CSC 205 - SPRING 2016
2D GRAPHICS AND IMAGE PROCESSING
ASSIGNMENT 5
UNIVERSITY OF VICTORIA

Due: Tuesday, March 29th, 2016 by noon.

1 Assignment Overview

The assignment involves implementing three image processing programs to process 8-bit grayscale images. References to the Burger and Burge textbook in the specification below refer to Volumes 1 and 2 of *Principles of Digital Image Processing* by Burger and Burge.

You may implement your solution in any language that can be run (using non-proprietary software tools) in ECS 354, ECS 242 or ECS 266, but you are encouraged to write code which is not platform-dependent. A simple image processor template (the same one provided for Assignment 4) has been provided in C++, Java and Python. As with the previous assignments, you are not required to use any of the starter code in your final submission.

If you use a language which relies on platform-specific libraries, you may find that it is easier to complete the entire assignment on a lab machine, instead of working on it on your own machine, since it can be difficult to resolve library version conflicts in some cases. The sample code for C++ and Python has been tested on the ECS 354 machines and a recent Debian-based linux distribution. Talk to your instructor if you need help compiling or running the sample code on your own machine. The compile/build/run process for the starter code will be covered in the lectures and labs.

1.1 Morphological Filtering

Write a program which reads a grayscale image and, interpreting the grayscale pixels as binary pixels (by assuming that pixels with intensities less than 128 are black and pixels with intensities greater than 128 are write), applies the morphological *closing* operation to the image using the structuring element H given below.

$$H = \begin{array}{|c|c|c|c|c|}\hline & & \bullet & & \\ \hline & \bullet & \bullet & \bullet & \\ \hline \bullet & \bullet & \blacksquare & \bullet & \bullet \\ \hline & \bullet & \bullet & \bullet & \\ \hline & & \bullet & & \\ \hline \end{array}$$

Recall that for binary images, black pixels are assumed to have the intensity value 1 and white pixels are assumed to have the intensity value 0, which is the opposite convention from that followed by 8-bit grayscale images. The most convenient way to accommodate this difference may be to convert the input image to a 0/1 binary image before processing, then convert back before saving the output image. The closing operation is defined by $I \bullet H = (I \oplus H) \ominus H$, where $\oplus$ denotes dilation and $\ominus$ denotes erosion.

1.2 Image Scaling

Write a program which reads an 8-bit grayscale image with resolution $w \times h$ and produces a scaled output image with dimensions $3w \times 3h$ using bilinear interpolation.

1.3 Non-linear Transformations

Section 10.1.6 of Volume 2 of Burger and Burge (pages 202-207) defines three non-linear transformations: *twirl*, *ripple* and *sphere* in terms of their inverse $T^{-1}(x', y')$ (which takes the coordinates (x', y') of a pixel in the output image and returns the corresponding (x, y) coordinates in the input image). Algorithm 10.1 in Volume 2 (page 211) describes the process of computing every pixel of an output image under such a transformation using an interpolation scheme to determine the intensity values of arbitrary input coordinates.

Choose one of the three transformations in Section 10.1.6 and implement an image processing program which applies the transformation to an input image, using bilinear interpolation as the interpolation scheme. Out-of-bounds pixels in the input image should be set to a constant intensity value of 128. Each of the transformations has various parameter values (for example, the radius of the spherical transformation); you are not required to use specific values for these parameters as long as the transformation is visible (the book gives sample values for each parameter which you can use as starting points).

If you implement two of the three transformations in Section 10.1.6, you will receive extra marks (see Section 3 below).

2 Externally-Sourced Code

You are permitted to use appropriately licensed code from an external source, if full attribution is given (including the name of the original author, the source from which the code was obtained and an indication of the terms of the license). Note that using copyrighted material without an appropriate license (such as the GPL, LGPL, BSD License or MIT License) is not permitted. Short fragments of code found on public websites (such as StackOverflow) may be used without an explicit license (but with the usual attribution). If you embed externally sourced code in your own code files, add a citation in your code file and include a README file detailing the sources of all code used. If you use entire, unmodified source files from an external source, a README file is not necessary as long as the file in question contains complete authorship and licensing information. If you submit the work of others without proper attribution, it will be considered plagiarism. Additionally, for assignments in CSC 205, the following two caveats apply to externally sourced code.

- You will not receive any marks for the work of others. That is, if you use externally sourced code to implement the core objectives of the assignment, you will only be marked on the parts of the code that you personally wrote. Therefore, you should only use externally sourced code in a supplementary capacity (such as including a third-party hash table implementation which your code uses to store data, or a third party Gaussian elimination package to solve matrix equations). If you have any doubts about what is considered supplementary, contact your instructor.

- You may not use externally sourced code from another CSC 205 student, or share your code with another CSC 205 student (whether they intend to use it or not), without explicit permission from the instructor.

3 Evaluation

Submit all of your code electronically through the Assignments tab on conneX. Your submission should include a complete set of files needed to compile, run and demonstrate your implementation's functionality for marking. Your implementation will be marked out of 80 during an interactive demo with an instructor. You may be expected to explain some aspects of your code to the evaluator. Demos must be scheduled in advance (through an electronic system available on conneX). If you do not schedule a demo time, or if you do not attend your scheduled demo, you will receive a mark of zero.

You may receive up to 90 marks on this assignment, but the final assignment mark will be taken out of 80 (so any marks you receive beyond 80 will be treated as bonus).

The marks are distributed among the components of the assignment as follows.

Marks	**Feature**
25	The morphological filtering program functions correctly on a variety of test images.
25	The image scaling program (with a fixed scale factor of 3) functions correctly on a variety of test images
5	The image scaling program allows the user to specify the size of the output image (which may be smaller than the input image size).
25	The non-linear transformation program functions correctly on a variety of test images.
10	A second non-linear transformation (in a separate program) has been implemented and functions correctly on a variety of test images.

Additionally, you may receive extra marks (with the total overall mark capped at 90) for other features of your own design, if you have received permission from the instructor before the due date.

Ensure that all code files needed to compile and run your code in an ECS lab are submitted. Only the files that you submit through conneX will be marked. The best way to make sure your submission is correct is to download it from conneX after submitting and test it. You are not permitted to revise your submission after the due date, and late submissions will not be accepted, so you should ensure that you have submitted the correct version of your code before the due date. conneX will allow you to change your submission before the due date if you notice a mistake. After submitting your assignment, conneX will automatically send you a confirmation email. If you do not receive such an email, your submission was not received. If you have problems with the submission process, send an email to the instructor **before** the due date.