Assignment 1: Simple High Dynamic Range (HDR)

Due date: *Thursday, October 29, 2020 at 22:00*

> *This is an individual assignment. You may discuss it with others, but your code and documentation must be written on your own.*

High Dynamic Range (HDR) is a technique used by photographers and camera software to obtain digital photographs with more details and sharp and balanced colors. Often digital photographs have spots that are too bright (pure white) or too dark (pure black), perhaps because the photo was overexposed or underexposed. The idea of HDR is to capture multiple photos with different exposures in a relatively short period of time and then combine them in such a way as to preserve the maximum details possible.

You must write a C program, in a single source file called `hdr.c`, that merges two images with different exposures. The rest of this document provides you with the technical details and specifications necessary to write your code.

PPM Picture Format The program must read and write images in *portable pixmap format (PPM)* specified as follows. The first three lines of the image file are formatted as ASCII text. These three lines contain, respectively: (1) a "magic number" that identifies the PPM format, specifically the two-character string "P6"; (2) two positive integers separated by a space indicating the width and height of the image; (3) a positive integer indicating the maximum levels of colors for each of three color channels (red, green, blue).

For the purpose of this assignment, you must support only 255 color levels, so that each pixel of the image consists of three consecutive *bytes* representing the red, green, and blue color values for that pixel. If the color level is not 255, your program must reject the input files and exit with an error message (see below).

After the first three lines, meaning immediately after the newline character immediately following the color levels, the file contains $3 \times width \times height$ bytes, where three consecutive bytes represent a single pixel, and the $width \times height$ pixels in the image are listed from top-left to bottom-right, row-by-row. Below is an example:

`P6\n`
`1024 768\n`
`255\n`
$r_{1,1}\ g_{1,1}\ b_{1,1}\ r_{2,1}\ g_{2,1}\ b_{2,1}\ \ldots r_{1024,1}\ g_{1024,1}\ b_{1024,1}$
$r_{1,2}\ g_{1,2}\ b_{1,2}\ r_{2,2}\ g_{2,2}\ b_{2,2}\ \ldots r_{1024,2}\ g_{1024,2}\ b_{1024,2}$
$\ldots r_{1,768}\ g_{1,768}\ b_{1,768}\ \ldots r_{1024,768}\ g_{1024,768}\ b_{1024,768}$
	First line: magic number, followed by newline.
	Second line: *width*, space, *height*, newline.
	Third line: color levelts, followed by newline.
	Red, green, and blue levels for each pixel.
	All $width \times height$ pixels are consecutive triples of bytes, $r_{x,y}\ g_{x,y}\ b_{x,y}$, without line breaks.

HDR Algorithm. We use a simple pixel-based HDR algorithm to merge two pictures. To determine the value of each pixel in the output picture, we calculate a weighted average of its two corresponding pixels in the given input pictures. The weights are obtained based on the luminosity of the original pictures. Having weights based on the luminosity of the original pictures increase the contrast and therefore results in a better looking picture.

Input: $r_1, g_1, b_1, r_2, g_2, b_2$ — pixel values from input images
Output: r, g, b — pixel values for output image

$$w_1 = |luminance(r_1, g_1, b_1) - 128|;$$
$$w_2 = |luminance(r_2, g_2, b_2) - 128|;$$
$$r = (w_1 r_1 + w_2 r_2)/(w_1 + w_2 + 1);$$
$$g = (w_1 g_1 + w_2 g_2)/(w_1 + w_2 + 1);$$
$$b = (w_1 b_1 + w_2 b_2)/(w_1 + w_2 + 1);$$

where
$luminance(r, g, b) = 0.2126r + 0.7152g + 0.0722b$

Program Usage. Your program must take the following command-line arguments.

`./hdr` *filename1 filename2*

or

`./hdr` *filename1 filename2 output-name*

The first two arguments, *filename1* and *filename2*, are the file names of the two input pictures. The third argument, *output-name* is optional and indicates the name of the output file. In case the output name is not specified, your program must output a file called "`hdr.ppm`" in the current directory.

Error Handling. If the user provides invalid arguments, in particular if the file does not exist, if it is not readable or is otherwise invalid, or if the "magic number" is not `P6`, or if the color level is not `255`, then the program must exit with a non-zero return code. Furthermore, if the dimensions of the two input pictures do not match, the program must also exit with a failure code and print an error message "`Invalid Dimension.`". If there are no errors, the program must exit with return code 0.

Submission Requirements and Instructions

To grade your implementation, we will run automated tests. To ensure you receive a full grade, your program must behave exactly as specified here. Below is a check-list of the most important requirements:

- *clean compilation:* your code must compile without warnings (`-Wall`);
- *usage and arguments:* your program must use the exact command-line arguments specified above;
- *output:* your program must generate an output image file exactly as specified above.

Submit one source file named `hdr.c`. Add comments to your code to explain sections of the code that might not be clear. You may also add comments at the beginning of the source file to refer to any and all external sources of information you may have used, including code, suggestions, and comments from other students. If your implementation has limitations and errors you are aware of and were unable to fix, list those as well in the initial comments.

You may use an integrated development environment (IDE) of your choice. However, *do not submit any IDE-specific file,* such as project description files. Also, *make absolutely sure that the file you submit can be compiled and tested with a simple invocation of the standard C compiler.*

Submit your solution through the iCorsi system.