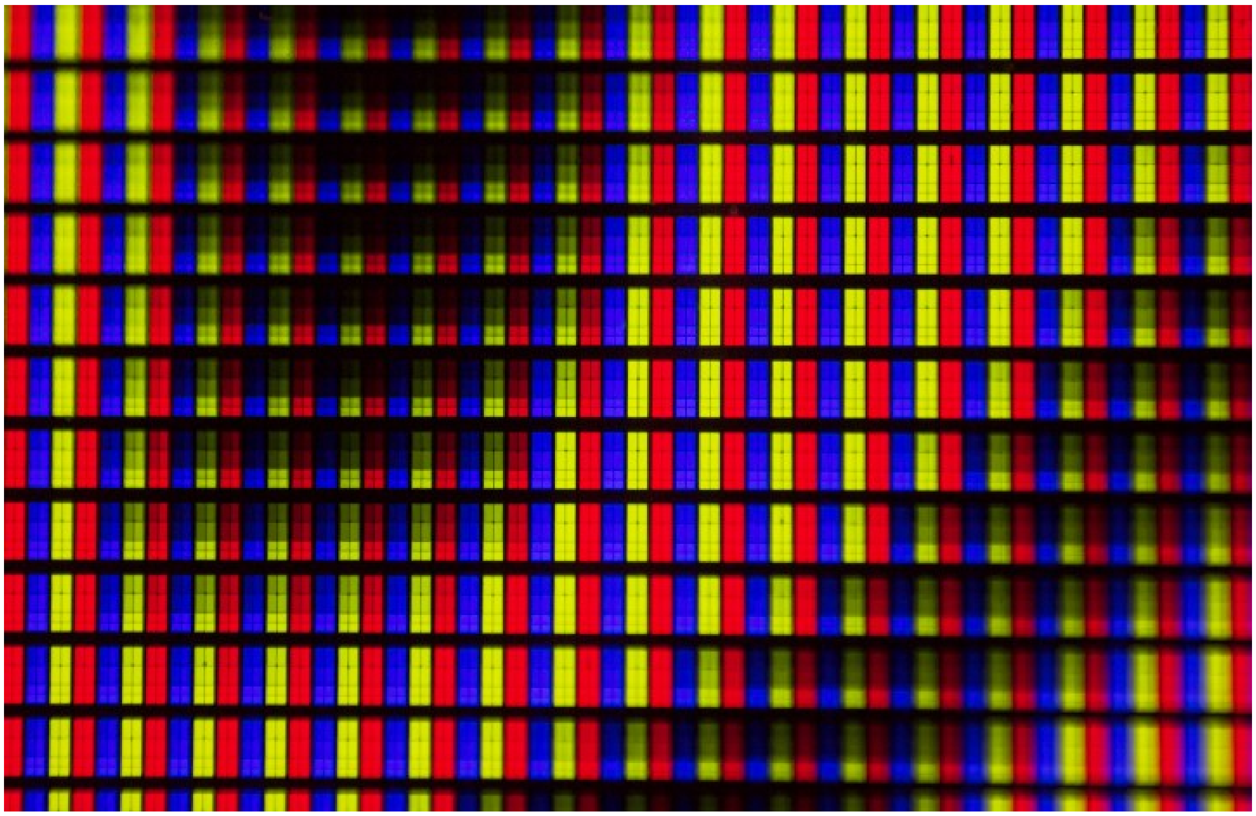
Image Processing and Pixel Manipulation: Photo Filters

We use filters all the time, from social media apps like Instagram and Snapchat to professional software like Photoshop. But have you ever wondered how filters work?

Pixels and Color Spaces



Images are stored digitally as an **array of pixels**. A pixel (short for picture element) is the smallest element of an image. Every pixel has a color and the way colors are defined is called the **color space**.

The most used color space is the **RGB color space**. In this model, every color is defined by three values, one for **red**, one for **green**, and one for **blue**. Usually, these values are 8-bit unsigned integers (a range of 0–255), this is called the **color depth**. The color **(0, 0, 0)** is **black**, **(0, 255, 0)** is **green**, **(127, 0, 127)** is **purple**.

Filters

Filters are mathematical functions that take the image as input and return a new image as output.

Filters can be applied on a pixel-level, that is, the output pixel at **(x, y)** depends only on the input pixel at **(x,y)**. Or on a global level, where the output of a certain pixel depends on multiple input pixels.

Filters can also be applied on a channel-level, that is, applied on every color channel (red, green, and blue) separately. For example, the red channel of the output depends only on the red channel the input.

We will cover the following filters:

- **Grayscale**
- **Brightness**
- **Contrast**
- **Saturation**
- **Gamma correction**
- **Blur**

Grayscale

Converting an image into grayscale is very straight forward. We need to convert three color values into a single color value. We can do this by taking their average (also known as the brightness).

$$\text{grayscale} = \frac{r + g + b}{3}$$

We will then put this value in red, green, and blue of the new image. Though, we have to make sure the new values are still integers.

Another way to do this is by taking a weighted sum of the red, green, and blue values.

$$\text{grayscale} = \alpha_1 r + \alpha_2 g + \alpha_3 b$$

where $\boldsymbol{\alpha_1}$, $\boldsymbol{\alpha_2}$, and $\boldsymbol{\alpha_3}$ are positive constants, such that $\boldsymbol{\alpha_1 + \alpha_2 + \alpha_3 = 1}$.

These values are usually chosen to be (YUV color encoding system):

$$\text{grayscale} = 0.299r + 0.587g + 0.114b$$

Brightness

The brightness of a pixel **μ** is a value between 0 and 255. defined as follows:

$$\mu = \frac{r+g+b}{3}$$

To increase **μ** by **Δμ**, we can increase every color by **Δμ**.

$$\frac{(r+\Delta\mu)+(g+\Delta\mu)+(b+\Delta\mu)}{3} = \frac{r+g+b+3\Delta\mu}{3} = \mu + \Delta\mu$$

But we have to make sure the values are still between 0 and 255.

The brightness increase **Δμ** is between -255 and 255, negative for darkening the image and positive for lightning the image.

Contrast

The contrast ratio of an image **C** is defined as follows:

$$C = \frac{\mu_{\max} - \mu_{\min}}{\mu_{\max} + \mu_{\min}}$$

A contrast value of **0** means that all pixels have the same brightness value. A contrast value of **1** means that the difference between the highest brightness and lowest brightness is 255 (maximum difference).

There are may ways we can use to increase or decrease the contrast ratio. One way is to spread brightness values away from 128 (middle of [0–255]), using the formula:

$$\mu' = \alpha(\mu - \mu*) + \mu*$$

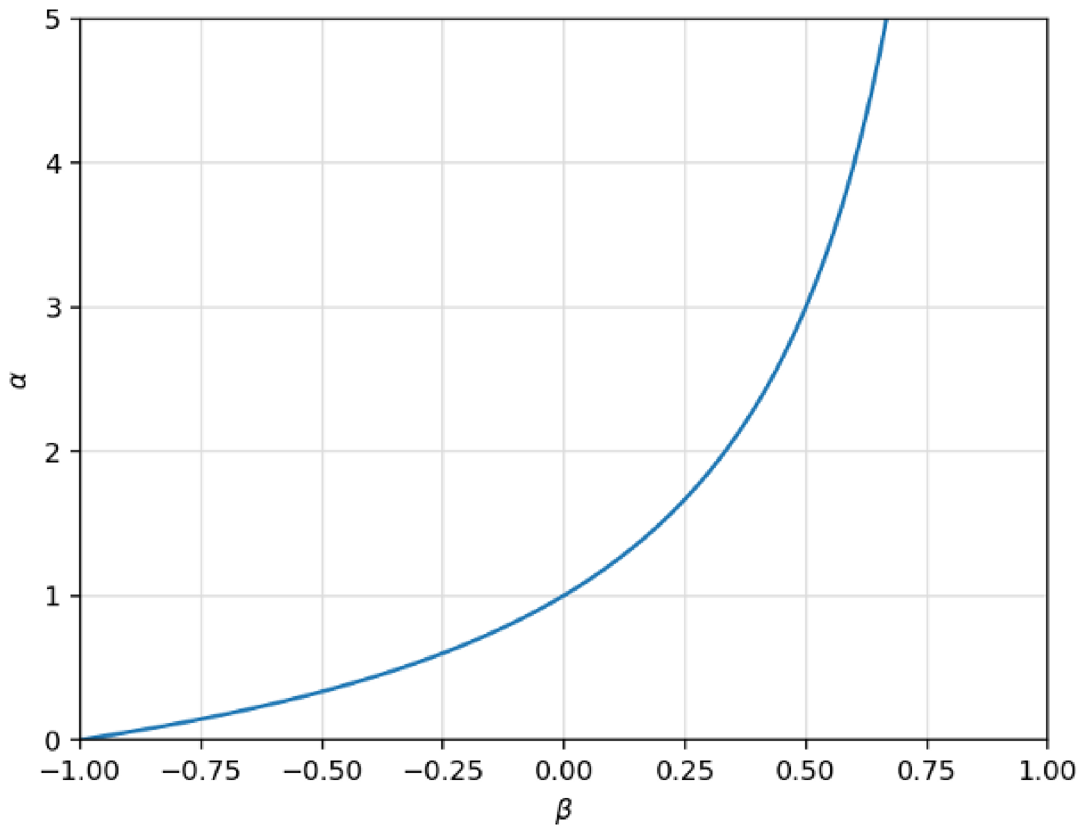
with **μ*** being the average brightness across the whole image. This formula spreads or scales the brightness values around **μ***, with a factor **α** greater than or equal to **0**.

A factor **α=1** doesn’t change anything, **α<1** decreases contrast, and **α>1** increases contrast.

It may be impractical for **α** to not have an upper bound. To overcome this we use a factor **β** between -1 and 1, such that:

$$\alpha = \frac{1+\beta}{1-\beta}$$

This means that when **β** changes from -1 to 1, **α** changes from 0 to infinity. And for **β=0**, **α=1**.



If we want it to be between -255 and 255, we can just replace **β** by **β/255**, this yields the following formula:

$$\alpha = \frac{255+\beta}{255-\beta}$$

We have to make sure we isolate the case where **β=255** since we can't divide by 0, in this case, we set **α=∞**.

Saturation

To change the saturation of a pixel, one may think about converting the **RGB** colors into **HSV/HSL** (in which **S** stands for saturation), change the saturation value, and then convert back to **RGB**. While this method is accurate, it is complicated and computationally expensive. I propose a simpler (less accurate) method.

Changing the saturation should not affect the brightness of the color. So if we spread the color values around their brightness (the mean of their **rgb** components, unlike the mean in contrast where it is taken across the whole image), that wouldn't change it.

Gamma Correction

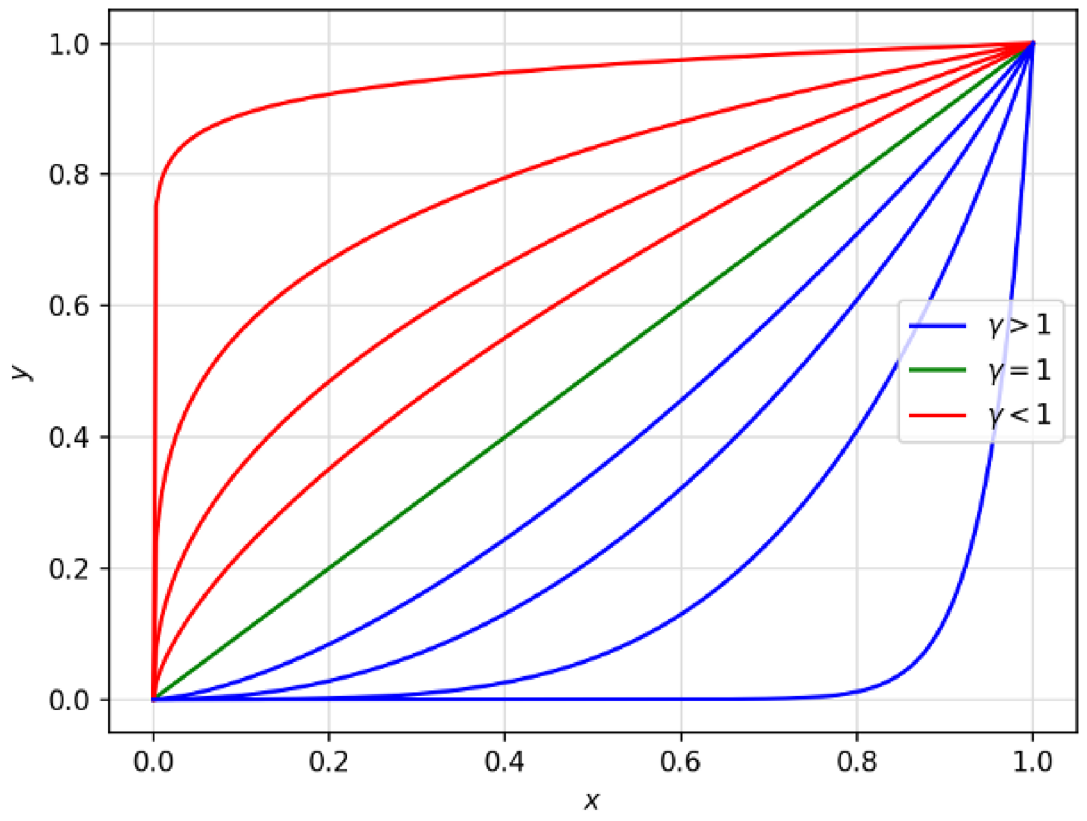
Humans don't perceive light linearly. We tend to have a higher sensitivity to dark colors than light ones. Gamma correction is set to allocate more bits to shadows than highlights by storing a gamma-encoded value instead of the original value. We won't go into details about how this saves space, but we will implement a gamma correction filter.

A gamma filter is the operation:

$$y = x^{\gamma}$$

where the exponent **γ** (gamma) is a positive real number. x is a number between 0 and 1.

For **γ>1** low values are squeezed and high values and stretched, and vice-versa for **γ<1.**



If we divide color values by 255, we get numbers between 0 and 1. Then after applying gamma, we can multiply by 255. Hence, applying a gamma filter is the operation:

$$x' = 255\left(\frac{x}{255}\right)^{\gamma}$$

Blur

There are many types of blur filters. Most common are **Box Blur** and **Gaussian Blur**. We will implement Box Blur.

Box Blur consists of taking the average color of pixels inside a square of a certain size (called **kernel size**) around a certain pixel as the color value of that pixel.

We can't draw a square around pixels close to the edges, so we will take the average of the pixels around it (that are in the image bounds).