CSC442/542 Image Processing – Spring 2017
Programming Assignment #2: Neighborhood Processing

Write a program that allows the user to apply a variety of spatial-domain filters and neighborhood operators to an image. Offer a menu with the following options:

1. convert color image to gray scale*
2. display image histogram*
3. perform histogram-based contrast stretch on image*
4. perform histogram equalization on image*
5. perform binary thresholding on the image (prompt user for threshold)*
6. add Gaussian noise (prompt user for sigma value)*
7. add impulse noise (prompt user for probability)*

8. 3x3 smoothing filter:

1	2	1
2	4	2
1	2	1

9. 3x3 sharpening filter:

0	-1	0
-1	5	-1
0	-1	0

10. plus-shaped median filter:

0	1	0
1	1	1
0	1	0

11. out-of-range noise cleaning filter (prompt user for threshold)
12. *n* x *n* neighborhood mean filter (prompt user for *n*)
13. *n* x *n* neighborhood median filter (prompt user for *n*)
14. *n* x *n* neighborhood minimum filter (prompt user for *n*)
15. *n* x *n* neighborhood maximum filter (prompt user for *n*)
16. *n* x *n* neighborhood range filter (prompt user for *n*)
17. *n* x *n* neighborhood standard deviation filter (prompt user for *n*)
18. Sobel edge magnitude
19. Sobel edge direction
20. Kirsch edge magnitude: max of 8 rotations of

-3	-3	-3
-3	0	-3
5	5	5

21. Kirsch edge direction (±22.5°, using the same masks)
22. Laplacian edge operator (offset by 128 and clip)
23. embossing

* Use predefined LuaIP routines for these operations.

Apply the selected action to the image tab that currently has focus. In general, these operations should function correctly with either color or grayscale images. Edge detection results should be mapped to grayscale values.

Extra credit possibilities:

1. Gaussian smoothing filter (prompt user for σ)
2. Marr-Hildreth edge operator: Laplacian of Gaussian or Difference of Gaussians (prompt user for σ), followed by detection of zero crossings
3. Canny edge operator (will cover in lecture)
4. Implement filters using separability (significant speedup for large filter sizes)
5. Handle image borders by reflection (instead of ignoring them)

Implementation Notes

- Sobel edge direction values need to be scaled to the range 0-255 for display purposes, with intensities ranging from black at 0° to white at 360°. The *atan2* documentation indicates that this function returns values in the range $-\pi$ to $+\pi$ radians (-180° to +180°). As discussed in class, adding 2π to negative values $[-\pi,0)$ generates angles in the range 0 to 2π. This range can easily be rescaled to 0 to 255 for display purposes.
- The Kirsch edge magnitude operator in LuaIP takes the maximum of the 8 template operators, divides by 3, and then clips the result at 255. This combination of scaling and clipping produces an edge image that generally looks better (closer to gradient magnitude operator results) than just clipping (too bright) or just scaling (too dark).
- Do not use any predefined LuaIP filter routines in your program. You may use any other LuaIP routines, including *gaussianNoise()* and *impulseNoise().*

Program Submission

- You are encouraged to work in teams of two on this assignment.
- When you are finished writing, testing, and debugging your program, turn in your source code using the MCS website submit program. Splitting your source code into multiple files for separate compilation is highly recommended (i.e., do not submit one monolithic source file). Use zip (or tar) to package multiple files for submission.
- The MCS website submit program is accessed via the MCS Department Web page (http://www.mcs.sdsmt.edu), by selecting the list item on the left entitled "Submit it!". Usage is self-explanatory: enter your name, choose the instructor and click "Select Instructor", choose the appropriate course (CSC442 or CSC542), type in (or browse to) the filename you wish to submit, and click "Upload". Submissions will be date- and time-stamped. You must submit your program by the due date (**Tuesday March 14**) in order to receive credit for this assignment. Late programs will not be accepted for partial credit unless prior arrangements have been made with the instructor. If you have any problems with the submit program, report them to your instructor and submit your program by email instead.
- To receive full credit, your code must be readable, modular, and well-documented, as well as correct. Try to make it reasonably efficient in terms of both execution time and space utilization. Your program must compile successfully using Lua and ZeroBrane Studio, on both Linux and Windows. If your program does not run correctly, indicate why. This will make it easier to give you partial credit.
- You may use sample code given in class or found on the course website as a basis for your program, but be sure to give credit for any code that you did not write. (As a general rule, *always* credit the author of any code that you use.)