CSE 273: Computational Photography, Spring 2023 – Assignment 1

Instructor: Ben Ochoa

Due: Wednesday, April 12, 2023, 11:59 PM

Instructions

- Review the academic integrity and collaboration policies on the course website.
- This assignment must be completed individually.
- This assignment may be completed in the programming language of your choice.
- You may use third party libraries/packages for basic linear algebra, basic image processing, and image file I/O. But, you may not use third party libraries/packages that directly solve the problem. If you are uncertain about using a specific library/package, then please ask the instructional staff whether or not it is allowable.
- You must prepare a report as a pdf file. The report must describe the problems, and your solutions and results. Math must be done in Markdown/LaTeX.
- Additionally, you must create a zip file containing all of your source code, along with an automated build method (e.g., a makefile) and a `readme` file with clear and concise directions on how to build and execute your program.
- The zip file must also contain any output image files.
- You must submit both files (.pdf and .zip) on Gradescope. You must mark each problem on Gradescope in the pdf.
- It is highly recommended that you begin working on this assignment early.

Problems

When completing these problems, refer to the color encoding information at
`https://www.color.org/chardata/rgb/rgb_registry.xalter`

1. (15 points) The file `macbeth.png` contains an RGB color image (8 bits per sample (i.e., bits per pixel per channel), nonlinear sRGB color encoding). Read this file directly into an 8 bit unsigned integer per sample, 3 channel image. Develop a function/method named `sRGBToLinear` that converts a nonlinear sRGB color encoded 8 or 16 bit unsigned integer (i.e., integers in [0, 255] or [0, 65535], respectively) per sample image to a linear RGB color space 32 bit floating-point (i.e., real numbers in [0, 1]) per sample image. Using `sRGBToLinear`, convert the nonlinear sRGB color encoded 8 bit unsigned integer per sample, 3 channel image to a linear RGB color space 32 bit floating-point per sample, 3 channel image. Note the linear RGB color space image has the same chromaticities as sRGB. Write the resulting linear RGB color space 32 bit floating-point per sample, 3 channel image directly to the OpenEXR file `macbeth.exr`. Note unless chromaticities are explicitly set when writing the OpenEXR file, sRGB chromaticities are implicitly set.

2. (10 points) Develop a function/method named `transformationMatrixRGBToRGB` that calculates the transformation matrix that maps linear RGB values in a color space with one set of chromaticities to linear RGB values in a color space with another set of chromaticities, given both sets of chromaticities. Using `transformationMatrixRGBToRGB`, calculate the transformation matrix that maps from sRGB chromaticities to ProPhoto (also referred to as ROMM) RGB chromaticities. Include the numerical values of the resulting transformation matrix in your report with sufficient precision such that it can be evaluated. Using the transformation matrix, transform the linear RGB color space 32 bit floating-point per sample, 3 channel image with sRGB chromaticities to a linear RGB color space 32 bit floating-point per sample, 3 channel image with ProPhoto RGB chromaticities. Write the resulting linear RGB color space 32 bit floating-point per sample, 3 channel image directly to the OpenEXR file `macbeth_ProPhotoRGB.exr`, where the ProPhoto RGB chromaticities are set explicitly when writing the file.

3. (10 points) Develop a function/method named `linearToProPhotoRGB` that converts a linear RGB color space 32 bit floating-point per sample image with ProPhoto RGB chromaticities to a nonlinear ProPhoto color encoded 8 or 16 bit unsigned integer per sample image. Using `linearToProPhotoRGB`, convert the linear RGB color space 32 bit floating-point per sample, 3 channel image with ProPhoto RGB chromaticities to a nonlinear ProPhoto RGB color encoded 16 bit unsigned integer per sample, 3 channel image. Intentionally create an *incorrectly* color encoded image by writing the nonlinear *ProPhoto RGB* color encoded 16 bit unsigned integer per sample, 3 channel image directly to the PNG file `macbeth_ProPhoto_incorrect.png`, where the *sRGB* chromaticities are explicitly set (or implicitly set, if the third party library/package used for PNG file output defaults to sRGB chromaticities, which most do).

4. (5 points) The input file and all of the output files above can be opened in GIMP. Do so and discuss any visual differences between the images with respect to any numerical differences in the sample values contained in the different files. Additionally, change how `macbeth_ProPhoto_incorrect.png` is displayed in GIMP by assigning it the ProPhoto color profile `ISO22028-2_ROMM-RGB.icc` (i.e., Image → Color Management → Assign Color Profile...) and discuss any visual differences between the result with respect to any numerical differences in the sample values contained in the different files. Note GIMP will not display `macbeth_ProPhotoRGB.exr` correctly, so do not include it in the discussion. (If running macOS, then the built-in viewer may display `macbeth_ProPhotoRGB.exr` correctly.)