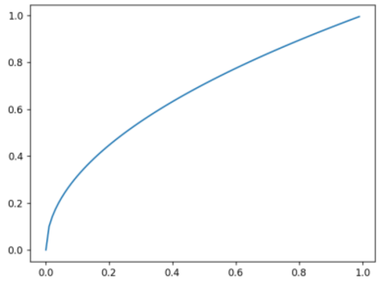
非线性变换

重要：本文假定图像在 0-1 范围内，图像最大值为 1，像素值设为 x

非线性变换，就是下面这个曲线，做一个映射，让亮的值稍微缓和一点，暗的值提升大一点：



一、基础：Gamma 和 Log 变换

图像想要如上图所示，一般有两种方法：Gamma 和 Log。

Gamma 老生常谈了，这也是最容易想到的，就是一个 x^{α} 函数，图像就不画了，反正就是上面的那张图样子。其中参数 α 是调节的系数。

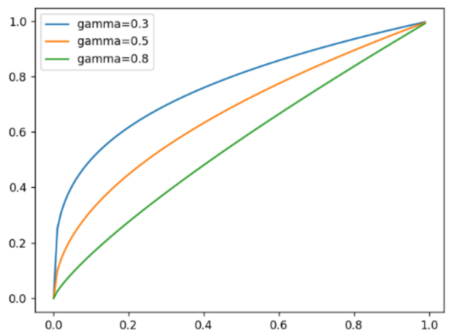
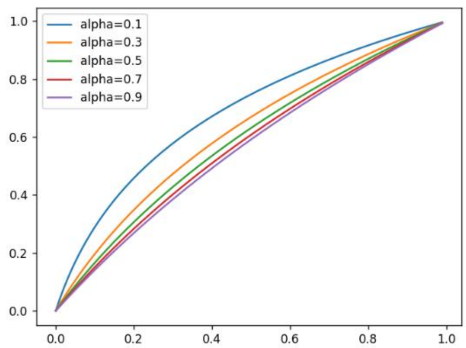
Log 变换也是上图所示，所以人们就想到是不是可以用 Log 做类似的事情，而由于像素是 0-1 之间，所以里面又加个 1，即 $\log(1+x)$。而为了最后归一化，所以还要除以一个 log2：

$$newx = \frac{log(1+x)}{log(2)}$$

看起来很好，然而仔细比较一下这个公式和上面的 Gamma：它怎么没有参数呀？也就是说它的**曲线形状已经是固定**了，我想曲线陡峭一点怎么办？所以实际 Log 变换需要加入调节参数：

$$newx = \frac{log(1+x/\alpha)}{log(1+1/\alpha)}$$

Gamma 变换和 Log 变换的图像如下所示，可以看到两个变换还是有所区别的，Gamma 在低值下急速上升，相比之下 Log 则保守一些，要根据实际情况选择使用：

Gamma	Log
	

二、Log 变换的另一种实现

在论文 *Adaptive Logarithmic Mapping for Displaying High Contrast Scenes* 中，作者提出了一种新的 Log 变换，看了好久才体会到，其总体公式如下：

$$L_d = \frac{L_{dmax} \cdot 0.01}{\log_{10}(L_{wmax}+1)} \cdot \frac{\log(L_w+1)}{\log\left(2+\left(\left(\frac{L_w}{L_{wmax}}\right)^{\frac{\log(b)}{\log(0.5)}}\right)\cdot 8\right)} \quad (4)$$

首先 $L_{dmax} * 0.01$ 直接忽略掉，它其实就是一个 gain，当成 1 即可，而这个那么乱的公式，可以化简为：

$$L_d = \frac{log_{base}(L_w+1)}{log_{10}(L_{wmax}+1)}$$
$$base = 2 + \left(\left(\frac{L_w}{L_{wmax}}\right)^{\frac{log(b)}{log(0.5)}}\right) \times 8$$

我们假定最大值是 1，然后公式里面的 $log(b)/log(0.5)$ 直接当成一个参数 γ，这样公式为：

$$L_d = \frac{log_{base}(L_w+1)}{log_{10}(2)}$$
$$base = 2 + 8 * (L_w)^{\gamma}$$

能看到分母是 log10 是因为现在的 base 最大为 10，现在有两个问题：

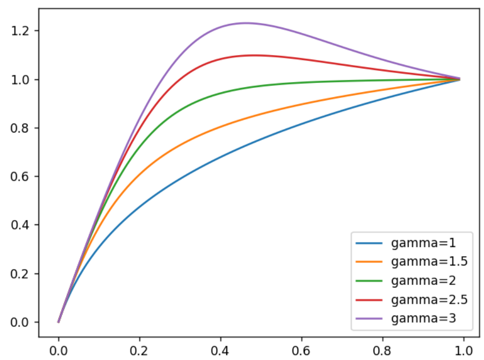
1. 为什么 base 中用 8 相乘，然后用 2 相加？
2. 这里的 γ 对函数曲线的影响？

第一个问题，为什么用 8 相乘，用 2 相加？为了下面的讲述，我们把乘数叫做 α，加数叫做 β，两者也决定了分母底数是 $\alpha+\beta$，作者是这样解释的：

> In the denominator decimal logarithm is used since the maximum luminance value in the scene is always re-sampled to decimal logarithm by the bias function.

没错，就是经验。根据经验，最大亮度一般用 log10 处理，所以 base 里面 $\alpha+\beta$ 要是 10。那么为什么偏偏一个是 8，一个是 2 呢？我推测这里还是经验。因为一般对数函数就三个最常见：log2，log10，ln。而 base 的取值范围是 $[\beta, \beta+\alpha]$，最大值是 10，那最小值就取 2 吧，所以 β 就是 2，α 就是 8。当然这里只是推测和经验，完全可以大胆的用别的数来处理。

第二个问题，γ 对函数曲线的影响？画图结果如下：



也是一种调整变换强度的参数。从图上也能看出来，要注意不能超过 2，否则变换函数不是单调函数了。论文里面是用 $\log(b)\ /\ \log(0.5)$ 来代替 γ 的，没有必要感觉。因此，最终这种公式下，有一个参数 γ 来调整变换强度，其曲线如上图所示。

三、其他非线性函数

$$I_n' = \frac{I_n^{(0.75z+0.25)} + (1 - I_n)0.4(1 - z) + I_n^{(2-z)}}{2}. \tag{3}$$

其中的参数 z 值由图像本身的内容决定，如下所示：

$$z = \begin{cases} 0 & \text{for } L \leqslant 50 \\ \dfrac{L-50}{100} & \text{for } 50 < L \leqslant 150 \\ 1 & \text{for } L > 150, \end{cases}$$

式中的 L 表示亮度图像的累计直方图（CDF）达到 0.1 时的色阶值，也就是说如果亮度图像中 90%的像素值都大于 150，则 Z=1，如果 10%或者更多的像素值都小于 50，则 Z 取值为 0，否则其他情况 Z 则根据 L 的值线性插值。如果 Z=0，说明图像中存在大量的偏暗像素，图像有必要变亮一些，如果 Z=1，则说明图像已经很亮了，则此时图像无需继续加亮处理。

函数的形状：

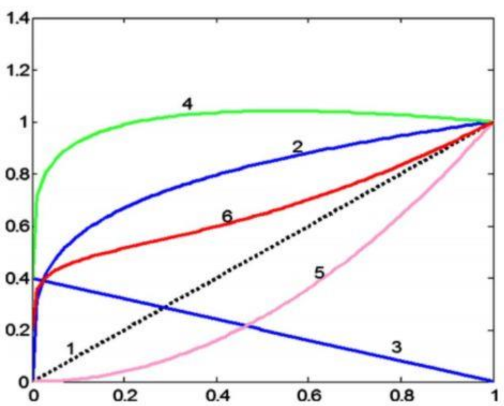


Fig. 2 Nonlinear transfer function (curve 6: when z=0) for luminance enhancement.

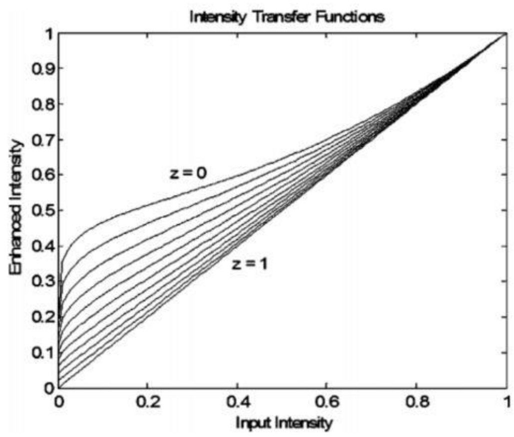


Fig. 3 Nonlinear transfer functions with different z values (z=0, top curve; z=1, bottom line).

原文：https://cloud.tencent.com/developer/article/1521677

四、自适应调节

在 gamma 和 log 的基础下，有一些论文提出自适应调节的方法，记录一下。这里的自适应的手段也可以类比到其他方法上。

全局均值作为参数

论文：Adaptive Local Tone Mapping Based on Retinex for High Dynamic Range Images

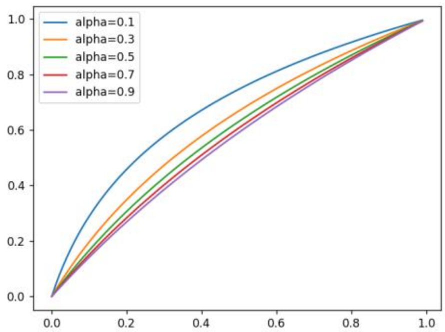
这篇文章的公式如下，其实就是 Log 变换中 α 替换成全局均值 $\bar{L}_w$：

$$L_g(x,y)=\frac{\log(L_w(x,y)/\bar{L}_w+1)}{\log(L_{w\max}/\bar{L}_w+1)}, \qquad (4)$$

论文用的均值是取完 log 后的均值，图中的 δ 是一个很小的数，防止溢出的。这个均值计算方式也没什么特别的科学依据，知道这个就好。

$$\bar{L}_w=\exp\left(\frac{1}{N}\sum_{x,y}\log(\delta+L_w(x,y))\right), \qquad (5)$$

回顾最开始说的 Log 函数图像（下图），能看到这里用均值的意义，如果图片整体偏暗，那么均值小从而图片变换会狠一点：



这个方法有一个弊端，有些图片对比度不好但平均值接近 128。比如下面的图，上半部分为天空，下半部分比较暗，且基本各占一半，因此其平均值非常靠近 128，此时没有均值没有太大作为调节参数的意义。

像素灰度值作为参数

论文：Local Color Correction

论文的原公式如下：

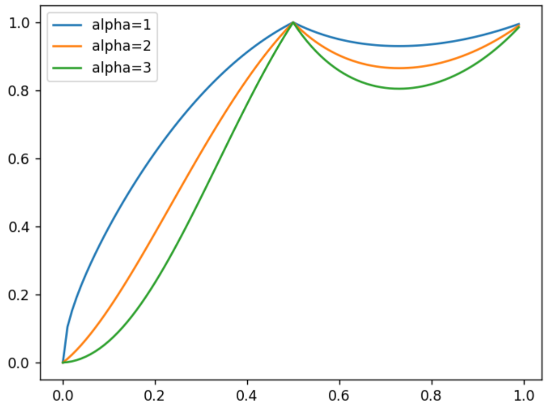
$$gamma(x,y)=\frac{128-gaussian(in(x,y))}{128}*2$$

$$out(x,y)=255*(\frac{in(x,y)}{255})^{gamma(x,y)}$$

本篇文章默认像素值是 0-1 之间，变换一下公式如下，为了便于讲述，这里用 α 代替 2，然后用原像素代替高斯后的像素值：

$$y = x^{(1-x)*\alpha}$$

研究了一下，说实话这个公式真的没道理，它的变换取下如图：



这个都不是单调函数，但是实验中用了这个处理一张图片，还可以，但估计也是凑巧。

之后还有一篇文章在此基础上提出改进：

1. 用双边滤波代替高斯模糊。
2. gamma 计算不用固定的系数 2（经验值），而是用一个 α 值来代替。见下文

当图像的整体平均值小于 128 的时候，$\alpha = ln(I_{avg}/255) \ / \ ln(0.5)$;当图像的整体平均值大于 128 的时候，$\alpha = ln(1 - I_{ave}/255) \ / \ ln(0.5)$。哈哈，看到这里是不是想起了上一小节。没错，其实就是上一小节的方式，同样的，对于上面那种天空图，效果也一般。

某个代码块：

```
import matplotlib.pyplot as plt
import numpy as np

# x 0-1 每隔 0.01, L 取 0.3、0.5、0.8, 绘制 y=log(x/L + 1)/log(1/L + 1) 图像
x = np.arange(0, 1, 0.01)
for L in [0.1, 0.3, 0.5, 0.7, 0.9]:
    y = np.log(x / L + 1) / np.log(1 / L + 1)
    plt.plot(x, y, label=f'alpha={L}')

plt.legend()
plt.show()
```

参考链接

1. https://www.cnblogs.com/yfor1008/p/15173963.html