Homeworks 7 and 8

Tom Kelliher, CS 325

60 points, due May 4, 2012 at 12:00 pm

1 The Assignment

For these assignments, you will write four image processing filters in OpenMP.

All programs read their image inputs from stdin and write their image outputs to stdout. Diagnostic output should be written to stderr. All programs should be written as OpenMP programs, utilizing as much parallelism as possible.

1.1 Grayscale Filter

This filter takes a color (P3) image and produces a grayscale (P2) image corresponding to the color image. It is an error for the input to be anything other than a P3 image. The program takes one command line argument, an integer in the range $[1, 3]$. The argument is interpreted as follows:

- 1: Use the lightness algorithm.
- 2: Use the average algorithm.
- 3: Use the luminosity algorithm.

The output image should have the same dimensions as the input image, and the output image should have the same maximum pixel component value as the input image.

1.2 Threshold Filter

This filter takes a grayscale (P2) image and produces a bitmap (P1) image corresponding to the grayscale image. It is an error for the input to be anything other than a P2 image. The program takes one command line argument, the threshold, which is an integer in the range $[0, 255]$. Any input pixel value less than the threshold value is clamped to black, which is the value 1 in the bitmap image. Any other input pixel value is clamped to white, which is the value 0 in the bitmap image. The output image should have the same dimensions as the input image.

1.3 Edge Detection Filter

This filter takes a grayscale (P2) image and produces a grayscale (P2) image corresponding to the input grayscale image. It is an error for the input to be anything other than a P2 image. The program takes one command line argument, an integer in the range $[1, 3]$. The argument is interpreted as follows:

- 1: Apply the Prewitt mask.
- 2: Apply the Sobel mask.

- 3: Apply the Laplace mask.

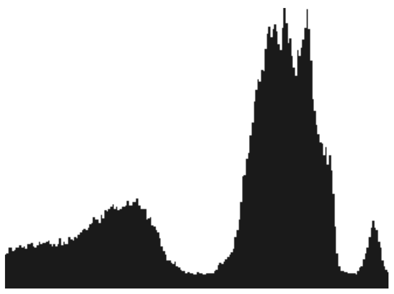
The output image should have the same dimensions as the input image, and the output image should have the same maximum pixel color component value as the input image.

1.4 Histogram Function

This filter takes a grayscale (P2) image and produces a bitmap (P1) histogram of the input image. The histogram itself should be black and the background should be white. It is an error for the input to be anything other than a P2 image. The program takes two command line arguments:

- Argument 1: The width, w, in pixels, of the histogram image.
- Argument 2: The height, h, in pixels, of the histogram image.

If the maximum pixel component value is n, the histogram should have columns corresponding to the pixel component values 0 to n. The tallest column of the histogram should extend the entire height of the histogram image. The heights of the remaining columns should be scaled accordingly. The widths of the columns should be scaled so that the histogram is w pixels wide. (For example, if n is 255 and w is 1024, then each column should be four pixels wide.) Here's the general idea:



There's an extension that can be made for an extra credit opportunity. See me if you're interested.

2 Documentation Requirements

Minimally, your program must contain your name and an overview at its top similar to this:

```
/*********************************************************************
 * Tom Kelliher, Goucher College.
 *
 * mandelbrot.c --- Serial program to generate a PPM image
 *                  representation of the Mandelbrot set for some
 *                  rectangular portion of the complex plane.
 *
 * Due to the use of sqrtf(), the math library needs to be compiled-in:
 *
 *    gcc -lm -o mandelbrot mandelbrot.c
 *
 * The PPM image is written to stdout.  It will be HUGE, so it is
 * advisable to pipe the output to pnmtojpeg (on a Linux system)
 * and redirect the filter's output to a file:
 *
 *    ./mandelbrot | pnmtojpeg > image.jpg
 *
 * The pack()/unpack() routines, unnecessary in a serial environment,
 * _might_ be useful in a message passing environment, but that is a
 * claim that ought to be tested.
 *********************************************************************/
```

If the program you submit for a grade does not work, I will need further documentation within the program itself in order to assign partial credit. (You risk receiving **no** partial credit if you do not provide sufficient documentation.) One component of this documentation should be an analysis, to the best of your ability, of why the program is not working.

3 Compiling and Running Your Program

To have the compiler recognize the OpenMP directives, use the `-fopenmp` switch:

```
gcc -fopenmp -o togs togs.c
```

To compile you program for serial execution, include the GNU OpenMP library:

```
gcc -lgomp -o togs togs.c
```

Use the following pipeline to convert common image formats to PPM images:

```
giftopnm < foo.gif | pnmtopnm -plain > foo.ppm
jpegtopnm < foo.jpg | pnmtopnm -plain > foo.ppm
pngtopnm < foo.png | pnmtopnm -plain > foo.ppm
```

Seeking Assistance

I strongly encourage you to begin this assignment as soon as it is issued and to come to my office if you need assistance. If you email me to describe a problem, describe the problem carefully and attach your source code. If extensive debugging will be required to determine the cause of the problem, I will let you know that you will need to visit me in person to resolve the problem. (In other words, I will not debug your program for you.)

Submitting Your Assignment

By noon on the 4th, send me your source code. You should anticipate that I will test your program with several different image files. Debugging messages in your source code should be commented-out. Late work will **not** be accepted, except for College-recognized reasons.