CCM矩阵

色彩校正的时候通常使用CCM矩阵作为从线性化的输入色彩空间到线性绝对色彩空间变换的近似。

CCM矩阵的形状通常3×3和4×3两种[1-2]。前者对色彩的数值进行线性变换，而后者做仿射变换。换言之，色彩空间在前者的变换后保持原点不变，而后者可以发生平移。可见，3×3的CCM矩阵的变换集合是4×3的真子集，这意味着使用4×3CCM矩阵拟合的解集更大。然而，最新的论文更愿意使用3×3的CCM矩阵而非后者。

3x3 CCM矩阵

用以拟合和推断

由于在拟合时，输入可能是色彩的列表，而在推断时，是一副图像。我们先以简单的二维矩阵为例，设输入的线性化色彩空间是：

$$S_l = \begin{bmatrix} R_{l1} & G_{l1} & B_{l1} \\ R_{l2} & G_{l2} & B_{l2} \\ \cdots & \cdots & \cdots \end{bmatrix}$$

输出的线性绝对色彩空间为：

$$D_l = \begin{bmatrix} R'_{l1} & G'_{l1} & B'_{l1} \\ R'_{l2} & G'_{l2} & B'_{l2} \\ \cdots & \cdots & \cdots \end{bmatrix}$$

则有：

$$D_l = S_l \times M_{CCM}$$

上式中，乘法符号被显示的写出来了。此时，该乘法与矩阵乘法相同，在numpy中可以使用numpy.dot, numpy.matmul或者@符号来实现。此外，上式CCM矩阵为：

$$M_{CCM} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

如果输入是多阶张量，则表示为：

$$T_D = T_S \times M_{CCM}$$

此时的乘法是扩展的矩阵乘法，依然可以使用numpy.dot, numpy.matmul或者@符号来实现，此时numpy.dot, numpy.matmul的结果是一致的。

初始值

程序提供了2种方法。

一种是白平衡法[1]。初始值设为：

$$M_{CCM} = \begin{bmatrix} k_R & 0 & 0 \\ 0 & k_G & 0 \\ 0 & 0 & k_B \end{bmatrix}$$

其中：

$$k_R = mean(R'_{li})/mean(R_{li})$$
$$k_R = mean(G'_{li})/mean(G_{li})$$
$$k_R = mean(B'_{li})/mean(B_{li})$$

第二种是最小二乘法，即距离函数使用线性RGB距离后的最优解。初始值为：

$$M_{CCM} = (S_l^T S_l)^{-1} S_l^T D_l$$

在numpy中使用numpy.linalg.lstsq实现，这里可以简单表示为：

$$M_{CCM} = ls(S_l, D_l)$$

如果有权重：

$$w = [w_1, w_2, \dots]$$

记：

$$W = \begin{bmatrix} \sqrt{w_1} & 0 & \dots \\ 0 & \sqrt{w_2} & \dots \\ \dots & \dots & \dots \end{bmatrix}$$

则：

$$M_{CCM} = ls(W \times S_l, W \times D_l)$$

逆

在评估模型时，会求色彩在乘以CCM前的原像。对于3x3矩阵，矩阵的逆存在，直接使用逆来计算原像。

$$S_l = D_l M_{CCM}^{-1}$$

4x3 CCM矩阵

用以拟合和推断

4x3 CCM矩阵可表示为：

$$M_{CCM} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

记：

$$S_l^+ = [\, S_l \quad 1_{col} \,] = \begin{bmatrix} R_{l1} & G_{l1} & B_{l1} & 1 \\ R_{l2} & G_{l2} & B_{l2} & 1 \\ \dots & \dots & \dots & \dots \end{bmatrix}$$

则有：

$$D_l = S_l^+ \times M_{CCM}$$

对于多阶张量：

$$T_D = T_S^+ \times M_{CCM}$$

初始值

程序提供了2种方法。

一种是白平衡法[1]。初始值设为：

$$M_{CCM}=\begin{bmatrix} k_R & 0 & 0 \\ 0 & k_G & 0 \\ 0 & 0 & k_B \\ 0 & 0 & 0 \end{bmatrix}$$

其中：

$$\begin{aligned} k_R &= mean(R'_{li})/mean(R_{li}) \\ k_R &= mean(G'_{li})/mean(G_{li}) \\ k_R &= mean(B'_{li})/mean(B_{li}) \end{aligned}$$

第二种是最小二乘法，即距离函数使用线性RGB距离后的最优解。初始值为：

$$M_{CCM}=(S_l^{+T}S_l^+)^{-1}S_l^{+T}D_l$$

在numpy中使用numpy.linalg.lstsq实现，这里可以简单表示为：

$$M_{CCM}=ls(S_l^+,D_l)$$

如果有权重：

$$w=[w_1,w_2,\dots]$$

记：

$$W=\begin{bmatrix} \sqrt{w_1} & 0 & \dots \\ 0 & \sqrt{w_2} & \dots \\ \dots & \dots & \dots \end{bmatrix}$$

则：

$$M_{CCM}=ls(W\times S_l^+,W\times D_l)$$

逆

在评估模型时，会求色彩在乘以CCM前的原像。由于：

$$D_l=S_l^+M_{CCM}=\begin{bmatrix} S_l & 1_{col} \end{bmatrix}\begin{bmatrix} Up_{(3,3)} \\ Down_{(1,3)} \end{bmatrix}=S_lUp_{(3,3)}+1_{col}Down_{(1,3)}$$

因此有：

$$S_l=(D_l-1_{col}Down_{(1,3)})Up_{(3,3)}^{-1}$$

也可以等价的表示成：

$$S_l=\begin{bmatrix} D_l & 1_{col} \end{bmatrix}\begin{bmatrix} Up_{(3,3)}^{-1} \\ -Down_{(1,3)}Up_{(3,3)}^{-1} \end{bmatrix}$$

参考文献

1. https://www.imatest.com/docs/colormatrix/
2. http://www.its.bldrdoc.gov/pub/ntia-rpt/04-406/