Assignment 4: Image Processing (Part 1)

Due Thursday by 8:59pm **Points** 0

IME: Image Manipulation and Enhancement

Starter files: **code**

1 Introduction

In this series of assignments, you will build an image processing application. You will create text-based and interactive GUI-based user interfaces for this program. Although you will implement a few simple and not-so-simple image processing effects, your system will have "good bones" through design that will streamline adding more image manipulations.

1.1 Images

Whatever the file format (jpg, png, bmp, etc.) an image is basically a sequence of pixels. Each pixel has a position in the image (row, column) and a color. For a greyscale image there is only one value per pixel. On a scale of 0-1, 0 indicates black and 1 indicates white. Values in between indicate shades of grey. This value is traditionally represented using an integer. The number of bits used to store this value dictates how many "levels of grey" are supported. For example, an 8-bit representation creates 256 distinct levels (including black and white).

1.1.1 Color Representation

For a color image, the pixel's color is represented by breaking it into individual components (usually 3) in some way. The most common representation is the red-green-blue (RGB) model. In this model, a color is represented by three numbers (components): red, green, blue. Any color is a combination of these three *base* colors. On a scale of 0-1, black is represented as (0,0,0), white as (1,1,1) and bright, pure yellow is represented as (1,1,0). There are several ways of measuring "brightness" or "intensity":

- Value: the maximum value of the three components for each pixel
- Intensity: the average of the three components for each pixel
- Luma: the weighted sum $0.2126r + 0.7152g + 0.0722b$

The three components can also be represented using integers. Each component is also called a "channel". If 8 bits are used per channel, this creates the (traditionally-termed) 24-bit image! Some formats also support transparency (e.g. the PNG format): this is done by adding a fourth component.

To see how various colors can be made using these base colors: open Intellij -> Type "Ctrl+Shift+A" on Windows or "Command+Shift+A" on Mac and type "show Color Picker". When selected, you can pick

different colors and observe their red, green and blue values to get an intuition for how this works).

1.2 Image Processing

The ubiquity of images is matched by the plethora of image manipulation programs and services. All phones with a camera also include apps that allow us to brighten, darken, blur or crop photos. Instagram has "filters" that convert a picture into something more interesting. Programs on traditional computers range from very simple photo manipulators to sophisticated Photoshop-like applications. The technical field of image processing itself is deeply rooted in math and signal processing theory. But the basics of image representation and manipulation are quite simple to understand. All image processing applications boil down to manipulating individual pixel colors. Some operations require computing properties or statistics on an image. For example, to increase the contrast in an image, one has to ensure that the intensities of pixels span a wider range (and hence can be thought of as an operation on the histogram of an image).

1.3 Sample Image Application

If you have not worked with image processing applications before, we recommend Gimp(http://www.gimp.org). Gimp is a free, open-source powerful image processing application. If you have access to Photoshop then that will be enough as well. We encourage you to find and try out the image processing operations that we will introduce in these assignments using these applications, to understand them better.

2 A sampling of image processing operations

2.1 Example Image Processing: Visualizing components

One of the easiest ways to analyze an image is to inspect the components (channels) of its pixels. This provides insight into the overall color balance of the image, or which components seem to store more information than others. This information can be used to compress the image better (by discarding data from or using fewer bits for its less-important channels) or make image processing faster (by only working on some channels).

The easiest way to visualize the channel of an image is to create a greyscale image, where the red, green and blue components of a pixel are equal to the specific channel value of that pixel in the original image. For example, if a pixel in the original image has the color (120,234,23), then the corresponding pixel to visualize the red component would have the color (120,120,120) and so on. In this way, one can create images that show individual components of an image (red, green, blue).

In the starter code, you will find several examples of greyscale images obtained by visualizing individual components, or value/intensity/luma.

2.2 Example Image Processing: Image Flipping

Another relatively straightforward operation is to flip the image horizontally or vertically. This does not change the size of the image. Flipping can be combined too (horizontal followed by vertical flip, or vice versa). In the starter code, you will find some examples of flipped images.

2.3 Example Image Processing: Brightening or darkening an image

Brightening or darkening an image corresponds to doing arithmetic on its colors. This operation is implemented by simply adding a positive constant to the red, green and blue components of all pixels and clamping to the maximum possible value (i.e 255 for an 8-bit representation) if needed. Similarly darkening corresponds to subtracting a positive constant from the values of all pixels and clamping to the minimum value (e.g. 0) if needed.

2.4 Example Image Processing: Converting to greyscale

There are many ways to convert a color image to greyscale, depending on what the goal is. Visualizing individual channels as above creates greyscale images. One could also use the value, intensity or luma (as defined in the RGB model above).

3 What to do

For this assignment, you must create a complete, fully-functional, tested program that has the following features:

- Load an image from an ASCII PPM file (see below).
- Create images that visualize individual R,G,B components of an image.
- Create images that visualize the value, intensity or luma of an image as defined above.
- Flip an image horizontally or vertically.
- Brighten or darken an image.
- Save an image to an ASCII PPM file (see below).
- Allow a user to interact with your program to use these operations, using simple text-based scripting (see below).

3.1 The PPM image format

The PPM format is a simple, text-based file format to store images. It basically contains a dump of the red, green and blue values of each pixel, row-wise.

Some code has been provided to you to read a PPM file. You may use and manipulate this code in any way you see fit.

3.2 Text-based scripting

Your program should support loading, manipulating and saving images using simple text-based commands. Here is a list of example test commands you should support. The syntax for your commands may be different than this: this is just an example:

1. load image-path image-name: Load an image from the specified path and refer it to henceforth in the program by the given image name.
2. save image-path image-name: Save the image with the given name to the specified path which should include the name of the file.
3. red-component image-name dest-image-name: Create a greyscale image with the red-component of the image with the given name, and refer to it henceforth in the program by the given destination name. Similar commands for green, blue, value, luma, intensity components should be supported.
4. horizontal-flip image-name dest-image-name: Flip an image horizontally to create a new image, referred to henceforth by the given destination name.
5. vertical-flip image-name dest-image-name: Flip an image vertically to create a new image, referred to henceforth by the given destination name.
6. brighten increment image-name dest-image-name: brighten the image by the given increment to create a new image, referred to henceforth by the given destination name. The increment may be positive (brightening) or negative (darkening)

The following shows a sample sequence of script commands using the above syntax. Support several ways of entering these commands into your program. The part of the line after the '#' symbol are comments shown only for illustration:

```
#load koala.ppm and call it 'koala'
load images/koala.ppm koala

#brighten koala by adding 10
brighten 10 koala koala-brighter

#flip koala vertically
vertical-flip koala koala-vertical

#flip the vertically flipped koala horizontally
horizontal-flip koala-vertical koala-vertical-horizontal

#create a greyscale using only the value component, as an image koala-greyscale
value-component koala koala-greyscale

#save koala-brighter
save images/koala-brighter.ppm koala-brighter

#save koala-greyscale
save images/koala-gs.ppm koala-greyscale

#overwrite koala with another file
load images/upper.ppm koala
```

Your program is expected to handle errors suitably (i.e. you cannot assume that each text command will be input accurately).

3.3 Important points to think about

- What does the model represent?
- What are possible ways to represent image data? Which ones seem more helpful for this application?
- Although you are implementing some specific operations in this assignment, you will likely implement others in subsequent assignments.
- Remember to think from not only the implementor's perspective (the person that is implementing a class) but also the client perspective (the people or classes that are using the class). What operations might they reasonably want to perform? What observations might they reasonably want to make?
- Remember not to fall into the trap of assuming that there is only one way to use your model. You do not know if there will be only one view, and what kind of user interaction will need to be supported.
- There may come a time in the semester when you will be asked to share your code with other students, or see others' code. So your design, code and documentation is not "for your eyes only".

4 High-level Design Principles

This assignment's open-endedness is strategic, to simulate a real-world situation where you may have a high-level idea of what is to be accomplished, but details are known progressively. This is what makes design challenging. As you make your design, keep the following in mind:

- Try to extrapolate from this assignment. What kinds of things may be coming? How can you make your design flexible to withstand changes?
- Your design may need to be enhanced as you add features. Adding a new feature without changing any of your design is ideal: prepare to achieve this goal as much as possible. But in cases where you cannot, strive towards isolating and minimizing changes. Editing an existing design is the worst option, because it can break other things.
- This series of assignments expects you to *combine* the design techniques you have learned in this and previous semesters. It would help to review what you have learned, and identify when they may be useful.

5 What to submit

A complete submission must contain:

- All your code in the src/ folder

- All your tests in the test/ folder
- At least one example PPM image (different from any of the ones provided to you in this assignment), its greyscaled, brighter, darker and flipped (one of each type) versions, named suitably in a res/ folder
- An image of a class diagram showing your design (excluding tests). The class diagram should show names of classes and interfaces, signature of methods and inheritance relationships. Do not show field names or "has-a" relationships because they may clutter the diagram. **We expect these to be not hand-scribbled).**
- A text README file explaining your design. Your README file should give the graders an overview of what the purposes are for every class, interface, etc. that you include in your submission, so that they can quickly get a high-level overview of your code. It does *not* replace the need for proper Javadoc!
- **A script of commands that your program will accept, that will load at least one file, run some operations on it and save results in other files. Include in your README file instructions on how we can run this script using your program (e.g. "provide this file as a command line argument", "type this script when the program runs", etc.).**
- A citation for the source of your image (in the README file). It is your responsibility to ensure that you are legally allowed to use any images, and your citation should be detailed enough for us to access the image and read its terms of usage. If an image is your own photograph or other original work, specify that you own it and are authorizing its use in the project. **Please do not use images provided or shown in this assignment as your examples.**

6 Grading standards

For this assignment, you will be graded on

- adherence to an MVC design
- the design of your interface(s), in terms of clarity, flexibility, and how plausibly it will support needed functionality;
- the appropriateness of your representation choices for the data, and the adequacy of any documented class invariants (please comment on why you chose the representation you did in the code);
- the forward thinking in your design, in terms of its flexibility, use of abstraction, etc.
- the correctness and style of your implementation, and
- the comprehensiveness and correctness of your test coverage.