Project 8.1: K-Curvature edge detector

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

Language: C++

Due date: soft copy 7.1: 4/28/2018 Saturday before Midnight

Early submission: 4/25/2018 Wednesday before Midnight

Due date: hard copy 7.1: 5/1/2018 Tuesday in class

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

I. Inputs:

- Input1 (argv[1]) : A text file contains the boundary points of an object in an image. The format of the input is as follows:

#rows #cols minVal maxVal // image header

label // the label of the object

r1 c1

r2 c2

r3 c3

:

:

- Input2 (console): ask the user from condole for K, the length of neighborhood to be used in the K-curvature computation

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

II. Outputs:

- Output1 (argv[2]): The result of the K-curvature of the object boundary points. The format of this output file is as follows:

#rows #cols minVal maxVal // image header

label // the label of the object.

#pts // the number of boundary points

r1 c1 1 // not a corner

r2 c2 8 // a corner (use 8 for corner indicator for the K-curvature)

r3 c3 1 // not a corner

:

:

- Output2 (argv[3]): Pretty print (displaying)of the result of the K-curvature corner detection, as in an image, where corner points are printed as 8 and non-corner points are printed as 1.

- Output3 (argv[4]): for all debugging output

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

III. Data structure:

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

- An image class

- numRows (int)

- numCols (int)

- minVal (int)

- maxVal (int)

- constructor

- img (int\*\*) // a 2D array for display, initially set to 0

- plotPt2Img()

// put each point (x, y)’s corner value (1 or 8) at Img(x, y)

- prettyPrint (img) // print img, if pixel(i,j) == 0

// print blank, otherwise, print its value.

- A boundaryPt class

friend of kCurvature class

- x (int)

- y (int)

- curvature (double)

- localMax (int)

- corner (int)

- constructor

- A kCurvature class

- K (int) // Ask the user from console

- countPts (…) // reads and returns the count of the boundary points

// in the input file.

- numPts (int)

- PtAry (boundaryPt \*) // an 1D array of boundaryPt class,

// need to dynamically allocate.

// use mod function to compute the curvature for the beginning of

// the K points without extending the tail of the array,

// the array size is numPts.

- printPtAry() // print the content of the entire PtAry

- beginIndex // set to 0

- Q (int) // an index of the array, initially set to 0

- P (int) // an index of the array, initially set to K

- R (int) // an index of the array, initially set to 2\*K

- storePt (x, y, index) // the input (x, y) to boundPtAry[index]

- computeCurvature(Q,P,R) // taught in class

- computeLocalMaxima (PtAry)

// P(i) is a local maxima in its 1 X 5 neighborhood iff the curvature

// of p(i) is >= the curvatures of

// its linear neighbors: p(i-2), p(i-1), p(i+1), p(i+2)

- setCorner (PtAry) // a boundary point, p(i) is a corner (returns 8)

// if (a) p(i) is a local maxima

// AND (b) for all p(i-2), p(i-1), p(i+1), p(i+2)

// only p(i-1) or p(i+1) can be a local maxima,

// otherwise, p(i) is not a corner (return 1).

- constructor

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

III. Algorithm steps in main

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

step 0: - inFile 🡨 open input files

(numRows, numCols, minVal, maxVal, label) <- get from inFile

- numPts 🡨 countPts (inFile)

- close inFile

- inFile 🡨 open the input file the second time.

- K <-- get from the user from console

- index <-- 0

step 1: (x, y) <-- read from inFile

storePt (x, y, index)// store x, y to PtAry[index]

printPtAry()

step 2: index ++;

step 3: repeat step 1 and step 2 until the end of inFile

step 4: Q 🡨 0

P 🡨 K

R 🡨 2\* K

step 5: index <-- P

curvature <-- computeCurvature (Q, P, R)

store curvature to PtAry[index]

print Q, P, R, index, x, y, curvature of PtAry[index]to argv[4]

// for debugging to see the curvature.

step 6: Increment Q, P, R by 1 // each need to mod with numPts

step 7: repeat step 5 to step 6 until P == K-1

step 8: print the info (x, y, curvature) of the entire PtAry to argv[4]

step 9: computeLocalMaxima (PtAry) for all point in PtAry[index], index from 0 to numPts-1

step 10: setCorner (PtAry) do for all point in boundPtAry[index], index from 0 to numPts-1

step 11: output only (x, y, corner) of the entire PtAry to argv[2]

step 12: Img <-- create an image of size numRows by numCols

step 13: plotPt2Img()

// put each point (x, y)’s corner value (1 or 8) at Img(x, y)

step 14: prettyPrint (img) to argv[3]

step 15: close all files