Graphics and Imaging Lab

Imaging

Informática Gráfica

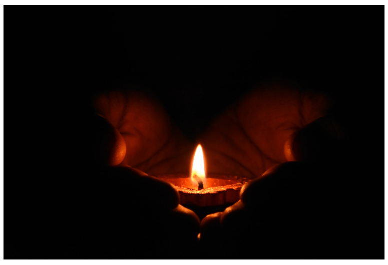
Grado en Ingeniería Informática

Goal

- Apply general knowledge about raster image and image formats.
- Understand the issues with the dynamic range of an image, and how and where to apply tone mapping techniques.
- Design and implement image loading and saving (that will be used for later assignments).

1 Problem statement

Images in nature typically have high dynamic range, with luminance values ranging from a small candle fire to a powerful burning sun. When using a physically based renderer, the resulting images can contain the same wide range of values as natural images.



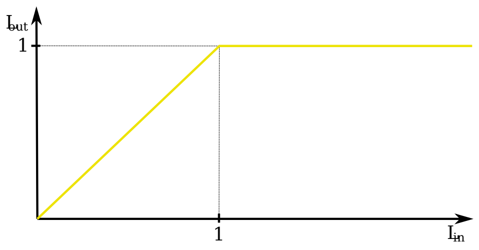
Such wide range of luminance values do not fit standard image formats nor the display ranges, resulting in the brigther areas of the image being overexposed (becoming all white) while darker zones are underexposed (fading to black). Therefore, there is a need for *tone mapping* the input image so it fits the conventional low dynamic range of the corresponding image format/display, making it possible to visualize all the elements of the image. This problem is rather important and this assignment treats it lightly, but there are widely acclaimed books on the topic [RWPD05].

2 Tone mapping

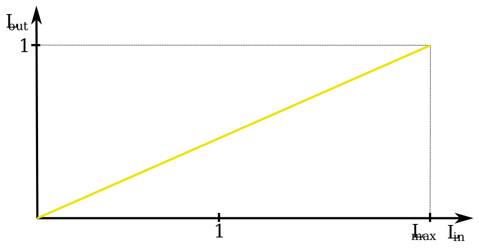
Your assignment is develop a set of tone mapping strategies as standalone applications which have as parameters the input HDR image, the output LDR image and the corresponding parameters of the tone mapping operator. Once loaded into the computer memory, a image will be stored as a 2D array of RGB tuples in floating point precision, to avoid losing information when opening and saving the image.

The tone mapping operators are simple:

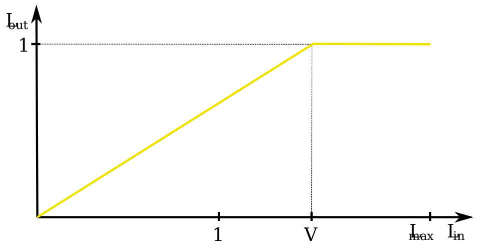
- **Clamping:** Discard all values greater than 255 (1 in floating point precision).



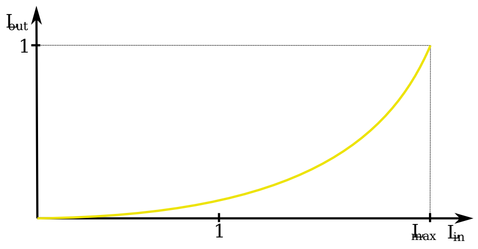
- **Equalization:** Linear transformation of values from minimum to the maximum (normalization).



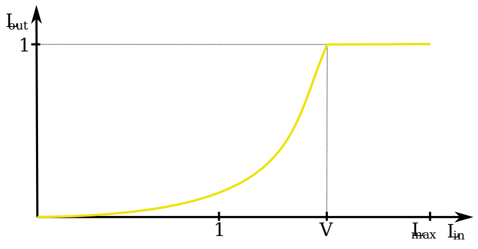
- **Equalize and clamp:** Combine both of the above according to a *clamping* parameter. Note that the previous two operators can be considered particular cases of this one.



- **Gamma curve:** Apply a gamma curve to all the values (you need to equalize first).



- **Clamp and gamma curve:** Apply a gamma curve after clamping the values (you need to equalize first). Note that all previous operators can be considered particular cases of this one.



3 Image format

We will use the plain `.ppm` image format for writing the output of the ray tracer into an image. It is a very simple text file that can be read by most viewers and is very simple to read and write. Find the details in `http://netpbm.sourceforge.net/doc/ppm.html`.

The idea is to use this image format both for storing HDR and LDR images. The standard image format does not support HDR images, but it can be *tweaked and bent* in order to do so. A standard image follows an structure such as this:

```
P3
# feep.ppm
4 4
15
 0  0  0    0  0  0    0  0  0   15  0 15
 0  0  0    0 15  7    0  0  0    0  0  0
 0  0  0    0  0  0    0 15  7    0  0  0
15  0 15    0  0  0    0  0  0    0  0  0
```

This includes the format identification P3, comments (lines starting with #) and resolution, both in terms of width and height (which in the case of the above image are both 4) and the resolution in color space, which translates into the number of potential color values. Then this is followed by all the ordered RGB tuples. When the RGB tuples start, no comments are allowed.

Given that in memory the images will be stored as floating point precision, for each stored value s (R, G or B of a RGB tuple) and given the color resolution c, the value to store when loading this format will be $\frac{s}{c}$ (so therefore it will range from 0 to 1). The standard only allows values $c \leq 65535$ as resolution in color space, but for storing HDR images we will lift such restriction. Generally speaking, when saving LDR images use $c = 255$ and when saving HDR images (when rendering in later assignment) use for instance $c = 2^{30}$ for a huge resolution in color space range.

Still, with this HDR format, the maximum of any image would be only 1. Instead we will enable an optional comment line such as `#MAX=18.35` that sets up the **real** maximum of the image as a real number. As a consequence, if c is the color resolution, s is the value stored in the file (R, G or B), m is the maximum value of the image and v is the real value stored in memory of the image then when loading (from disk to memory) you need to apply the following equation:

$$v = s\frac{m}{c} \tag{1}$$

and when saving (from memory to disk) you need to apply the inverse one:

$$s = v\frac{c}{m} \tag{2}$$

We provide a set of HDR-ppm images as supplemental material of this assignment in order to test loading and saving and the different operators.

4 Optional extensions

The students can choose one (or more) of the following extensions, that will enhance further evaluable assignments:

Other LDR image formats to save the image to. There are other LDR formats that are binary and are reasonably easy to save and load from without any external library. This includes, for instance, `bmp` format. Note, however, that many image editing applications can load standard `ppm` files.

Advanced tone mapping operators. Add other more advance tone mapping operators, including other global operators or even local ones. Any operator can be chosen, from the ones discussed during class [RSSF02, MDK08] or any other operator found in the literature.

5 Implementation details

Programming language. The students can choose any programming language they consider. However, we recommend C++ due to the following:

- The compiled code should be more efficient than other languages.
- The code that will be provided for the photon mapping work is in C++ (unless you choose to stick with your own code for both of them).
- C++ enables to define operators such as + or * for any new data type. This makes the code more readable for data types such as vectors.

External libraries. The use of any kind of external library or downloaded source code is forbidden. The students are only allowed to use the standard library that comes with the chosen programming language. The only exception to this rule is to include code that is your own previous work (maybe coming from previous assigments at previous computer science subjects). If you include code that is your own previous work, you are required to document it propperly (when did you generate it, for what assignment and what subject).

Submission

This assignment does not have to be submitted for grading. However, it is advised that it is finished by the recommended deadline in order to balance the workload.

References

[MDK08] RafałMantiuk, Scott Daly, and Louis Kerofsky. Display adaptive tone mapping. *ACM Trans. Graph.*, 27(3):68:1–68:10, August 2008.

[RSSF02] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. *ACM Trans. Graph.*, 21(3):267–276, July 2002.

[RWPD05] Erik Reinhard, Greg Ward, Sumanta Pattanaik, and Paul Debevec. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting (The Morgan Kaufmann Series in Computer Graphics).* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.