**Camera**

For a simplified model, we can assume that cameras intensity spectrum is linear, so the objects x times as bright in real life would be x times as bright to the camera sensor.

Using such a simplified model, we can assume that a camera has sensitivity floor sf and sensitivity ceiling sc. This means that everything under sf would not be registered on sensor and everything above sc would lead to overexposure (any additional brightness is cut off). Dynamic range of this camera would be:

Let’s assume, we have an image, where each pixel is represented as follows:

Then, to capture this pixel on camera we have to:

1. Shift the color values so that 0 of the color value would align with camera’s sensitivity floor.
2. Convert the color space so that the dynamic range of the camera would fit within the range of [0;1]
3. Trim values outside of the range [0;1]

So the conversion formula would be:

where

**Object intensity. Basics**

Each object can be represented as a color or an array of colors and intensity. It is useful, if we store these separately, because this allows us to:

1. Work with intensity-specific operations with more ease and effectiveness. When we need to work with a huge array of pixels, having multiple color channels increases our overhead, so working with intensity only saves us some time
2. For single-color objects, such as stars, we are allowed to store a single color value, yet lots of intensity values, allowing us to decrease the size of an object.

So, the final color channel would be:

**Color channel conversion. Basics.**

Let’s say, we have a pixel with its color values (represented as raw data) in source color channels:

Out target is to obtain a pixel with its values in target color channels:

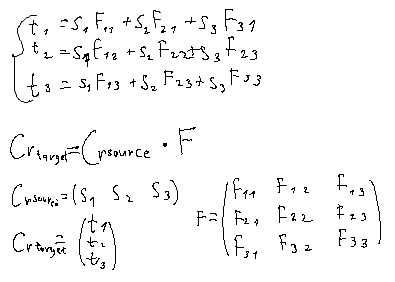
To achieve this we would need to apply next algorithm:

For each si in Crsource we need to obtain a color translation filter of si into a color space of Crtarget:

and multiply it by a value of si. Thus, we can obtain the target pixel as follows:

**Color channel conversion. Final transformation**

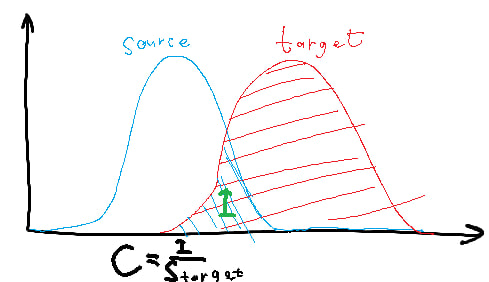
In the end, we can write the entire transformation as a matrix multiplication:



**Color channel conversion. Obtaining filter**

Let’s assume we have two spectrums: source spectrum and target spectrum. Each spectrum is represented as:

Each spi is an array where each element represents sensitivity towards the light at a certain wavelength.



The coefficient is obtained as an area of intersection divided by an area of target sensitivity.

Since our functions can’t go below zero, intersection can be represented as:

So out function can be rewritten as: