Concurrency:
Multi-core Programming and Data Processing

Assignment 05

Professor: Pascal Felber
Assistant: Isabelly Rocha

May 7, 2020

Submission: Java code and a pdf file containing your answers.

An image is represented as a matrix m x n of numbers. Each element of the image is called pixel and has a value that indicates the color.

There is a great number of image processing algorithms and most of them can benefit from concurrency. The need for concurrency becomes even more evident when the same operation is applied multiple times over a stream of images. Image processing operations vary from single-pixel oriented algorithms (e.g., contrast adjustment), local operations on groups of pixels (e.g., noise reduction), to global operations on all pixels (e.g., coding, decoding).

This assignment consists of 2 individual parts, as follows.

1 Smoothing the edges

The goal of this exercise is to alter an image to "smooth the edges" and remove "hot pixels". The initial image is modified by "turning off" (set to 0) all "live" pixels (value != 0) that do not have at least d live neighbors. We consider neighbors of pixel p all pixels surrounding p. Thus, a corner pixel will have 3 neighbors, an edge pixel will have 5, all the others having 8. Starting with the newly formed image this time, the algorithm is repeated and all pixels that don't have at least d live neighbors will be turned off. The execution ends when there are no more differences between the new image and the old image.

Write a program that solves the above problem using the heartbeat paradigm. The heartbeat paradigm controls how the communication takes place between threads. It is explained in greater detail in the additional file *heartbeat.pdf*. Test your program on different images, number of threads and values for d. Describe which tests you ran and explain the results (run at least on one image, with 1, 2, 4 and 8 threads, and d=3, 5, 7).

2 Applying a filter

Write a program that applies a filter on an image of size m x n. You can choose any filter from the following:

- Blur
- Sharpen
- Edge enhancement
- Find edges
- Emboss

The recommended size for the filter is 3x3. Only one filter is required for maximum points on this exercise. Image processing has to be done in parallel and tested in a similar manner to exercise 1. More details on filters can be found in *filters.pdf*.

Tip: When applying a filter, you can either copy the same pixel values for the corner and the edges, or pad the matrix with additional black lines (to simulate that, for example, the corner pixels also have 8 neighbors like all the others).

Further instructions:

1. Data input format: the matrix that represents the image will be stored in a file read by a single thread. There are no constraints regarding the image file: you can use RGB for each pixel (that is, 3 bytes per pixel), but for simplicity it is highly recommended to use gray-scale pixels (i.e., with a single value, between 0 and 255). Moreover, in order to be able to physically see your results, it is recommended to store your image in a PGM file (that can be visualized with various graphics programs). Details about the format of PGM files can be found here. A PGM file can be easily parsed manually. Otherwise, there are a lot of sample implementations online (if you use any of them, just mention this in the pdf file).
2. In order to obtain a PGM test file, you can save any JPG image as PGM in an image editor (such as Photoshop, Inkscape, Gimp, etc.). It can also be opened in any text editor (see Notepad++) in order to check the format/data.