

# Auto gamma软件算法流程理解

2022年9月2日 3:12

## Auto gamma软件算法流程

TravisYang

### 原理概述

电-光转换实质就可以是RGB坐标系到CIE1931XYZ坐标系的转换。知道了这个原理，只要知道这个屏电光特性，就可以简单的利用一个3\*3矩阵进行转换，从而得到需要的RGB值。另外，还要考虑此非线性特性——也就是gamma特性，需要进行反gamma校正，得到线性的RGB值。

我们要做的就是通过测量，计算出这片屏的转换矩阵RGB to XYZ，再求其逆矩阵得到XYZ to RGB，和测量gamma特性。有了这些参数，就可以计算出任意亮度/任意坐标的RGB值。

最后，根据目标色温坐标和目标gamma计算出RGB查找表。

此算法简单便捷，只需测量显示屏的红绿蓝三点的XYZ坐标，得到转换矩阵，再测量最亮白场和若干个灰场的XYZ坐标。

### 软件算法流程

1. 首先通过测量R/G/B三个场的XYZ值，即可直接当作矩阵。——为什么？

因为归一化的 $R=(1, 0, 0)$ ,  $G=(0, 1, 0)$ ,  $B=(0, 0, 1)$ , R/G/B构成的矩阵刚好是主对角线为1单位矩阵，

$$RGB = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

**单位矩阵性质：任何矩阵与单位矩阵相乘都等于本身。**

故当 $A \cdot RGB = XYZ$ ，且RGB是单位矩阵时， $A = XYZ$ 。

2. 测量出转换矩阵A（RGB to XYZ）后，求逆可得到逆矩阵A'（XYZ to RGB）

我们需要的目标 $RGB = A' \cdot target\_XYZ$ ，下面继续求target\_XYZ

3. 我们首先要计算最大可行的target\_XYZ。

target\_x, target\_y 是已知条件; X, Y, Z与x,y的关系如下：

- $Y = L_v$  (亮度)
- $X = Y \cdot target\_x / target\_y$
- $Z = Y \cdot (1 - target\_x - target\_y) / target\_y$

-----

$$\text{设 } A' = \begin{bmatrix} a1 & b1 & c1 \\ a2 & b2 & c2 \\ a3 & b3 & c3 \end{bmatrix}$$

$$\text{target\_XYZ} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

-----

则通过公式  $A' \cdot \text{target\_XYZ} = \text{RGB}$  , 有

$$\begin{aligned} a_1 \cdot X + b_1 \cdot Y + c_1 \cdot Z &= R \leq 1 \\ a_2 \cdot X + b_2 \cdot Y + c_2 \cdot Z &= G \leq 1 \\ a_3 \cdot X + b_3 \cdot Y + c_3 \cdot Z &= B \leq 1 \end{aligned}$$

以上, 我们发现只有Y是未知数, 故可分别求出三个等式的解Y1, Y2, Y3。

**那么, 最大可行亮度值  $Y_{\max} = \min \{ Y_1, Y_2, Y_3 \}$**

4. 得到最大可行亮度值  $Y_{\max}$  后, 即可按10%亮度递减, 算出11个梯度的XYZ

```
for (i=0, i ≤ 10, i++)
{
    Yi = Y_max * (i/10)^gamma_target
    Xi = (Yi · target_x) / target_y
    Zi = Yi · (1 - target_x - target_y) / target_y
}
```

5. 得到以上11个梯度的XYZ后, 再通过  $\text{RGB} = A' \cdot \text{target\_XYZ}$  得到RGB (0~1)

6. 测量出11个灰阶的XYZ, 再插值为256个

7. 通过gamma算法求出11个点实际的RGB:

```
for n=1:3
    for m=1:11
        temp=abs(XYZ256(n,:)-XYZ(n,m));
        [M,l]=min(temp);    //找出目标XYZ里的点对应XYZ256最接近的点,l是Index,M是Min用
        不到
        panelga=log(XYZ256(n,l)/XYZ256(n,256))/log(l/255);    //求这个点的gamma值
        RGB(n,m)=RGB(n,m)^(1/panelga);    //用反gamma求出实际的RGB值
    end
end
```

测量XYZ256矩阵:

X1	X2	X3	...	...	X256
Y1	Y2	Y3	...	...	Y256
Z1	Z2	Z3	...	...	Z256

原目标XYZ矩阵:

X1	X2	X3	...	X11
Y1	Y2	Y3	...	Y11
Z1	Z2	Z3	...	Z11

实际RGB:

R1(0~1)	R2	R3	...	R11
G1(0~1)	G2	G3	...	G11
B1(0~1)	B2	B3	...	B11

. \*255 =

最终RGB

R1(0~255)	R2	R3	...	R11
G2(0~255)	G2	G3	...	G11
B3(0~255)	B2	B3	...	B11

最后，将这11个实际算出的RGB中做插值运算，扩展为256个RGB值，  
将结果导出至gamma.ini