Project 5 (in Java): Robert, Sobel and Gradient Edge Detectors

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

I. Language: Java

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

Points: 10 pts

Due Date: Soft copy: 3/10/2020 Tuesday before midnight

1 day late: -1 pt 3/11/2020 Wednesday before midnight

1 and 1/2 days late: -2 pts 3/12/2020 Thursday before NOON

-10 pts: after 3/12/2020 noon

Due Date: All Hard copy: 3/12/2020 Thursday in class

-1 pt for late hard copy submission after 3/12/2020.

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

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

I. Input (args[0]): a txt file representing a grey-scale image with header.

II. RobertEdgeOut: Use args[1]- result of the sum of two Robert's edge image.

SobelEdgeOut: Use args[2]- result of the sum of two Sobel edge images.

GradientEdgeOut: Use args[3]- result of the gradient image.

prettyOut: Use args[4] for the pretty printing all files.

deBugOut: Use args[5] for all debugging print.

\*\*\*\* Using your existing programs

(histogram and threshold and prettyPrint and Bi-Gaussian Fitting): to do

- For each of the three edge image files:

RobertEdgeOut, SobelEdgeOut, and GradientEdgeOut

- Compute the histogram of the file;

- Use your auto-threshold program to select

the "best" threshold value to do thresholding on the file

- print the overlay bi- Gaussian curves on histogram.

- prettyPrint the result of the threshold

- The hard copies include:

- cover sheet

- source code.

- for each edge image,

- print the Histogram

- print the overlay bi-Gaussian curves on histogram.

- pretty print the best threshold result

// So, you will print three sets of the above.

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

III. Data structure:

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

- An imageEdge Class

- numRows (int)

- numCols (int)

- minVal (int)

- maxVal (int)

- mirrorFramedAry (int \*\*) // a 2D array, need to dynamically allocate at run time

of size numRows + 2 by numCols + 2.

- maskRobertRightDiag[2][2] (int)// may be hard coded.

- maskRobertLeftDiag[2][2] (int)// may be hard coded.

- maskSobelRightDiag[3][3] (int)// may be hard coded.

- maskSobelLeftDiag[3][3] (int)// may be hard coded.

- RobertRightDiag (int \*\*)

// a 2D array, need to dynamically allocate at run time

of size numRows+2 by numCols+2.

- RobertLeftDiag (int \*\*)

// a 2D array, need to dynamically allocate at run time

of size numRows+2 by numCols+2.

- SobelRightDiag (int \*\*)

// a 2D array, need to dynamically allocate at run time

of size numRows+2 by numCols+2.

- SobelLeftDiag (int \*\*)

//a 2D array, need to dynamically allocate at run time

of size numRows+2 by numCols+2.

- GradiantEdge (int \*\*)

// a 2D array, need to dynamically allocate at run time

of size numRows+2 by numCols+2.

- edgeSum (int \*\*)

// a 2D array, need to dynamically allocate at run time

of size numRows+2 by numCols+2.

- methods:

- constructor(s) // need to dynamically allocate the all arrays

- loadImage // load input file to mirrorFramedAry begin at (1,1)

- mirrorFramed (...) // do as in your previous project

- (int) convoluteRobert (i,j, mask) // as taught in class

// 2x2 convolution on img[i,j] with a Robert's mask and

// returns the result respective Robert array[i,j].

- (int) convoluteSobel (i,j, mask) // as taught in class

// 3x3 convolution on img[i,j] with a Sobel's mask and

// returns the result respective Sobel array[i,j].

- (int) computeGradient(i,j) // as taught in class

// Compute the gradient on img[i,j] and

// returns the result to GradiantEdge array[i,j].

- addTwoArys (Ary1, Ary2, Ary3)

// Ary3[i, j] <-- Ary1[i,j] + Ary2[i,j] for all i, j

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

IV. main (...)

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

step 0: open the image and read the image header

dynamically allocate mirrorFramedAry and all the edge arrays

step 1: loadImage (mirrorFramedAry)

// load input file to mirrorFramedAry begin at (1,1)

step 2: mirrowFramed (mirrorFramedAry)

step 3: process the mirrorFramedAry, from left to right and top to bottom

begin at (1, 1) // process all pixels!!!

RobertRightDiag(i,j)🡨 abs(convoluteRobert (i,j, maskRobertRightDiag))

RobertLeftDiag(i,j) 🡨 abs (convoluteRobert (i,j, maskRobertLeftDiag))

SobelRightDiag(i,j) 🡨 abs(convoluteSobel (i,j, maskSobelRightDiag))

SobelLeftDiag(i,j)🡨 abs (convoluteSobel (i,j, maskSobelLeftDiag))

GradiantEdge(i,j) 🡨 computeGradient(i,j)

step 4: repeat step 3 until all pixels inside of the frame are processed.

step 5: addTwoArys (RobertRightDiag, RobertLeftDiag, edgeSum)

output RobertRightDiag to deBugOut file // with caption

output RobertLeftDiag to deBugOut file // with caption

output input image header to RobertEdgeOut file

output edgeSum to RobertEdgeOut file // begin at edgeSum[1][1]

step 6: addTwoArys (SobelRightDiag, SobelLeftDiag, edgeSum)

output SobelRightDiag to deBugOut file // with caption

output SobelLeftDiag to deBugOut file // with caption

output input image header to SobelEdgeOut file

output edgeSum to SobelEdgeOut file // begin at edgeSum[1][1]

Step 7: output input image header to GradiantEdgeOut file

output GradiantEdge to GradiantEdgeOut file //begin at GradiantEdge[1][1]

step 8: close all files