APCS Lab Assignment Sheet

APCS :: Lab 12 - Image Manipulation

Description:
In this Lab you will make a GUI application that modifies a grayscale image in a number of different ways. NOTE: if you load a color image, and just treat it like a grayscale image, it will still work; it will just turn the color image into a grayscale one.

Resources:
Most of the things asked of you in this lab are covered in the Squint textbook on pages: 299-303, 307-311. If you’re stuck, read that.

You will also find the GuiManager and SImage API documentation online quite helpful.

Starting Files:
I have given you some code to start with. It provides a template for how to setup the GUI. If you follow the pattern, you should be fine. The comments in the code should be helpful.

A new bit of code is in there involving a java component called JFileChooser. It’s a built-in component that GREATLY simplifies the process of having the user choose a file from the system.

I’ve also used a component called a JScrollPane so that if you choose to enlarge the image, the image will stay inside a scrollable panel rather than having the window frame grow without bound. We’ll talk more about JScrollPane in the future.

Ultimately, the focus of this assignment is on manipulating 2D arrays more than it is about producing a spiffy GUI (though you can spiff it up if you want to).
Minimum Requirements:

Your program **must have** buttons that do the following to any grayscale image loaded by the user:

0. Negative (done in class)

1. Flip horizontal =>

2. Flip vertical =>

3. Rotate right (or left) =>

And at least one of:

0. **Blur**
   Compose a new 2D array of pixels where each cell (x,y) in the new array is the average of the cells around (x,y) in the original.

1. **Reduce** (by half)
   Compose a new 2D array of size (N/2) x (M/2) where NxM is the size of the original image. The average of the values in each 2x2 block in the original becomes the value for one cell in the reduced version.

2. **Enlarge** (2x)
   Compose a new 2D array of size (N*2) x (M*2) where NxM is the size of the original image. Each cell in the original image should become a 2x2 block in the enlarged version.

3. **Find Edges** (with fixed tolerance)
   Compose a new 2D array of the same size as the original, where all cells are black except for "edge lines" from the original, which should be white. If the difference in values between any neighboring cells is high, it's likely that that is an “edge” in the image. For the John Dewey image, try using a difference of 15 for starters.

4. **Superimpose**
   Create a button that allows the user to choose a different image file, which you then superimpose on the original. You'll compose a new 2D array where for every (x,y) that the two loaded images share, the cell (x,y) in the output will be the average of the values in (x,y) from the two loaded images. The dimensions
of the output image should be the same as the original regardless of the dimensions of the superimposed image.

Advanced Options:

0. Of course, you may invent your own kind of filter for the image. Go wild. If you do create your own just be sure to clearly document what it’s supposed to do.

1. If you write all of the the filters above (or more) your GUI will become cluttered with buttons. Clean it up, maybe by better organizing the buttons into panels, or by moving the commands into one or more JComboBoxes instead.

2. Each of blur, reduce, enlarge, find edges and superimpose could accept user-supplied values for the “amount” of each one to apply. For example, the user may wish to reduce the image by 20%, or 10%, instead of by half every time, or the user may wish to supply a higher threshold value for finding edges. Design a way to do this. You may want to experiment with a component called JSlider. GUIManager has an event handler for JSlider called sliderChanged. Check it out.

3. Program design. Your program probably has A LOT of code in the buttonClicked method inside your GUIManager. Similar to what we did for Carnival Game, break out the image manipulation code into its own class called, say, ImageManipulator. This class will maintain the state of the current image, allowing the GUI to simply call methods and get back a new SImage to display.