynamically allocate // see description in the above

step 3: zero2DAry(zeroFramedAry) // see description in the above

step 4: loadImg (imgFile, zeroFramedAry) // see description in the above

prettyPrint (zeroFramedAry, prettyPrintFile) // see description in the above

step 5: zero2DAry(structAry)

loadstruct (structFile, structAry) // see description in the above

prettyPrint (structAry, prettyPrintFile) // see description in the above

step 6: zero2DAry(morphAry)

ComputeDilation (…) // see algorithm below

AryToFile (morphAry, dilateOutFile) // see description in the above

prettyPrint (morphAry, prettyPrintFile)

step 7: zero2DAry(morphAry)

ComputeErosion (…) // see algorithm below

AryToFile (morphAry, erodeOutFile)

prettyPrint (morphAry, prettyPrintFile)

step 8: zero2DAry(morphAry)

ComputeOpening (…) // see algorithm below

AryToFile (morphAry, openingOutFile)

prettyPrint (morphAry, prettyPrintFile)

step 9: zero2DAry(morphAry)

ComputeClosing (…) // see algorithm below

AryToFile (morphAry, closingOutFile)

prettyPrint (morphAry, prettyPrintFile)

step 10: close all files

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

V. ComputeDilation (…)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// process dilation on each pixel in the entire zeroFramedAry

step 1: i 🡨 rowFrameSize

step 2: j 🡨 colFrameSize

step 3: if zeroFramedAry[i,j] > 0

dilation (i, j)

step 4: j++

step 5: repeat step 3 to step 4 while j < (numImgCols + colFrameSize)

step 6: i++

step 7: repeat step 2 to step 6 while i < (numImgRows + rowFrameSize)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

VI. ComputeErosion (…)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// process dilation on each pixel in the entire zeroFramedAry

step 1: i 🡨 rowFrameSize

step 2: j 🡨 colFrameSize

step 3: if zeroFramedAry[i,j] > 0

erosion (i, j)

step 4: j++

step 5: repeat step 3 to step 4 while j < (numImgCols + colFrameSize)

step 6: i++

step 7: repeat step 2 to step 6 while i < (numImgRows + rowFrameSize)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

VII. dilation (i, j)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

on your own

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

VIII. erosion (i, j)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

on your own